

图神经网络导论

卷积图神经网络

授课教师：周晟

浙江大学 软件学院

2024.11.19



课程内容

- ① 经典图神经网络
- ② 卷积
- ③ 图傅立叶变换
- ④ 卷积图神经网络实例



1 经典图神经网络

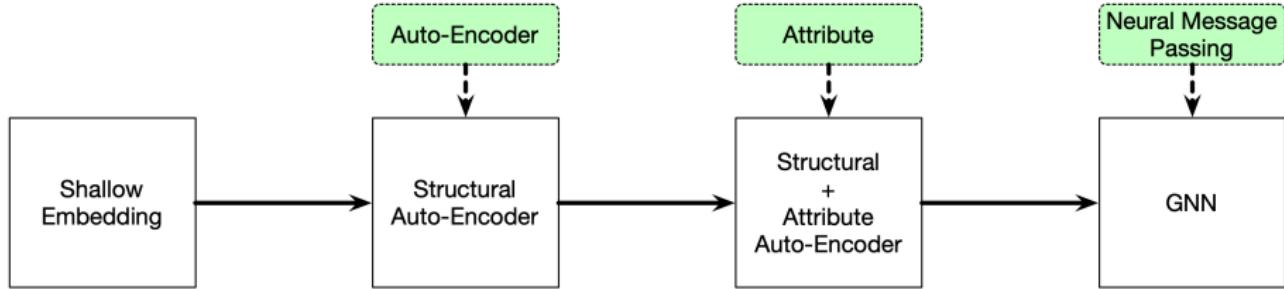
2 卷积

3 图傅立叶变换

4 卷积图神经网络实例



上节课回顾



图神经网络发展历史

图神经网络的设计

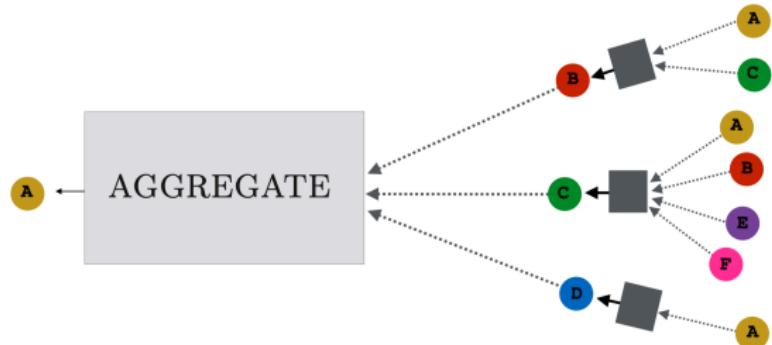
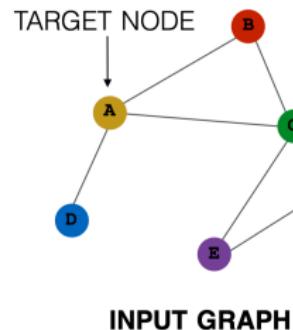
- ① Aggregation
- ② Update

What? How? Why?

神经消息传递 (Neural Message Passing)

神经消息传递 (Neural Message Passing)

神经网络传递 (Neural Message Passing) 是指节点间传递特征信息并使用神经网络更新的一种机制。



神经消息传递 (Neural Message Passing)

神经消息传递的数学形式：

$$\begin{aligned}\mathbf{h}_u^{(k+1)} &= \text{UPDATE}^{(k)} \left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)} \left(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\} \right) \right) \\ &= \text{UPDATE}^{(k)} \left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)} \right),\end{aligned}$$

核心模块：

- ① Aggregate
- ② Update

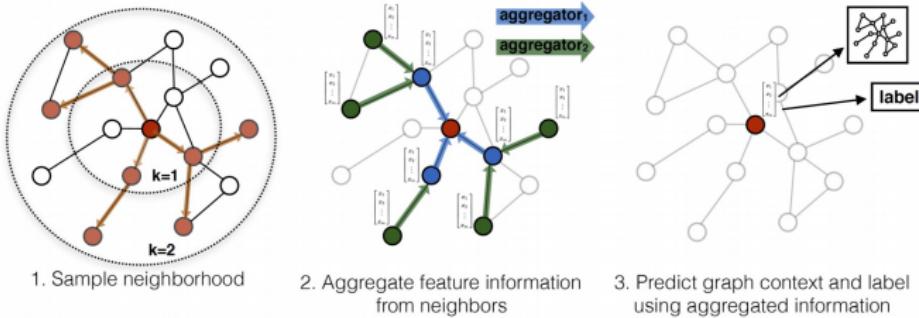


GraphSAGE[Hamilton et al., 2017]

研究动机

现有的 Graph Embedding 的方法大多要求图中所有的点在训练阶段出现过，对于训练阶段未出现的节点学习节点表征，需要对整个网络进行重新训练。

Transductive VS Inductive



GraphSAGE 由邻居采样、邻居特征聚集和训练三部分组成

GraphSAGE——邻居节点采样

研究动机

- ① 网络中存在度比较大的节点，整合邻居信息复杂度高
- ② 网络中节点度差异较大，模型训练不稳定

GraphSAGE 邻居节点采样

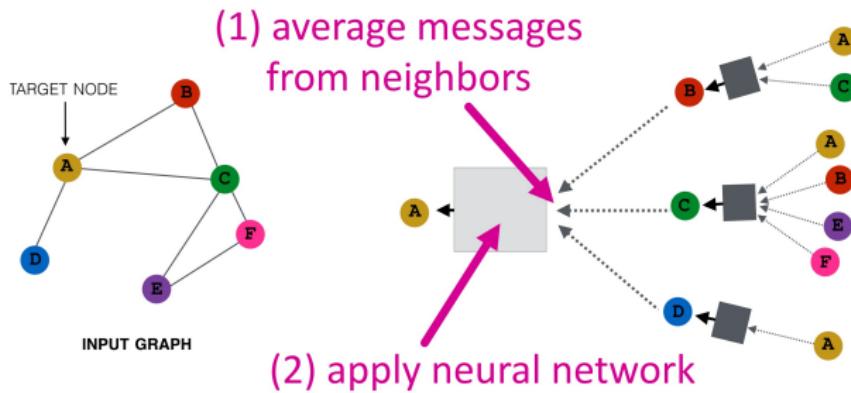
对于每个节点采样固定数量 S 的邻居，若节点邻居数小于 S ，则采用有放回的抽样方法，否则采用无放回的抽样。

网络的不同层之间使用独立的 Random Walk 采样，同一个节点在不同层采样得到邻居的概率是不同的。

GraphSAGE——Aggregation

在 GraphSAGE 的每一层，都执行两个操作：

- ① 从邻居节点中聚集特征
- ② 通过神经网络生成目标节点的特征



GraphSAGE——Aggregation

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 $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$; non-linearity σ ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, \dots, K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

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

```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;  
2 for  $k = 1 \dots K$  do  
3   for  $v \in \mathcal{V}$  do  
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;  
5      $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k))$   
6   end  
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$   
8 end  
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

GraphSAGE 的前向传播

GraphSAGE——Aggregation

GraphSAGE aggregation 函数：

$$h_v^k = \sigma \left(W^k \cdot \text{CONCAT} \left(h_v^{k-1}, \text{AGG} \left(\{h_u^{k-1} \mid \forall u \in \mathcal{N}(v)\} \right) \right) \right)$$

GraphSAGE 在聚合过程中，对于每一层的节点表示都进行了 L2 的标准化。

$$h_v^k \leftarrow \frac{h_v^k}{\|h_v^k\|_2}, \forall v \in \mathcal{V} \text{ where } \|u\|_2 = \sqrt{\sum_i u_i^2}$$

表征的标准化使得所有的向量都有了相同的尺度。一定程度上可以使模型训练更稳定，并带来一定的效果提升。

GraphSAGE——Aggregation

为了在 Aggregation 过程中保留图的 permutation invariant，
GraphSAGE 中尝试了三种聚集函数：

- Mean aggregator
- LSTM aggregator
- Pooling aggregator



GraphSAGE——Mean AGG

$$h_v^k \leftarrow \sigma(W_k \cdot \text{MEAN}(\{h_v^{k-1}\} \cup \{h_u^{k-1}, \forall u \in \mathcal{N}(v)\}))$$

- 在 Mean aggregator 中， MEAN 操作替代了 CONCAT 操作。除了对邻居节点的特征求平均，还将目标节点本身也看作了它的邻居节点，权重是相同的。
- 与 CONCAT 的区别： CONCAT 对于一个目标节点输出的 size 是 (1, 2D)，再通过 W_k 来变换成 (1, D)。而 Mean aggregator 的结果 size 是 (1, D)。



GraphSAGE——LSTM AGG

- 相较于 Mean aggregator, LSTM aggregator 具有更强的表示能力，但它本质上并不是对称的，需要按照输入的顺序来处理数据。
- 因此，作者将邻居节点的向量集合随机打乱顺序，作为 LSTM aggregator 的输入。

$$AGGREGATE_k^{LSTM} = \text{LSTM}([h_u^{k-1}, \forall u \in \pi(\mathcal{N}(v))])$$

其中 $\pi(\cdot)$ 表示随机打乱。



GraphSAGE——Pooling AGG

$$AGGREGATE_k^{pool} = \max(\{\sigma(W_{pool}h_u^{k-1} + b_{pool}), \forall u \in \mathcal{N}(v)\})$$

- 每个邻居节点都通过一个全连接网络，然后整体做 max-pooling。权重 W_{pool} 和 b_{pool} 是共享的。
- 任何满足对称性的算子都可以用来替换 max，比如 mean。在实验中，作者发现 mean-pooling 和 max-pooling 并没有显著的差别。



GraphSAGE——训练

无监督训练

节点的低维表征可以拟合节点之间的相似性：

$$J_{\mathcal{G}}(z_u) = -\log(\sigma(z_u^T z_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-z_u^T z_{v_n}))$$

半监督训练

节点的低维表征可以最大化半监督数据的似然：

$$\mathcal{L} = - \sum_{l \in \mathcal{Y}_L} \sum_{f=1}^F Y_{lf} \ln Z_{lf}$$

GraphSAGE——数据集

- Citation: 论文分类任务 (节点分类)
包含 2000 年到 2005 年六个生物学相关领域的论文，每篇论文属于六种类别之一。数据集包含 302424 个节点，平均度数为 9.15。
- Reddit: 帖子分类任务 (节点分类)
包含 Reddit 上发布帖子的一个大型图数据集，节点标签为帖子所属的社区。数据集包含 232965 个节点，平均度数为 492。
- PPI: 蛋白质分类任务 (Graph 分类)
每个图对应一个不同的人体组织，共有 121 种标签。
平均每个图包含 2373 个节点，平均度数为 28.8。



Graph Attention Network[Veličković et al., 2017]

GraphSAGE-mean 也可以写为

$$h_v^k = \sigma \left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu} W^k h_u^{k-1} \right)$$

其中 $\alpha_{vu} = \frac{1}{|\mathcal{N}(v)|}$ 为节点 u 对于 v 的权重因子（重要性）。

在 GraphSAGE 中，所有邻居节点 $u \in \mathcal{N}(v)$ 对于节点 v 的重要性都是相同的。

节点从相似的邻居搜集更多的信息，从不相似的邻居搜集更少的信息

GAT——attention

Attention 机制

Attention 机制通过在模型训练过程中动态调整模型对数据/特征不同部分的权重提升模型的学习效果。在计算机视觉，自然语言处理中取得了广泛的应用。

input image



attention map



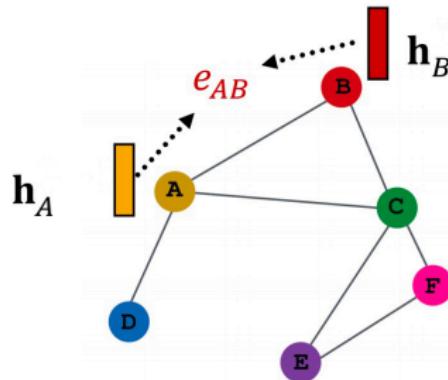
GAT 在图神经网络中引入了 Attention，希望关注邻居节点集合中比较重要的部分，淡化其余部分。

GAT——注意力机制

- GAT 对节点执行一个共享的注意力机制 $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$,
由此计算 attention 系数:

$$e_{ij} = a(W\vec{h}_i, W\vec{h}_j)$$

表示节点 j 的特征对节点 i 的重要性。



GAT——注意力机制

- 在最一般的机制中，所有节点对之间的 attention 系数都会被计算，这种做法显然会丢失**结构信息**。
- GAT 采用了一种 masked attention 的方式——仅将注意力分配到节点 i 的邻居节点集 \mathcal{N}_i 上，并用 softmax 对 attention 系数进行了标准化：

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



GAT——注意力机制

- 在 GAT 中，注意力机制 a 是由一个单层前馈神经网络来实现的，其权重向量 $\vec{a} \in \mathbb{R}^{2F'}$ ，并使用 LeakyReLU 作为非线性激活函数。
- 由此，权重因子 α 可以表示为：

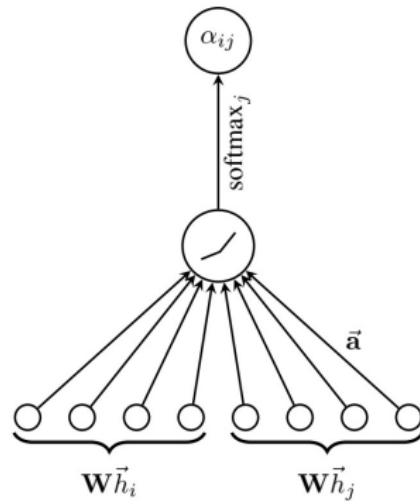
$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\vec{a}^T [W\vec{h}_i || W\vec{h}_j])))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(\vec{a}^T [W\vec{h}_i || W\vec{h}_k]))}$$

其中 $||$ 表示拼接操作。



GAT——注意力机制

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

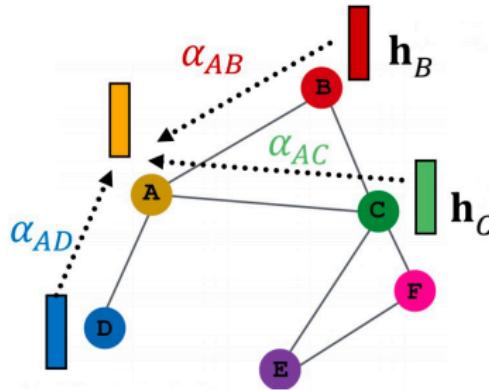


GAT——注意力机制

- 网络的输出特征可以通过下式得到：

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

σ 为一个非线性激活函数。



GAT——multi-head attention

- 为了提高模型的拟合能力，GAT 还引入了 multi-head attention，即同时使用多个独立的注意力机制来计算 attention 因子，并将它们的结果合并（拼接或求和）：

$$\vec{h}'_i = \parallel_{k=1}^K \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k \vec{h}_j \right)$$

其中 \parallel 表示拼接操作， α_{ij}^k 为第 k 个注意力机制 a^k 产生的 attention 因子。

由于 $W^k \in \mathbb{R}^{F' \times F}$ ，所以这里的 $\vec{h}'_i \in \mathbb{R}^{KF'}$ 。



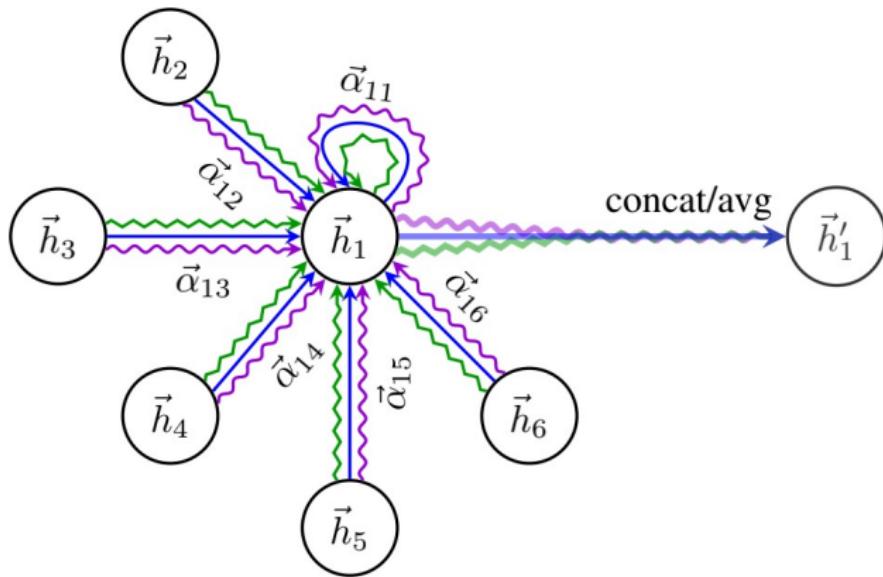
GAT——multi-head attention

- 对于网络的最终层，拼接操作就不再合适了。因此，可以用求均值的方式来代替：

$$\vec{h}'_i = \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k \vec{h}_j \right)$$

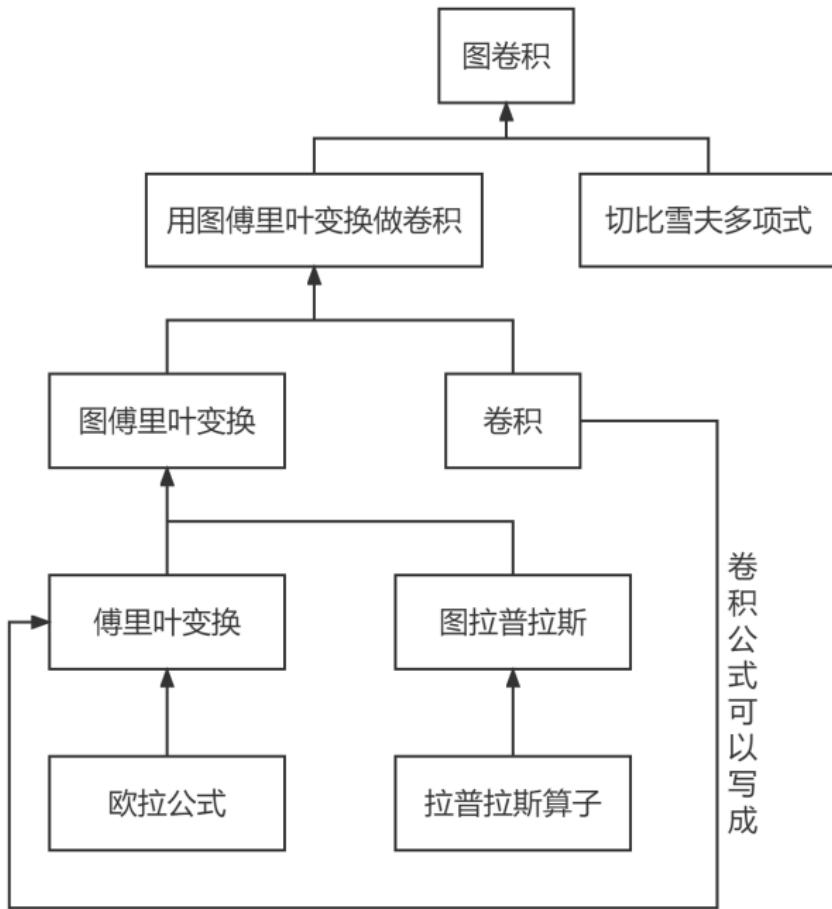


- 下图为 $K=3$ 的 multi-head attention 示意。不同颜色的线表示不同的 attention 机制。



- 1 经典图神经网络
- 2 卷积
- 3 图傅立叶变换
- 4 卷积图神经网络实例





生活中的卷积



抛骰子游戏

抛两个质地均匀的骰子，
总数为 8 的概率是多少？

离散卷积：抛骰子游戏



生活中的卷积

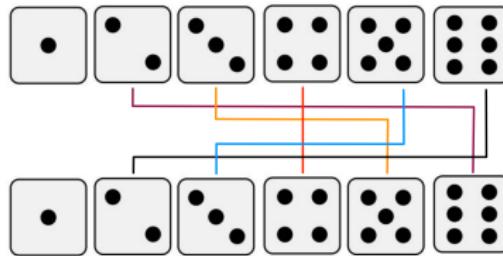


离散卷积：抛骰子游戏

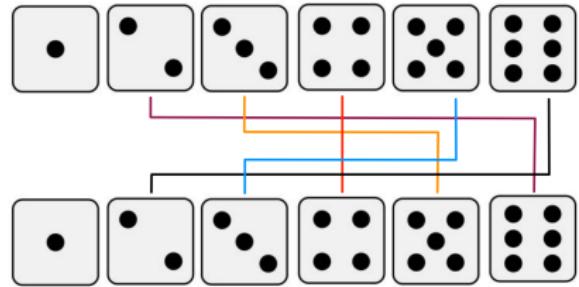
抛骰子游戏

抛两个质地均匀的骰子，
总数为 8 的概率是多少？

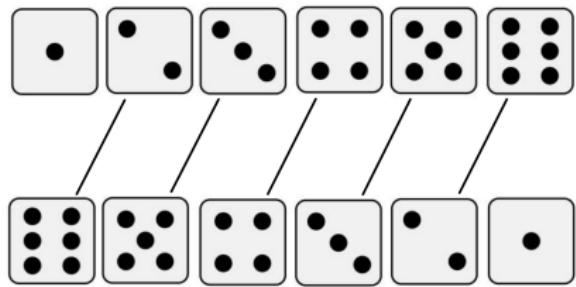
$$\sum_{m=2}^6 f(8-m)g(m)$$



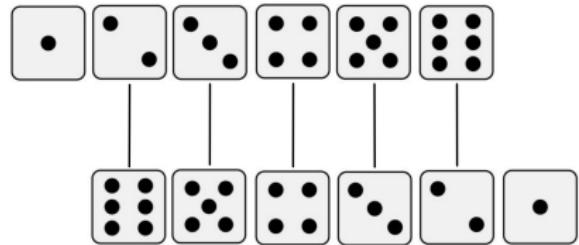
生活中的卷积



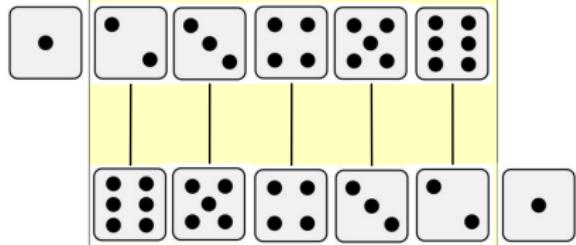
解



卷



移



积

生活中的卷积

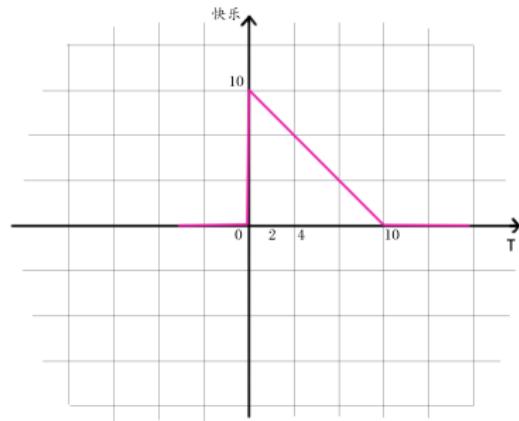


肥宅快乐问题

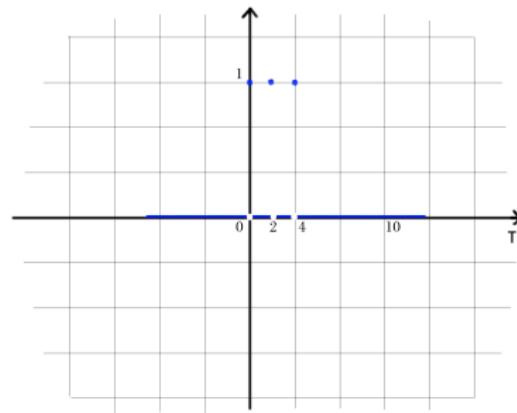
如果喝一口肥宅水，可以让快乐指数瞬间增加了 10 点，但是快乐指数马上开始下降，并在 10 分钟后消失。

若张三在第 0, 2, 4 分钟喝了三口快乐水，他在第 8 分钟的快乐指数是多少？

生活中的卷积



快乐消失曲线 $10 - t$



间隔的快乐



生活中的卷积

张三在第 8 分钟的快乐指数：

$$h(8) = [10 - (8 - 0)] + [10 - (8 - 2)] + [10 - (8 - 4)]$$

张三在小于 10 的任意时刻 t 的快乐指数：

$$h(t) = [10 - (t - 0)] + [10 - (t - 2)] + [10 - (t - 4)]$$

定义函数 $f(t) = 1, t = 0, 2, 4$ 和 $g(t) = 10 - t$, 张三的快乐指数为：

$$h(T) = \int_{\mathbb{R}} f(t)g(T-t)dt$$



生活中的卷积

快乐函数：

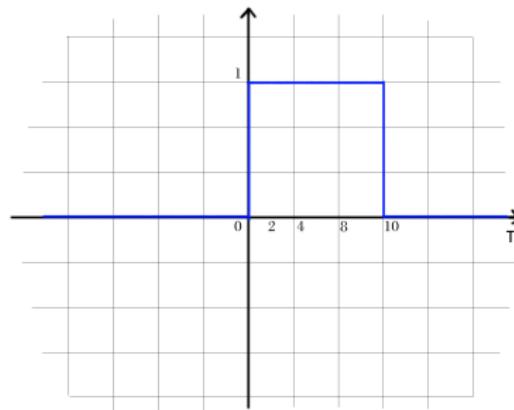
$$h(T) = \int_{\mathbb{R}} f(t)g(T-t)dt$$

- ① $h(T)$ 表示张三在 T 时刻的快乐度
- ② $f(t)$ 表示喝可乐的信号，也就是行为
- ③ $g(t)$ 表示快乐程度的信号，也就是后果

如果一个函数表示行为，另一个函数表示后果，卷积公式得到的就是所有行为 f 在效果函数 g 的作用下产生的后果。



生活中的卷积



连续的可乐

连续的快乐

假设究极肥宅李四在 $0 - 10$ 分钟内一直在喝可乐，那么他在第 8 分钟的快乐度有多少？

生活中的卷积

- 我们同样可以写出计算式：

$$h(8) = \int_0^8 [10 - (8 - t)] dt$$

- 拓展到任意时刻 T

$$h(T) = \int_0^T [10 - (T - t)] dt$$

- $T \leq 10$ 时，用 $f(t) = 1, t \in [0, T]$, $g(t) = (10 - t)$ 来表达：

$$h(T) = \int_{\mathbb{R}} f(t)g(T - t)dt$$



数学上的卷积

卷积的定义

卷积是一种定义在两个函数 f, g 上的数学操作 $(f * g)(n) :$

① 连续卷积:

$$(f * g)(n) = \int_{-\infty}^{\infty} f(\tau)g(n - \tau)d\tau$$

② 离散卷积:

$$(f * g)(n) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(n - \tau)$$

卷积的三个基本操作: 翻转, 滑动, 叠加

数学上的卷积

卷积的由来：

$$(f * g)(n) = \int_{-\infty}^{\infty} f(\tau)g(n - \tau)d\tau$$

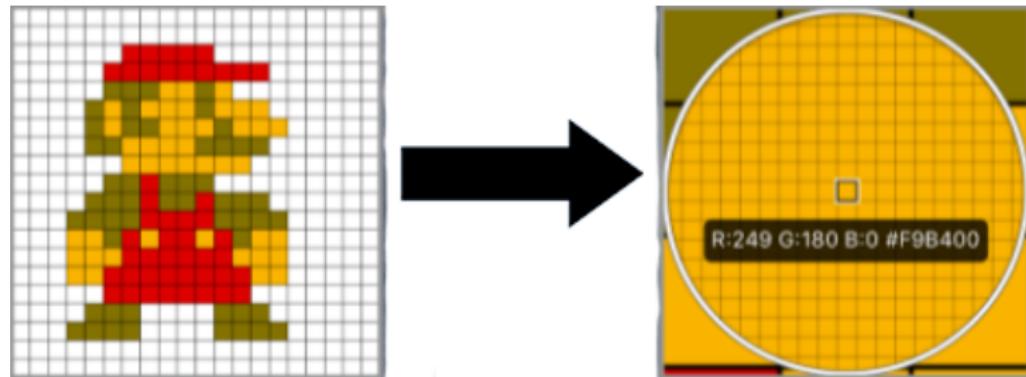
- **卷**: 先对 g 函数进行翻转，相当于在数轴上把 g 函数做个轴对称，也就是卷积的“卷”的由来。
- **积**: 在连续情况下，积指的是对两个函数的乘积求积分，在离散情况下就是加权求和，为简单起见就统一称为积。



计算机视觉上的卷积

计算机视觉训练问题

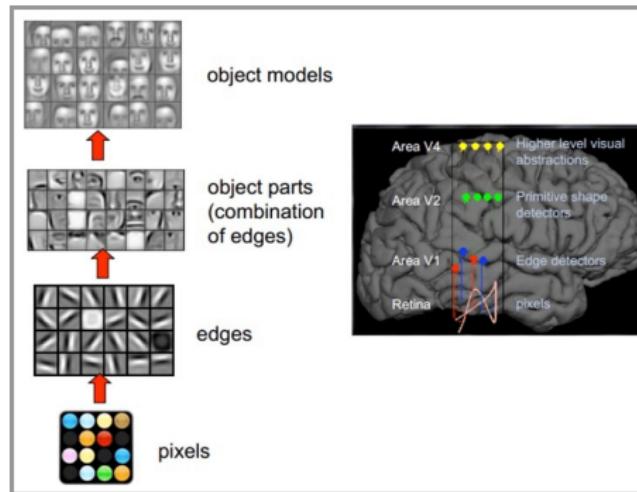
- ① 全连接神经网络参数过多，难以训练
- ② 像素间矢量位置关系容易丢失
- ③ 与人类视觉原理难以匹配



计算机视觉上的卷积

人类的视觉原理

从原始信号摄入开始（瞳孔摄入像素 Pixels），接着做初步处理（大脑皮层某些细胞发现边缘和方向），然后抽象（大脑判定，眼前的物体的形状，是圆形的），然后进一步抽象（大脑进一步判定该物体是只气球）。



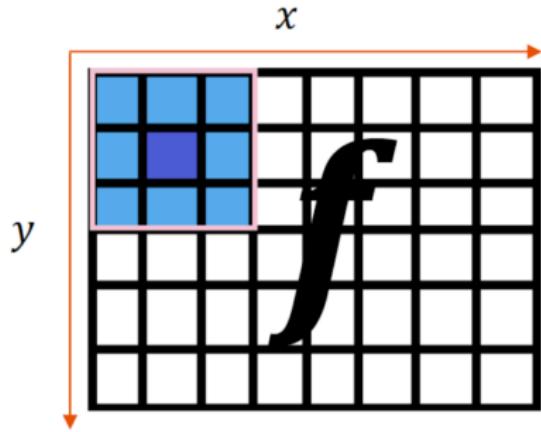
计算机视觉上的卷积

卷积神经网络的设计动机

- ① 平移不变性 (translation invariance)：不管检测对象出现在图像中的哪个位置，神经网络的前面几层应该对相同的图像区域具有相似的反应，即为“平移不变性”。
- ② 局部性 (locality)：神经网络的前面几层应该只探索输入图像中的局部区域，而不过度在意图像中相隔较远区域的关系，全局的特征依靠堆叠的多层卷积进行抽样。



计算机视觉上的卷积



$g(1,1)$	$g(0,1)$	$g(-1,1)$
$g(1,0)$	$g(0,0)$	$g(-1,0)$
$g(1,-1)$	$g(0,-1)$	$g(-1,-1)$

$$\begin{aligned} h(1,1) = & f(0,0)g(1,1) + f(1,0)g(0,1) + f(2,0)g(-1,1) \\ & + f(0,1)g(1,0) + f(1,1)g(0,0) + f(2,1)g(-1,0) \\ & + f(0,2)g(1,-1) + f(1,2)g(0,-1) + f(2,2)g(-1,-1) \end{aligned}$$

计算机视觉上的卷积

$$h(x, y) = (f * g)(x, y) = \sum_{m,n} f(x - m, y - n)g(m, n)$$

从图像卷积到图卷积

为何要讲卷积操作从图像（Image）迁移到图（Graph）上？

图像卷积的成功

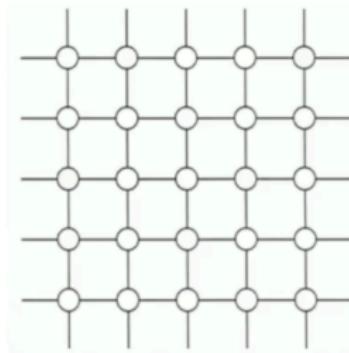
- ① 平移不变性（一举多得）
- ② 矢量信息保留（图像数据特点）

卷积用于图数据

- ① 共享结构特征（一举多得）
- ② 置换不变性（图数据特点）



从图像卷积到图卷积



Grid-like network



Irregular network

图卷积的困难

- ① 节点没有顺序（排列不变性）
- ② 节点邻居没有固定个数（难以定义统一的特征提取函数）

- 1 经典图神经网络
- 2 卷积
- 3 图傅立叶变换
- 4 卷积图神经网络实例



生活中的傅立叶变换



高音甜 中音准 低音沉

时域信号难以直接分析出高、中、低音，他是怎么做到的？



卷积与傅立叶变换

傅立叶变换的定义

傅立叶变换是一种将时域信号转换为频域信号的线性积分变换：

$$\hat{f}(v) = \int_{t_1}^{t_2} f(t) e^{-2\pi i vt} dt$$

经过傅立叶变换得到的函数 \hat{f} 称为原函数 f 的频谱。傅立叶变换是一种可逆变换，即可通过 \hat{f} 得到其原函数 f 。



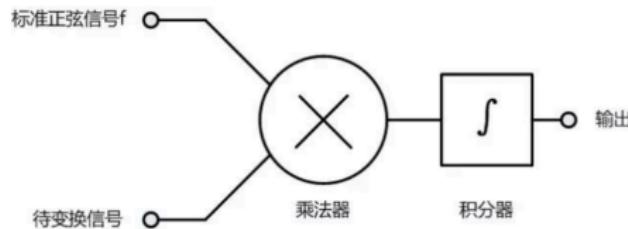
傅立叶变换

傅立叶变换的假设

任何连续周期信号可以由一组适当的正弦曲线组合而成。

正弦函数的特性：正交性

任意两个不同频率的正弦波的乘积，在两者的公共周期内的积分等于零。



基于傅立叶变换的滤波器¹

¹<https://kknews.cc/education/p6qjkxp.html>

傅里叶变换

信号的变换



傅里叶变换

欧拉公式：

$$e^{i\theta} = i \sin \theta + \cos \theta \quad e^{i\pi} = -1$$

$(\cos \theta, i \sin \theta)$ 是复平面上单位圆上的点

复平面上的欧拉公式²

²<https://www.bilibili.com/video/BV1Vd4y1e7pj>

傅里叶变换

把 $\theta = -2\pi ft$ 代入， f 代表频率， t 代表时间。为了区分函数 $f()$ 与频率 f ，我们把频率记作 v

$$\int_{t_1}^{t_2} f(t) e^{-2\pi i v t} dt$$

傅立叶变换的三层操作

- ① $e^{-2\pi i v t}$: 用时间和频率描述的复平面上的**单位向量**, 负号表示该向量随时间朝顺时针转动
- ② $f(t)$ 表示作用于单位向量的变换
- ③ $\int_{t_1}^{t_2} () dt$ 表示对作用之后的向量的积分

傅里叶变换

余弦信号变换³

³<https://www.bilibili.com/video/BV1Vd4y1e7pj>

傅立叶变换

傅立叶变换理解：

$$\int_{t_1}^{t_2} f(t) e^{-2\pi i v t} dt$$

- 在上述例子中，傅里叶变换的第二层将一个余弦函数按照频率 f “卷”了起来，成为复平面上的一朵花（一条曲线）
- 不难发现，傅里叶变换是对单个函数的先“卷”后“积”，而卷积是对两个函数中的一个先“卷”后“积”，他们之间难道有什么关系吗？



卷积与傅里叶变换

傅里叶变换：将时间的函数转化为频率的函数

$$\mathcal{F}\{f\}(v) = \int_{t_1}^{t_2} f(t)e^{-2\pi i vt} dt$$

傅里叶逆变换：将频率的函数转化回时间的函数

$$\mathcal{F}\{f\}^{-1}(t) = \int_{t_1}^{t_2} f(v)e^{2\pi i vt} dv$$

卷积：

$$h(n) = (f * g)(n) = \int_{-\infty}^{\infty} f(\tau)g(n - \tau)d\tau$$

傅立叶变换与卷积的数学形式惊人相似！



卷积与傅里叶变换

$$\begin{aligned} h(n) &= \int_{-\infty}^{\infty} f(\tau)g(n - \tau)d\tau \\ \mathcal{F}\{f * g\}(v) &= \mathcal{F}\{h\}(v) \\ &= \int_{\mathbb{R}} h(n)e^{-2\pi i n \cdot v} dn \\ &= \int_{\mathbb{R}} \int_{\mathbb{R}} f(\tau)g(n - \tau)e^{-2\pi i n \cdot v} d\tau dn \\ &= \int_{\mathbb{R}} f(\tau) \left(\int_{\mathbb{R}} g(n - \tau)e^{-2\pi i n \cdot v} dn \right) d\tau \end{aligned}$$

此处进行了一次“交换积分次序”



卷积与傅里叶变换

代入 $y = n - \tau; dy = dn$

$$\begin{aligned}\mathcal{F}\{f * g\}(v) &= \int_{\mathbb{R}} f(\tau) \left(\int_{\mathbb{R}} g(y) e^{-2\pi i(y+\tau) \cdot v} dy \right) d\tau \\&= \int_{\mathbb{R}} f(\tau) e^{-2\pi i\tau \cdot v} \left(\int_{\mathbb{R}} g(y) e^{-2\pi iy \cdot v} dy \right) d\tau \\&= \int_{\mathbb{R}} f(\tau) e^{-2\pi i\tau \cdot v} d\tau \int_{\mathbb{R}} g(y) e^{-2\pi iy \cdot v} dy \\&= \mathcal{F}\{f\}(v) \cdot \mathcal{F}\{g\}(v)\end{aligned}$$

最后对等式的两边同时作傅里叶逆变换 \mathcal{F}^{-1} ，得到

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$



图卷积

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$

两个函数的卷积可以写成
这两个函数傅立叶变换后乘积的反卷积。

有了卷积和傅立叶变换的关系
图上的卷积转化为图上的傅立叶变换!

图上的卷积

- ① 定义图上的傅立叶变换/逆变换
- ② 定义图上的信号 f
- ③ 定义图上的操作 g (学习目标)

图傅里叶变换

- 用 w 表示 $2\pi v$, 傅立叶变换可转化为:

$$\mathcal{F}[f(t)] = \hat{f}(v) = \int f(t)e^{-iwt} dt$$

- e^{-iwt} 是复平面上的一个旋转向量 (基向量), 傅立叶变换等价于信号 $f(t)$ 在该基向量上投影的积分
- 基向量 e^{-iwt} 与图数据有何关系?



拉普拉斯算子

拉普拉斯算子

一个函数的拉普拉斯算子定义为函数梯度 (∇f) 的散度 ($\nabla \cdot$)，即

$$\Delta f = \nabla^2 f = \sum_{i=1}^n \frac{\partial^2 f}{\partial x_i^2}$$

将拉普拉斯算子作用于 e^{-iwt} 上，也就是求二阶导数，有

$$\Delta e^{-iwt} = \frac{\partial^2 e^{-iwt}}{\partial t^2} = -w^2 e^{-iwt}$$

拉普拉斯算子本质上是一种变换，可以用矩阵表示。

拉普拉斯算子对应的变换矩阵是什么？



从拉普拉斯算子到拉普拉斯矩阵

考虑离散拉普拉斯算子。离散函数的导数如下所示

$$\begin{aligned}\frac{\partial f}{\partial x} &= f'(x) = f(x+1) - f(x) \\ \frac{\partial^2 f}{\partial x^2} &= f''(x) \approx f'(x) - f'(x-1) \\ &= f(x+1) + f(x-1) - 2f(x)\end{aligned}$$



拉普拉斯算子-拉普拉斯矩阵

二维离散函数的二阶导数如下：

$$\begin{aligned}\Delta f &= \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \\&= f(x+1, y) + f(x-1, y) - 2f(x, y) \\&\quad + f(x, y+1) + f(x, y-1) - 2f(x, y) \\&= f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)\end{aligned}$$

二阶导数等于其在所有自由度上进行一次运动之后获得的增益的和。二维离散函数的自由度分别是 x, y 两个维度的 $+1$ 和 -1 共四个方向。

图上的自由度

图上的自由度

给定一个有 n 个节点的图，在离散的图上，从 i 节点出发的一次运动可以到达任意邻居。若该图为无权图，则所有邻居到达概率一致，若该图为有权图，则可达概率与权重成正比。

设 f 函数的定义域为离散的 $1, 2, \dots, n$ ，用 f_i 表示 f 在 i 节点处的取值。

根据图上自由度的定义，图上的拉普拉斯算子定义如下：

$$\Delta f_i = \sum_{j=1}^n w_{ij} (f_i - f_j) = d_i f_i - \sum_{j=1}^n w_{ij} f_j$$



图拉普拉斯

$$\Delta f_i = \sum_{j=1}^n w_{ij} (f_i - f_j) = d_i f_i - \sum_{j=1}^n w_{ij} f_j$$

对应的变换矩阵为：

$$\Delta f = \begin{pmatrix} \Delta f_1 \\ \vdots \\ \Delta f_n \end{pmatrix} = (D - W)f = Lf$$

结合拉普拉斯算子的定义和图上自由度的定义，得到了图上的拉普拉斯矩阵。

图相关矩阵定义：邻接矩阵

邻接矩阵，记作 W ，有时也记作 A ， i, j 位置的元素表示第 i 个节点和第 j 个节点的权重或连通信息

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



图相关矩阵定义：度矩阵

度矩阵 D , 仅对角线上有元素, 第 i 行表示第 i 个节点的度

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



图相关矩阵定义：拉普拉斯矩阵

拉普拉斯矩阵的定义为： $L = D - W$

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

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



拉普拉斯矩阵

拉普拉斯矩阵的性质

① 对称矩阵

$$L = L^T$$

② 半正定

$$x^T L x \geq 0$$

半正定性决定了拉普拉斯矩阵一定存在 N 个相互正交的特征向量
(正交基)



拉普拉斯矩阵半正定性

$$\begin{aligned}x^T L x &= x^T D x - x^T A x = \sum_{i=1}^n d_i x_i^2 - \sum_{i,j=1}^n x_i x_j A_{ij} \\&= \frac{1}{2} \left(\sum_{i=1}^n d_i x_i^2 - 2 \sum_{i,j}^n x_i x_j A_{ij} + \sum_{j=1}^n d_j x_j^2 \right) \\&= \frac{1}{2} \left(\sum_{i=1}^n \left(\sum_{j=1}^n A_{ij} \right) x_i^2 - 2 \sum_{i,j}^n x_i x_j A_{ij} + \sum_{j=1}^n \left(\sum_{i=1}^n A_{ji} \right) x_j^2 \right) \\&= \frac{1}{2} \left(\sum_{i,j=1}^n A_{ij} x_i^2 - 2 \sum_{i,j=1}^n x_i x_j A_{ij} + \sum_{j,i=1}^n A_{ji} x_j^2 \right) \\&= \frac{1}{2} \sum_{i,j=1}^n A_{ij} (x_i^2 - 2x_i x_j + x_j^2) = \frac{1}{2} \sum_{i,j=1}^n A_{ij} (x_i - x_j)^2 \geq 0.\end{aligned}$$

从拉普拉斯算子到拉普拉斯矩阵

拉普拉斯矩阵的特征向量

拉普拉斯算子的定义 $\Delta e^{-iwt} = -w^2 e^{-iwt}$ ，图上的拉普拉斯算子对应的变换矩阵是图拉普拉斯矩阵，因此 e^{-iwt} 是图拉普拉斯矩阵的特征向量。

e^{-iwt} 可通过特征分解求得：

$$L = U \Lambda U^T$$

其中

$$U = [u_1, \dots, u_n]$$

表示特征向量

$$\Lambda = \text{diag}([\lambda_1, \dots, \lambda_n])$$

表示特征值



图傅里叶变换

图傅立叶变换矩阵形式定义：

$$\hat{f} = U^T f$$

图傅立叶逆变换矩阵形式定义：

$$f = U \hat{f}$$

U 是图拉普拉斯矩阵的特征向量。

图傅里叶变换的实质就是对图上节点的信号做了一次线性变换，而这个线性变换包含了图拓扑的信息

图上的卷积

- 已知卷积可以写成傅里叶变换的形式

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$

- 已知图傅里叶变换及其逆变换：

$$\mathcal{G}\mathcal{F}\{f\} = U^T f \quad , \quad \mathcal{IG}\mathcal{F}\{f\} = U f$$

- 图卷积公式则呼之欲出：

$$f * g = U (U^T f \cdot U^T g)$$

$U^T g$ 怎么定义？



图傅里叶变换-卷积

$$f * g = U \left(U^T f \cdot U^T g \right)$$

通过简单的恒等变换:

$$U^T g = [\boldsymbol{\theta}_0, \dots, \boldsymbol{\theta}_{n-1}]^T$$

$$g_\theta = \text{diag}([\boldsymbol{\theta}_0, \dots, \boldsymbol{\theta}_{n-1}])$$

图上的卷积可以表示为:

$$\begin{aligned} f * g &= U \left((U^T f) \cdot (U^T g) \right) \\ f * g &= U g_\theta U^T f \end{aligned}$$



图傅里叶变换与图卷积

$$f * g = \textcolor{red}{U} g_{\theta} \textcolor{green}{U}^T f$$

图卷积操作可以通过如下的变换实现：

- ① 图信号的傅立叶变换，从时域转化为频域

$$\mathcal{G}\mathcal{F}\{f\} = \textcolor{green}{U}^T f$$

- ② 频域基变换操作

$$g_{\theta} = \text{diag}([\theta_0, \dots, \theta_{n-1}])$$

- ③ 傅立叶逆变换

$$\mathcal{I}\mathcal{G}\mathcal{F}\{\hat{f}\} = \textcolor{red}{U} \hat{f}$$



- 1 经典图神经网络
- 2 卷积
- 3 图傅立叶变换
- 4 卷积图神经网络实例



Spectral Networks and Deep Locally Connected Networks on Graphs

Joan Bruna

New York University

bruna@cims.nyu.edu

Wojciech Zaremba

New York University

woj.zaremba@gmail.com

Arthur Szlam

The City College of New York

aszlam@ccny.cuny.edu

Yann LeCun

New York University

yann@cs.nyu.edu

参考 CNN 中通过池化和子采样层减少网格大小的方式，在图上进行多尺度聚类

Spectral GCN

$$x * y = U g_\theta U^T x$$

- [Bruna et al., 2013] 中的谱图卷积神经网络：

$$x_{k+1,j} = h \left(\sum_{i=1}^{f_k} U F_{k,i,j} U^T x_{k,i} \right)$$

$$j = 1, \dots, f_{k+1}$$

$$i = 1, \dots, f_{k-1}$$

f_k 类比于 CV 里面的卷积核个数，

- h 表示非线性激活函数



谱域图卷积的缺陷

- ① 计算拉普拉斯矩阵的特征值特征向量时间复杂度高，尤其是在大规模图数据上。
- ② 频域卷积无约束，容易丢失结构信息

无约束频域卷积丢失结构信息：

$$g_\theta(L) = \mathbf{I} - \frac{L^2}{2!} + \frac{L^4}{4!} \dots$$

当 L 的幂次足够大时，每个节点将与网络中的所有节点共享信号。



谱域图卷积操作的恒等变换：

$$y = g_\theta(L)x = g_\theta(U\Lambda U^T)x = Ug_\theta(\Lambda)U^Tx$$

- 而 Chebyshev 网络就是把 $g_\theta(\Lambda)$ 近似为切比雪夫多项式的 K 阶截断：

$$g_\theta(\Lambda) \approx \sum_{k=0}^K \theta_k T_k(\tilde{\Lambda}), \tilde{\Lambda} = 2\Lambda_n/\lambda_{\max} - I_n$$

- 采用切比雪夫多项式是因为它可以循环递归求解。为了避免特征值分解，我们可将上述改写为关于 L 的函数如下：

$$y = U \sum_{k=0}^K \theta_k T_k(\tilde{\Lambda}) U^T x = \sum_{k=0}^K \theta_k T_k(\tilde{L}) x, \tilde{L} = 2L/\lambda_{\max} - I_n$$

换一种更清晰的写法

$$g_\beta(\Lambda) = \sum_{k=0}^K \beta_k \Lambda^k \quad \Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$$

- 因为 $U\Lambda^k U^T = (U\Lambda U^T)^k = \tilde{L}^k$, $\tilde{L} = \frac{2}{\lambda_{\max}}L - I_N$
- 卷积公式就可以写成

$$\mathbf{x} *_G \mathbf{y} = \mathbf{U} g_\beta(\Lambda) \mathbf{U}^T \mathbf{x} = \sum_{k=0}^K \beta_k \tilde{L}^k \mathbf{x}$$



Spectral GCN VS ChebyNET 异同

- 不再需要特征向量分解
- 时间复杂度从 $O(n^2)$ 降到了 $O(|E|)$

$$\mathbf{x} *_G \mathbf{y} = \mathbf{U} g_\theta(\Lambda) \mathbf{U}^T \mathbf{x} = \sum_{k=0}^K \beta_k \tilde{L}^k \mathbf{x}$$

- 卷积的范围仅限于距离中心节点 k 跳的邻居节点范围内



ChebyNET-GCN

$$\mathbf{x} *_G \mathbf{y} = \mathbf{U} g_\theta(\Lambda) \mathbf{U}^T \mathbf{x} = \sum_{k=0}^K \beta_k \tilde{L}^k \mathbf{x}$$

- 设定 $k = 1$ 那卷积公式可以简化为

$$\mathbf{x} *_G \mathbf{y} \approx \beta'_0 \mathbf{x} + \beta'_1 (L - I_N) \mathbf{x}$$

- 进一步减少参数

$$\beta = \beta'_0 = \beta'_1$$

$$\mathbf{x} *_G \mathbf{y} \approx \beta(L) \mathbf{x}$$



- 设定 $k = 1$, 也就是只考虑一阶邻居信息, 卷积公式可以简化为:

$$\mathbf{x} *_G \mathbf{y} \approx \beta(L) \mathbf{x}$$

- 当图中每个节点的表示不是单独的标量而是一个大小为 C 的向量时, 可以使用其变体进行处理:

$$Z = LX\Theta$$

- 其中, $\Theta \in \mathbb{R}^{C \times F}$ 表示参数矩阵, $Z \in \mathbb{R}^{N \times F}$ 为相应的卷积结果。此时, 每个节点的节点表示被更新成了一个新的 E 维向量, 该 F 维向量包含了相应的一阶邻居上的信息。



$$Z = LX\Theta$$

- 再加上非线性激活函数，我们可以得到图卷积网络的最终公式：

$$H^{(l+1)} = \sigma(LH^{(l)}W^{(l)})$$

- 其中，第 l 层网络的输入为 $H^{(l)} \in \mathbb{R}^{N \times D}$ (初始输入为 $H^{(0)} = X$)， N 为图中的节点数量，每个节点使用 D 维的特征向量进行表示。



$$H^{(l+1)} = \sigma(LH^{(l)}W^{(l)})$$

- 这样做有什么问题？



归一化拉普拉斯

- 采用加法规则时，对于度大的节点特征越来越大，而对于度小的节点却相反，这可能导致网络训练过程中梯度爆炸或者消失的问题
- 所以我们需要对 L 进行归一化：**定义**归一化拉普拉斯如下：

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

$$\hat{A} = A + I$$

$$\hat{D} = D + I$$

- 简单的归一化方法: $\hat{D}^{-1} \hat{A}$
- 但是这样得到的矩阵是非对称阵
- 所以使用 $\hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2}$



归一化拉普拉斯

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}, \hat{A} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$
$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}, \hat{D} = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$



图傅里叶变换

$$\hat{D}^{-1} = \begin{bmatrix} 1/3 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 0 \\ 0 & 0 & 0 & 0 & 1/3 \end{bmatrix}$$

$$\hat{D}^{-1}\hat{A} = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 1/4 \\ 0 & 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \end{bmatrix}$$



图傅里叶变换

$$\hat{D}^{-1}\hat{A} = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 0 \\ 1/4 & 0 & 1/4 & 1/4 & 1/4 \\ 0 & 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \end{bmatrix}$$

该矩阵不满足拉普拉斯矩阵的对称性和半正定！

解决方案

对 \hat{A} 左乘 $\hat{D}^{-1/2}$, 右乘 $\hat{D}^{-1/2}$, 得到 $\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ 。

这样既得到了近似的归一化也保持了矩阵对称性。(左乘是行变换, 右乘是列变换。)

图傅里叶变换

$$\hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2} = \begin{bmatrix} \frac{1}{\sqrt{3}\sqrt{3}} & \frac{1}{\sqrt{3}\sqrt{3}} & \frac{1}{\sqrt{3}\sqrt{4}} & 0 & 0 \\ \frac{1}{\sqrt{3}\sqrt{3}} & \frac{1}{\sqrt{3}\sqrt{3}} & 0 & \frac{1}{\sqrt{3}\sqrt{4}} & 0 \\ \frac{1}{\sqrt{4}\sqrt{3}} & 0 & \frac{1}{\sqrt{4}\sqrt{4}} & \frac{1}{\sqrt{4}\sqrt{4}} & \frac{1}{\sqrt{4}\sqrt{3}} \\ 0 & \frac{1}{\sqrt{4}\sqrt{3}} & \frac{1}{\sqrt{4}\sqrt{4}} & \frac{1}{\sqrt{4}\sqrt{4}} & \frac{1}{\sqrt{3}\sqrt{4}} \\ 0 & 0 & \frac{1}{\sqrt{3}\sqrt{4}} & \frac{1}{\sqrt{3}\sqrt{4}} & \frac{1}{\sqrt{3}\sqrt{3}} \end{bmatrix}$$



图傅里叶变换

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

等价于：

$$h_v = f \left(\frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} Wx_u + b \right), \quad \forall v \in \mathcal{V}$$



References I

-  Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.
-  Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035.
-  Tang, S., Li, B., and Yu, H. (2019). Chebnet: Efficient and stable constructions of deep neural networks with rectified power units using chebyshev approximations. *arXiv preprint arXiv:1911.05467*.



References II

-  Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017).
Graph attention networks.
arXiv preprint arXiv:1710.10903.

