实验五：图卷积神经网络

姓名：刘威

学号：PB18010469

Click here to finish reading:-)

实验目的

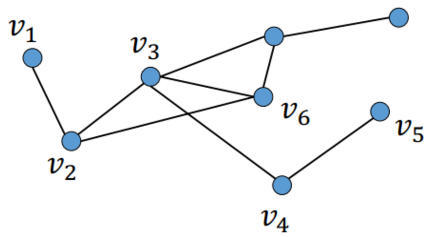
- 熟悉图卷积神经网络的基本原理
- 了解网络层数对图卷积神经网络性能的影响
- 了解不同激活函数，Add self loop, DropEdge, PairNorm等技术对图卷积神经网络性能的影响。

实验原理

图的基本概念

- 图的矩阵$\boldsymbol{A}$表示
  - 邻接矩阵 $A_{ij} = 1$, 如果v_i和v_j相邻
  - 度矩阵 $\boldsymbol{D} = \mathrm{diag}(d(v_1), \ldots, d(v_N))$

$$d(v_i) = \sum_{v_j \in \mathcal{N}_{v_i}} A_{ij}$$



度矩阵

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$\boldsymbol{D}$

邻接矩阵

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$\boldsymbol{A}$

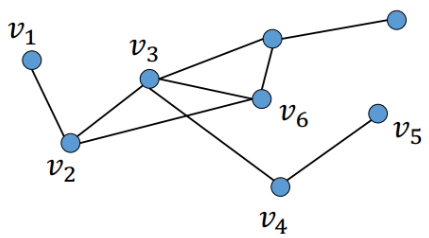
- 拉普拉斯矩阵的性质 $\boldsymbol{L}=\boldsymbol{D}-\boldsymbol{A}$
- 1） $\boldsymbol{L}\,\boldsymbol{1}=\boldsymbol{D}\boldsymbol{1}-\boldsymbol{A}\boldsymbol{1}=\boldsymbol{d}\;-\boldsymbol{d}=\boldsymbol{0}$ $\qquad\qquad \boldsymbol{d}=\mathrm{diag}(\boldsymbol{D})$
- 2） $\boldsymbol{1}\,\boldsymbol{L}=\boldsymbol{0}$
- 3） $\boldsymbol{L}$ 是半正定的。最小特征值为0，其特征向量为$\boldsymbol{1}$

证明： 对于任意的向量$\boldsymbol{z}\in\mathbb{R}^n$，那么

$\boldsymbol{L}$的特征值中0的个数等同于原图的连通块数量

$$
\begin{aligned}
\boldsymbol{z}^\top\boldsymbol{L}\,\boldsymbol{z} &= \sum_i d_i z_i^2-\sum_{ij}A_{ij}z_i z_j \\
&=\frac{1}{2}\left(\sum_i d_i z_i^2-2\sum_{ij}A_{ij}z_i z_j+\sum_j d_j z_j^2\right) \\
&=\frac{1}{2}\left(\sum_i\sum_j A_{ij}z_i^2-2\sum_{ij}A_{ij}z_i z_j+\sum_j\sum_i A_{ij}\,z_j^2\right) \\
&=\frac{1}{2}\sum_i\sum_j A_{ij}\left(z_i-z_j\right)^2\geq 0
\end{aligned}
$$

- 归一化后的拉普拉斯矩阵
  - 对称归一化: $\boldsymbol{L}^{sym}=\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}=\boldsymbol{I}-\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$
  - 随机游走归一化: $\boldsymbol{L}^{rw}=\boldsymbol{D}^{-1}\boldsymbol{L}=\boldsymbol{I}-\boldsymbol{D}^{-1}\boldsymbol{A}$



度矩阵 邻接矩阵 拉普拉斯矩阵

$$
\underbrace{\begin{pmatrix}
1&0&0&0&0&0&0&0\\
0&3&0&0&0&0&0&0\\
0&0&4&0&0&0&0&0\\
0&0&0&2&0&0&0&0\\
0&0&0&0&2&0&0&0\\
0&0&0&0&0&4&0&0\\
0&0&0&0&0&0&3&0\\
0&0&0&0&0&0&0&1
\end{pmatrix}}_{\boldsymbol{D}}
-
\underbrace{\begin{pmatrix}
0&1&0&0&0&0&0&0\\
1&0&1&0&0&1&0&0\\
0&1&0&1&0&1&1&0\\
0&0&1&0&1&0&0&0\\
0&0&0&1&0&1&0&0\\
0&1&1&0&1&0&1&0\\
0&0&1&0&0&1&0&1\\
0&0&0&0&0&0&1&0
\end{pmatrix}}_{\boldsymbol{A}}
=
\underbrace{\begin{pmatrix}
1&-1&0&0&0&0&0&0\\
-1&3&-1&0&0&-1&0&0\\
0&-1&4&-1&0&-1&-1&0\\
0&0&-1&2&-1&0&0&0\\
0&0&0&-1&2&-1&0&0\\
0&-1&-1&0&-1&4&-1&0\\
0&0&-1&0&0&-1&3&-1\\
0&0&0&0&0&0&-1&1
\end{pmatrix}}_{\boldsymbol{L}}
$$

- 归一化后的拉普拉斯矩阵$\boldsymbol{L}^{sym}$的性质:
- 1) $\boldsymbol{L}^{sym}$特征值的介于0到2之间

图傅里叶变换

- $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\top}$为其特征值分解， $\boldsymbol{U}$的列向量类比于傅里叶变换中的基
- 对图上信号$\boldsymbol{z}$的图傅里叶变换：

$$\mathcal{F}(\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^{n} f(i)u_l(i) \ = \boldsymbol{u}_l{}^{\top}\boldsymbol{f}$$

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

$$\hat{\boldsymbol{f}} = \boldsymbol{U}^{\top}\boldsymbol{f}$$

图傅里叶逆变换

- $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\top}$为其特征值分解， $\boldsymbol{U}$的列向量类比于傅里叶变换中的基
- 对图上信号$\boldsymbol{z}$的图傅里叶逆变换：

$$f(i) = \sum_{l=1}^{n} \hat{f}(\lambda_l)u_l(i) \ = \boldsymbol{u}(i)^{\top}\hat{\boldsymbol{f}}$$

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

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

任一信号$\boldsymbol{f}$是不同频率的特征向量的组合

谱域上的图卷积

- 空域上很难定义卷积-->转到谱域

- 从图信号处理的角度考虑图卷积
- 卷积公式 $\boldsymbol{f} * \boldsymbol{g} = \mathbb{F}^{-1}\{\mathbb{F}\{\boldsymbol{f}\} \cdot \mathbb{F}\{\boldsymbol{g}\}\}$

卷积定理：函数卷积的傅里叶变换是函数傅立叶变换的乘积

- 给一个图信号$\boldsymbol{x}$和一个卷积核$\boldsymbol{g}$ $\quad \boldsymbol{x} * \boldsymbol{g} = \boldsymbol{U}(\boldsymbol{U}^{\top}\boldsymbol{x} \odot \boldsymbol{U}^{\top}\boldsymbol{g})$
- $\boldsymbol{U}^{\top}\boldsymbol{g}$当成整体的卷积核，用参数$\boldsymbol{\theta}$表示

$$\boldsymbol{x} * \boldsymbol{g} = \boldsymbol{U}(\boldsymbol{U}^{\top}\boldsymbol{x} \odot \boldsymbol{U}^{\top}\boldsymbol{g}) = \boldsymbol{U}(\boldsymbol{U}^{\top}\boldsymbol{x} \odot \boldsymbol{\theta}) \ = \boldsymbol{U}\boldsymbol{g}_{\boldsymbol{\theta}}\boldsymbol{U}^{\top}\boldsymbol{x}$$

用$\boldsymbol{g}_{\theta}$表示对角线为$\boldsymbol{\theta}$的矩阵

- 回顾:

$$\text{GFT}: \hat{\boldsymbol{f}} = \boldsymbol{U}^{\top} f \qquad\qquad \text{IGFT}: \boldsymbol{f} = \boldsymbol{U}\hat{\boldsymbol{f}}$$

- 滤波信号：空域信号→谱域信号→滤波→空域信号

$$\boldsymbol{f} \xrightarrow[\text{Decompose}]{\text{GFT}} \underset{\text{Coefficients}}{\boxed{\boldsymbol{U}^{\top} f}} \xrightarrow[\text{Filter}]{\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})} \underset{\text{Filtered coefficients}}{\boxed{\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})\boldsymbol{U}^{\top}\boldsymbol{f}}} \xrightarrow[\text{Reconstruct}]{\text{IGFT}} \boxed{\boldsymbol{U}\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})\boldsymbol{U}^{\top}\boldsymbol{f}}$$

最简单的谱域图卷积网络

- Spectral Graph CNN
  - 无参数化: $\boldsymbol{g}_{\theta}$与$\boldsymbol{L}$的特征向量无关

$$\boldsymbol{x} * \boldsymbol{g} = \boldsymbol{U}\boldsymbol{g}_{\theta}\boldsymbol{U}^{\top}\boldsymbol{x} = \boldsymbol{U}\begin{pmatrix} \theta_1 & & & \\ & \theta_2 & & \\ & & \ddots & \\ & & & \theta_n \end{pmatrix}\boldsymbol{U}^{\top}\boldsymbol{x}$$

- 缺点
  - 参数多，总共有n个参数，n为节点数
  - 需要对拉普拉斯矩阵进行特征分解，$O(n^3)$时间复杂度
  - 不能局部化：每个节点上信号依赖于其他全部节点信号，而不只是邻居

多项式卷积核（ChebyNet）

- 局部化：$\boldsymbol{L}$对图信号$\boldsymbol{f}$的操作$\boldsymbol{L}\,\boldsymbol{f}$相当于在图上传播一步

常用归一化的$\boldsymbol{L}$，比如$\boldsymbol{L}_{sym}$

$$\boldsymbol{f} \xrightarrow[\text{Decompose}]{\text{GFT}} \underset{\text{Coefficients}}{\boxed{\boldsymbol{U}^{\top} f}} \xrightarrow[\text{Filter}]{\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})} \underset{\text{Filtered coefficients}}{\boxed{\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})\boldsymbol{U}^{\top}\boldsymbol{f}}} \xrightarrow[\text{Reconstruct}]{\text{IGFT}} \boxed{\boldsymbol{U}\,\underbrace{\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})}_{\boldsymbol{g}_{\theta}(\boldsymbol{L})}\,\boldsymbol{U}^{\top}\boldsymbol{f}}$$

假设滤波器是拉普拉斯矩阵的函数

- 多项式卷积核:

$$\boldsymbol{g}_{\theta}(\boldsymbol{\Lambda}) = \sum_{k=0}^{K}\theta_k\boldsymbol{\Lambda}^k$$

$$\hat{g}(\Lambda) = \begin{bmatrix} \sum_{k=0}^{K}\theta_k\lambda_1^k & & & \\ & \sum_{k=0}^{K}\theta_k\lambda_2^k & & \\ & & \cdots & \\ & & & \sum_{k=0}^{K}\theta_k\lambda_N^k \end{bmatrix}$$

$$\boldsymbol{U}\hat{g}(\boldsymbol{\Lambda})\boldsymbol{U}^{T}\boldsymbol{f} = \boldsymbol{U}\sum_{k=0}^{K}\theta_k\boldsymbol{\Lambda}^k\,\boldsymbol{U}^{T} = \sum_{k=0}^{K}\theta_k\boldsymbol{L}^k\,\boldsymbol{f}$$

$$\boldsymbol{L}^k = \boldsymbol{U}\boldsymbol{\Lambda}^k\boldsymbol{U}^{T}$$

局部性：实际上在图上传播了K步，因此一个节点只影响它周围距离为K以内的邻居
效率高：不需要进行特征值分解
参数少：只有K+1个参数

- 切比雪夫多项式，通过如下递归定义
  - $T_0(x) = 1;\ T_1(x) = x$
  - $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$

- 由于其定义域为$[-1,1]$，因此$\hat{g}(\boldsymbol{\Lambda})$通过如下方式定义

$$\hat{g}(\boldsymbol{\Lambda}) = \sum_{k=0}^{K} \theta_k T_k(\widetilde{\boldsymbol{\Lambda}}) \qquad \text{其中 } \widetilde{\boldsymbol{\Lambda}} = \frac{2\boldsymbol{\Lambda}}{\lambda_{max}} - \boldsymbol{I}$$

- 在归一化之后，任然可以不用计算特征值分解

- 有结论 $\boldsymbol{U}\hat{g}(\boldsymbol{\Lambda})\boldsymbol{U}^{\top}\boldsymbol{f} = \sum_{k=0}^{K} \theta_k T_k(\tilde{\boldsymbol{L}})\,\boldsymbol{f}$, 其中 $\tilde{\boldsymbol{L}} = \frac{2\boldsymbol{L}}{\lambda_{max}} - \boldsymbol{I}$

归一化图拉普拉斯矩阵$\boldsymbol{L}^{sym}$
最大特征值约为2，$\lambda_{max} \approx 2$

$$\boldsymbol{L}^{sym} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$$

$$\Rightarrow \tilde{\boldsymbol{L}} = \frac{2\boldsymbol{L}^{sym}}{\lambda_{max}} - \boldsymbol{I} = \boldsymbol{L}^{sym} - \boldsymbol{I} = -\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$$

- 假设我们只取1阶切比雪夫多项式，且$\lambda_{max} \approx 2$
- 此时节点只能被它周围的1阶邻接点所影响，但只需要叠加K层这样的图卷积层，就可以把节点的影响力扩展到K阶邻居节点

$$\boldsymbol{y} = \sum_{k=0}^{1} \theta_k T_k(\tilde{\boldsymbol{L}})\,\boldsymbol{x} \approx \theta_0 T_0(\tilde{\boldsymbol{L}})\boldsymbol{x} + \theta_1 T_1(\tilde{\boldsymbol{L}})\boldsymbol{x}$$

$$\approx \theta_0\boldsymbol{x} - \theta_1\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{x}$$

取$\theta' = \theta_0 = -\theta_1$ $\quad \approx \theta'\left(\boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\right)\boldsymbol{x}$

根据类似方法，可证明可得 $\boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$的特征值在$[0,2]$之间，叠加多层图卷积层会多次迭代这个操作，可能造成数值不稳定和梯度爆炸的问题

$$\boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}} \xrightarrow{\text{做归一化}} \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\,\widetilde{\boldsymbol{D}}^{-\frac{1}{2}} \Rightarrow \boldsymbol{y} = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\,\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{X}\boldsymbol{\Theta}$$

$\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$
$\widetilde{\boldsymbol{D}} = \boldsymbol{D} + \boldsymbol{I}$

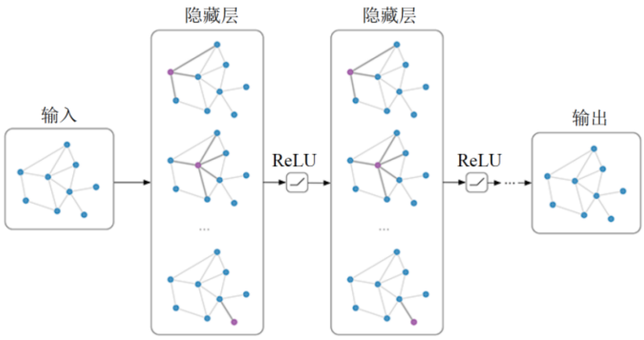
特征值在$[0,1]$

$\boldsymbol{X} \in R^{n\times d}$: 节点属性矩阵

从ChebyNet到GCN

- 叠加多层GCN得到最终的模型(常取两层)

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



其中

$$\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}$$
$$\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_n$$
$$\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$$

实验内容

- 给定三个图数据，实现GCN算法用于节点分类和链路预测，分析自环、层数、DropEdge、PariNorm、激活函数等因素的分类性能和链路预测性能的影响
- 请助教准备Cora、Citeseer、PPI的数据

实验结果

- 本实验使用`PyTorch`进行，并主要使用了`pytorch_geometric`库。
- 本实验再`Cora`和`Citeseer`两个数据集上进行了节点分类，并比较了自环，层数，DropEdge，PairNorm，激活函数对其分类性能的影响。

源码结构及说明

数据处理部分

数据集概览：

Dataset	Nodes	Edges	Classes	Features
Citeseer	3327	4732	6	3703
Cora	2708	5429	7	1433

数据处理方法：

按照`pytorch_geometric`的数据输入格式，将顶点关联的`features`组织成一个二维矩阵`x: shape=(Nodes, Features)`，将图结构，即顶点的连接关系用 `COO` 格式组织成一个二维矩阵`edge_index: shape=(2, Edges)`(邻接矩阵的稀疏表示). 将标签处理为一维向量`y: shape=(Nodes,)`其取值范围为`range(Classes)`.

通过`mask`将顶点划分为`train, val, test`. 其中`train_mask`覆盖每个类别分别20个顶点，`val_mask`覆盖除`train_mask`外的随机500个顶点，`test_mask`覆盖除前两者外的随机1000个顶点。图结构难以拆解成三个部分，因此图是一整个输入到网络中的，也即所有的顶点都会参与计算。`train_mask`的作用是，在计算损失时，将其他顶点mask掉，只计算训练顶点的损失。同样地，通过`val_mask,test_mask`我们可以分别计算`val`和`test`顶点的分类准确率。

模型部分

网络用`n_layers`**层**`GCN`堆叠而成，在每层`GCN`后都紧跟一层**可选的**`PairNorm`**层**；除了最后一层外，每层的`PairNorm`后还有有激活函数和`dropout`，**激活函数可以选择**`relu, tanh, sigmoid`，`dropout`可以调节drop的概率p.

其中`GCN`直接使用`pytorch_geometric`库中的`GCNConv`层，它可以**通过参数**`add_self_loop`**设置是否添加自环。**在输入`GCN`之前，还可以通过**设置**`drop_edge`**的drop比例**去掉部分`edge_index`.

完整的模型定义如下：

```
import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv, PairNorm
from torch_geometric.utils import dropout_adj

activations = {
    'relu': torch.relu,
    'sigmoid': torch.sigmoid,
    'tanh': torch.tanh,
}

class GCN(torch.nn.Module):
    def __init__(self, in_channels: int, hidden_channels: int, num_classes: int,
                 n_layers: int, act: str = 'relu', add_self_loops: bool = True,
                 pair_norm: bool = True, dropout: float = .0, drop_edge: float =
.0):
        super(GCN, self).__init__()
        self.dropout = dropout
        self.drop_edge = drop_edge
        self.pair_norm = pair_norm
        self.act = activations[act] if isinstance(act, str) else act

        self.conv_list = torch.nn.ModuleList()
        for i in range(n_layers):
            in_c, out_c = hidden_channels, hidden_channels
            if i == 0:
                in_c = in_channels
            elif i == n_layers - 1:
                out_c = num_classes
            self.conv_list.append(GCNConv(in_c, out_c,
add_self_loops=add_self_loops))

    def forward(self, x, edge_index):
        edge_index, _ = dropout_adj(edge_index, p=self.drop_edge)

        for i, conv in enumerate(self.conv_list):
            x = conv(x, edge_index)
            if self.pair_norm:
                x = PairNorm()(x)
            if i < len(self.conv_list) - 1:
                x = self.act(x)
                x = F.dropout(x, p=self.dropout, training=self.training)

        return x
```

结果及分析

参数设置

本实验的可选参数及其默认值为

```
default_cfg = {
    'data_root': './GNN/',  # 数据根目录
    'data_name': 'cora',  # citeseer or cora
    'num_train_per_class': 20,  # 训练集包含的每个类别的顶点数目
    'num_val': 500,  # 验证集顶点数目
    'num_test': 1000,  # 测试集顶点数目
    'seed': 114514,
    'device': 'cuda:0',
    'epochs': 1000,
    'patience': 5,  # 早停的等待轮数
    'lr': 5e-3,
    'weight_decay': 5e-4,
    'hidden_dim': 32,
    'n_layers': 2,
    'activations': 'relu',
    'dropout': 0.5,
    'drop_edge': 0.,
    'add_self_loop': True,
    'pair_norm': False,
    'test_only': False
}
```

其中本实验进行调节的参数及调节的范围为

```
cfg_grid = {
    'data_name': ['citeseer', 'cora'],
    'add_self_loop': [True, False],
    'n_layers': [1, 2, 3, 5, 10],
    'drop_edge': [0, .1, .2, .3, .5],
    'pair_norm': [True, False],
    'activations': ['relu', 'tanh', 'sigmoid']
}
```

共有600种可能的参数组合。 在每种参数组合下，分别训练模型，并通过验证集`val_loss`进行早停，以`val_loss`最低时的模型权重对测试集进行测试，以其分类准确率`test_acc`作为最终评价指标。

结果对比分析

所有的组合下的`test_acc`结果可以在附件[`result.csv`]中查看，下面仅列举出部分结果。

两个数据集上的最好结果及对应参数

`data_name`	`add_self_loop`	`n_layers`	`drop_edge`	`pair_norm`	`activations`	`test_acc`
'cora'	True	2	0.	False	'relu'	0.797
'citeseer'	True	2	0.	False	'relu'	0.685

> **Note**: 下面的对比均以`citeseer`数据集为例, 即`data_name='citeseer'`

是否添加自环的对比

Selected Compairson:

data_name	add_self_loop	n_layers	drop_edge	pair_norm	activations	test_acc
citeseer	True	3	0.0	False	relu	0.645
citeseer	False	3	0.0	False	relu	0.628
citeseer	True	2	0.0	False	relu	0.685
citeseer	False	2	0.0	False	relu	0.667

分析：添加自环效果好，在某些参数下提升非常显著。

不同层数的对比

Selected Compairson:

data_name	add_self_loop	n_layers	drop_edge	pair_norm	activations	test_acc
citeseer	True	1	0.0	False	relu	0.68
citeseer	True	2	0.0	False	relu	0.685
citeseer	True	3	0.0	False	relu	0.645
citeseer	True	5	0.0	False	relu	0.522
citeseer	True	10	0.0	False	relu	0.176

分析：两层效果最好， 层数多难以优化。

drop edge的对比

data_name	add_self_loop	n_layers	drop_edge	pair_norm	activations	test_acc
citeseer	True	5	0.0	False	relu	0.522
citeseer	True	5	0.1	False	relu	0.351
citeseer	True	5	0.2	False	relu	0.182
citeseer	True	5	0.3	False	relu	0.201
citeseer	True	5	0.5	False	relu	0.188
citeseer	True	3	0.0	False	relu	0.645
citeseer	True	3	0.1	False	relu	0.671
citeseer	True	3	0.2	False	relu	0.655
citeseer	True	3	0.3	False	relu	0.663
citeseer	True	3	0.5	False	relu	0.609

分析：层数少时drop edge 有点效果，层数深时效果不好。

是否使用PairNorm的对比

data_name	add_self_loop	n_layers	drop_edge	pair_norm	activations	test_acc
citeseer	True	1	0.0	False	relu	0.68
citeseer	True	1	0.0	True	relu	0.443
citeseer	True	2	0.0	False	relu	0.685
citeseer	True	2	0.0	True	relu	0.526
citeseer	True	3	0.0	False	relu	0.645
citeseer	True	3	0.0	True	relu	0.568
citeseer	True	5	0.0	False	relu	0.522
citeseer	True	5	0.0	True	relu	0.545
citeseer	True	10	0.0	False	relu	0.176
citeseer	True	10	0.0	True	relu	0.3

分析：层数少时加`PairNorm`效果变差，层数多时`PairNorm`有效果。

不同激活函数的对比

data_name	add_self_loop	n_layers	drop_edge	pari_norm	activations	test_acc
citeseer	True	2	0.0	False	relu	0.685
citeseer	True	2	0.0	False	tanh	0.683
citeseer	True	2	0.0	False	sigmoid	0.207
citeseer	True	3	0.0	False	relu	0.645
citeseer	True	3	0.0	False	tanh	0.667
citeseer	True	3	0.0	False	sigmoid	0.207
citeseer	True	5	0.0	False	relu	0.522
citeseer	True	5	0.0	False	tanh	0.588
citeseer	True	5	0.0	False	sigmoid	0.207
citeseer	True	10	0.0	False	relu	0.176
citeseer	True	10	0.0	False	tanh	0.472
citeseer	True	10	0.0	False	sigmoid	0.195

分析：2层和3层时`relu~=tanh>>sigmoid`, 3层和5层`tanh>relu>>sigmoid`, 10层`tanh>>sigmoid~=relu`。

实验总结

本次实验的最大收获在于了解的图神经网络的原理，以及学会了使用`torch_geometric`库。

原GCN论文里面`citeseer`和`cora`数据集的最好结果（%）分别为 70.3 和 81.5， 我这里略差，分别是 68.5 和 79.7。其实网络结构是一样的，也是两层，用`relu`作为激活函数。我对比了一下才发现原因：**它那个数据集划分是某种特定的划分。**虽然划分比例相同，但在它那个划分下结果就是好不少，主要差别就在于这里。~~(你们这些做学术的人都在调些什么啊，dataset split is all you need ?)~~

(完)