



Chapter 9 Graph Neural Networks

第9章 图神经网络(1)



李政伟

中国矿业大学 计算机科学与技术学院

基于异质图注意力网络的 miRNA 与 疾病关联预测算法

李政伟^{1,2}, 李佳树^{1,2}, 尤著宏³, 聂 茹², 赵 欢², 钟堂波²



Volume 23, Issue 2
March 2022

Predicting miRNA - disease associations based on graph random propagation network and attention network

Get access >

Tangbo Zhong, Zhengwei Li, Zhu-Hong You, Ru Nie, Huan Zhao

Briefings in Bioinformatics, Volume 23, Issue 2, March 2022, bbab589,

<https://doi.org/10.1093/bib/bbab589>

Published: 25 January 2022 Article history ▼

Molecular Therapy

ORIGINAL ARTICLE | VOLUME 30, ISSUE 4, P1775-1786, APRIL 06, 2022

Hierarchical graph attention network for miRNA-disease association prediction

Zhengwei Li ✉ • Tangbo Zhong • Deshuang Huang ✉ • Zhu-Hong You ✉ • Ru Nie ✉

Published: February 01, 2022 • DOI: <https://doi.org/10.1016/j.ymthe.2022.01.041> • Check for updates



Purchase



Subscribe



Save



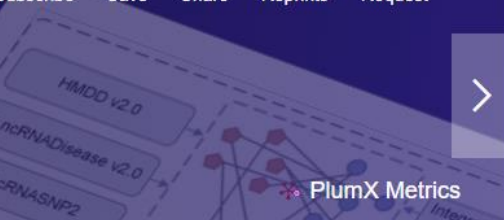
Share



Reprints



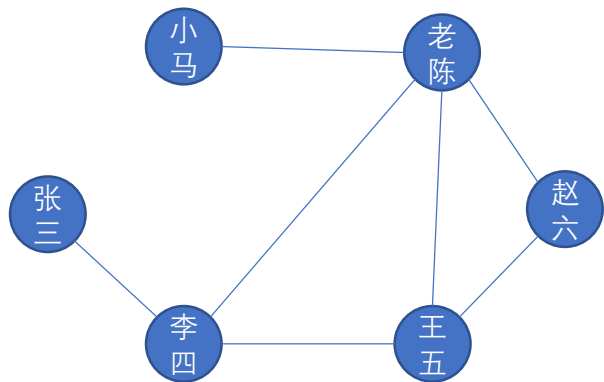
Request



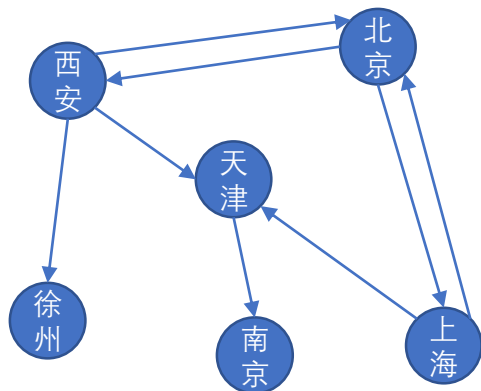
PlumX Metrics

图(Graph)

图 $G = \{\mathcal{V}, \mathcal{E}\}$ 包含一个节点集合 $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ 和一个边集合 $\mathcal{E} = \{e_1, e_2, \dots, e_M\}$ 。



朋友关系网络



城市之间的航线图

加权图
无权图
连通图
非连通图

图的基本术语及概念

- 如果存在一条边连接顶点 v_i 和 v_j ，则称 v_j 是 v_i 的**邻居**，反之亦然。
- 顶点 v_i 的**邻域**（集合） $N(v_i) = \{v_j | \{v_i, v_j\} \in \mathcal{E} \text{ or } \{v_j, v_i\} \in \mathcal{E}\}$
- 以顶点 v_i 为端点的边的数目称为 v_i 的**度**，记为 $\deg(v_i) = |N(v_i)|$
- 有向图中有**出度**和**入度**之分，顶点的度数等于该顶点的出度与入度之和。
- **定理**：图中所有结点的度数之和是边数的两倍。

$$\sum_{v_i \in \mathcal{V}} d(v_i) = 2|\mathcal{E}|$$

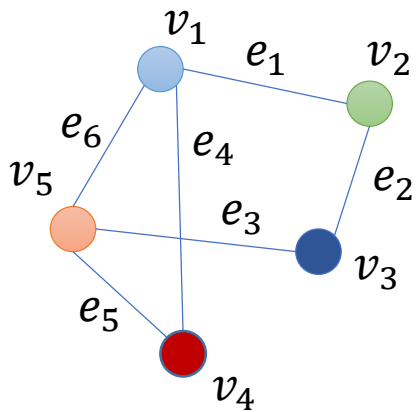
图的基本术语及概念

- 若从顶点 v_i 出发，沿着一些边经过一些顶点 $v_{p1}, v_{p2}, \dots, v_{pm}$ 达到顶点 v_j ，则称边序列 $P_{ij} = (v_{ip1}, v_{p1p2}, \dots, v_{pmj})$ 为从顶点 v_i 到 v_j 的一条**路径**。
- **途径(walk)**：结点和边的交替序列。
- **迹(trail)**：边各不相同的路径。
- **路径 (Path)**：顶点各不相同的途径。
- **路径的长度**：路径中**边的数目**， $L(P_{ij}) = |P_{ij}|$ 。
- **顶点间的距离**：若存在至少一条路径由顶点 v_i 到达顶点 v_j ，则定义 v_i 到 v_j 的距离为： $d(v_i, v_j) = \min |P_{ij}|$

图的基本术语及概念

- **k阶邻居**: 若 $d(v_i, v_j) = k$, 则称 v_j 是 v_i 的 k 阶邻居。
- **k阶子图**: 顶点 v_i 的 k 阶子图为: $G_{v_i}^{(k)} = (V', E'), V' = \{v_j \mid d(v_i, v_j) \leq k\}$
- **定理**: 对于图 G 及其邻接矩阵 A , 用 A^n 表示 A 的 n 次幂。 A^n 的第 i 行和第 j 列等于长度为 n 的 v_i 到 v_j 途径的个数。(提示: 可用数学归纳法证明)
- 若图 $G' = (\mathcal{V}', \mathcal{E}')$ 的顶点集和边集分别是图 $G = \{\mathcal{V}, \mathcal{E}\}$ 的顶点集和边集的子集, $\mathcal{V}' \subset \mathcal{V}, \mathcal{E}' \subset \mathcal{E}$, 则称图 G' 是图 G 的**子图**(subgraph)。
- **连通分量**: 给定图 $G = \{\mathcal{V}, \mathcal{E}\}$, 如果子图 $G' = (\mathcal{V}', \mathcal{E}')$ 中任意一对节点之间都至少存在一条路, 且 \mathcal{V}' 中的节点不与任何中 $\mathcal{V} - \mathcal{V}'$ 的节点相连, 那么 G' 是一个连通分量。
- **连通图**: 如果图 $G = \{\mathcal{V}, \mathcal{E}\}$ 只有一个连通分量, 那么 G 是连通图。

图(Graph)



$$\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$$

$$\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5, e_6\}$$

邻接矩阵 (adjacency matrix)

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

度矩阵 (degree matrix)

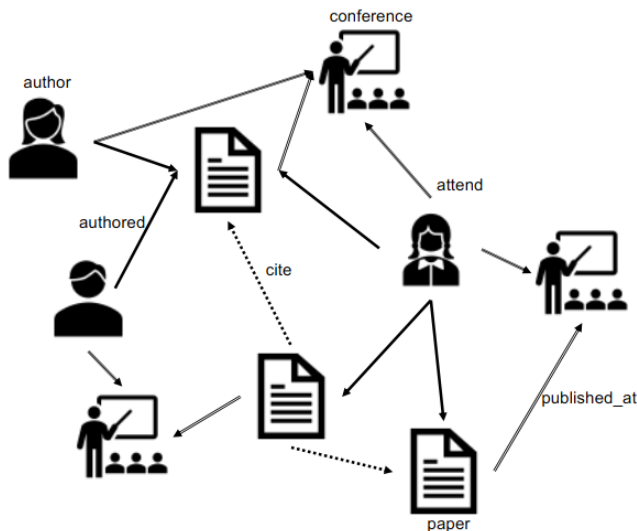
$$d(v_i) = \sum_{v_j \in \mathcal{V}} \mathbb{I}((v_i, v_j) \in \mathcal{E}) = \sum_{j=1}^N A_{ij}$$

$$D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Complex Graphs

■ Heterogeneous Graphs

A heterogeneous graph \mathcal{G} consists of a set of nodes $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ and a set of edges $\mathcal{E} = \{e_1, e_2, \dots, e_M\}$ where **each node** and **each edge** are associated with a **type**. Let \mathcal{T}_n denote the set of node types and \mathcal{T}_e indicate the set of edge types. There are two mapping functions $\varphi_n: \mathcal{V} \rightarrow \mathcal{T}_n$ and $\varphi_e: \mathcal{E} \rightarrow \mathcal{T}_e$ that map each node and each edge to their types, respectively.



A heterogeneous academic graph

Complex Graphs

■ Bipartite Graphs

Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, it is bipartite if and only if $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$ and $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ and $v_e^1 \in \mathcal{V}_1$ while $v_e^2 \in \mathcal{V}_2$ for all $e = (v_e^1; v_e^2) \in \mathcal{E}$.



An e-commerce bipartite graph

Complex Graphs

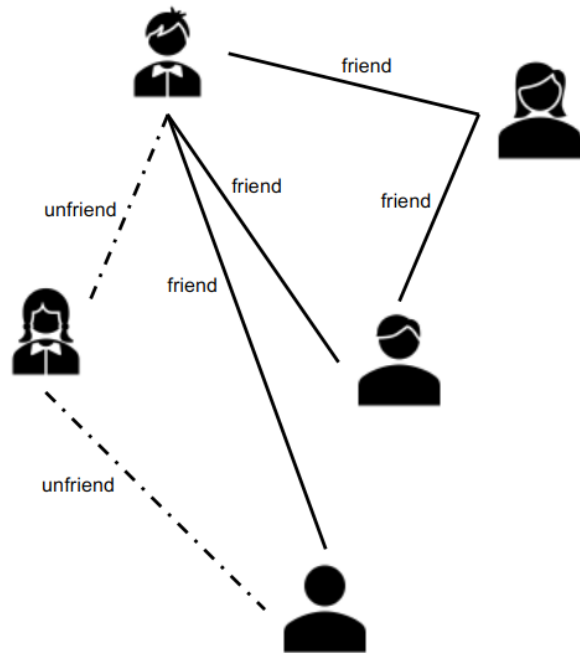
■ Multi-dimensional graphs

A multi-dimensional graph consists of a set of N nodes $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ and D sets of edges $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_D\}$. Each edge set \mathcal{E}_d describes the d -th type of relations between the nodes in the corresponding d -th dimension. These D types of relations can also be expressed by D adjacency matrices $A^{(1)}, A^{(2)}, \dots, A^{(D)}$. In the dimension d , its corresponding adjacency matrix $A^{(d)} \in \mathbb{R}^{N \times N}$ describes the edges \mathcal{E}_d between nodes in \mathcal{V} . Specifically, the i, j -th element of $A^{(d)}$, denoted as $A_{i,j}^{(d)}$, equals to 1 only when there is an edge between nodes v_i and v_j in the dimension d (or $\{v_i, v_j\} \in \mathcal{E}_d$), otherwise 0.

Complex Graphs

■ Signed Graphs

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-\}$ be a signed graph, where $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ is the set of N nodes while $\mathcal{E}^+ \subset \mathcal{V} \times \mathcal{V}$ and $\mathcal{E}^- \subset \mathcal{V} \times \mathcal{V}$ denote the sets of positive and negative edges, respectively. Note that an edge can only be either positive or negative, i.e., $\mathcal{E}^+ \cap \mathcal{E}^- = \emptyset$. These positive and negative edges between nodes can also be described by a signed adjacency matrix \mathbf{A} , where $A_{ij} = 1$ only when there is a positive edge between node v_i and node v_j , $A_{ij} = -1$ denotes a negative edge, otherwise $A_{ij} = 0$.



An illustrative signed graph

Complex Graphs

■ Hypergraphs

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ be a hypergraph, where \mathcal{V} is a set of N nodes, \mathcal{E} is a set of **hyperedges** and $\mathbf{W} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$ is a diagonal matrix with $W[j, j]$ denoting the weight of the hyperedge \mathcal{E}_j . The hypergraph \mathcal{G} can be described by an incidence matrix $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$, where $H_{i,j} = 1$ only when the node v_i is incident to the edge e_j . For a node v_i , its degree is defined as $d(v_i) = \sum_{j=1}^{|\mathcal{E}|} H_{i,j}$, while the degree for a hyperedge e_j is defined as $d(e_j) = \sum_{i=1}^{|\mathcal{V}|} H_{i,j}$. Furthermore, we use \mathbf{D}_e and \mathbf{D}_v to denote the diagonal matrices of the edge and node degrees, respectively.



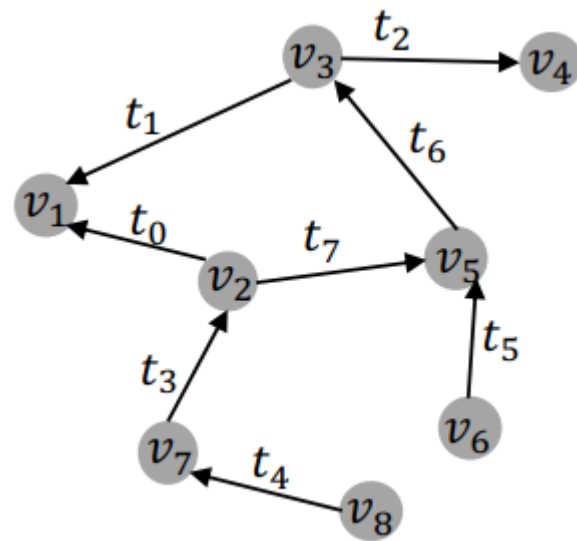
An illustrative hypergraph

Complex Graphs

Dynamic Graphs

A dynamic graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ consists of a set of nodes $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ and a set of edges $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_D\}$ where each node and/or each edge is associated with a **timestamp** indicating the time it emerged. Specifically, we have two mapping functions ϕ_v and ϕ_e mapping each node and each edge to their emerging timestamps.

(Discrete Dynamic Graphs) A discrete dynamic graph consists of T graph snapshots, which are observed along with the evolution of a dynamic graph. Specifically, the T graph snapshots can be denoted as $\{\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_T\}$ where \mathcal{G}_0 is the graph observed at time 0.

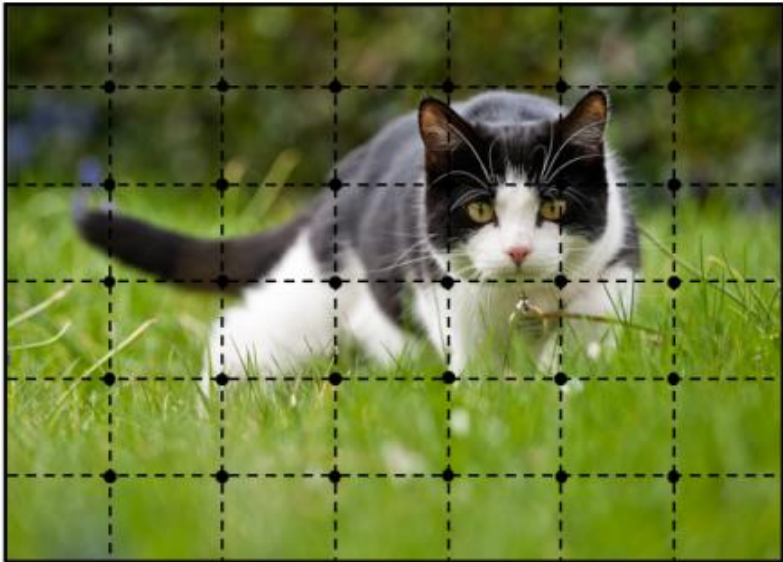


An illustrative example of dynamic graphs

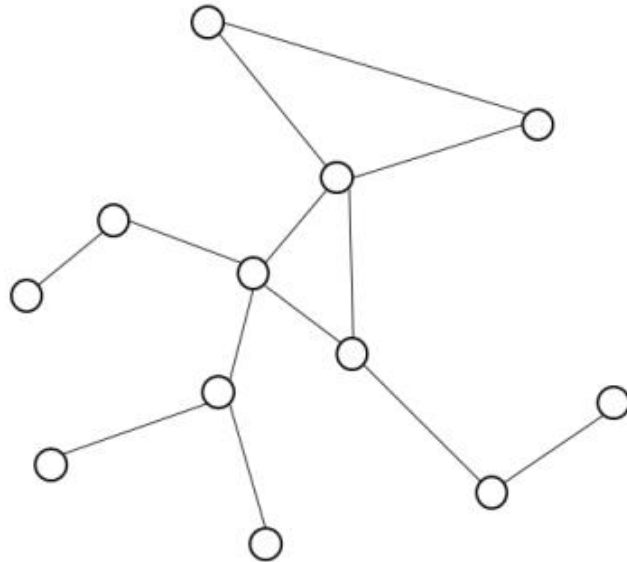
|| CNN的由来

- 2005年，Marco Gori等人首次提出图神经网络的概念。在此之前，处理图数据的方法是在数据预处理阶段将图转换为向量表示。这种处理方法最大的问题就是图中的结构信息可能会丢失，并且得到的结果会严重依赖于对图的预处理。
- 在2009年的两篇论文中又进一步阐述了图神经网络，并提出了一种监督学习的方法来训练GNN。
- 2013年Bruna等首次将卷积引入GNN中，基于频域卷积操作的概念提出一种图卷积网络模型，首次将可学习卷积操作作用于图数据之上。
- 2016年，Kipf等将频域图卷积的定义进行简化，使得图卷积的操作能够在空域进行，极大地提升了图卷积模型的计算效率，
- 更多的基于空域图卷积的神经网络模型的变体被提出，统称为GNN。

Graph Neural Networks



Image(Euclidean space)

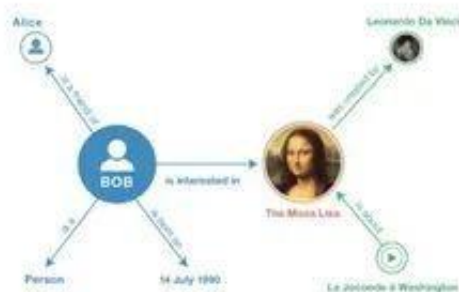


Graph(non-Euclidean space)

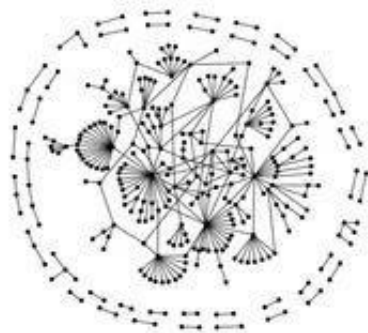
为什么需要图神经网络?



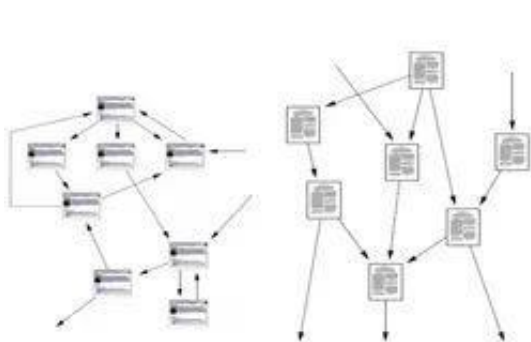
Social networks



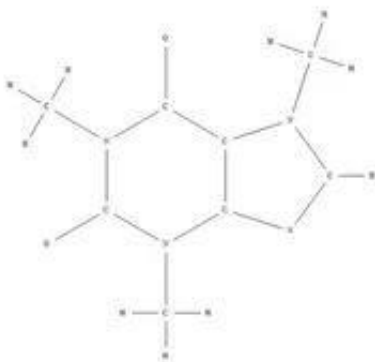
Knowledge graphs



Biological networks

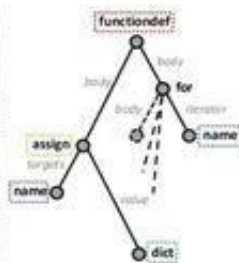


Complex Systems



Molecules

```
def encode(obj):  
    """  
    Encode a Python object  
    dictionary containing complex values  
    into a form that can be serialized  
    using JSON.  
    """  
    e = {}  
    for key, value in obj.items():  
        if isinstance(value, dict):  
            e[key] = encode(value)  
        elif isinstance(value, complex):  
            e[key] = {'type': 'complex',  
                    'r': value.real,  
                    'i': value.imag}  
    return e  
  
import ast  
tree = ast.parse("x = 1")
```



Code

矩阵乘法的三种方式

设矩阵 $A \in R^{M \times K}$, $B \in R^{K \times N}$, 则 $C = AB$ 的三个计算视角。

1、内积视角（正常）

$$C_{ij} = A_{i,:} B_{:,j}$$

2、行向量视角（对于理解空域图卷积有很大意义）

将B看作行向量矩阵，A为系数矩阵

$$C_{i,:} = \sum_{k=1}^K A_{ik} B_{k,:}$$

3、列向量视角

将A看作列向量矩阵，B为系数矩阵

$$C_{:,j} = \sum_{k=1}^K B_{kj} A_{:,k}$$

例9.1, 矩阵乘法的视角(续)

$$\text{设 } A = \begin{bmatrix} 1 & -1 & 2 \\ 0 & 2 & 1 \end{bmatrix}, B = \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & -1 \end{bmatrix}, C = AB = \begin{bmatrix} 3 & -3 \\ -2 & 1 \end{bmatrix}$$

行视角, 以 C 的第一行计算过程为例

$$\begin{bmatrix} 3 & -3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & -1 \end{bmatrix}$$

$$= 1 \begin{bmatrix} 2 & 0 \end{bmatrix} + (-1) \begin{bmatrix} -1 & 1 \end{bmatrix} + 2 \begin{bmatrix} 0 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 \end{bmatrix} + \begin{bmatrix} 1 & -1 \end{bmatrix} + \begin{bmatrix} 0 & -2 \end{bmatrix}$$

$$C_{i,:} = \sum_{k=1}^K A_{ik} B_{k,:}$$

例9.1，矩阵乘法的视角(续)

$$\text{设 } A = \begin{bmatrix} 1 & -1 & 2 \\ 0 & 2 & 1 \end{bmatrix}, B = \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & -1 \end{bmatrix}, C = AB = \begin{bmatrix} 3 & -3 \\ -2 & 1 \end{bmatrix}$$

列视角，以 C 的第一列计算过程为例

$$\begin{bmatrix} 3 \\ -2 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 2 \\ 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}$$

$$= 2 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + (-1) \begin{bmatrix} -1 \\ 2 \end{bmatrix} + 0 \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ -2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$C_{:,j} = \sum_{k=1}^K B_{kj} A_{:,k}$$

9-2 图信号与拉普拉斯矩阵

设图 $G = (V, E)$ ，顶点数为 N ，图信号可以表示成**向量**形式：

$$\mathbf{x} = [x_1, x_2, \dots, x_N]^T$$

其中， x_i 表示节点 v_i 上信号的强度。

■ 普通形式的拉普拉斯矩阵

$$L = D - A$$

矩阵元素：

$$L_{ij} = \begin{cases} \text{diag}(v_i) & i = j \\ -1 & i \neq j \text{ 且 } v_i \text{ 与 } v_j \text{ 相邻} \\ 0 & \text{其他} \end{cases}$$

其中， $\text{diag}(v_i)$ 表示顶点 v_i 的度。

对称归一化的拉普拉斯矩阵

■ 对称归一化拉普拉斯矩阵(Symmetric normalized Laplacian)

$$L_{sym} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

$$\begin{aligned} L_{sym} &= D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}} \\ &= D^{-\frac{1}{2}}DD^{-\frac{1}{2}} - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \\ &= I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \end{aligned}$$

矩阵元素：

$$L_{sym}[i, j] = \begin{cases} 1 & i = j \text{ 且 } \text{diag}(v_i) \neq 0 \\ -\frac{1}{\sqrt{\text{diag}(v_i)\text{diag}(v_j)}} & i \neq j \text{ 且 } v_i \text{ 与 } v_j \text{ 相邻} \\ 0 & \text{其他} \end{cases}$$



随机游走归一化拉普拉斯矩阵

■ 随机游走归一化拉普拉斯矩阵(Random walk normalized Laplacian)

$$L_{rw} = D^{-1}L = I_N - D^{-1}A$$

■ 矩阵元素

$$L_{rw} [i, j] = \begin{cases} 1 & i = j \text{ 且 } \text{diag}(v_i) \neq 0 \\ -\frac{1}{\text{diag}(v_i)} & i \neq j \text{ 且 } v_i \text{ 与 } v_j \text{ 相邻} \\ 0 & \text{其他} \end{cases}$$

例9-2，拉普拉斯矩阵的计算

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$L = D - A = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

$$D^{-1/2} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = \begin{bmatrix} 1 & -\frac{1}{\sqrt{6}} & 0 & 0 & -\frac{1}{\sqrt{6}} & 0 \\ -\frac{1}{\sqrt{6}} & 1 & -\frac{1}{\sqrt{6}} & 0 & -\frac{1}{\sqrt{9}} & 0 \\ 0 & -\frac{1}{\sqrt{6}} & 1 & -\frac{1}{\sqrt{6}} & 0 & 0 \\ -\frac{1}{\sqrt{6}} & 0 & -\frac{1}{\sqrt{6}} & 1 & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ 0 & -\frac{1}{\sqrt{9}} & 0 & -\frac{1}{\sqrt{9}} & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{\sqrt{6}} & 0 & 1 \end{bmatrix}$$



图信号上的总变差

$$L\mathbf{x} = (D - A)\mathbf{x} = [\dots, \sum_{v_j \in N(v_i)} (x_i - x_j), \dots]^T$$

$$\mathbf{x}^T L\mathbf{x} = \sum_{v_i} \sum_{v_j \in N(v_i)} x_i (x_i - x_j)$$

$$= \sum_{e_{ij} \in E} (x_i - x_j)^2$$

图信号上的总变差

$$TV(\mathbf{x}) = \mathbf{x}^T L\mathbf{x} = \sum_{e_{ij} \in E} (x_i - x_j)^2$$

各条边上信号的差值平方加和，用于衡量图信号的整体平滑度。



拉普拉斯矩阵的半正定性证明

$$\mathbf{x}^T L \mathbf{x} \geq 0$$

$$\mathbf{x}^T L \mathbf{x} = \mathbf{x}^T D \mathbf{x} - \mathbf{x}^T A \mathbf{x}$$

$$= \mathbf{x}^T * \text{diag}(\mathbf{d}) * \mathbf{x} - \mathbf{x}^T A \mathbf{x}$$

$$= \sum_{i=1}^N d_i x_i^2 - \sum_{j=1}^N \left[\sum_{i=1}^N x_i * a_{ij} \right] x_j$$

$$= \sum_{i=1}^N d_i x_i^2 - \sum_{i,j=1}^N x_i * x_j * a_{ij}$$

$$= \frac{1}{2} \left[\sum_{i=1}^N d_i x_i^2 - 2 \sum_{i,j=1}^N x_i * x_j * a_{ij} + \sum_{i=1}^N d_j x_j^2 \right]$$

$$= \frac{1}{2} \sum_{i=1}^N a_{ij} (x_i - x_j)^2$$

9-3 图傅里叶变换

$$L = V\Lambda V^{-1} = V \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} V^T$$

$$= \begin{bmatrix} \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_N \\ \vdots & \vdots & \cdots & \vdots \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix} \begin{bmatrix} \cdots & \boldsymbol{v}_1 & \cdots \\ \cdots & \boldsymbol{v}_2 & \cdots \\ \cdots & \vdots & \cdots \\ \cdots & \boldsymbol{v}_N & \cdots \end{bmatrix}$$

其中, $L \in R^{N \times N}$, $V \in R^{N \times N}$, $VV^T = I$ 。

λ_k 是 V 的第 k 个特征值, \boldsymbol{v}_i 为对应的特征向量(傅里叶基)。

$L\mathbf{1} = 0$, 因此 L 具有最小的特征值0。



图傅里叶变换

图傅里叶变换 (GFT)

$$\tilde{x}_k = \sum_{i=1}^N V_{ki}^T x_i = \langle \mathbf{v}_k, \mathbf{x} \rangle$$

逆图傅里叶变换 (IGFT)

$$x_k = \sum_{i=1}^N V_{ki} \tilde{x}_i$$

其中, \tilde{x}_k 为**傅里叶系数**。

$$\tilde{\mathbf{x}} = \mathbf{V}^T \mathbf{x}, \quad \tilde{\mathbf{x}} \in \mathbb{R}^N$$

$$\mathbf{x} = \mathbf{V} \tilde{\mathbf{x}}$$

所有傅里叶系数合称为 $\tilde{\mathbf{x}}$ 的**频谱**。

$$\mathbf{V} \tilde{\mathbf{x}} = \mathbf{V} \mathbf{V}^T \mathbf{x}$$

$$\mathbf{x} = \mathbf{V} \tilde{\mathbf{x}}, \quad \mathbf{x} \in \mathbb{R}^N$$

$$= \begin{bmatrix} \vdots & \vdots & \cdots & \vdots \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N \\ \vdots & \vdots & \cdots & \vdots \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_N \end{bmatrix}$$

$$= \tilde{x}_1 \mathbf{v}_1 + \tilde{x}_2 \mathbf{v}_2 + \dots + \tilde{x}_N \mathbf{v}_N$$

$$= \sum_{k=1}^N \tilde{x}_k \mathbf{v}_k$$



图傅里叶变换

对总变差进行重新改写：

$$TV(\mathbf{x}) = \mathbf{x}^T L \mathbf{x} = \mathbf{x}^T V \Lambda V^T \mathbf{x}$$

$$= (V \tilde{\mathbf{x}})^T V \Lambda V^T (V \tilde{\mathbf{x}})$$

$$= \tilde{\mathbf{x}}^T V^T V \Lambda V^T V \tilde{\mathbf{x}}$$

$$= \tilde{\mathbf{x}}^T \Lambda \tilde{\mathbf{x}}$$

$$= \sum_{k=1}^N \lambda_k \tilde{x}_k^2$$

结论：总变差是图的所有特征值（图信号的频率）的一个线性组合，权重是图信号相对应的傅里叶系数的平方。

什么样的图信号具有最小的总变差

将图信号限定为单位向量，特征值 $\lambda_1, \lambda_2, \dots, \lambda_N$ 按升序排列。

$$TV(\mathbf{x}) = \sum_{k=1}^N \lambda_k \tilde{x}_k^2$$

$$\tilde{\mathbf{x}} = \mathbf{V}^T \mathbf{x}, \quad \tilde{\mathbf{x}} \in \mathbb{R}^N$$

总变差取最小值的条件

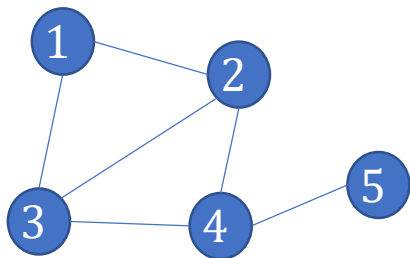
图信号与最小特征值 λ_1 对应的特征向量 \mathbf{v}_1 完全重合此时仅有 $\tilde{x}_1 \neq 0$ ，其他项的傅里叶系数为0，总变差 $TV(\mathbf{v}_1) = \lambda_1$ 。

选择一组彼此正交的图信号，使得各自的总变差依次取得最小值，那么这组图信号就是 $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_N$ 。

$$\lambda_k = \min_{\mathbf{x}: \|\mathbf{x}\|=1, \mathbf{x} \perp \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{k-1}} \mathbf{x}^T \mathbf{L} \mathbf{x}$$



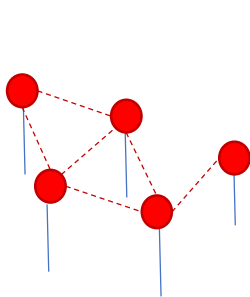
例9-3， 计算图拉普拉斯矩阵的特征值和特征向量



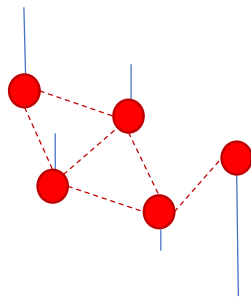
$$V = \begin{bmatrix} -0.447 & 0.438 & -0.703 & 0 & 0.338 \\ -0.447 & 0.256 & 0.242 & 0.707 & -0.419 \\ -0.447 & 0.256 & 0.242 & -0.707 & -0.419 \\ -0.447 & -0.138 & 0.536 & 0 & 0.702 \\ -0.447 & -0.811 & -0.318 & 0 & -0.202 \end{bmatrix}$$

$$\Lambda = \text{diag}(0 \quad 0.8299 \quad 2.689 \quad 4 \quad 4.481)$$

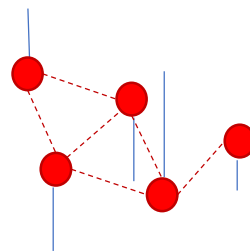
$$L = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$



第1列



第2列



第5列

9-4 图滤波器

设图滤波操作:

$$\mathbf{y} = H\mathbf{x} = \sum_{k=1}^N (h(\lambda_k) \tilde{x}_k) \mathbf{v}_k$$

$$= \begin{bmatrix} \vdots & \vdots & \cdots & \vdots \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N \\ \vdots & \vdots & \cdots & \vdots \end{bmatrix} \begin{bmatrix} h(\lambda_1) \tilde{x}_1 \\ h(\lambda_2) \tilde{x}_2 \\ \cdots \\ h(\lambda_N) \tilde{x}_N \end{bmatrix} = V \begin{bmatrix} h(\lambda_1) & & & \\ & h(\lambda_2) & & \\ & & \ddots & \\ & & & h(\lambda_N) \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \cdots \\ \tilde{x}_N \end{bmatrix} = V \begin{bmatrix} h(\lambda_1) & & & \\ & h(\lambda_2) & & \\ & & \ddots & \\ & & & h(\lambda_N) \end{bmatrix} V^T \mathbf{x}$$

$$H = V \begin{bmatrix} h(\lambda_1) & & & \\ & h(\lambda_2) & & \\ & & \ddots & \\ & & & h(\lambda_N) \end{bmatrix} V^T = V \Lambda_h V^T$$

从算子角度, $H\mathbf{x}$ 描述了作用在每个节点一阶子图上的变换操作。

9-4 图滤波器

滤波器又可定义为：

$$H = h_0 L^0 + h_1 L^1 + \cdots + h_K L^K = \sum_{k=0}^K h_k L^k$$

1) 空域角度

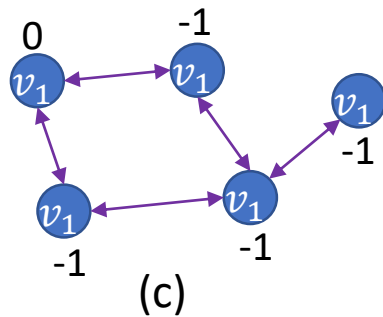
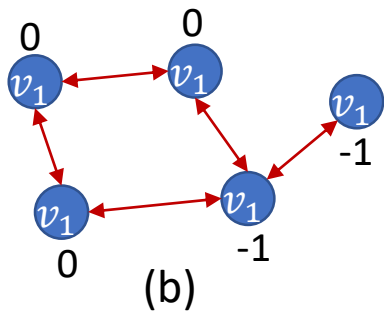
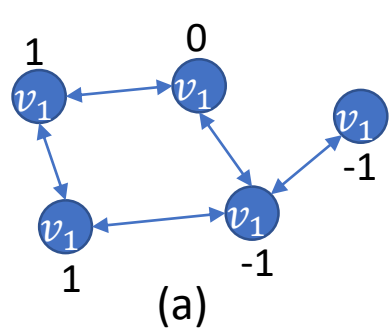
$$\mathbf{y} = H\mathbf{x} = \sum_{k=0}^N h_k L^k \mathbf{x}$$

若设定 $\mathbf{x}^{(k)} = L^k \mathbf{x} = L\mathbf{x}^{(k-1)}$

则

$$\mathbf{y} = \sum_{k=0}^K h_k \mathbf{x}^{(k)}$$

例9.5，图信号滤波在空域上的计算



$$H = A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

给定图信号 $\mathbf{x} = [1 \ 0 \ 0 \ -1 \ -1]^T$ ，系数向量 $\mathbf{h} = [1 \ 0.5 \ 0.5]^T$ ，得到滤波器输出的图信号为 $\mathbf{y} = h_0 \mathbf{x}^{(0)} + h_1 \mathbf{x}^{(1)} + h_2 \mathbf{x}^{(2)}$

$$\mathbf{x}^{(0)} = \mathbf{x} = [1 \ 0 \ 0 \ -1 \ -1]^T, \quad \mathbf{x}^{(1)} = H\mathbf{x}^{(0)} = [0 \ 0 \ 0 \ -1 \ -1]^T, \quad \mathbf{x}^{(2)} = H\mathbf{x}^{(1)} = [0 \ -1 \ -1 \ -1 \ -1]^T$$

$$\mathbf{y} = 1\mathbf{x}^{(0)} + 0.5\mathbf{x}^{(1)} + 0.5\mathbf{x}^{(2)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \\ -1 \end{bmatrix} + 0.5 \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ -1 \end{bmatrix} + 0.5 \begin{bmatrix} 0 \\ 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ -0.5 \\ -0.5 \\ -2 \\ -2 \end{bmatrix}$$

2) 频域角度

由 $L = V\Lambda V^T$, 则

$$H = \sum_{k=0}^K h_k L^k = \sum_{k=0}^K h_k (V\Lambda V^T)^k = V \left(\sum_{k=0}^K h_k \Lambda^k \right) V^T = V \begin{bmatrix} \sum_{k=0}^K h_k \lambda_1^k & & \\ & \sum_{k=0}^K h_k \lambda_2^k & \\ & & \ddots \\ & & & \sum_{k=0}^K h_k \lambda_N^k \end{bmatrix} V^T$$

$$\mathbf{y} = H\mathbf{x} = V \left(\sum_{k=0}^K h_k \Lambda^k \right) V^T \mathbf{x} \quad (9.25)$$

变换过程:

- (1) 通过图傅里叶变换, 即 $V^T \mathbf{x}$ 将图信号变换到频域空间;
- (2) 通过 $\Lambda_h = \sum_{k=0}^K h_k \Lambda^k$ 对频率分量的强度进行调节, 得到 $\tilde{\mathbf{y}} = \Lambda_h V^T \mathbf{x}$;
- (3) 通过逆图傅里叶变换, 即 $V \tilde{\mathbf{y}}$ 将 $\tilde{\mathbf{y}}$ 反解成图信号 \mathbf{y} 。

2) 频域角度

$$\Lambda_h = \sum_{k=0}^K h_k \Lambda^k = \text{diag}(\Psi \mathbf{h})$$

其中,

$$\Psi = \begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^K \\ 1 & \lambda_2 & \cdots & \lambda_2^K \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_N & \cdots & \lambda_N^K \end{bmatrix}$$

多项式系数:

$$\mathbf{h} = \Psi^{-1} \text{diag}^{-1}(\Lambda_h)$$

总结:

- (1) 从频域视角能够更加清晰地完成对图信号的特定滤波操作;
- (2) 图滤波器如何设计具有显式的公式指导;
- (3) 对矩阵进行特征分解是一个非常耗时的操作, 具有 $O(N^3)$ 的时间复杂度, 相比空域视角中的矩阵向量乘法而言, 有工程上的局限性。

9.5 图卷积神经网络

两组图信号 \mathbf{x}_1 和 \mathbf{x}_2 的卷积运算

$$\mathbf{x}_1 * \mathbf{x}_2 = IGFT(GFT(\mathbf{x}_1) \odot GFT(\mathbf{x}_2))$$

其中 \odot 表示哈达玛积，就是频域的矩阵点乘运算。

$$\mathbf{x}_1 * \mathbf{x}_2 = V((V^T \mathbf{x}_1) \odot (V^T \mathbf{x}_2)) = V((\tilde{\mathbf{x}}_1) \odot (V^T \mathbf{x}_2))$$

$$= V(\text{diag}(\tilde{\mathbf{x}}_1)(V^T \mathbf{x}_2))$$

$$= V(\text{diag}(\tilde{\mathbf{x}}_1)V^T)\mathbf{x}_2$$

令 $H_{\tilde{\mathbf{x}}_1} = V(\text{diag}(\tilde{\mathbf{x}}_1)V^T)$ ，则

$$\mathbf{x}_1 * \mathbf{x}_2 = H_{\tilde{\mathbf{x}}_1} \mathbf{x}_2$$

结论：图卷积等价于图滤波。

1) 对频率响应矩阵进行参数化

定义神经网络层

$$X' = \sigma\left(V \begin{bmatrix} \theta_1 & & \\ & \theta_2 & \\ & & \ddots \\ & & & \theta_N \end{bmatrix} V^T X\right)$$
$$= \sigma(V \text{diag}(\boldsymbol{\theta}) V^T X)$$
$$= \sigma(\Theta X)$$

其中 σ 是激活函数， $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_N]$ 是待学习参数， Θ 是对应的待学习图滤波器， X 是输入图信号矩阵， X' 是输出图信号矩阵。

(1) **空域视角**，该层引入了一个自适应图位移算子，通过学习手段指导该算子的学习，从而完成对输入图信号的针对性变换操作。

(2) **频域角度**，该层在 X 与 X' 之间训练了一个可自适应的图滤波器，图滤波器的频率响应函数通过任务与数据之间的对应关系来进行监督学习。

缺点：学习参数过多，需要学习的参数量与图中的节点数一致，对大规模图数据易发生过拟合。

2) 对多项式系数进行参数化

$$X' = \sigma \left(V \begin{bmatrix} \sum_{k=0}^K \theta_k \lambda_1^k \\ \sum_{k=0}^K \theta_k \lambda_1^k \\ \vdots \\ \sum_{k=0}^K \theta_k \lambda_1^k \end{bmatrix} V^T X \right) = \sigma \left(V \begin{bmatrix} K \\ \sum_{k=0}^K h_k \Lambda^k \end{bmatrix} V^T X \right) = \sigma(V \text{diag}(\Psi \boldsymbol{\theta}) V^T X)$$

其中 $\boldsymbol{\theta} = [\theta_0, \theta_1, \dots, \theta_K]$ 是多项式系数向量，也是该网络层真正需要学习的参数。与前述方法不同的是，这个方法的参数量 K 可以自由控制。一般设 $K \ll N$ ，这将大降低模型过拟合的风险。

3) 设定固定的滤波器

限制 $K = 1$, 则 $X' = \sigma(\theta_0 X + \theta_1 L X)$

令 $\theta_0 = \theta_1 = \theta$

则 $X' = \sigma(\theta(I + L)X) = \sigma(\theta \tilde{L} X) = \sigma(\tilde{L} X)$ 设 $\theta = 1$

令 $\tilde{L}_{sym} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$ $\tilde{A} = A + I$

$$\tilde{D}_{ii} = \sum_j A_{ij}$$

则图卷积层为 $X' = \sigma(\tilde{L}_{sym} X W)$

图卷积层在节点层面的计算公式

$$x_i = \sigma \left(\sum_{v_j \in \tilde{N}(v_i)} \tilde{L}_{sym}[i, j] (W x_j) \right) \quad (9.36)$$

Cora数据集: 由2708篇论文及5429条引用关系组成。根据主题划分为7类, 分别是神经网络、强化学习、规则学习、概率方法、遗传算法、理论研究、案例相关。每篇论文的特征是通过词袋模型得到, 维度为1433, 每一维表示一个词, 1 表示该词在这篇文章中出现过, 0表示未出现。

x: 顶点特征, 维度为 2708×1433 。

y: 顶点标签, 包括7个类别。

adjacency: 邻接矩阵, 维度 2708×2708 。

train_mask: 训练集掩码向量, 维度为 2708, 当节点属于训练集时, 相应位置为True, 否则False。

val_mask: 验证集掩码向量, 维度为 2708, 当节点属于验证集时, 相应位置为True, 否则False。

test_mask: 测试集掩码向量, 维度为 2708, 当节点属于测试集时, 相应位置为True, 否则False。


```
class GraphConvolution(nn.Module):
```

```
    def __init__(self, input_dim, output_dim, use_bias=True):
```

```
        """图卷积:  $L \times X \times \theta$ """
```

```
        Args:
```

```
            input_dim: int            输入特征的维度
```

```
            output_dim: int          输出特征维度
```

```
            use_bias : bool, optional 是否使用偏置
```

```
        """
```

```
        super(GraphConvolution, self).__init__()
```

```
        self.input_dim = input_dim
```

```
        self.output_dim = output_dim
```

```
        self.use_bias = use_bias
```

```
        self.weight = nn.Parameter(torch.Tensor(input_dim, output_dim))
```

```
        if self.use_bias:
```

```
            self.bias = nn.Parameter(torch.Tensor(output_dim))
```

```
        else:
```

```
            self.register_parameter('bias', None)
```

```
        self.reset_parameters()
```

GCN层定义


```
def reset_parameters(self):
    init.kaiming_uniform_(self.weight)
    if self.use_bias:
        init.zeros_(self.bias)

def forward(self, adjacency, input_feature):
    """邻接矩阵是稀疏矩阵，因此在计算时使用稀疏矩阵乘法
    Args:
        adjacency: torch.sparse.FloatTensor 邻接矩阵
        input_feature: torch.Tensor 输入特征
    """
    support = torch.mm(input_feature, self.weight)
    output = torch.sparse.mm(adjacency, support)
    if self.use_bias:
        output += self.bias
    return output
```

两层GCN模型

```
class GcnNet(nn.Module):
    """
    定义一个包含两层GraphConvolution的模型
    """
    def __init__(self, input_dim=1433):
        super(GcnNet, self).__init__()
        self.gcn1 = GraphConvolution(input_dim, 16)
        self.gcn2 = GraphConvolution(16, 7)

    def forward(self, adjacency, feature):
        h = F.relu(self.gcn1(adjacency, feature))
        logits = self.gcn2(adjacency, h)
        return logits
```



```
def train():
    loss_history = []
    val_acc_history = []
    model.train()
    train_y = tensor_y[tensor_train_mask]
    for epoch in range(EPOCHS):
        logits = model(tensor_adjacency, tensor_x) # 前向传播
        train_mask_logits = logits[tensor_train_mask] # 只选择训练节点进行监督
        loss = criterion(train_mask_logits, train_y) # 计算损失值
        optimizer.zero_grad()
        loss.backward() # 反向传播计算参数的梯度
        optimizer.step() # 使用优化方法进行梯度更新
        train_acc, _, _ = test(tensor_train_mask) # 计算当前模型训练集上的准确率
        val_acc, _, _ = test(tensor_val_mask) # 计算当前模型在验证集上的准确率
        loss_history.append(loss.item())
        val_acc_history.append(val_acc.item())
        print("Epoch {:03d}: Loss {:.4f}, TrainAcc {:.4}, ValAcc {:.4f}".format(
            epoch, loss.item(), train_acc.item(), val_acc.item()))

    return loss_history, val_acc_history
```

GCN与CNN的关系

- **图像是一种特殊的图数据**，CNN的卷积运算相较于GCN，最大区别是没有显式的表达出邻接矩阵。GCN的卷积计算用来处理更普遍的非结构化的图数据。
- **从网络连接方式看，二者都是局部连接**。GCN计算作用在其一阶子图上；CNN计算作用在中心像素的 $N*N$ 像素栅格内；节点下一层特征计算只依赖于自身邻域的方式，在网络连接上表现为局部连接。
- **二者卷积核的权重处处共享**。均作用于全图所有节点。参数共享大大减少了每层网络的参数量，有效地避免过拟合现象。
- **从模型层面看，感受野随着卷积层的增大而变大**。每多一层卷积运算，中心节点能多融合进更外一圈的信息。

GCN 对图结构数据进行端到端的学习

GCN计算过程

- XW 对属性信息仿射变换，学习属性特征间的交互模式。
- $\tilde{L}_{sym}XW$ 从空域看是聚合邻居节点的过程，代表了对节点局部结构信息的编码。

GCN的优势

- 对表示学习和任务学习一起进行端到端的优化。
- 对结构信息和属性信息的学习同步进行。

总结：GCN通过堆叠图卷积层，属性信息的编码学习与结构信息的编码学习不断交替进行，完成对图数据中更加复杂的模式学习。

GCN是一个低通滤波器

$$\tilde{L}_{sym} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$$

$$= \tilde{D}^{-1/2} (\tilde{D} - L) \tilde{D}^{-1/2}$$

$$= I - \tilde{D}^{-1/2} L \tilde{D}^{-1/2}$$

$$= I - \tilde{L}_s$$

$\tilde{L}_s = V \tilde{\Lambda} V^T$ 可被正交对角化, $\tilde{\lambda}_i$ 是 $\tilde{\Lambda}$ 的特征值, 可证明 $\tilde{\lambda}_i \in [0, 2)$

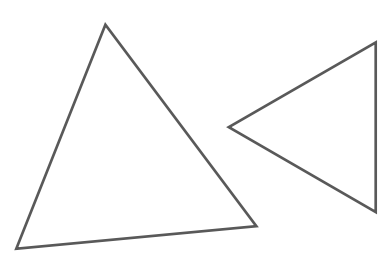
$$\tilde{L}_{sym} = I - \tilde{L}_s = I - V \tilde{\Lambda} V^T = V (I - \tilde{\Lambda}) V^T$$

优点

- 1) 权值共享。从 AXW 可以看出每个节点的参数矩阵都是 W ;
- 2) 局部连接, 每次聚合的只是一阶邻居;
- 3) 感受野正比于卷积层层数, 层数越多, 感受野越大, 参与运算的信息量越充分。
- 4) 复杂度大大降低, 不再计算拉普拉斯矩阵和进行特征分解。

不足

- 1) 扩展性差: 由于训练时需要关于训练节点、测试节点在内的所有节点的邻接矩阵 A , 因此是transductive的, 不能处理大图;
- 2) 局限于浅层: 实验中使用2层效果最好, 为了加深, 需要使用残差连接等, 但是即用了这些trick, 也只能勉强保持性能不降, 并没有提高。
- 3) 无法处理有向图: 推导过程中用到拉普拉斯矩阵的特征分解需要满足拉普拉斯矩阵是对称矩阵的条件。



The end

