深度学习第四次实验作业报告

课程/2023 深度学习　学号/SA23229086　学生/郎文翀
大数据学院计算机技术班
2024 年 1 月 7 日星期日

1 实验要求

使用 pytorch 或者 tensorflow 的相关神经网络库，编写图卷积神经网络模型 (GCN)，并在相应的图结构数据集上完成节点分类和链路预测任务，最后分析自环、层数、DropEdge 、PairNorm 、激活函数等因素对模型的分类和预测性能的影响 。

2 实验设计

2.1 设计需求

本次实验选择 Pytorch 实现，并使用 GPU 版本进行训练，最后采用 tensorboard 完成绘图与实验结果分析，具体程序设计需求如下：

数据集处理：本次实验数据集选用 cora, citeseer 数据集，需要自行划分训练集，验证集和测试集。

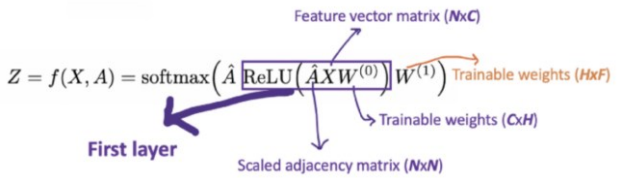
模型参数设置：设置不同的 config 配置选项构建模型，对比分析自环，层数，DropEdge 等对模型在不同任务表现上的影响。

分批次训练：每 Epoch 数据分为 64 一组进行训练。

2.2 网络模型设计

本次实验主要参考论文《SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS》实现，其推理过程可参考博客讲解：

图卷积网络(Graph Convolutional Networks, GCN)详细介绍-CSDN 博客。具体的数学公式推导为：



这里的 Ahat 矩阵就是图神经网络的核心推导公式，其具体组成见 下图，其功能是参考各节点的度数生成度矩阵与 A 矩阵相乘得到节点特征融合功能。

本次实验主要是使用 cora,citeseer 数据集搭建 GCN 网络，进一步基于搭建的模型完成节点分类与链路预测两个任务，虽然任务类型不同，但是均使用同一网络模型实现，只是部分网络超参数不同，具体可以分为 11 组不同的 config 配置对应多个不同超参数对比，分别为：

1. 网络模型深度：num_layers 设置为 2，4，6 分析信息交换次数对模型的影响
2. 网络是否包含自环：在网络模型深度均为 2 的基础上 add_self_loops 分别设置为 true 和 false 分析节点信息交换时包含本身信息的必要性
3. 数据标准化：探讨度矩阵系数对数据标准化后的模型表现影响
4. 边保留：对比分析每轮迭代节点交换信息时邻居选取策略对模型的影响
5. 激活函数：探讨 Relu,tanh,leakyRelu 对模型的性能影响

综上，本次共进行了 11 次对比实验具体模型参数见下表，基于 cora 与 citeseer 数据集不同配置模型的表现情况分析超参数影响。

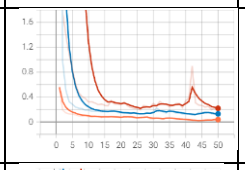
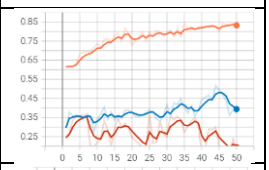
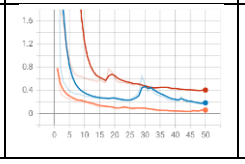
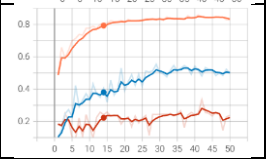
网络深度 num_layers	是否自环 add_self_loops	标准化 use_pairnorm	边保留 Drop_edge	激活函数 Activation
2	√	×	1	Relu
4	√	×	1	Relu
6	√	×	1	Relu
2	×	×	1	Relu
2	√	√	1	Relu
4	√	√	1	Relu
6	√	√	1	Relu
2	√	×	0.6	Relu
4	√	×	0.6	Relu
2	√	×	1	tanh
2	√	×	1	leakyRelu

最终实验结果可视化图像与输出日志见文件夹 logs 与 output。

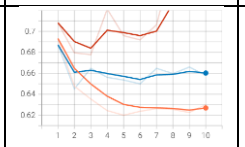
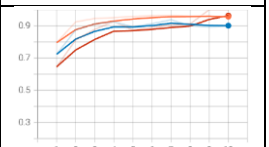
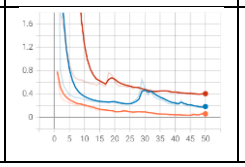
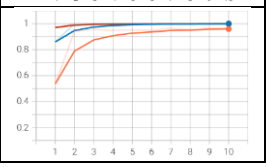
3 实验分析

3.1 模型深度对比

节点分类任务：

Config 配置	训练集损失	验证集准确率
Cora 数据集+深度=2,4,6		
Citeseer 数据集+深度=2,4,6		

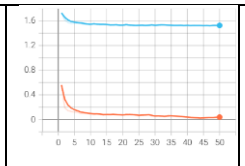
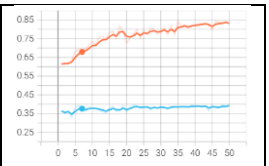
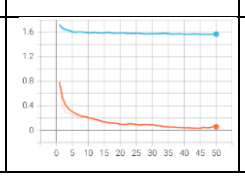
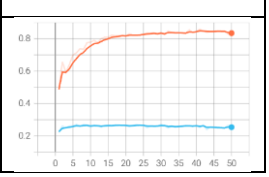
链路预测任务：

Config 配置	训练集损失	验证集准确率
Cora 数据集+深度=2,4,6		
Citeseer 数据集+深度=2,4,6		

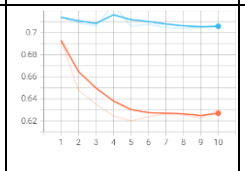
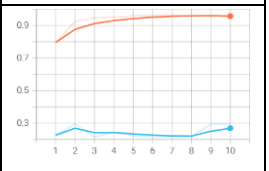
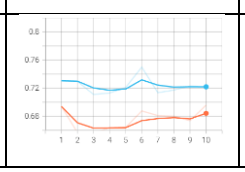
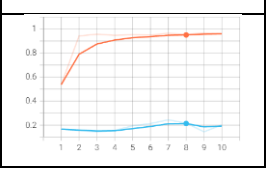
对比在 Cora 和 Citseer 数据集上模型对节点分类任务与链路预测任务的结果对比图不难发现深度=2 的模型表现最佳，而深度=6 的模型表现反而较差，分析原因可能是因为伴随着网络深度增加，节点间交换信息频率增加，但是由于图规模较小，在 2 次后即已经可以实现节点间特征信息的充分交换融合，进一步增加网络深度扩大节点交换信息频率可能导致各节点信息注意分散无法充分关注局部信息，因此导致深度更大的模型节点分类表现性能反而降低。

3.2 自环影响

节点分类任务：

Config 配置	训练集损失	验证集准确率
Cora 数据集+自环/无自环		
Citeseer 数据集+自环/无自环		

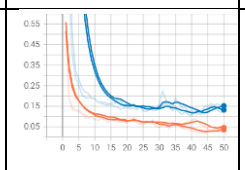
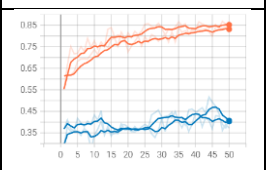
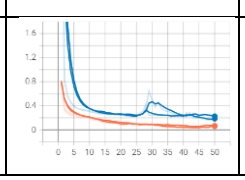
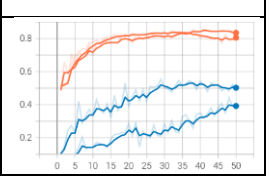
链路预测任务：

Config 配置	训练集损失	验证集准确率
Cora 数据集+自环/无自环		
Citeseer 数据集+自环/无自环		

对比上图不难发现，无自环的模型 loss 居高不下同时在验证集上准确率极低，分析原因是因为在于相邻节点交换融合特征信息时，为了维持自身所携带的信息特征，节点有必要加入自环以维持自身特征，无自环的图节点在互相交换融合特征后可能会丢失自身特征属性导致在最终节点分类时表现性能下降。

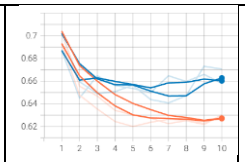
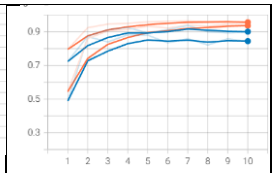
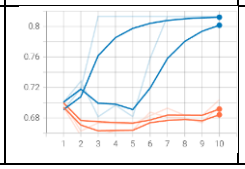
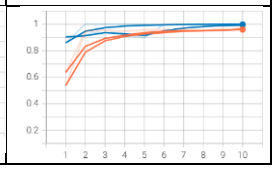
3.3 DropEdge

节点分类任务：

Config 配置	训练集损失	验证集准确率
Cora+有/无 DropEdge		
Citeseer +有无 DropEdge		

链路预测任务：

Config 配置	训练集损失	验证集准确率

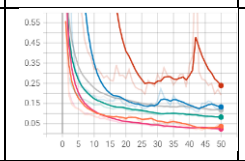
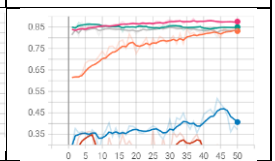
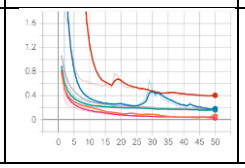
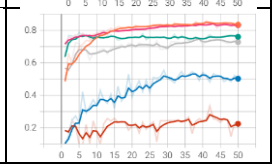
Cora+ 有/无 DropEdge		
Citeseer +有无 DropEdge		

这里首先注意 DropEdge 数值表示的是保存的边比例，因此 1-DropEdge 才是实际上丢弃的边比例，同时在真正的训练时是采用 Neighbors 提供的接口逐 num_layer 开放临边采样数量限制，对比上图结果发现无论是深度=2 或 4，有 Dropout 的模型的表现性能更加，推测可能是因为当面临每一个节点邻接节点很多时，训练初期抛弃部分临边，使得每一个节点之和少量几个相邻节点交换信息，而后期逐步开放所有临边节点交换信息更有助于分层从局部->全局学习信息，使得模型特征信息交换更加全面合理，因此加入适当的 Dropout 后有助于提升模型收敛后的任务表现性能。

3.4 标准化影响

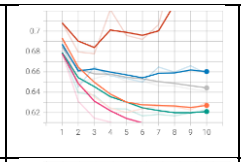
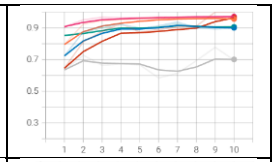
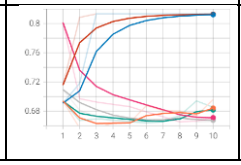
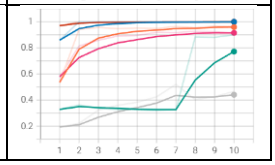
PairNorm 标准化是一种常用的用于维持节点在于相邻节点交换特征信息时维持数据分布稳定的一种手段，加入 PairNorm 后有助于节点在多次迭代交换信息后特征信息分布不变，理论上可以使得模型推理更加合理。

节点分类任务：

Config 配置	训练集损失	验证集准确率
Cora+ 有/无 标准化		
Citeseer +有无 标准化		

链路预测任务:

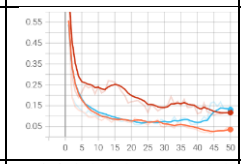
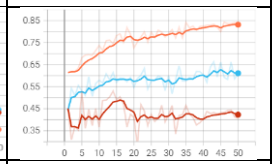
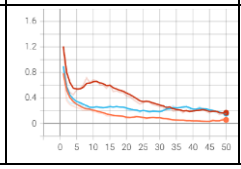
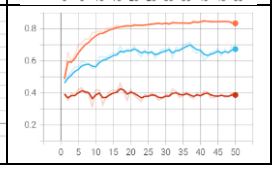
Config 配置	训练集损失	验证集准确率

Cora+ 有/无 标准化		
Citeseer +有无 标准化		

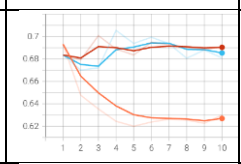
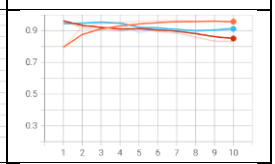
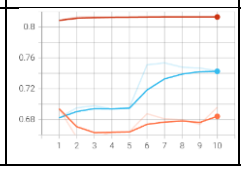
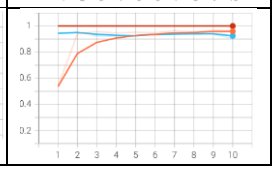
结合上图实验结果对比图，也可以发现加入 PairNorm 后的模型在验证集上具有更加的表现尤其是深度=2+PairNorm 配置标准率达到了最高。

3.5 激活函数

节点分类任务：

Config 配置	训练集损失	验证集准确率
Cora+Relu/tanh/leakyRelu		
Citeseer +有无 标准化		

链路预测任务：

Config 配置	训练集损失	验证集准确率
Cora+Relu/tanh/leakyRelu		
Citeseer +有无 标准化		

这里还进一步探讨了 Relu, tanh, leakyRelu 等不同的主流激活函数对网络的性能影响。对比上图发现 Relu 函数综合表现最佳，但是整体上不同的激活函数对网络模型性能影响并不起到决定性作用。

3.6 总结

以上各模块分别控制变量探讨研究了加入设置不同的模型超参数包括模型深度，是

否包含自环，是否进行标准化，是否以一定概率抛弃邻接边交换信息等后的网络模型基于 Cora 和 Citeseer 数据集在节点分类与链路预测两个任务的性能表现，综上所有不同的模型对比(以 Cora 数据集节点分类任务为例分析)，发现在网络深度=2+自环+边丢弃+标准化的网络模型是表现最佳的模型，可以使得训练参数充分学习并且没有过拟合在测试集达到最高准确率，而网络深度=4+无自环+无边丢弃+无标准化的网络模型表现最差测试集的准确率极低基本无推理能力。这也验证了网络模型并不是无休止的越深越好，需要经过合理的构建与设计的网络模型与充分分布合理的数据集充分迭代训练后才能在未训练的测试集上表现出最佳的性能，同时加入标准化，边丢弃，自环等是必要的，有助于在相同的数据集上训练出性能更加强大的推理模型。这也启发我们以后在未来的科研工作中要着重注意网络模型设计的合理性。

最后对以上使用的模型的准确率进行总结（每一个模型具体的训练过程请参考同文件夹下的 log 日志）：

节点分类任务(Cora)：

网络深度 num_layers	是否自环 add_self_loops	标准化 use_pairnorm	边保留 Drop_edge	激活函数 Activation	测试集准确率
2	√	×	1	Relu	84.96%
4	√	×	1	Relu	52.92%
6	√	×	1	Relu	35.28%
2	×	×	1	Relu	38.72%
2	√	√	1	Relu	89.88%
4	√	√	1	Relu	87.74%
6	√	√	1	Relu	85.24%
2	√	×	0.6	Relu	86.54%
4	√	×	0.6	Relu	48.10%
2	√	×	1	tanh	53.11%
2	√	×	1	leakyRelu	66.11%

节点分类任务(Citeseer)：

网络深度 num_layers	是否自环 add_self_loops	标准化 use_pairnorm	边保留 Drop_edge	激活函数 Activation	测试集准确率
2	√	×	1	Relu	84.57%
4	√	×	1	Relu	53.71%
6	√	×	1	Relu	33.73%
2	×	×	1	Relu	25.89%
2	√	√	1	Relu	84.81%
4	√	√	1	Relu	78.02%
6	√	√	1	Relu	74.95%
2	√	×	0.6	Relu	83.35%
4	√	×	0.6	Relu	44.03%
2	√	×	1	tanh	48.71%
2	√	×	1	leakyRelu	73.98%

Tips:链路预测任务输出请查看 output 下日志，同时 ppi 数据集由于暂未成功跑通忽略展示。

4. Requirements.txt

```
absl-py==2.0.0
cachetools==5.3.1
certifi==2023.7.22
charset-normalizer==3.3.0
colorama==0.4.6
contourpy==1.1.1
cycler==0.12.1
fonttools==4.43.1
importlib-metadata==6.8.0
importlib-resources==6.1.0
kiwisolver==1.4.5
Markdown==3.5
MarkupSafe==2.1.3
matplotlib==3.8.0
netron==7.2.5
numpy==1.26.1
oauthlib==3.2.2
packaging==23.2
Pillow==10.1.0
protobuf==4.23.4
pyasn1==0.5.0
pyasn1-modules==0.3.0
pycodestyle==2.11.1
pyparsing==3.1.1
python-dateutil==2.8.2
requests==2.31.0
requests-oauthlib==1.3.1
rsa==4.9
six==1.16.0
tensorboard==2.15.0
tensorboard-data-server==0.7.1
tomli==2.0.1
tqdm==4.66.1
typing_extensions==4.8.0
urllib3==2.0.7
Werkzeug==3.0.0
zipp==3.17.0
torch_geometric==2.4.0
torch-sparse==0.6.18+pt20cu118
```

作者本次实验均在单张 RTX 3090 运行完成(CU118)

提示：在本次安装 torch_geometric 库后会报错确实 pyg-lib 或者 torch-sparse,这需要在如下网站参考 cuda 版本选择对应版本 whl 文件手动安装。