**中山大学计算机学院**

**人工智能**

**本科生实验报告**

**（2022学年春季学期）**

课程名称：Artificial Intelligence

|  |  |  |  |
| --- | --- | --- | --- |
| 教学班级 | **计科2班** | 专业（方向） | **计算机科学与技术** |
| 学号 | **20337263** | 姓名 | **俞泽斌** |

# 实验题目

以Cart Pole为环境， 实现DQN和PG算法， 要求进

行可视化(reward, loss, entropy 等）

# 实验内容

1. 算法原理

DQN：

Q-learning算法采用一个Q-tabel来记录每个状态下的动作值并进行比较来算出最佳策略，但在当状态空间或动作空间较大时，需要的存储空间也会较大DQN是采取用一个人工神经网络来进行深度强化学习的算法，主要的思想是每一次都判断一下所有下一个状态和动作的价值，然后根据值的大小来选择动作策略。同时为了保证训练的收敛性及稳定性，一般都采用一个经验池来保存已经做完的动作并学习更新人工神经网络，来得到更佳的结果。

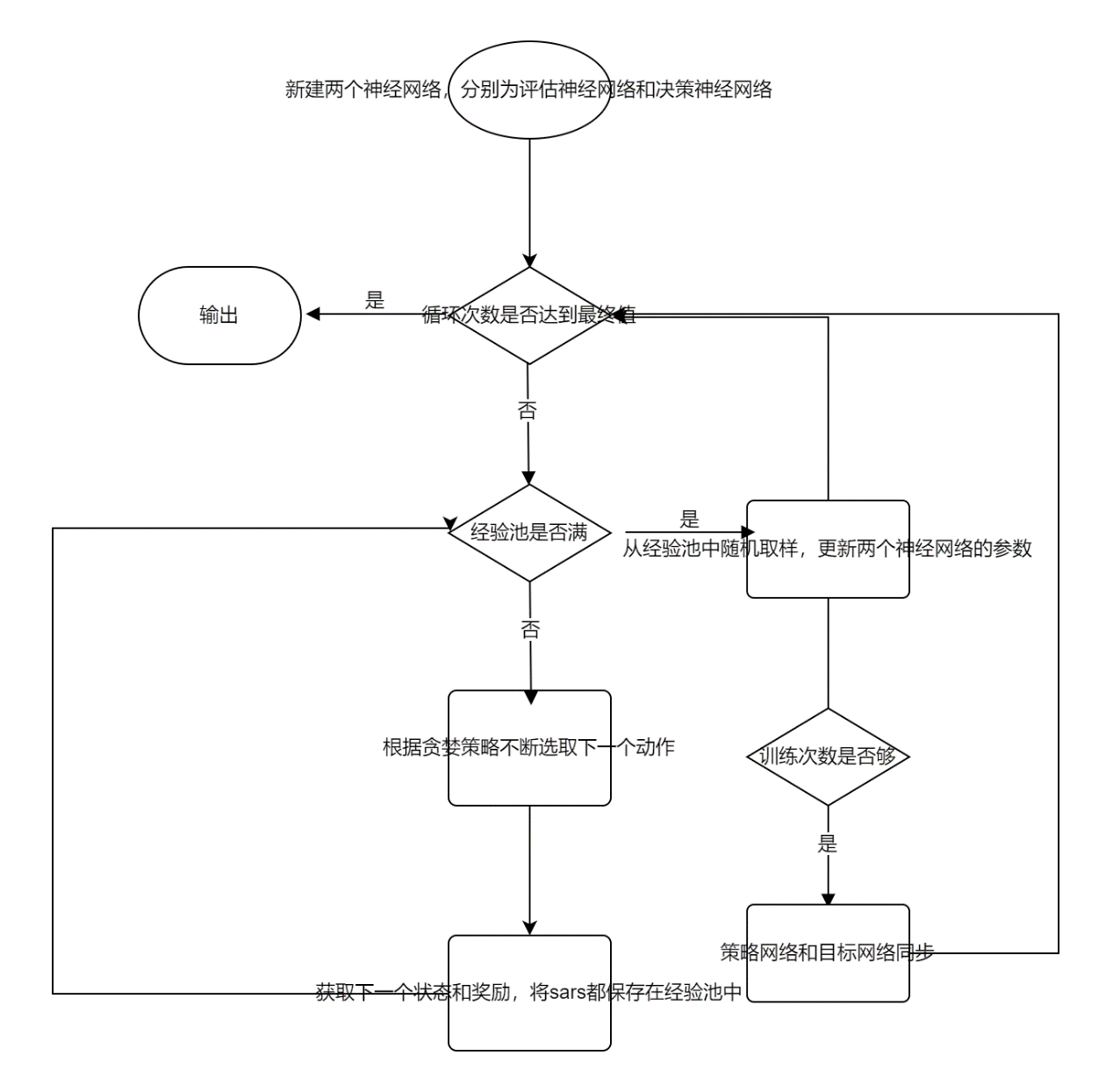
我这里使用了两个神经网络，其中一个是用来估计的，另一个是实际的决策操作，先动作和状态通过估计的神经网络来得到策略，有一个update\_target参数来判断是否到了取样的结束阶段，如果是就将估计神经网络的参数赋值给决策神经网络，进行下一步的策略是两种，有概率性的随机进行或者是通过评估的神经网络来得到奖励最大的那一个动作，然后每一次进行下一步的时候都会计算并保存损失，差距等参数，然后保存其中的奖励，等到奖励值到一定限度，也就是经验池满了的时候，就在经验池中采样来更新两个神经网络。

PG:

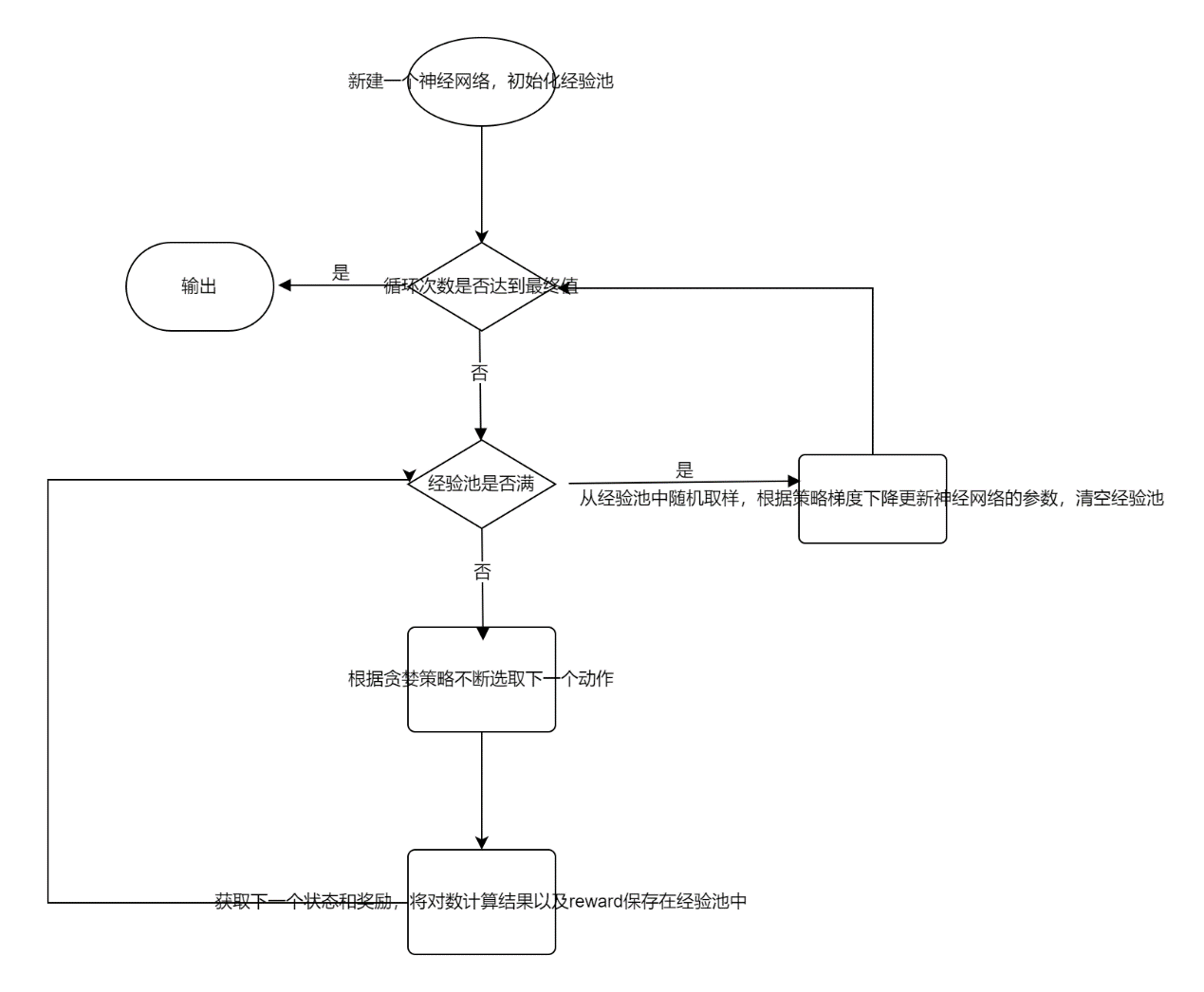
PG算法直接计算每个状态对应的动作或者动作的概率，使得它可以在一个连续区间内选取动作，他对策略函数进行建模，然后用梯度下降更新网络的参数。因为所需要的是最大化累计奖励的期望值，所以可以选择一个loss函数来设成-函数的奖励值与动作策略概率的乘积的对数值，可以方便相关运算，同时在pytorch中也有相对应的库函数来调用，如果一个动作的奖励值较大，则下次选取该动作的可能性增加的幅度也大，反之选取该动作的可能性增加的幅度小。

1. 伪代码

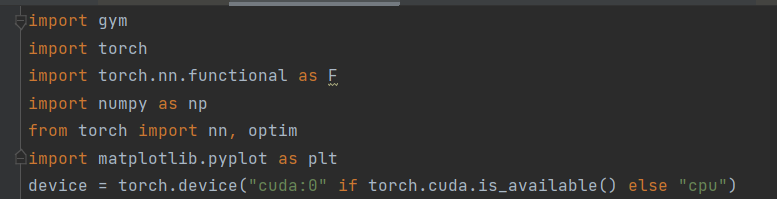
DQN



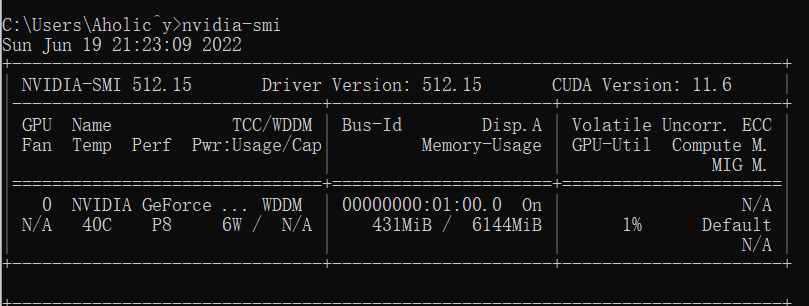
PG：



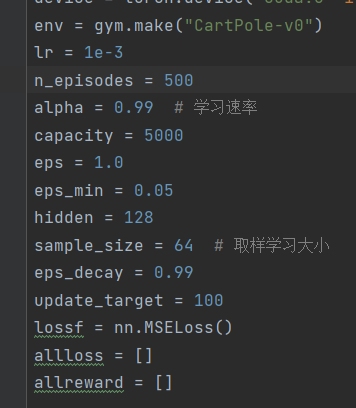
1. 关键代码展示（带注释）



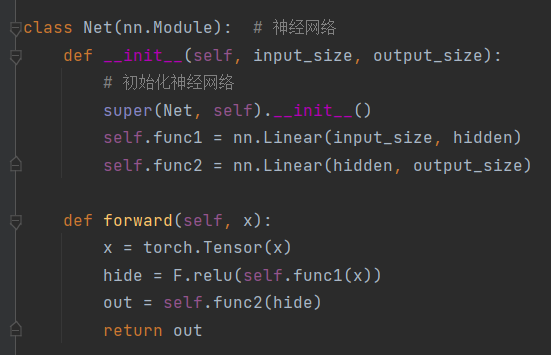
首先是一些库函数的引入以及为了提高矩阵计算的速度而使用GPU来进行计算，这里使用的是cuda，具体教程都是参考网上的代码和实现，安装完cuda的包后可以在命令行中输入nvidia-sml得到对应的cuda版本号来代表安装成功



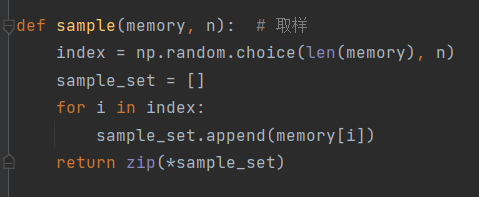
这样就可以使用cuda来调用pytorch从而使用gpu进行有关的计算和操作，但后来发现gpu计算的加速一般都在于参数比较大的情况，而本次实验中参数较为小的时候也可以得到基本收敛的结果，所以还是大部分使用cpu来进行计算



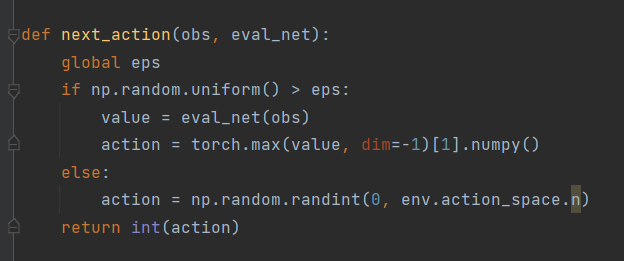
一些基本的参数，学习率，采样次数，采样的大小，随机选择进行下一步的概率等等。



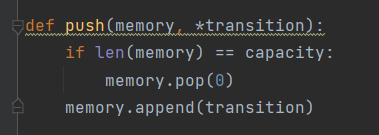
基本的神经网络的类，这也是本次实验中唯一封装的一个类，因为在DQN算法当中，我们所需要的是两个人工神经网络，一个是估计的神经网络，另一个是实际做决策的神经网络。



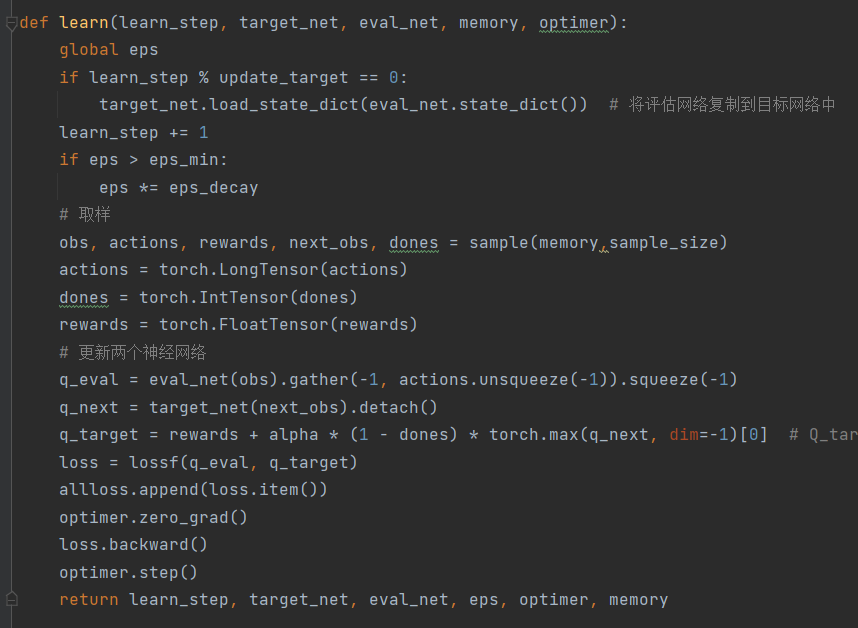
取样函数，因为DQN中关于估计神经网络的更新是通过从经验池中取样然后学习的过程来实现的，所以需要一个随机的选取memory其中的几个参数的函数，并且将他们封装好发出来。



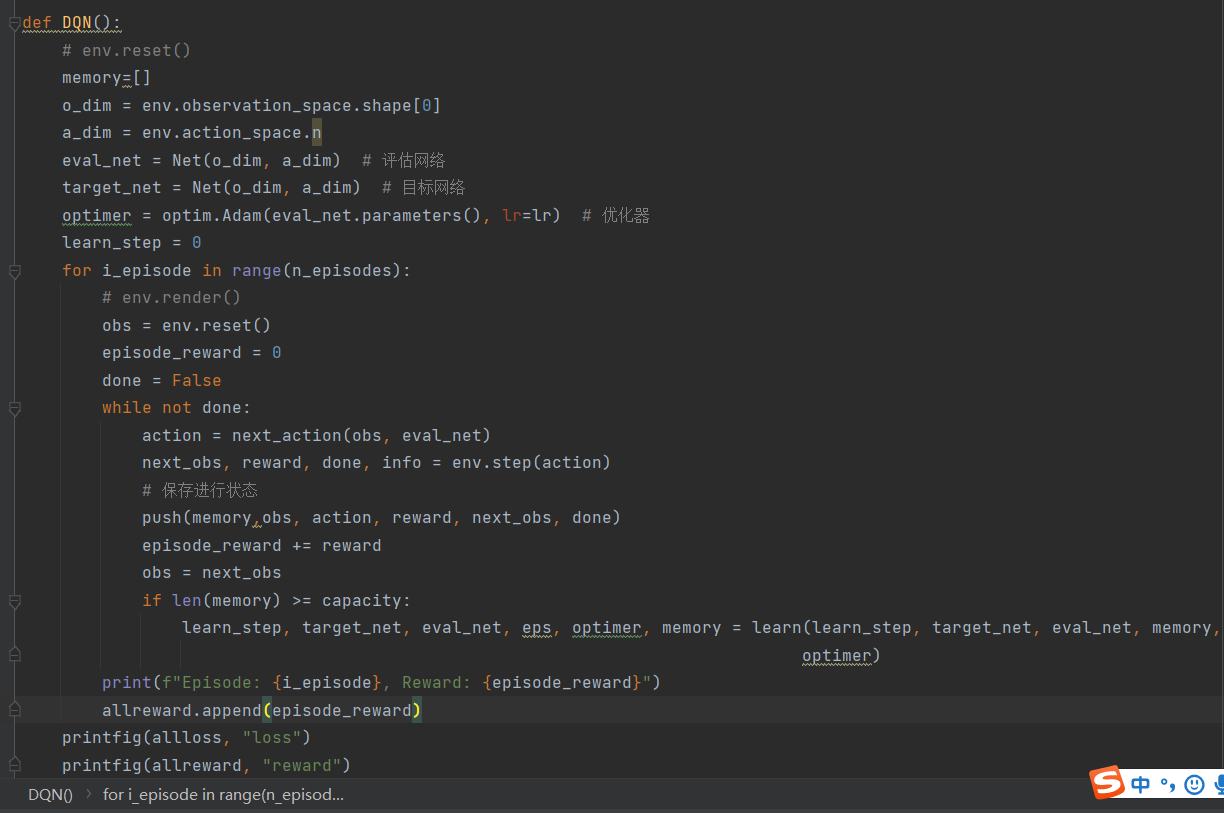
进行下一步的函数，主要操作是通过一个eps概率，有概率性的随机操作，或者是通过评估神经网络来获得一个奖励值最大的动作来进行，在main函数中会保存这个动作对应的一些信息到经验池中。



这是往经验池中添加元素的操作，因为所添加的东西是很多信息的一个集合，所以不能简单的使用一个append函数，然后需要通过传递地址来，同时使用一个capacity来判断经验池是否溢出，如果满了就排除最早的那个经验，因为最不重要。



这是DQN算法里的核心内容了，也就是关于两个网络的更新方面的，前面提到，关于下一步动作，我们一般大概率都是采用的估计网络来输出，当经验池中的步骤和奖励足够多的时候，我们就要开始学习了，从经验池中取样，然后取出里面奖励最大的那个值来更新两个网络的参数，同时如果学习的次数到达了update\_target，就将评估网络复制到目标网络中。



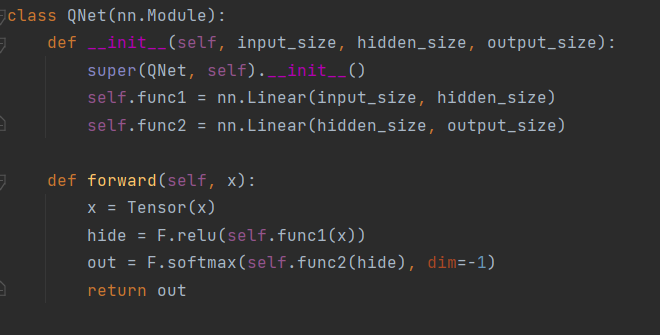
剩下的方面就是DQN函数的处理，其实大部分也就是拼接操作了，首先初始化两个网络为目标和评估网络，然后调用pytorch库的那个的optim的优化器，在规定的循环次数内，不断进行下一步的操作，并保存相关的经验参数，如果经验的数量超过了经验池的容量，就通过学习来更新两个网络。

最后输出两个图表来展示实验过程中的reward和loss

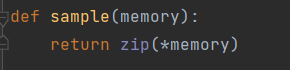
PG:



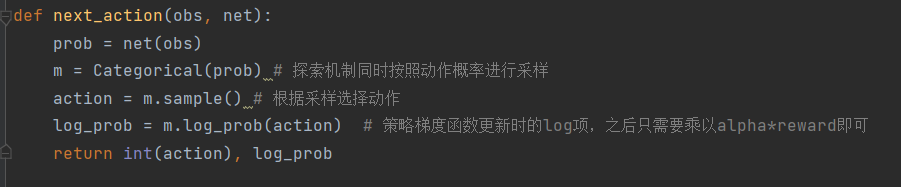
首先依旧是一些基本的库函数的引入以及基本参数的确定。



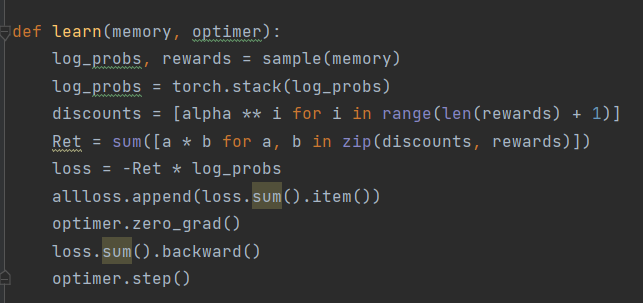
这是PG算法中的神经网络的确定，与DQN算法中的神经网络不同的是，这里的隐层到输出层中的操作用的是softmax函数，其优点主要是预测的概率为非负数；并且各种预测结果概率之和等于1，方便之后的策略梯度下降。



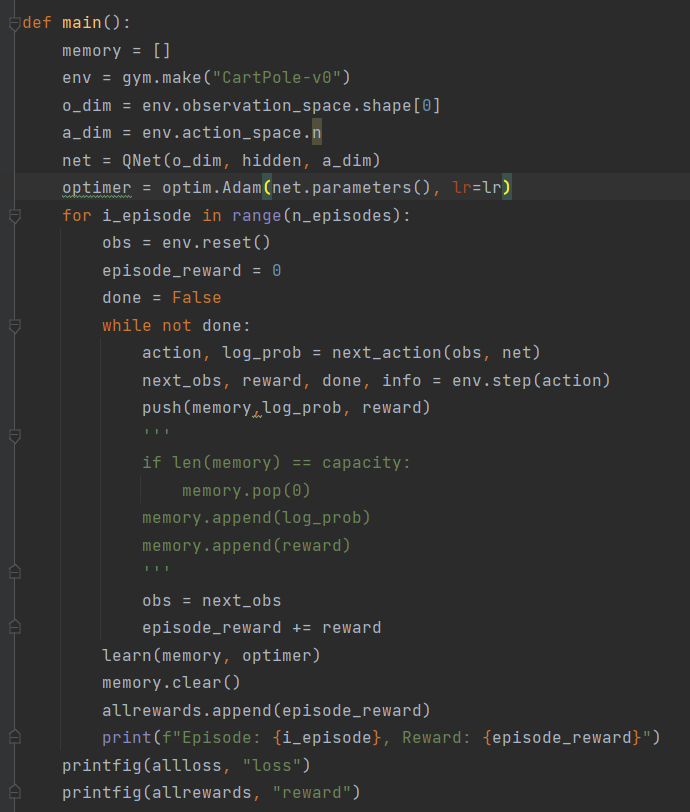
这里的取样只需要取其中一个，就写一个简单的取样函数。



下一步的操作，我们这里所需要保存的参数有两个，一个是reward，另一个是我们策略梯度下降的时候有一个对数项（-函数的奖励值与动作策略概率的乘积的对数值）要每次先把他计算出来保存，方便更新的时候可以直接取出来然后进行计算。

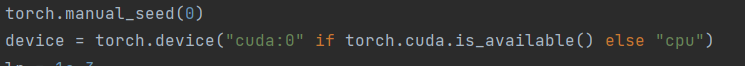


学习函数，主要也是在经验池满的时候开始进行操作，从我们上述的下一步函数可以看出，我们每进行下一步，就会计算一次reward和对数值并保存到经验池中，现在我们就取样并进行学习，主要操作是计算每一次的折扣函数和reward的乘积从而得到对数值，然后计算出loss，并且通过loss来返回修改神经网络中的几个参数。



这是主体的代码，主要的操作也就是和DQN一样的拼接了，首先初始化神经网络，然后调用pytorch库的那个的optim的优化器，在规定的循环次数内，不断进行下一步的操作，并保存相关的reward和对数参数，如果经验的数量超过了经验池的容量，就通过学习来计算策略梯度并更新网络。

1. 创新点&优化（如果有）

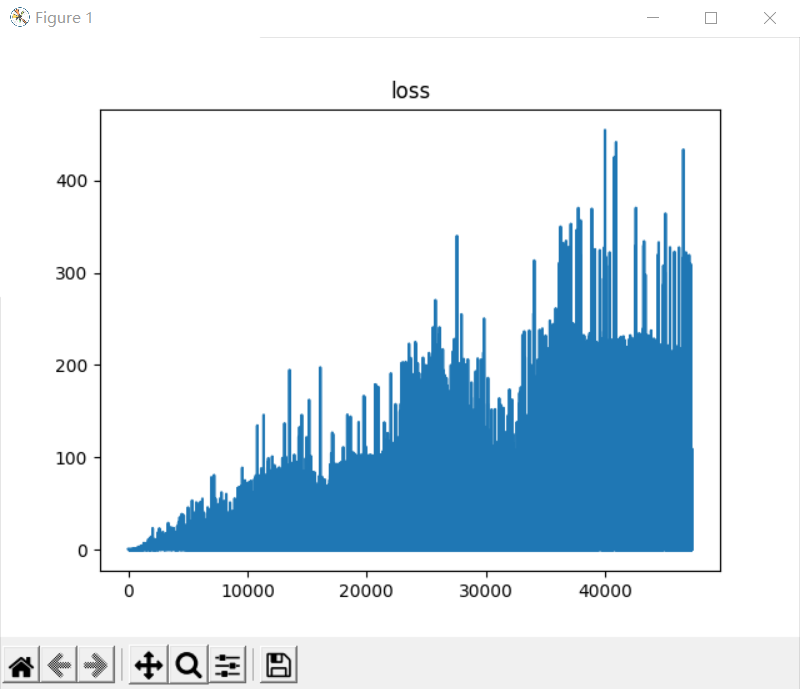


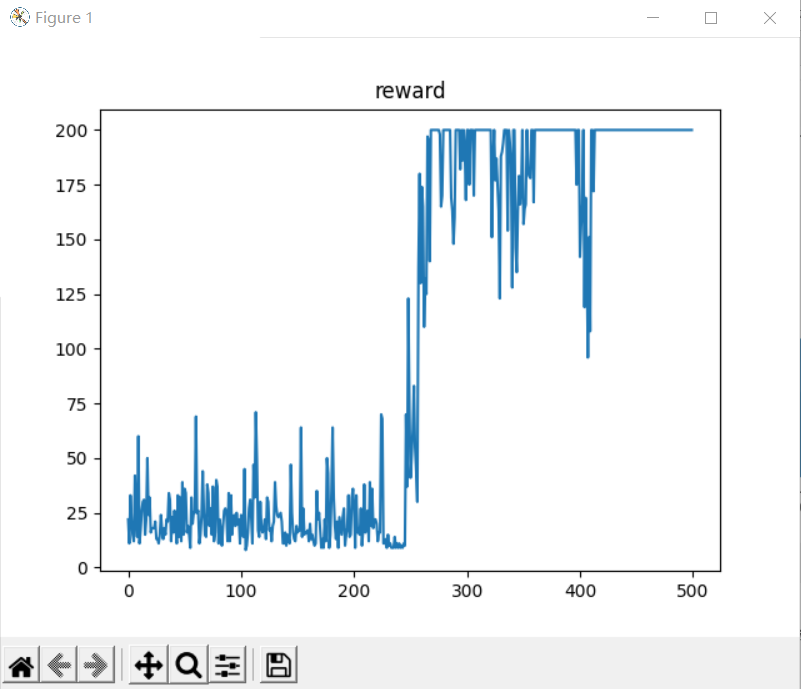
尝试采用了cuda来调用pytorch从而使用gpu进行有关的计算和操作，但后来发现gpu计算的加速一般都在于参数比较大的情况，而本次实验中参数较为小的时候也可以得到基本收敛的结果，所以还是大部分使用cpu来进行计算

# 实验结果及分析

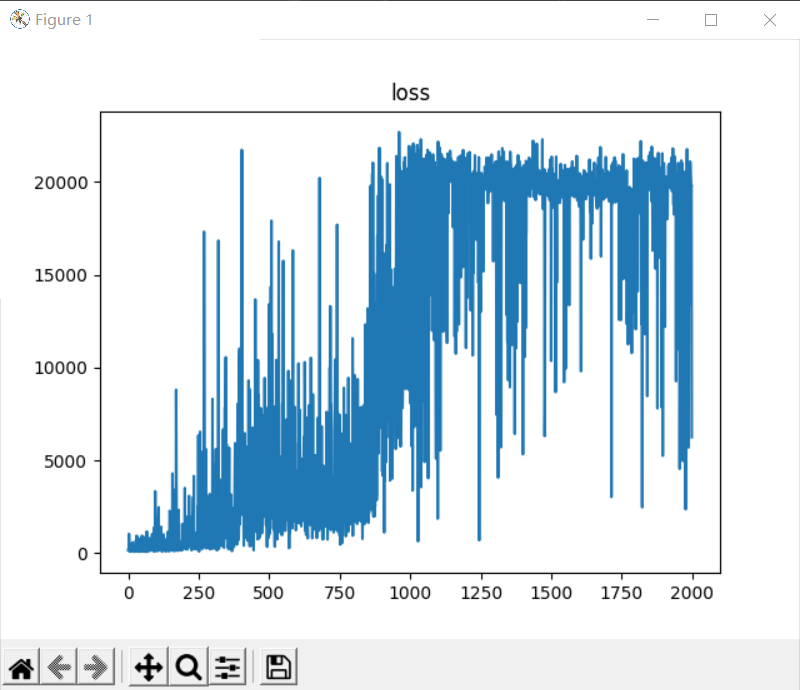
1. 实验结果展示示例（可图可表可文字，尽量可视化）

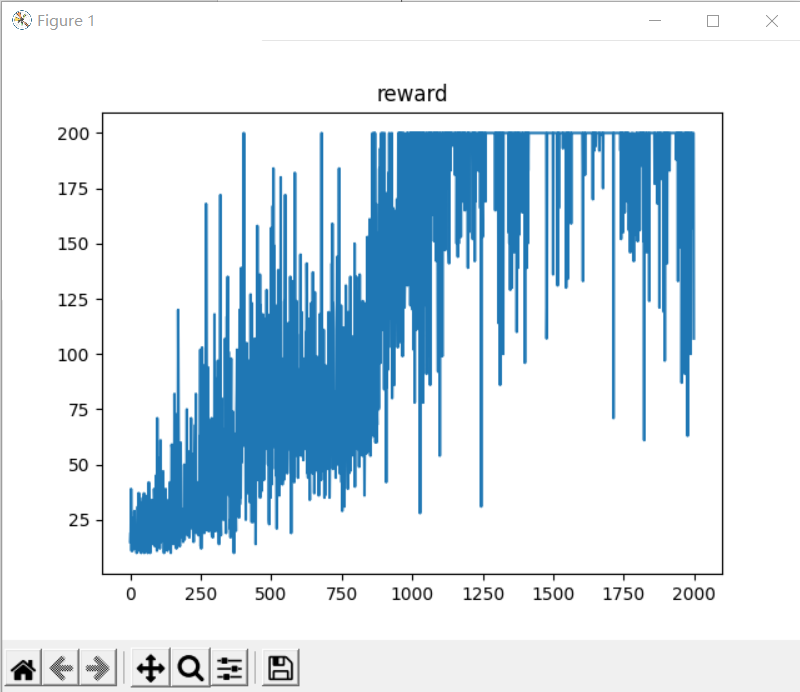
DQN





PG





2. 评测指标展示及分析（机器学习实验必须有此项，其它可分析运行时间等）

可以看到，本次实验中DQN算法比起PG算法来说，DQN算法更快收敛，并且收敛后的波动较小，应该是因为更新的是两个网络中的参数，同时取样的时候是取了很多的经验池样本来学习，比起PG算法的一个网络以及学习时值抽取少量的样本，DQN算法更容易收敛。但同时是因为上述的两个原因，DQN算法所需要取样的以及计算的步骤就特别多，导致DQN算法的速度可以说是慢了许多，特别是运行到后面更甚，PG算法计算操作少，运行速度更快。

DQN：循环次数500次



PG：循环次数2000次



**|-----------如有优化，请重复1，2，分析优化后的算法结果-----------------------|**

# 参考资料

**https://blog.csdn.net/lz\_peter/article/details/84574716?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%2522165568325216781435465380%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request\_id=165568325216781435465380&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~top\_positive~default-2-84574716-null-null.142^v17^control,157^v15^new\_3&utm\_term=softmax&spm=1018.2226.3001.4187**

**https://blog.csdn.net/zuzhiang/article/details/103180919?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%2522165564461516781483720888%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request\_id=165564461516781483720888&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~baidu\_landing\_v2~default-2-103180919-null-null.142^v17^control,157^v15^new\_3&utm\_term=pg%E7%AE%97%E6%B3%95%E5%8E%9F%E7%90%86&spm=1018.2226.3001.4187**

**https://github.com/joenghl/BaseRL/tree/da2b68e4bc30d18c0791c5c71a33a083ff5e3b1f**

PS：可以自己设计报告模板，但是内容必须包括上述的几个部分，不需要写实验感想