Openai gym 最终报告

1. 游戏规则：

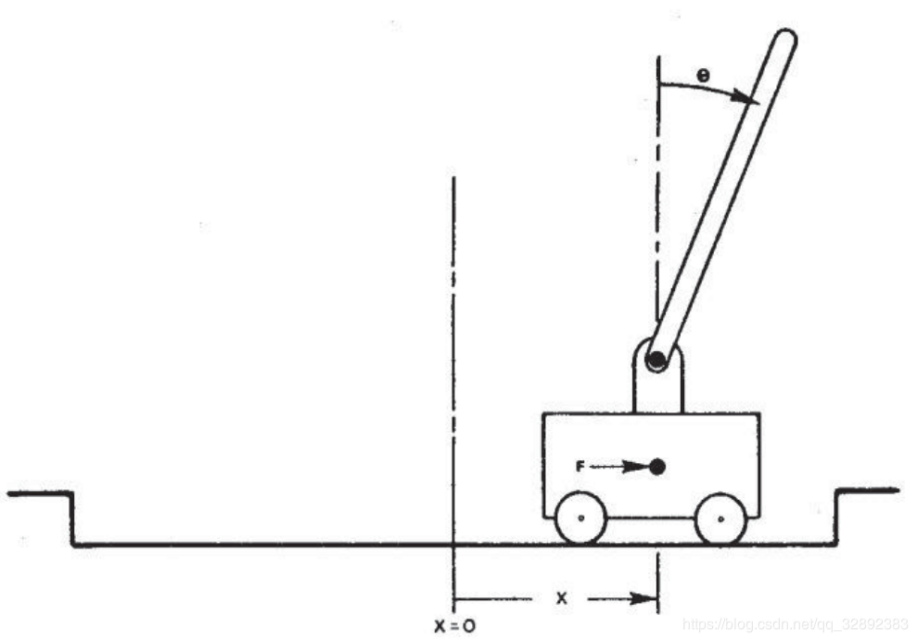
Car游戏里面有一个小车，上有竖着一根杆子，每次重置后的初始状态会有所不同。小车需要左右移动来保持杆子竖直，为了保证游戏继续进行需要满足以下两个条件：

杆子倾斜的角度*θ*

θ不能大于15°

小车移动的位置*x*

x需保持在一定范围（中间到两边各2.4个单位长度）



**动作**：

左移（0）

右移（1）

**状态变量**：

x：小车在轨道上的位置

θ：杆子与竖直方向的夹角

x˙：小车速度

θ˙：角度变化率

二．参数的设置

Observation：

Type: Box(4)

Num Obeservation Min Max

0 Cart Position -2.4 2.4

1 Cart Velocity -Inf Inf

2 Pole Angle -24 deg 24 deg

3 Pole Velocity At Tip -Inf Inf

Acions：

Type: Discrete(2)

Num Action

1. Push cart to the left

1 Push cart to the left

三．利用爬山算法研究cartpole

简单来说，爬山算法分为

1.随机选择一个符合条件的解

2.按某一个间隔n递增

3.得出一个解集，找出解集中的最优解

4.将这个最优解依据上面的方法再构造一个解集，再求最优解

5.直到前一次的最优解和后一次的最优解相同才结束爬山

这五个步骤。

对于CartPole倒立摆这个具体问题，其控制输出a很简单，就是向左或者向右。输入p就是车的位置，速度，杆的角度，角速度。首先需要构造一个从输入到输出的映射:a=f（p）

那么这里显然我们可以使用最简单的映射，就是线性映射： a=wp，w即为参数向量。然后我们就模拟修改这些参数，从而得到一个新的“点”。

最后的问题就是如何判断w好坏了，想法也很简单，每次运行一次episode即一次实验，累加计算得到的reward值，用Score得分表示，代表这个参数效果更好。

显然，这是一个局部最优的算法，无法保证结果最优

结果截图：

1. 利用dqn研究cartpole

我们先讨论Q-learning。Q-Learning是强化学习算法中value-based的算法，Q即为Q（s,a）就是在某一时刻的 s 状态下(s∈S)，采取 动作a (a∈A)动作能够获得收益的期望，环境会根据agent的动作反馈相应的回报reward r，所以算法的主要思想就是将State与Action构建成一张Q-table来存储Q值，然后根据Q值来选取能够获得最大的收益的动作。

Q-Table a1 a2

s1 q(s1,a1) q(s1,a2)

s2 q(s2,a1) q(s2,a2)

s3 q(s3,a1) q(s3,a2)

在cartpole问题中，我们用obeservation来表示cartpole的状态。四个参数分别为小车在轨道上的位置，杆子与竖直方向的夹角，小车速度，角度变化率。

参数设置：

Observation：

Type: Box(4)

Num Obeservation Min Max

0 Cart Position -2.4 2.4

1 Cart Velocity -Inf Inf

2 Pole Angle -24 deg 24 deg

3 Pole Velocity At Tip -Inf Inf

将其离散化，可得q矩阵

另一个参数action：

Acions：

Type: Discrete(2)

Num Action

1. Push cart to the left

1 Push cart to the left

整个实验可按照以下步骤进行

A.传输当前的状态

B.离散化当前状态，并根据当前状态决定action

C.执行action后，获得新的状态

D.更新q矩阵

而dqn相对于qlearing的改进是，dqn不用q表记录q值，而是用神经网络来预测q值，并通过不断更新神经网络从而学习到最优的行动路径。

Dqn中，q值表中表示的是当前已学习到的经验。而根据公式计算出的 q 值是agent通过与环境交互及自身的经验总结得到的一个分数（即：目标 q 值）。最后使用目标 q 值(target\_q)去更新原来旧的 q 值(q)。而目标 q 值与旧的 q值的对应关系，正好是监督学习神经网络中结果值与输出值的对应关系。