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

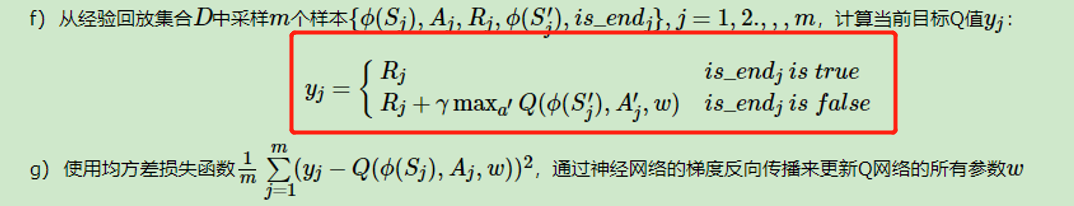
def step

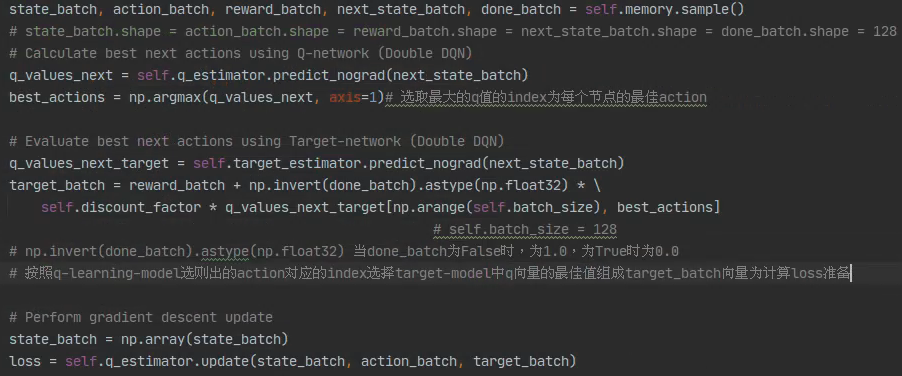
def eval\_step：使用q-learning-model预测验证集节点最佳action（通过选择最大q值确定），

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

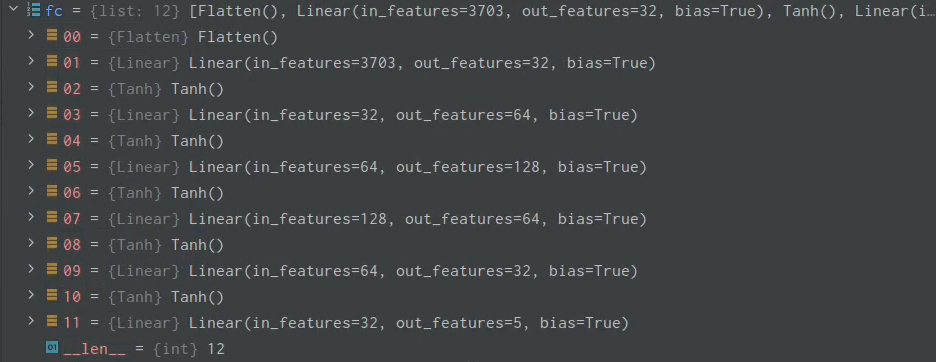
def feed\_memory

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

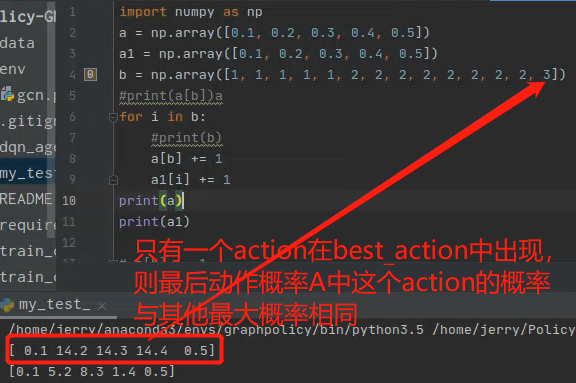
def train

def eval\_batch

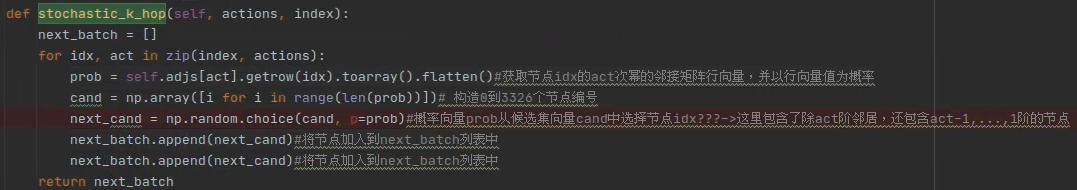
def test\_batch：使用已经训练好的q-learning-model确定测试集上的最佳action，并基于与action使用GNN对测试集进行测试，得到测试acc的均值

def check

**源码疑点：**

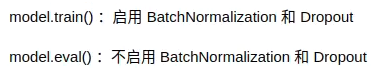


**下一个状态的节点选取存在问题**

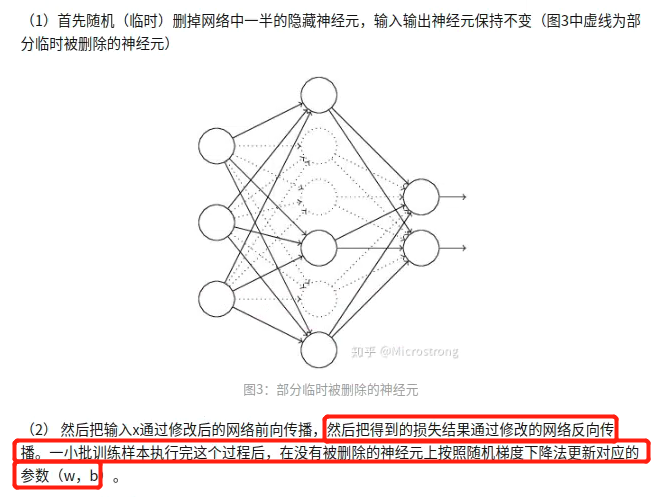


**model.train()/model.eval()解释：**

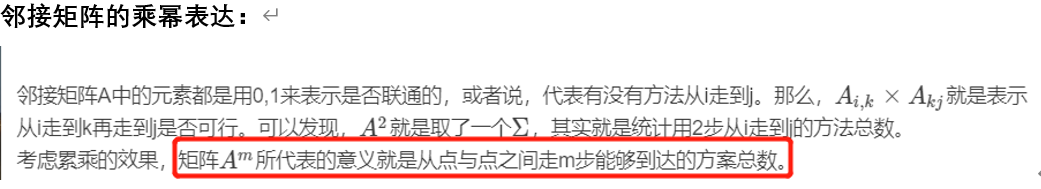


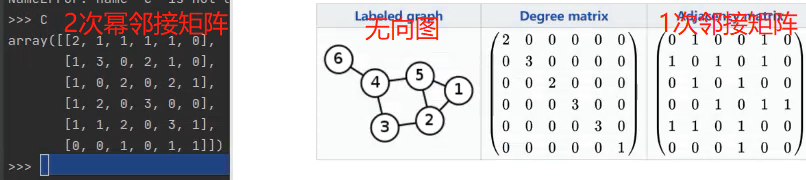


**dropout解释：**



**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**