# RL-1

Abhilash Jain
Student Number: 722074

September 27, 2019

## 1 Increase time steps

Can the same model, trained with 200 timesteps, balance the pole for 500 timesteps? Why/why not?

When the episode length is increased, in my experiments it did not succeed to balance the pole for 500 timesteps, one of the reasons for that maybe the weights it learned for balancing during training were optimal for maximum 200 timesteps only, there-fore whenever in an episode after reaching 200 timesteps it was unsure later therefore didn't succeed(it didn't train for this scenario). However, some training might produce a result where the weights do somehow include this scenario where 500 timesteps is successful.But this is a matter of training the model till it achieves that(Stochastic in nature).

## 2 Repeatibility

Why is this a case? What are the implications of this, when it comes to comparing reinforcement learning algorithms to each other?

The algorithm used in the exercise is stochastic in nature and, which logically explains the high variance between different trainings.This can also be taught as, there are a lot plans for success and each training instance, gets one or the other plan. i.e to maximize the reward, the weights are quite different in each training instance.

When comparing algorithms in this high variance setting is very difficult, more factors have to be looked into other than success/convergence, like the type of reward function, training time, even though there is high variance among runs, some idea of robustness can be made. Which can be used to compare algorithms.

# 3 Reward functions

The different types of reward functions have been commented in the code file cartpole.py

How does changing the reward function impact the time needed for training?

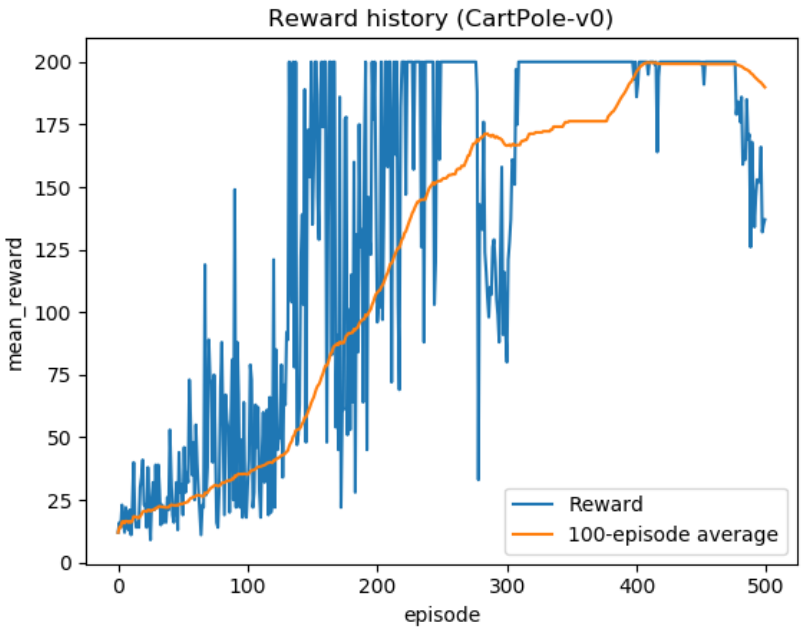
During the process of writing and experimenting with reward functions, few things which might affect the time needed are:

1) As the complexity of the reward function increases the model takes longer for success or may not converge in some cases, i.e. if the model is constrained with alot of more difficult tasks or incentive is given for more difficult tasks, the time needed to train increases by alot, for example in the third task, it requires alot of time to train such a model when compared to simple balancing problem. Also sometimes the high velocity task completely fails or the objective is not reached(very hard to reach) if the reward function is not smooth

2) Using smooth reward functions are also easier to train than sparse functions, there-fore it would be better if we can include smooth reward functions.

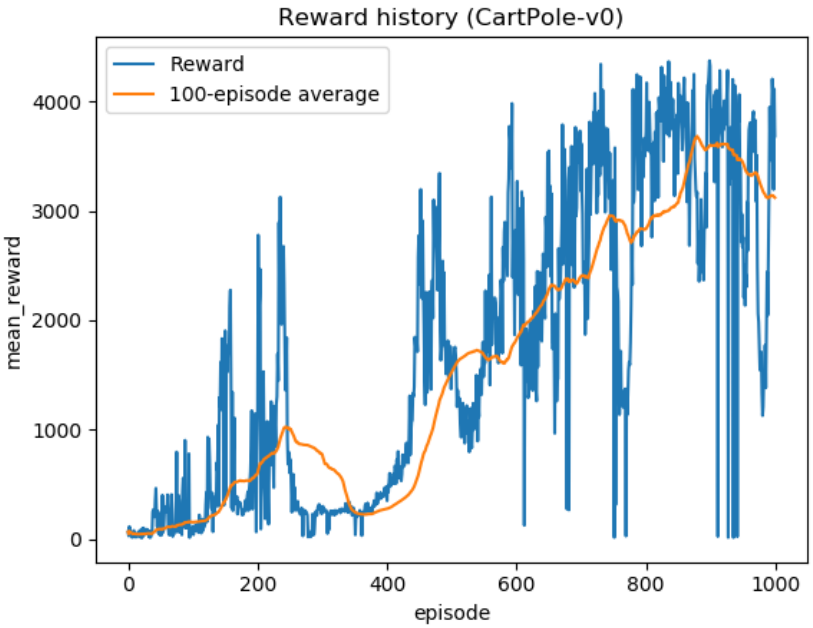
# 4 Results

## 4.1 Normal run with timesteps 200



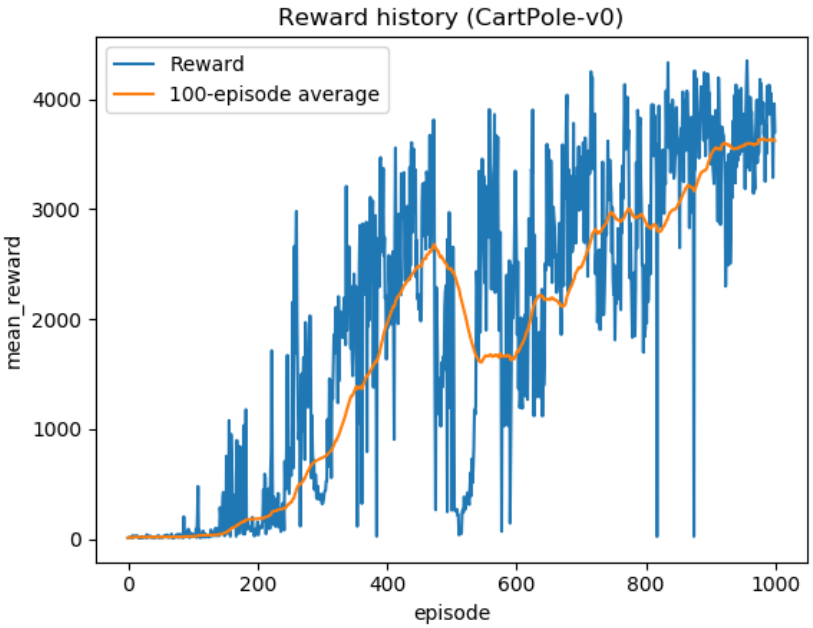

During testing fails with 500 timesteps.

## 4.2 Position 0



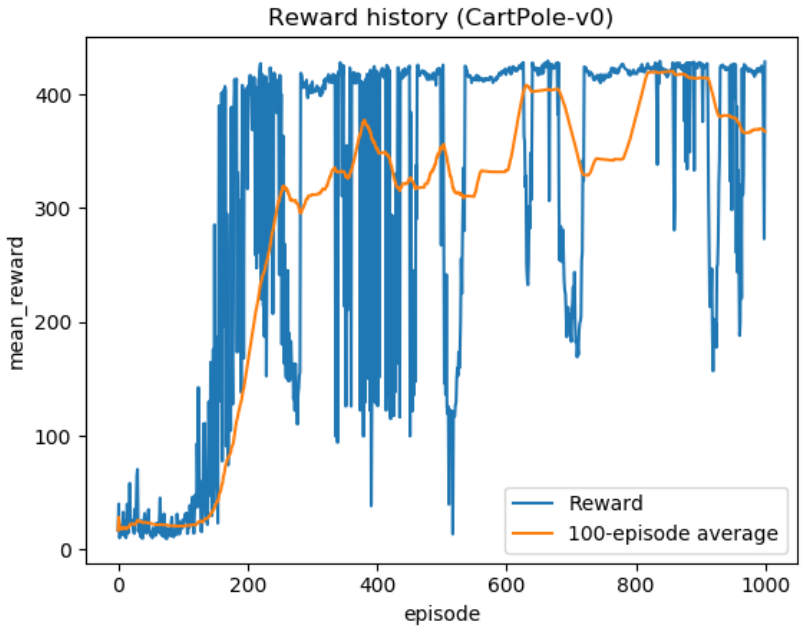

During testing it can be seen it tries to stay at position 0

### 4.3 Position X




During testing it can be seen it tries to balance at position X, which in our case is 1

### 4.4 High Velocity




During testing it was seen that it moved with a higher velocity than normal, but also didn't get out of the screen. i used a smooth function sigmoid for implementation of the reward function