# 残缺棋盘覆盖实验报告

简介：本实验使用强化学习方法去覆盖残缺棋盘，一共经历了6个版本的迭代过程，不同版本之间的差异为奖励函数的不同，强化学习算法的不同，动作空间和状态空间的不同，其中共使用了3中强化学习算法，分别为：DDPG，PPO2，DQN。本文没有对强化学习的理论内容过于阐述，只讲述了建模过程。本文采用openAI的gym库和stables-baselines 强化学习库，强化学习模型中所采用的策略网络为MlpPolicy和CnnPolicy。

## 强化学习研究现状

强化学习主要包括基于价值函数、策略梯度以及搜索和监督的方法。DQN代表了基于值的深度强化学习算法，Q-Learning中总会存在一些问题，由于max贪婪机制的存在，总是会过大的估计Q-VALUE。如果值函数每一点的值都被过估计了相同的幅度，即过估计量是均匀的，由于最优策略是贪婪策略，即找到最大的值函数对应的动作，这时候最优策略是保持不变的。然而，在实际情况中，过估计量并非是均匀的，因此值函数的过估计会影响最终的策略决策，从而导致最终的策略并非最优，而只是次优。

为了解决值函数过估计的问题，Hasselt提出了Double Q-learning的方法，是将动作的选择和动作的评估分别用不同的值函数来实现，但大多数DQN如duel-DQN只能处理离散的动作集，不能表达连续的行为。基于策略的强化学习弥补了DQN的不足。通过对策略函数进行建模，可以输出动作的分布，包括actor-critic、DDPG、A3C和PPO2算法。2017年，AlphaGo Zero以其惊人的成功成为国际头条新闻。从空白状态学习，AlphaZero 可以快速自学，无需任何人工输入。除了无样本和自训练外，AlphaGo Zero还采用了蒙特卡罗树搜索（MCTS）算法，为算法带来了新的发展。

## 二、master版本

master版本为最初版本，使用DDPG模型，Agent为对环境的建模，其中包含reset()，step()，render()，reward()。对棋盘进行如下建模：

1、棋盘初始化：棋盘大小为size，构建shape=(size,size)的初始值为0的数组state，随机选取残缺点state[x][y]=1，用done表示游戏的状态，游戏的初始状态为False，棋盘全部覆盖满是为True。

2、棋盘从位置（0，0）开始遍历，到（size-1，size-1）结束，其中在每个位置，需要将当前位置周围8个点的状态作为obs返回给DDPG模型，模型根据obs输出该位置执行的动作，同时环境模型agent会执行该动作，若能填入，则跳转到下一个位置，否则继续停留在原定。

3、奖励策略：首先判断当前输出的动作（选择的三格板）能否填入当前棋盘，若能填入，则给0.5的奖励，否则给-1的奖励，若能够全部填满，则给10的奖励。这是一个超稀疏的奖励策略。

在训练10000次后，累计奖励值随迭代次数的变化如下图所示，发现累计奖励值没有上升的趋势，失败！

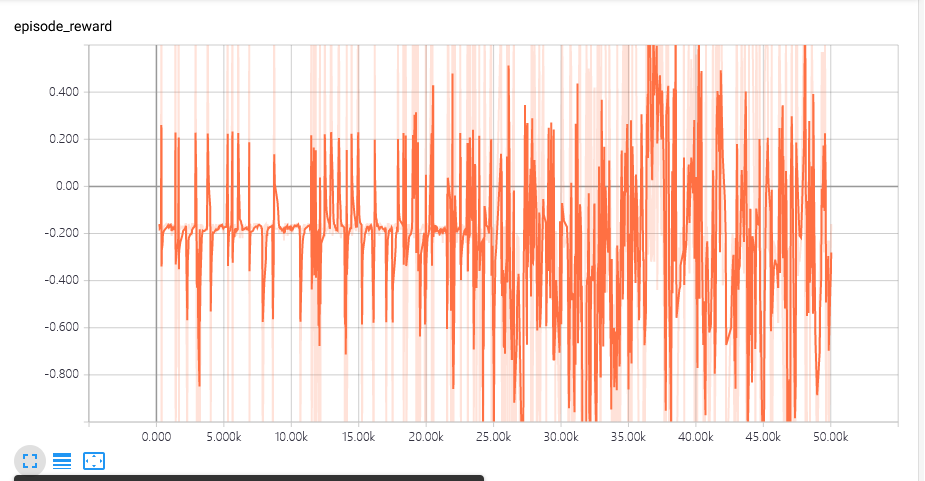
## 三、Policy1.0

与master的版本区别：

更改了奖励策略，现奖励策略如下：若能选定的动作(三格板)，能填进棋盘，则给予1的奖励，若选出的动作为0，表示不覆盖任何三格板，同时，若此时真的不能覆盖三格板，说明该预测动作正确，则给予-0.2奖励，若此时真的能覆盖三格板，说明该预测动作错误，则基于-0.5的奖励，如何改预测动作不为0并且不能填入任何动作，则基于-0.8的奖励。没执行一步给与-0.1的奖励。

更改了强化学习的可观察空间，可观测空间变为整个棋盘的大小。

在训练50000次后，累计奖励值随迭代次数的变化如下图所示，发现累计奖励值没有上升的趋势，失败！，但测试已经能够填满一种情况。

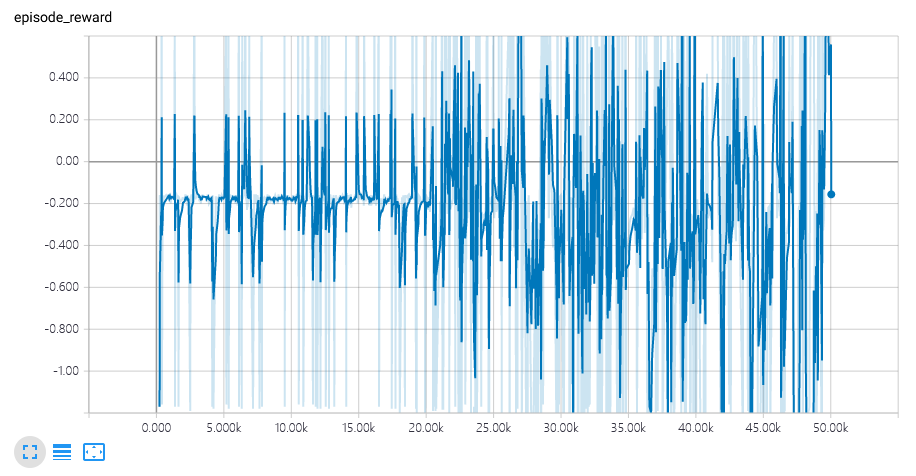


## 四、policy1.1

模型采用PPO2模型。相比较于之间的奖励策略，policy1.1版本在策略上采用了新的方法：

使用cnn卷积的思想，基本思路为：设置一卷积核，其权重固定为1，上下两个方向的步长全为2，其大小的变化从size\*size到4\*4、2\*2，其目的是计算棋盘中填满的2\*2、4\*4…的格子有多少，将这些值全部累加起来，在与执行该预测动作前的数值相减，得到的值作为reward，摒弃了以前只要能填进去就给予一个正奖励，这次的变化对棋盘建模更加接近事实。

在训练50000次后，累计奖励值随迭代次数的变化如下图所示，发现累计奖励值没有上升的趋势，失败！，但测试已经能够填满一种情况。相比较之前有了进步，同时在训练1e7后，累计奖励值存在上升的趋势（由于训练时间太长，且忘截图，故没图），但效果还是不很好，且由于状态空间为整个棋盘的大小，导致棋盘大小以改变就要重新训练一个模型，不是一个通用模型。



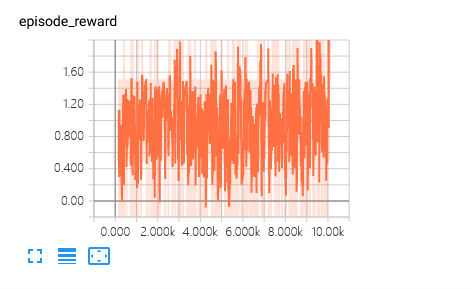
## 五、policy2.0

模型采用PPO2模型，受到动态规划思想的启发，改进之处：

更改了对棋盘位置的选择，不在采用遍历每个位置的方法，而是首先选择棋盘的中心点作为第一位置，之后将棋盘分成大小相同，状态类似的四个部分，选择其中心点作为接下来的要填的位置，之后再分，直到分成的四个小棋盘边长为2。棋盘的状态空间obs为每个棋盘矩阵state，动作空间没变。

更改了奖励的计算方法，卷积核的大小固定，为2\*2，只计算一次，不在计算大小为4\*4，8\*8，步长不变，之后的计算方式。

在训练10000次后，累计奖励值随迭代次数的变化如下图所示，发现累计奖励值没有上升的趋势，失败！，同时在训练1e7时，效果比版本policy1.1还差。



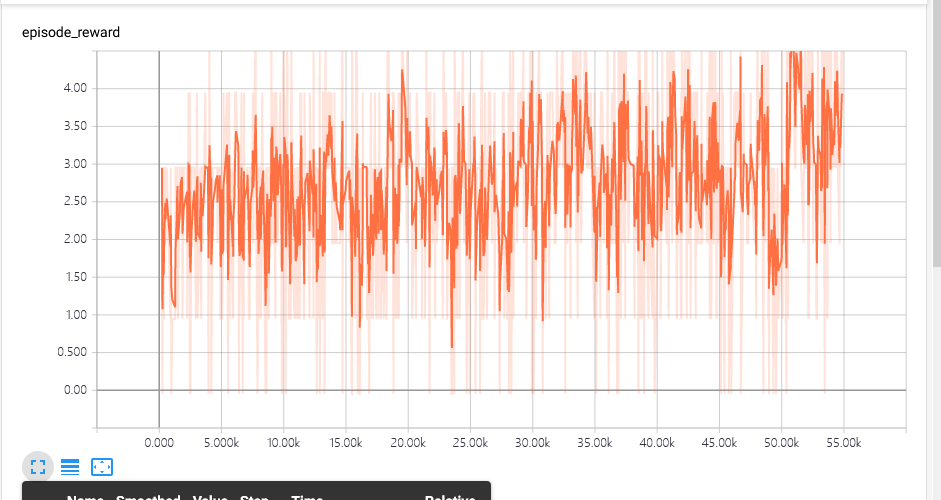
## 六、policy2.1

模型采用PPO2模型，与上一版的对比（改进）：

模型的状态空间：对当前棋盘按照其中心点进行划分为四个小棋盘，分别对这四个小棋盘的内的数进行求和，agent返回观察值obs为这四个小棋盘的和。举例：现有一8\*8的棋盘，第一个填值位置在棋盘中心，现将棋盘划分成4个4\*4的小棋盘，分别对这4个小棋盘进行求和，返回一个长度为4的列表；当执行完一次动作后，跳转到右上角棋盘大小为4\*4的棋盘，将4\*4的棋盘按中心点划分为4个2\*2的棋盘，对这4个2\*2的棋盘求和并返回obs。

模型的奖励方法没有更改。

在训练10000次后，累计奖励值随迭代次数的变化如下图所示，发现累计奖励值没有上升的趋势，失败！在训练1e7次后，累计奖励值没有上升的趋势，且测试效果不好。



## 七、policy2.2

此次，使用DQN强化学习模型，输入的obs为上述长度为4的obs中为1的下标。模型的奖励函数没有改变

在训练10000次后，累计奖励值随迭代次数的变化如下图所示，可以发现累计奖励值随迭代次数快速上升，并达到最大值。成功！且在测试过程中能够完美填入棋盘，且是一个通用模型。



## 八、总结

在几次实验的过程中，发现对棋盘的建模越具体，设置的奖励值越符合实际情况，则强化学习模型对动作的预测就越准确。在测试中的表现也就更好。