**Policy Gnn 代码检查**

**目录**

**一****、复现结论**

**二、代码检查**

**三、节点分类任务排名**

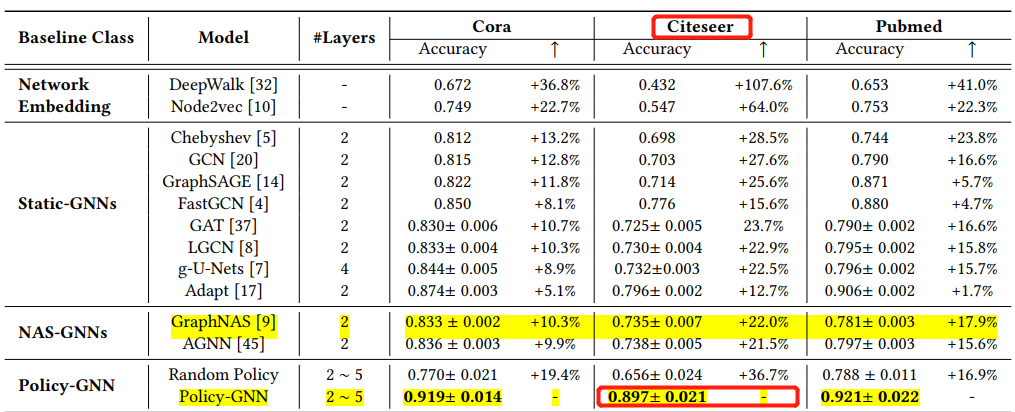
**四、计划**

**五、笔记**

**一、复现结论**

**结论：无法使用policy gnn源代码得到与原文相同的结果，结果与上周汇报一致，针对citeseer数据效果只能到70%，并且效果方差极大**





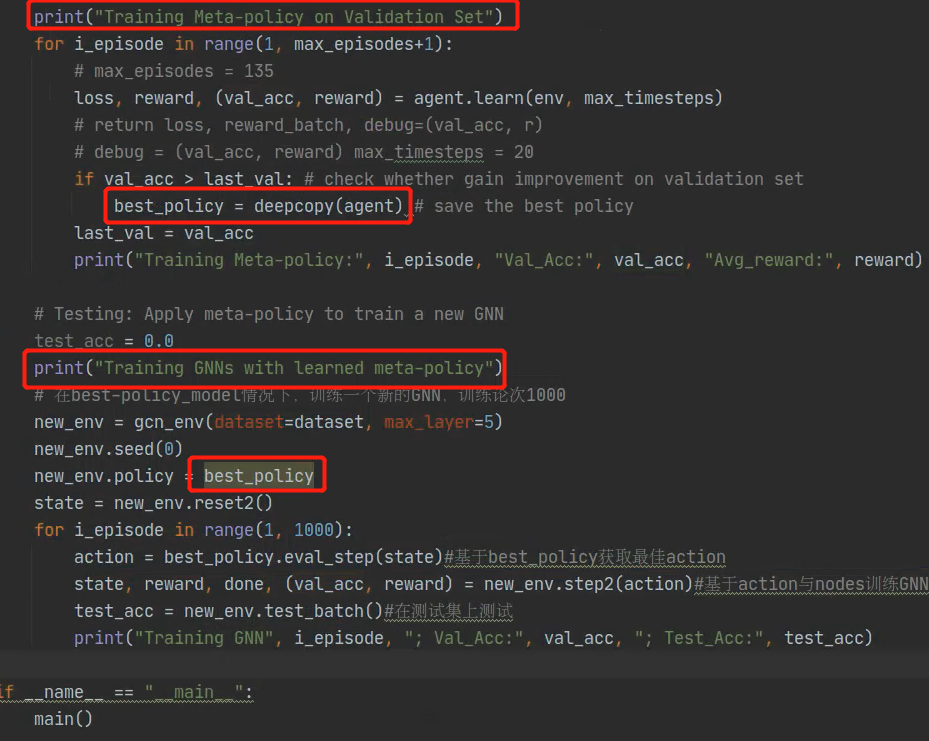
**二、代码检查**

**检查项：**

* **是否有开关控制 无**
* **超参配置是否与论文一致 一致**
* **reward function设置是否与论文一致？一致**
* **代码执行逻辑是否与论文理论一致？一致**

**1.是否有开关控制？**

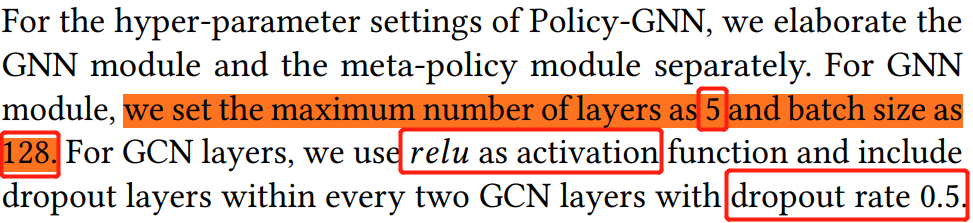
**否，使用训练好的best\_policy选择GNN邻居节点聚合的阶阶数**

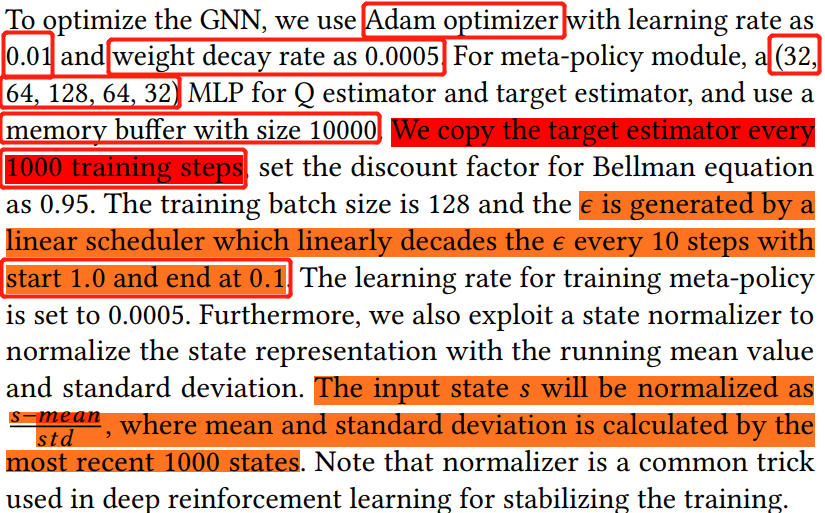


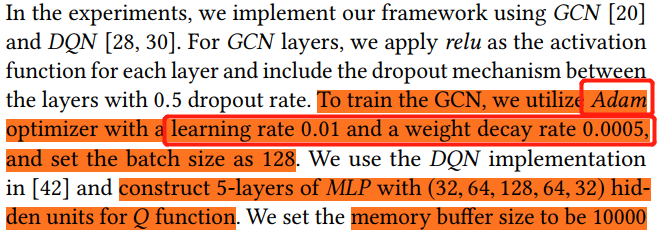
**2.超参设置是否与论文一致？**

**一致**

**论文给出超参：**





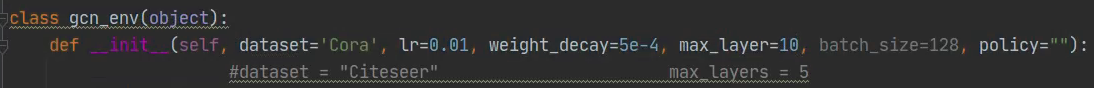


**论文给出GNN设置：**

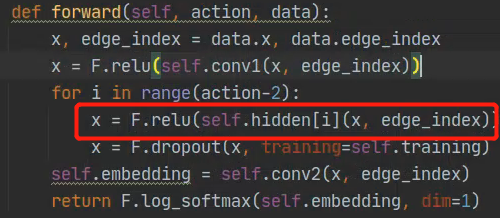
(1).最大层数为5层



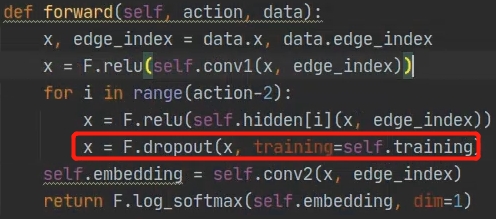
(2).训练batch\_size为128

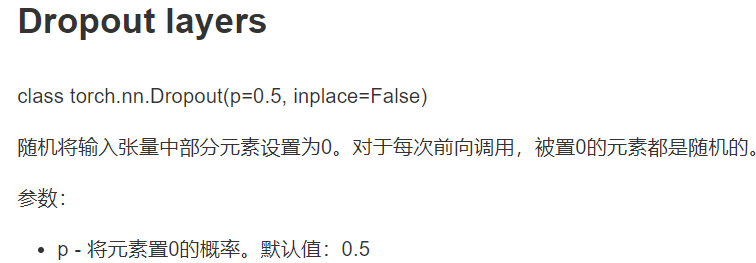


(3).激活函数relu



(4).dropout rate = 0.5

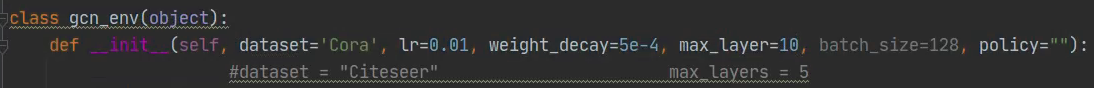




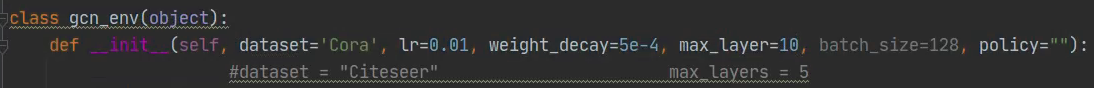
(5).优化算法Adam



(6).学习率：0.01



(7).权重衰减率0.0005



**论文给出policy-model设置：**

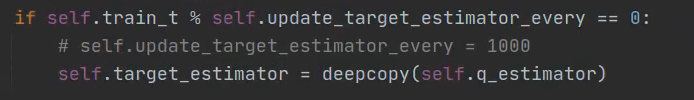
(1).policy-model是一个结构为[32,64,128,64,32]mlp



(2).memory size = 10000



(3).target\_estimator 每训练1000次policy model,更新一次参数



(4).epsilon探索率范围从[1,0.1]



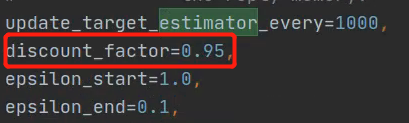
(5).policy model 训练batch\_size = 128



(6). 优化算法Adam

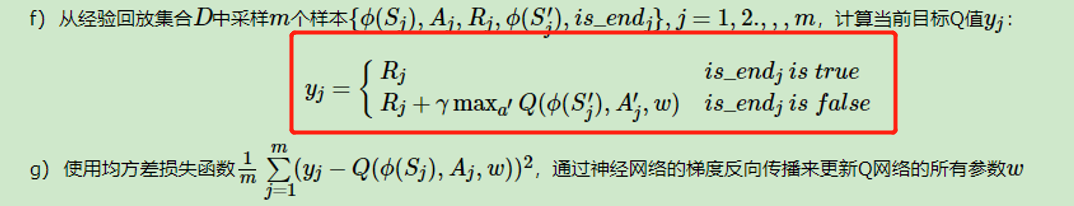


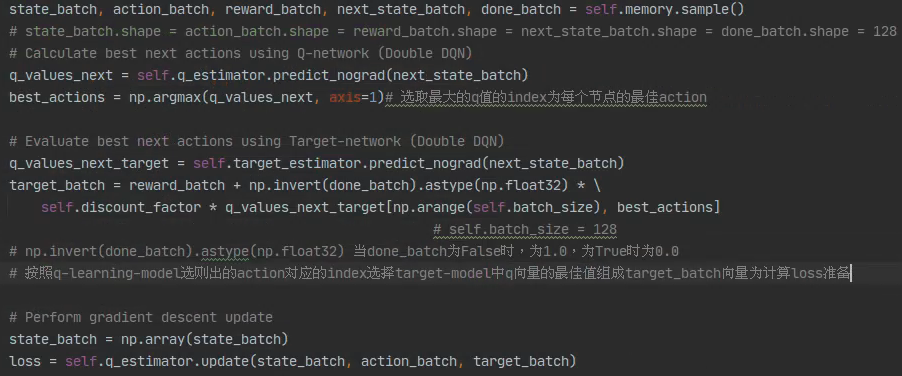
(7).折扣因子0.95



**3.reward function设置是否与论文一致？**

**一致**







**4.代码执行逻辑是否与论文理论一致？**

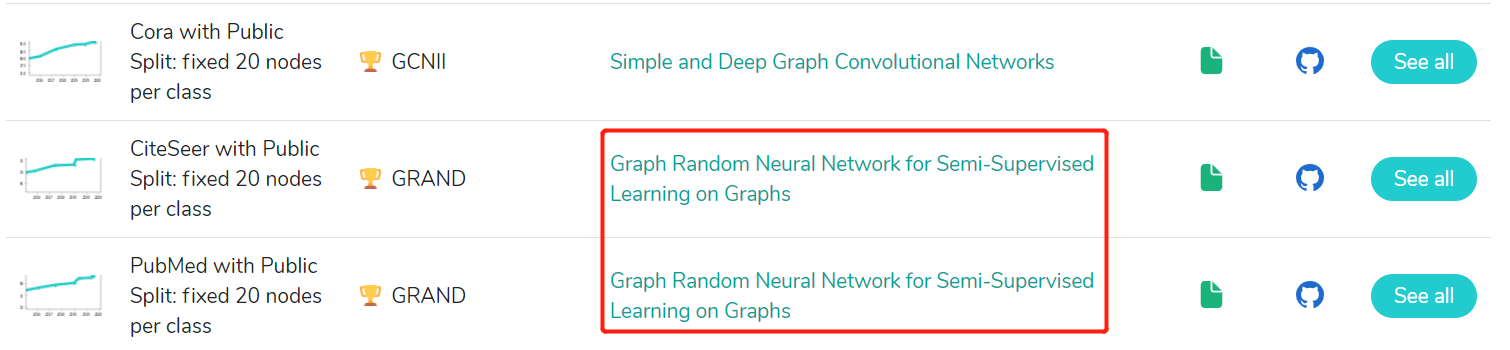
一致

**结论：大概率基于作者公开的代码无法复现论文的结果**

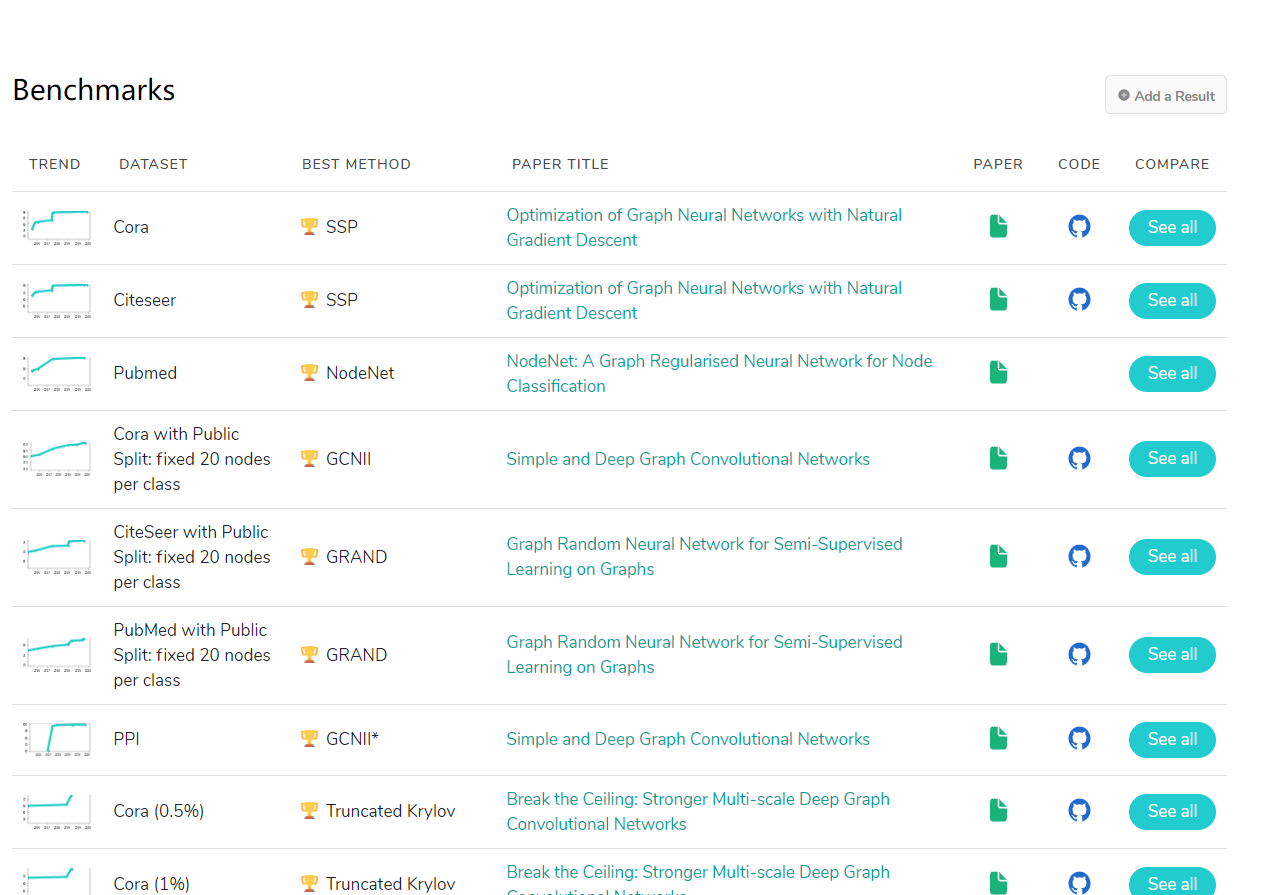


**三、节点分类任务排名**

**半监督任务排名（paperswithcode）**

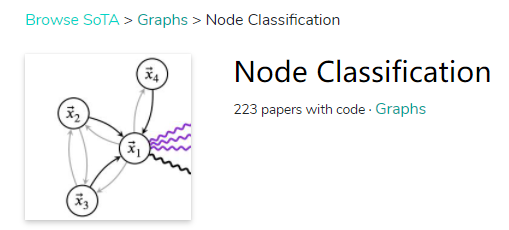


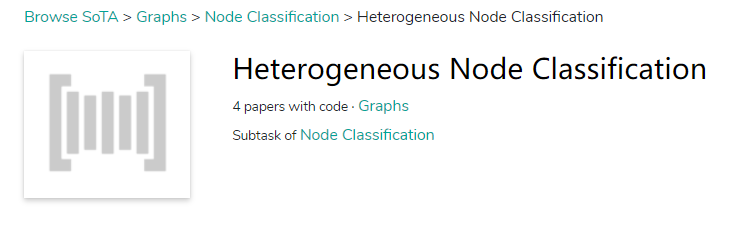
1.先用grand确定order然后使用graphnas确定结构

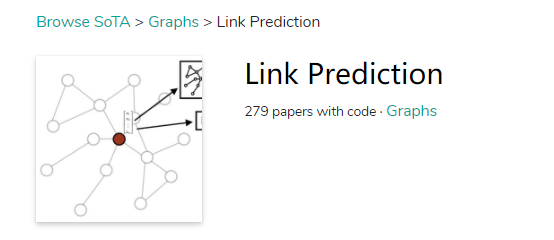


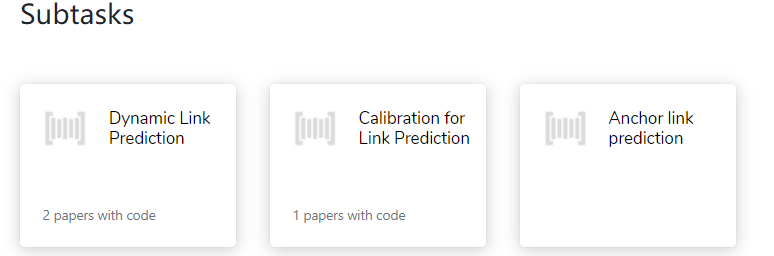
**四、计划**

**1.基于paperswithcode网站梳理出node classification/link prediction以及子任务中每个任务数据集、具体的任务设置——>（莫卡德）**









**目标：探索可以使用以graphnas为基础进行改进后可以应用的新任务，新数据集中**

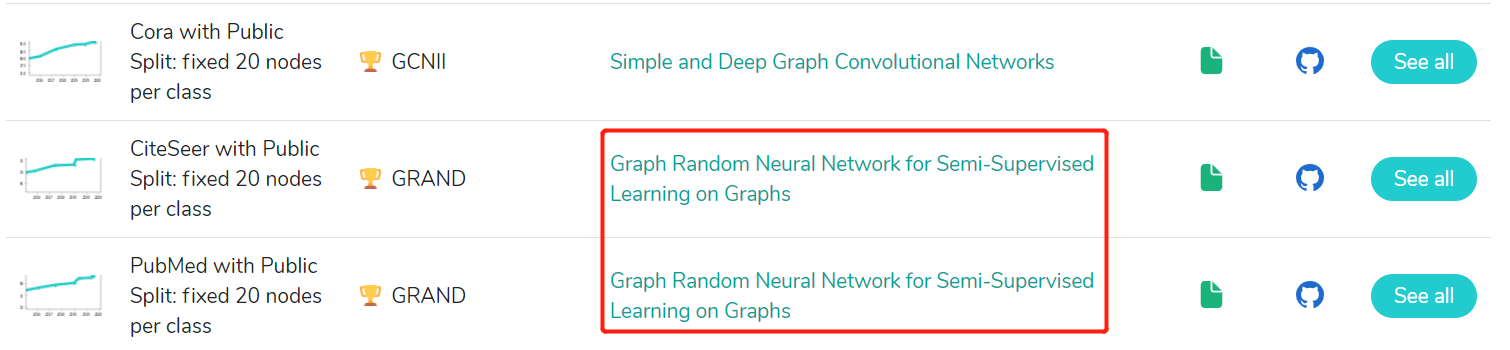
**2.读有关节点聚合方式论文，探索以graphnas结构搜索为基础结合最新节点聚合方式可能性?——>（陈家民）**

**难点1：是否能将两者通过一个损失函数，或多个损失函数联合的方式结合起来？**

**如果可以结合，使用节点分类任务验证初步效果。**

例如： [**1.Graph Random Neural Network for Semi-Supervised Learning on Graphs**](https://paperswithcode.com/sota/node-classification-on-pubmed-with-public)

**2.**[**Simple and Deep Graph Convolutional Networks**](https://paperswithcode.com/sota/node-classification-on-cora-with-public-split)



**目标：构建我们自己的图卷积神经模型搜索方式**

**五、笔记**

1.代码目录

train\_citeseer.py

def main()

dqn\_agent\_pytorch.py

class Normalizer

def normalize

def append

class Memory

def save

def sample

class DQNAgent

def learn：q\_estimator训练控制逻辑

def feed

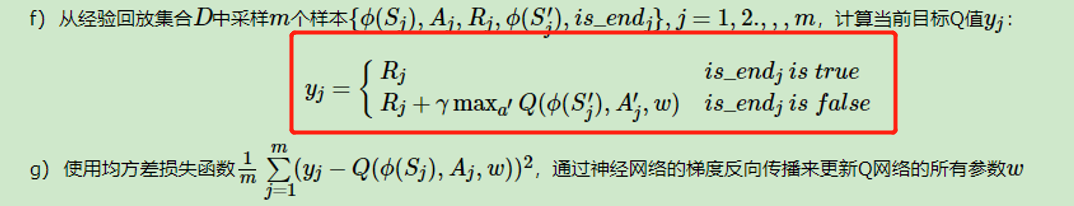
def step

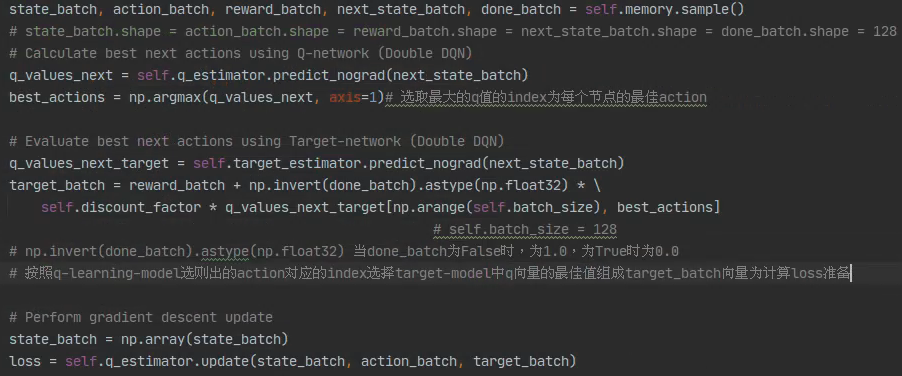
def eval\_step：使用q-learning-model预测验证集节点最佳action（通过选择最大q值确定），

def predict

def predict\_batch：基于q-learning-model获取每个节点最佳动作A向量

def train: 基于经验集训练q-learning-model







def feed\_norm

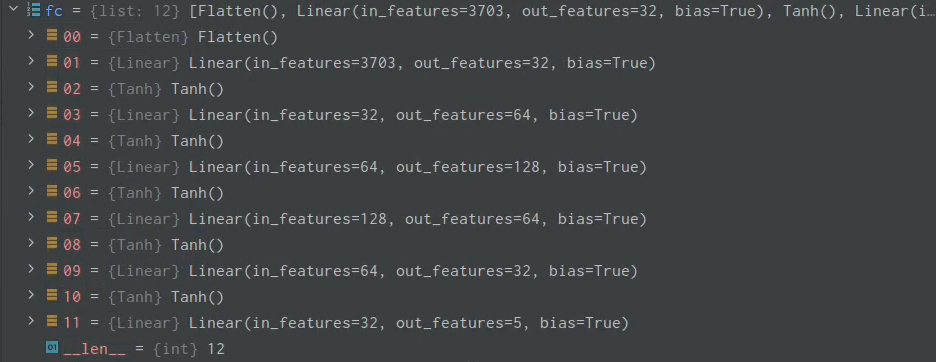
def feed\_memory

class Estimator:构建5层q-learning-model的mlp

def predict\_nograd：预测输入node\_batch的最佳action向量

def upate：基于标签y跟新q-learning-model参数

class EstimatorNetwork:构建 q-learning-model网络5层mlp



def forward：构建q-learning-model前向计算过程

gcn.py

class Net：构建GCN基本模块

def forward：构建前向传播过程

class gcn\_env

def seed

def init\_k\_hop: 获取1次到5次幂的邻接矩阵A

def reset

def \_set\_ation\_space

def \_set\_observation\_space

def step

def reset2：获取1号节点到119号节点index与节点state

def step2：

基于已选择的节点state、actions确定:

1.当满足训练触发条件时，选择对应的action与节点训练GNN model;

2.基于目前node\_batches的states获取下一个node\_batches的states、indexes;

3.计算此时在此参数下GNN网络在500个节点的验证集上的表现；

4.基于val\_acc与baseline计算rewards；

5.返回下一个node\_batch的states, rewards, done\_batch, (val\_acc\_mean ,reward\_mean)；

def stochastic\_k\_hop：基于概率与action挑选下一个节点

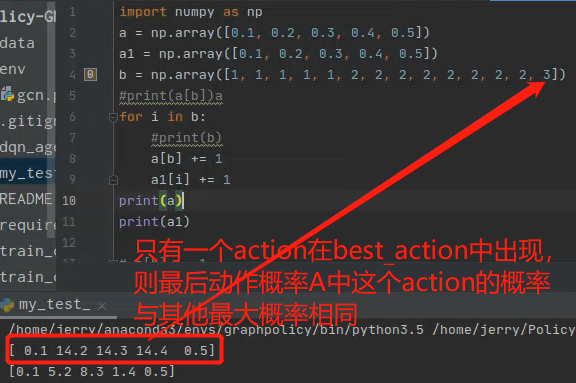
def train

def eval\_batch

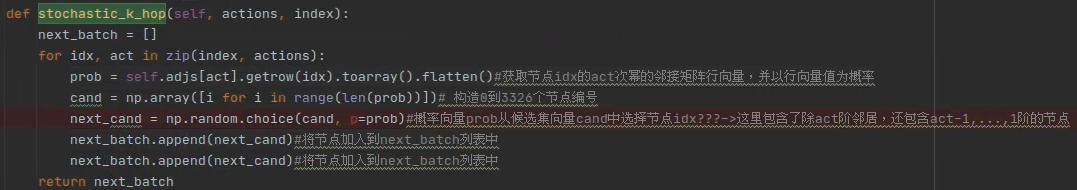
def test\_batch：使用已经训练好的q-learning-model确定测试集上的最佳action，并基于与action使用GNN对测试集进行测试，得到测试acc的均值

def check

**源码疑点：**

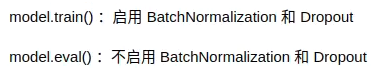


**下一个状态的节点选取存在问题**

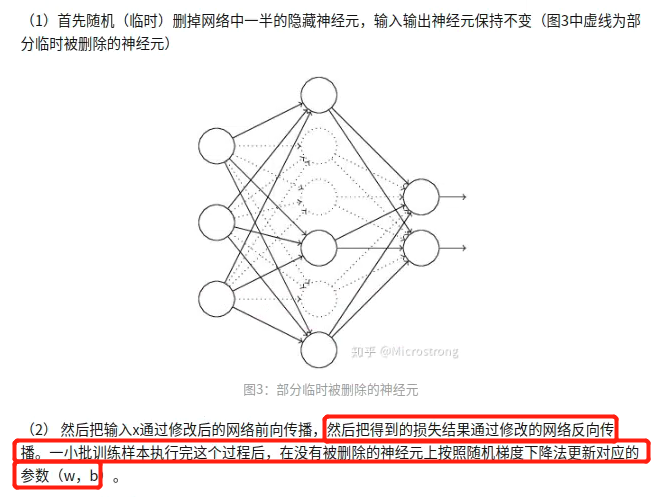


**model.train()/model.eval()解释：**

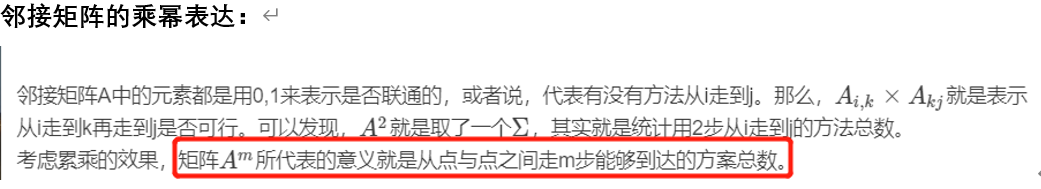


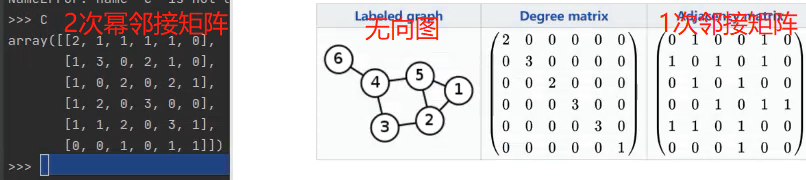


**dropout解释：**



**无向图的邻接矩阵A的幂含义**





**无向图邻接矩阵A的k次幂：一定能表达节点i与哪些节点j存在k阶邻居关系，不一定能表达节点i与节点j存在k-1,…,1阶邻居关系**