### 第三次实验报告

**实现目的：**基于深度学习完成游戏

**实验思路**：该次实验要求使用的深度学习的方法使得游戏更加智能。在实验二中曾使用分类器来实现游戏，在深度学习中，我同样使用神经网络搭建了一个分类器来尝试解决问题。相对于机械学习的数据要求，搭建神经网络训练时使用的数据更加麻烦一点，因此对之前收集的数据重新进行了编排。此外除了使用神经网络的分类器还使用了策略梯度和q-learning的算法去实现该游戏。

**实验准备**：对于神经网络搭建的分类器需要数据的准备，而对于策略梯度和深度学习的算法则需要更多的时间来训练模型得到一个较好的模型。数据格式；

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| -0.020597992 | 0.010312307 | -0.030507967 | 0.032676929 | 1 | 0 |

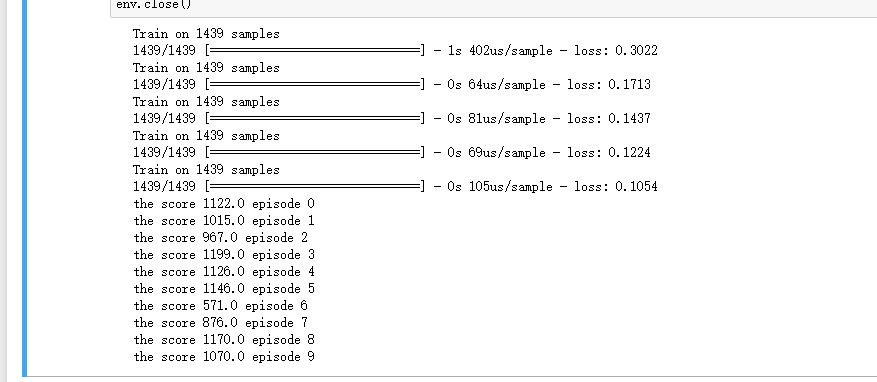
前四位是每次返回的参数，而之后的1，0作为一个二维数组来决定移动方向，因为模型预测出的结果是也是一个二维数组表示两个方向出现的概率，因此[1,0]表示向左，[0,1]表示向右。对预测结果选择概率大的作为方向的选择。

实验上交内容：数据集：l.csv（训练数据集，根据上次实验用的得分大于1000的数据集改版得到），三个python代码，分别实验分类器（classier.py），策略梯度(policy.py)和q-learning(q-learning.py)。一份实验报告。

**实验结果以及代码说明**

1.分类器实现：

搭建一个 4 x 64 x 20 x 2 的网络，输入层为4，输出层为2。对数据集进行训练得到数据集，得到模型之后对行为进行预测得到结果。

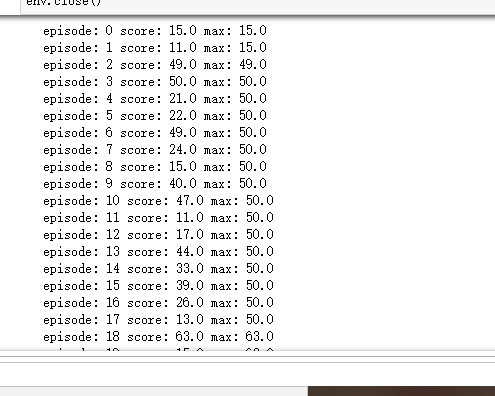


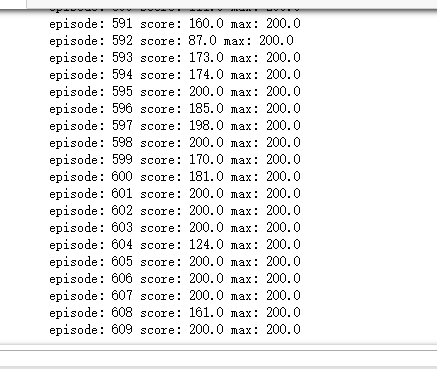
我对模型进行了五次训练可以看到，每增加一次训练loss会变得更小，模型的效果就会好一些，下面的数据是对模型对不同观测行为的预测结果的得分情况，可以看到结果分数平均在1000分左右可以看到模型效果不错。

1. 策略梯度算法

Policy Gradient 网络的输入也是状态(State)，输出是每个动作的概率。例如 [0.7, 0.3] ，这意味着有70%的几率会选择动作0，30%的几率选择动作1。所以导致了算法有更多的不确定性。该次实验搭建的神经网络位输入层为4，输出层为2，隐藏层为100。

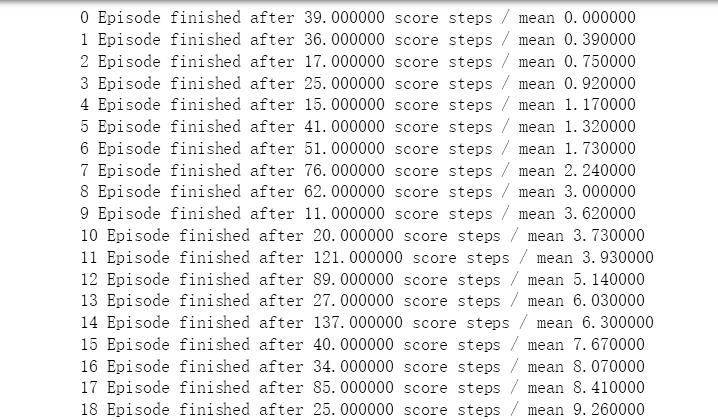
优化策略：累加期望的衰减，某一步的累加期望等于下一步的累加期望乘衰减系数gamma（0.95），加上reward。这使得得分越高之前步骤期望值就越大，说明该步骤出现的概率就越大，因此我借助损失函数来向期望大的方向优化。对于累加期望大的动作，可以放大loss的值，而对于累加期望小的动作，那么就减小loss的值。这样便得到优化策略，之后对游戏进行反复训练，在得分高于一定情况时便结束游戏。

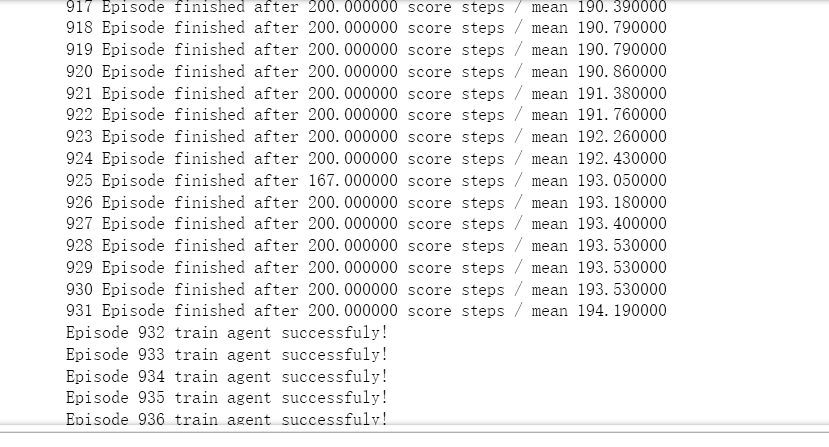


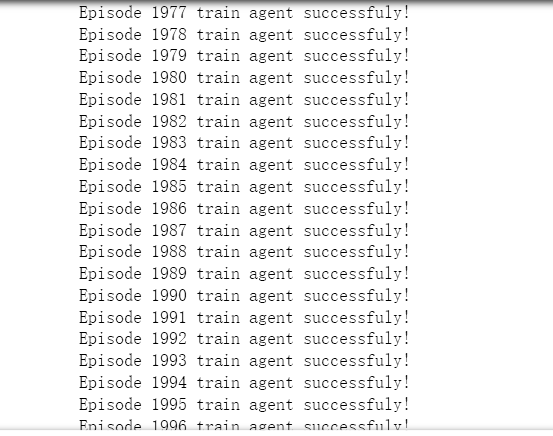


可以看到训练到600多次便可接近200分，分数也稳定在100以上，在经过更长的训练结果会更好。

1. q-learning 算法
2. Learning 就是要学习在一个给定的 state 时，采取了一个特定的行动后，能得到的奖励是什么。在这次实验中我们给定的状态就是小车的4个状态值，行动就是向左向右，奖励就是得分。之后对每一个步骤进行学习，来更新我们的q-table。下一步行动就从表格中选择。但这样还是难以取得好的结果，因此引入了惩罚机制和贪心策略让算法更加智能。具体在代码中实现。







实验共进行了200次游戏，当近100场游戏平均得分大于195时成功，可以看到开始时候每次游戏分很低，随着时间进行分数逐渐升高至稳定，由于学习模型最高分限制在200，如果放开限制那么模型在预测时随着时间进行效果应该会更好得分更高。

**实验小结**

该次实验加深了自己对神经网络以及q-learning的了解，同时也学到对神经网络的不同使用方法既可以作为分类器，也可以作为深度学习中的载体，来实现不同的学习方法和学习策略。同样许多的算法都是可以优化的，在原算法的基础上加以改进也会取得比原来更好的结果，要学会综合利用不同算法的优缺点，不同算法相互结合以达到更好的效果。