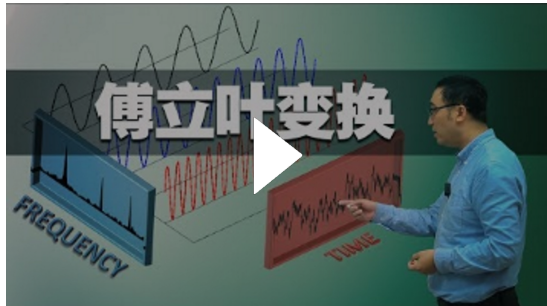


03 - 图卷积神经网络整理

2021年1月3日 18:23

文章地址: https://blog.csdn.net/weixin_39373480/article/details/90741121

相关视频: [傅立叶变换如何理解? 美颜和变声都是什么原理? 李永乐老师告诉你](#)



离散卷积

CNN中的卷积, 本质上就是利用共享参数的一个滤波器, 通过计算中心像素以及相邻像素点的加权求和来构成特征图, 来实现空间特征的提取。

卷积核的权重就是网络需要学习的参数。
卷积核的参数通过随机初始化再通过反向传播和梯度下降来迭代优化。

总结来说就是,
卷积核的参数, 是通过优化方法求出来的,
用于提取图像空间域上特征的一个工具。

它使得神经网络在这样的特征输入全连接层或者反卷积层以后能更好完成任务。

转折: 上面的这些假设都是在欧氏空间的。
欧式空间由一些很好的性质, 但是我们生活中有很多数据都是非欧氏空间的, 比如社交网络, 知识图谱, 3D的蛋白质结构或者点云之类的, 这样的数据, 我们称为图。

对于这些数据, 传统的CNN是无法对其进行局部的特征提取, 主要是因为它们具有非结构化的特点。
即, 每个顶点的邻居节点的个数可能不一样, 因此不能用传统的离散卷积去提取特征。

对于图来说

$$G = (V, E)$$

V 是图的节点集合

E 是图的边集合

设 W 为图的邻接矩阵

$|V|$ 表示节点的个数

D 叫做图的度矩阵 degree matrix

其中 V 是图的结点集合, E 是图的边集合, 设 W 是图的邻接矩阵, $|V|$ 表示结点的个数。 D 被称为图的**度矩阵Degree matrix**, 其中 $D = \text{diag}(d_1, \dots, d_N)$, 其中 $d_i = \sum_{j=1}^N W_{ij}$, 即每个结点的邻居的个数。

目前在图上的卷积定义基本分成两类: 基于谱的图卷积 + 基于空间域的图卷积

基于谱的图卷积:

通过傅里叶变化将节点映射到频域空间, 通过在频域空间上做乘积来实现时域上的卷积, 最后再将做完卷积的特征映射回时域空间;

基于空间的图卷积:

和传统的CNN类似, 只不过在图结构上更难定义节点的邻居以及于邻居之间的关系。

傅里叶变换

就是将时域上的函数, 转换为频域上的函数, 是一个函数的不同视角:

$$F(w) = \mathcal{F}(f(t)) = \int f(t) e^{-iwt} dt$$

傅里叶变换就是时域信号域拉普拉斯算子特征函数的积分。

我们由整体来推特殊, 由于离散积分就是一种内积的形式, 如果我们可以找到一个graph上的拉普拉斯算子, 我们可以仿照上面去定义graph上的傅里叶变换的形式:

$$F(\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^N f(i) u_l^*(i)$$

f 是graph上的 N 维向量, $f(i)$ 与graph的顶点一一对应。

f 是graph上的 N 维向量, $f(i)$ 与graph的顶点一一对应, $u_l(i)$ 表示第 l 个特征向量的第 i 个分量。那么特征值 λ_l 下的 f 的傅里叶变换就是与 λ_l 对应的特征向量 $u_l(i)$ 的内积运算。

因此对一个Graph上的拉普拉斯算子, 如果它由 N 个特征值, 那么其傅里叶变换可以推广到矩阵形式:

$$\begin{pmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{pmatrix} = \begin{pmatrix} u_1(1) & u_1(2) & \dots & u_1(N) \\ u_2(1) & u_2(2) & \dots & u_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ u_N(1) & u_N(2) & \dots & u_N(N) \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{pmatrix}$$

即

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

最后，什么是图的拉普拉斯算子呢？

其实就是拉普拉斯矩阵，对于一个图G来说，其拉普拉斯矩阵的定义为：


$$L = D - A$$

L 为拉普拉斯矩阵

D是顶点的度矩阵（对角矩阵），也就是对角线上的元素是以各个顶点的度

A为图的邻接矩阵

例子如下：

Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

在大部分GCN中，使用的拉普拉斯矩阵实际是Symmetric Normalized Laplacian

$$L^{sym} = D^{\frac{1}{2}} L D^{\frac{1}{2}}$$

类似地，**传统傅里叶变换的逆变换**是对 w 求积分

$$\mathcal{F}^{-1}[F(\omega)] = \frac{1}{2\Pi} \int F(\omega) e^{i\omega t} d\omega$$

迁移到graph上就是

$$f(i) = \sum_{l=1}^N \hat{f}(\lambda_l) u_l(i)$$

类似地也可以写成矩阵的形式：

$$\begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{pmatrix} = \begin{pmatrix} u_1(1) & u_2(1) & \dots & u_N(1) \\ u_1(2) & u_2(2) & \dots & u_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(N) & u_2(N) & \dots & u_N(N) \end{pmatrix} \begin{pmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{pmatrix}$$

即

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

推广卷积

基于谱的图卷积，通过傅里叶变换将节点映射到频域空间。

通过在频域空间上做乘积来实现时域上的卷积，最后再将卷积的特征映射回时域空间。

这里用到的就是卷积定理，也就是函数卷积的傅里叶变化是韩式傅里叶变换的乘积

$$\mathcal{F}(f * h) = \hat{f}(w) \cdot \hat{h}(w)$$

其中*为卷积操作，上式可以整理为：

$$f * h = \mathcal{F}^{-1}(\hat{f}(w) \cdot \hat{h}(w)) = \frac{1}{2\pi} \int \hat{f}(w) \cdot \hat{h}(w) e^{-iwt} dw$$

这就是基于谱的图卷积的最初版本，其也可以写成矩阵计算的形式：

$$(f * h)_G = U \begin{pmatrix} \hat{h}(\lambda_1) & & & \\ & \hat{h}(\lambda_2) & & \\ & & \dots & \\ & & & \hat{h}(\lambda_N) \end{pmatrix} U^T f$$

即

$$(f * h)_G = U((U^T f) \odot (U^T h))$$

其中 \odot 表示对应位置的乘积计算，即内积。

第一代GCN: Spectral Networks and Locally Connected Networks on Graphs 计算量大，每次卷积都要进行矩阵相乘；没有空间局部性，每次卷积都要考虑所有的节点；参数的个数为 $O(N)$ ，传统的CNN参数个数为常数；

第二代GCN: Convolutional Neural Networks on Graph with Fast Localized Spectral Filtering
引入空间局部性，降低了计算复杂度；

第三代GCN: Semi-supervised Classification with Graph Convolutional Networks
进一步降低计算复杂度，只考虑1-hot邻域，通过堆叠多层增加感受野

(出于数学上的一些推导，并没有仔细去看，所以原理还不是很清楚，只是知道了大概，所以等需要的时候再来研究吧)

关于自己实现图卷积的，还有一个Numpy的简单实现

[图卷积网络到底怎么做，这是一份极简的Numpy实现 \(qq.com\)](#)

简单看了下，也是比较好的介绍了原理，但是没有真正的进行计算。

值得研究

基于空间域的图卷积

Graph Attention Network 这个就是代表作，然后还有这个作者的一个解读
包含网络结构和代码

根据这个博客，了解了作者就是清华的一个计算机硕士。
是非常值得自己学习的。
看看人家的学术精神和态度，啧啧。