**cart-pole实验报告**

2016202106

张艺邻

1. **背景介绍**
   1. **cartpole**

cartpole-v0和cartpole-v1是openaigym上的一个系列的小车平衡游戏。小车上有一个可摇晃的杆子，小车可被操纵向左移或向右移，在杆子倾斜角度大于十五度或小车距离绝对值大于2.4时，游戏失败；否则，玩家获得点数为1的回报值，游戏继续。

这个平台的接口为玩家提供了方便的接口，其中env.step(parameter)接受action（值为0或1）作为参数，返回observation，reward，done，info。

observation是一个大小为4的一维数组，值分别是小车当前的位置，速度，旋转角度，和旋转速度；reward是action之后的报酬，若小车未失去平衡则为1；done为1则小车失去平衡，反之为0；info为调试信息，通常不用。

observation

| **Num** | **Observation** | **Min** | **Max** |
| --- | --- | --- | --- |
| 0 | Cart Position | -2.4 | 2.4 |
| 1 | Cart Velocity | -Inf | Inf |
| 2 | Pole Angle | ~ -12° | ~ 12° |
| 3 | Pole Velocity At Tip | -Inf | Inf |

action

| **Num** | **Action** |
| --- | --- |
| 0 | Push cart to the left |
| 1 | Push cart to the right |

玩家的任务即是通过观察值observation，制定策略决定左移（action = 0）或右移（action = 1），使得小车越长时间保持平衡越好

**2.pytorch**

Pytorch由Facebook开源的神经网络框架，Torch 是一个经典的对多维矩阵数据进行操作的张量（tensor ）库，在机器学习和其他数学密集型应用有广泛应用。与Tensorflow的静态计算图不同，pytorch的计算图是动态的，可以根据计算需要实时改变计算图。作为经典机器学习库 Torch 的端口，PyTorch 为 Python 语言使用者提供了舒适的写代码选择。

PyTorch的设计追求最少的封装，设计遵循tensor→variable(autograd)→nn.Module 三个由低到高的抽象层次，分别代表高维数组（张量）、自动求导（变量）和神经网络（层/模块），而且这三个抽象之间联系紧密，可以同时进行修改和操作。

在本次实验中，主要利用pytorch完成了机器学习的线性回归，逻辑回归和深度强化学习

1. **机器学习方法**
   1. **线性回归**

我们拥有观察值位置，速度，倾斜角，角速度，那么能否这些变量预测出下一个阶段小车的观察值呢？

我们将数据分为两部分，第一部分是在action = 0之下的一对观察值（即在采取一个动作之前的观察值和采取动作之后的观察值），一部分是在action = 1之下的一对观察值，分别对它们进行线性拟合

y = Ax + b

where x == observation.reshape(1, 4)

y == next\_predict\_observation

其中y，x，b是4维列向量,A是4\*4的矩阵，A，b是利用梯度下降算出的参数

最后得出在action=0时，矩阵A为

[

[ 1.0000000e+00, 2.0000011e-02, -8.6545292e-08, 1.9063853e-08],

[ 9.6184351e-08, 9.9999869e-01, -1.4286327e-02, -6.2227770e-07],

[-1.1246829e-08, 2.8609592e-09, 9.9999994e-01, 2.0000001e-02],

[-3.3518585e-04, 3.1545463e-03, 3.1878245e-01, 1.0026963e+00]

]

a为

[[ 3.6427844e-10],

[-1.9512123e-01],

[-1.3289457e-10],

[ 2.9277134e-01]]

在action = 1时，矩阵B为

[[ 9.9995482e-01, 2.0425035e-02, -3.9787360e-02, -5.7976576e-04],

[1.7955095e-07, 9.9999869e-01, -1.4293073e-02, -7.3635579e-07],

[-1.3007304e-08, 1.2111884e-07, 9.9998885e-01, 1.9999843e-02],

[6.4456863e-08, -5.1967859e-06, 3.1537077e-01, 9.9999470e-01]]

偏移量b为

[[ 1.2654233e-04],

[ 1.9512123e-01],

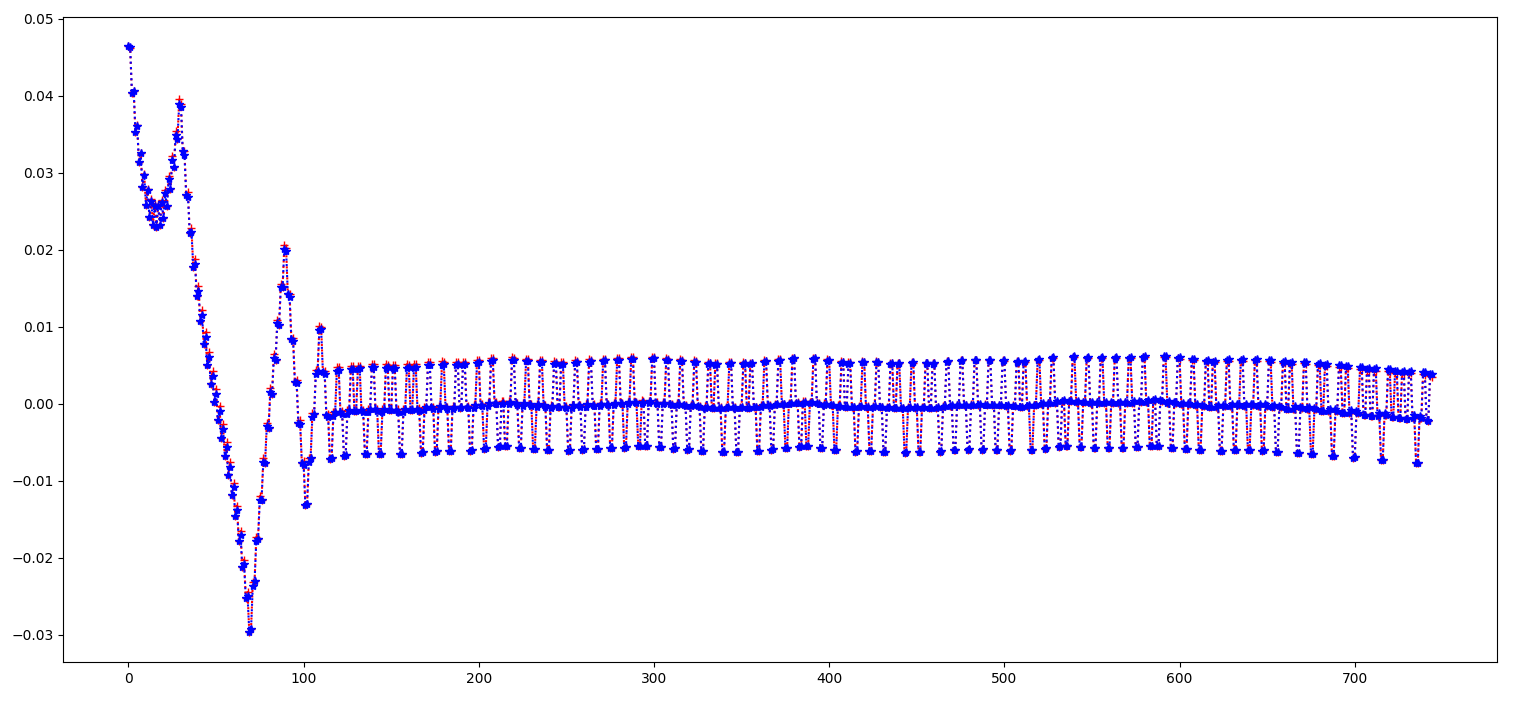
[ 3.4660967e-08],

[-2.9267576e-01]]

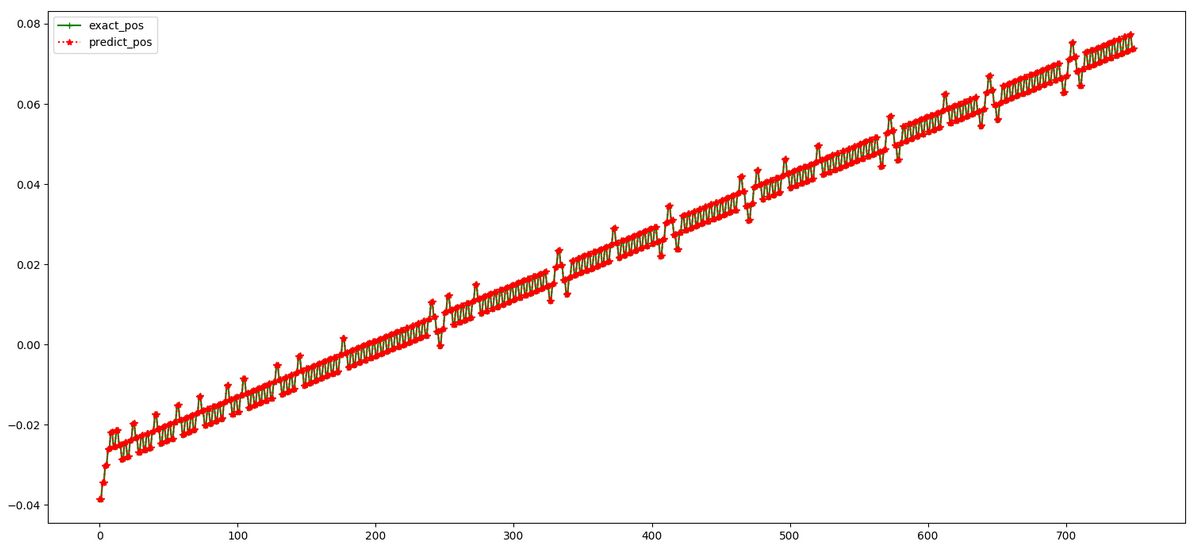
loss函数为MSE

我们开启一个游戏作为测试集，并计算预测值和实际值之间的平方误差和。在这个过程中，我们每次选择角度预期变化量最小的预测值。

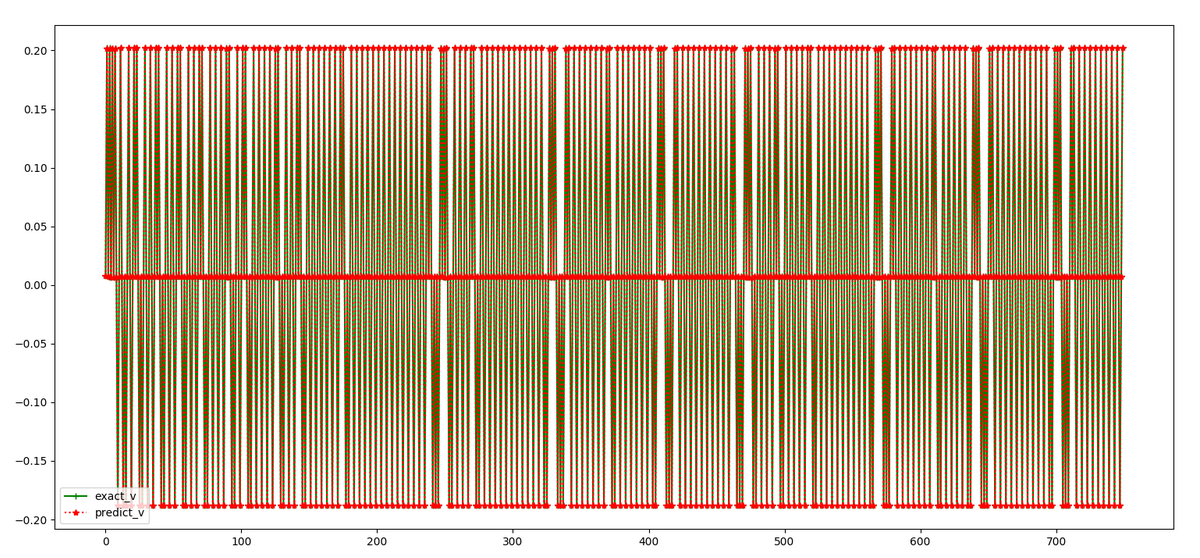




红蓝点分别为预测的角度值和在移动小车后实际的角度值，可以看到它们基本重合



红绿点分别为预测以及实际的小车位置



红绿点分别为预测以及实际的小车速度

* 1. **logistic二分类**

logistic regression是常用的二分类算法，可视为线性回归的一种变式。

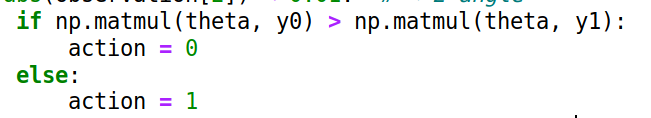
我们将observation作为input，经过sigmoid函数输出一个介于0， 1的值，我们希望达到的效果是这个值越接近于1，那么当前状态越容易失去平衡，这个值越接近0，那么当前状态越能保持平衡

首先，我们生成四维的数据，并分情况给它们贴上标签0或1：

如果该数据位于一局游戏的后百分之十，则贴上标签1，反之贴标签0



损失函数是交叉熵函数



theta是logistics返回的参数

事实证明，该方法虽然在特定的训练集上能对observation二分类，但在根据它做出决策时，小车坚持步数与随机选择无异。

结论是，可能logistic二分类在该情况下，仅仅对于分类有效果，但在精度要求更高的需要依据它做出决策的情况下，效果并不好。

* 1. **kmeans聚类**

由于logistic regression的效果并不显著，因此我觉得可能observation与它的平衡能力之间并没有线性关系，所以我尝试另外一种方法建立observation与其平衡能力的关系，即k平均聚类算法。

同样，最大的问题是数据的产生。我们以最佳状态[0, 0, 0, 0]为起始点，分值最高，根据上述线性部分y=Ax+b产生观察值集，并且分值根据它们到起始点的步长递减。



然而我们将y0和y1分类的尝试仍然失败了，在实际的游戏中与logistics算法的表现相差无几。

结论是，通过kmeans和logistics是很难对观察值这个四维变量进行分类的，我的猜测是，小车状态的好坏不能单纯地通过线性或者平方表示。