# 实验报告

学号: \_2017326603075\_姓名: \_陈浩骏\_ 班级\_17信科1班\_ 成绩\_\_\_\_\_\_\_

# 【实验目的】

解决OpenAI Gym中的cartpole-v0问题

## 【实验内容及要求】

采用深度学习、启发式搜索、进化算法、强化学习中的任一种或两种，解决gym的cartpole-v0问题。

## 【算法描述】

神经网络的搭建采用了Keras的API.

首先, observation是由4个值组成的list, 分别为小车位置, 小车速度, 棍的偏移角与棍的远端线速度. action是一个binary值, 他控制小车向左(0)或向右(1)移动从而将上方的棍置于平衡态. 每个frame结束后, 若棍未达到阈值条件(倒下)则reward+=1, solved的定义为在连续100次根据observation给定action的episode里达到平均reward >= 195. 其余参数与详情详见([openai-gym-cartpole-v0](https://gym.openai.com/envs/CartPole-v0/)).

在本次实验过程中, 首先拿games\_for\_learn=1500局游戏, 采集observation, 与action, 这里需要注意的是, 由于observation是传入action后的step产生的, 在模型的建立过程中, 上一次的observation与本次的action才有关联意义.即(last\_observation驱动action, 即传入last\_observation以predict action).拿games\_before\_learn=1200次随机游戏采集数据, 清洗并采用satisfied\_score=57以上的游戏策略提取特征(即高于上述分数的行为才”值得”学习).

清洗过程中, 对于action做一次one-hot编码, 让神经网络最后一层不采用binary输出, 转而采用激活函数为softmax的二维全连接层输出后, 再拿argmax.

训练完神经网络后, 根据要求, predict 100次游戏, 取平均reward输出.

## 【算法实现】

network = input\_data*(*shape=*[*None, len*(*train\_X*[*0*])*, 1*]*, name='input'*)*network = fully\_connected*(*network, 128, activation='relu'*)*network = dropout*(*network, 0.8*)*network = fully\_connected*(*network, 256, activation='relu'*)*network = dropout*(*network, 0.8*)*network = fully\_connected*(*network, 512, activation='relu'*)*network = dropout*(*network, 0.8*)*network = fully\_connected*(*network, 256, activation='relu'*)*network = dropout*(*network, 0.8*)*network = fully\_connected*(*network, 128, activation='relu'*)*network = dropout*(*network, 0.8*)*network = fully\_connected*(*network, 2, activation='softmax'*)*network = regression*(*network, optimizer='adam', learning\_rate=learning\_rate, loss='categorical\_crossentropy'*)*model = tflearn.DNN*(*network*)*model.fit*(*train\_X, train\_Y, n\_epoch=epoch, snapshot\_step=1000, show\_metric=True*)*

根据satisfied\_score的不同, 以及每次随机游戏的质量, 输入的向量维也会发生改变, 在上述参数下, train\_X的维数大概在(2000, 4, 1)左右以供学习. 全连接层会将其映射并压缩到一个新的维度上以提取特征值, 再输出一个类似one-hot编码二维的predict, 取argmax即可知.

## 【性能分析】

在压缩episodeBeforeSolved(即本实验报告中的games\_before\_learn)的时候, 发现在1000次到1200次左右都能得到接近200的平均reward. 并且在调节参数时, 发现该模型有较高的损失函数值, 且该模型要注意避免过拟合, 都是由于这个模型的特殊性, 因为它的输出结果越接近游戏(即游戏完成度越好), 它的action集的平均值就越趋向于0.5, 所以导致在激活函数中总会出现微小的偏差.

在调整参数后, 可以达到相对稳定的reward=199的输出, 认为该问题已solved.

**【实验小结】**

虽然仅取observation四个特征给进神经网络模型中进行训练, 感觉起来相对比较暴力, 但是该算法在数据量相对较小的时候也能得到较好的输出, 在最初建立模型时, 我曾想用大约过万次的数据来进行训练以达到结果. 但是该训练模式不足的是, 它太依赖于随机游戏时产出的数据了, 即要求越高, 数据越少, 训练效果越差, 毕竟gym对于该游戏的算法评分是“采用越少episodeBeforeSolved越好”.

运行console位于 console.log中, 源码位于gym-tf.py中.