# HW3

Brody Kendall

November 2022

## Problem 1

### a

$E_{\pi_\theta} \nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)$
$= E_{\pi_\theta} \nabla_\theta \log \pi_\theta(\tau) r(\tau) - E_{\pi_\theta} \nabla_\theta \log \pi_\theta(\tau) b$
$= \nabla_\theta J(\theta) - \nabla_\theta E_{\pi_\theta} b$
$= \nabla_\theta J(\theta)$

So this is an unbiased estimate of the policy gradient.

### b

$Var(\nabla_\theta J(\theta)) = E((\nabla_\theta J(\theta))^2) - E(\nabla_\theta (J(\theta))^2$
$= E(\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)^2) - E(\nabla_\theta \log \pi_\theta(\tau) r(\tau))^2$

### c

Take the derivative and set equal to 0:

$\frac{\partial}{\partial b} Var(\nabla_\theta J(\theta))$
$= \frac{\partial}{\partial b}(E(\nabla_\theta \log \pi_\theta(\tau) r(\tau)^2) + E(\nabla_\theta \log \pi_\theta(\tau) b^2) - E(\nabla_\theta \log \pi_\theta(\tau)^2 2 r(\tau) b))$
$= 2bE(\nabla_\theta \log \pi_\theta(\tau)^2) - 2E(\nabla_\theta \log \pi_\theta(\tau)^2 r(\tau)) = 0$
$\Rightarrow bE(\nabla_\theta \log \pi_\theta(\tau)^2) = E(\nabla_\theta \log \pi_\theta(\tau)^2 r(\tau))$
$\Rightarrow b = \frac{E(\nabla_\theta \log \pi_\theta(\tau)^2 r(\tau))}{E(\nabla_\theta \log \pi_\theta(\tau)^2)}$

Note that this b results in the minimum variance because the second derivative is always positive:

$\frac{\partial^2}{\partial b^2} Var(\nabla_\theta J(\theta)) = 2E(\nabla_\theta \log \pi_\theta(\tau)^2) > 0$

### d

The minibatch used to compute the policy gradient should be (ii) larger. This is because policy optimization techniques are general more versatile, but less sample-efficient. In addition, these techniques are less exploratory, so having a larger minibatch size should help compensate.

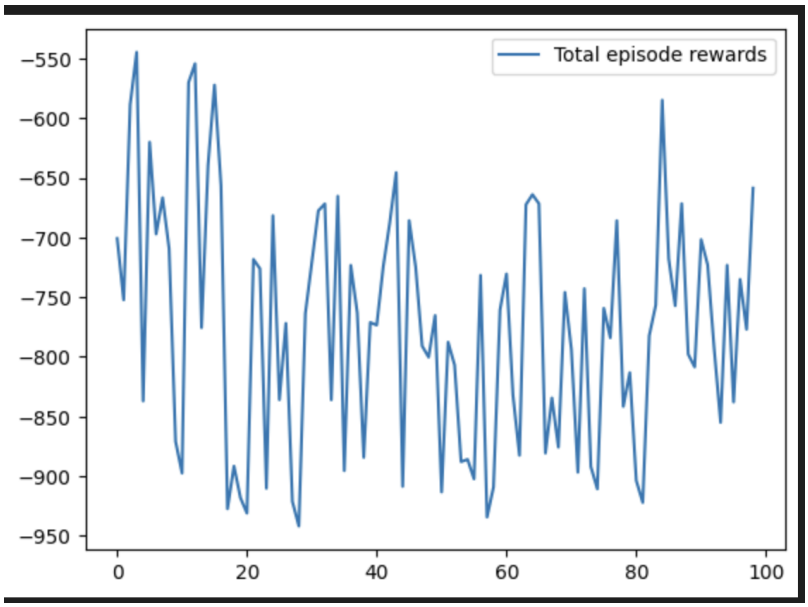
# Problem 2

## 1

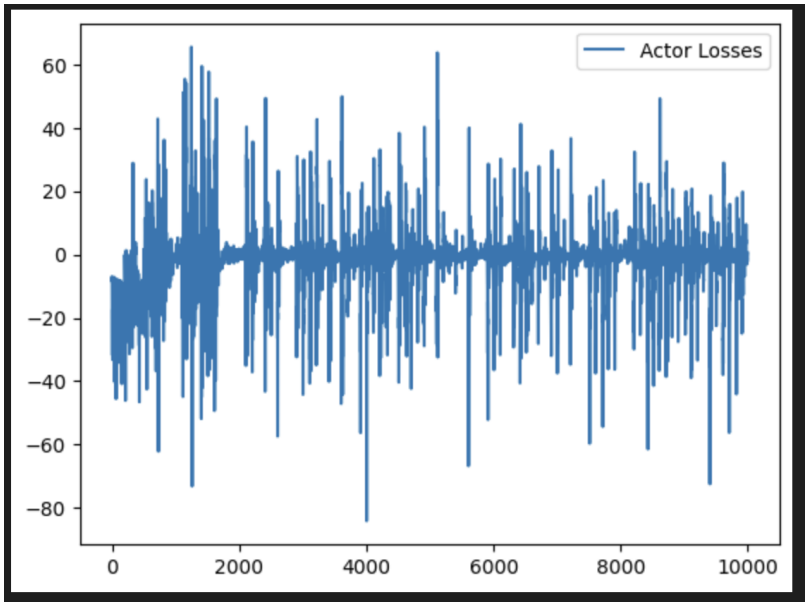
There are an infinite number of states and actions since the state and action spaces are both continuous.

## 2

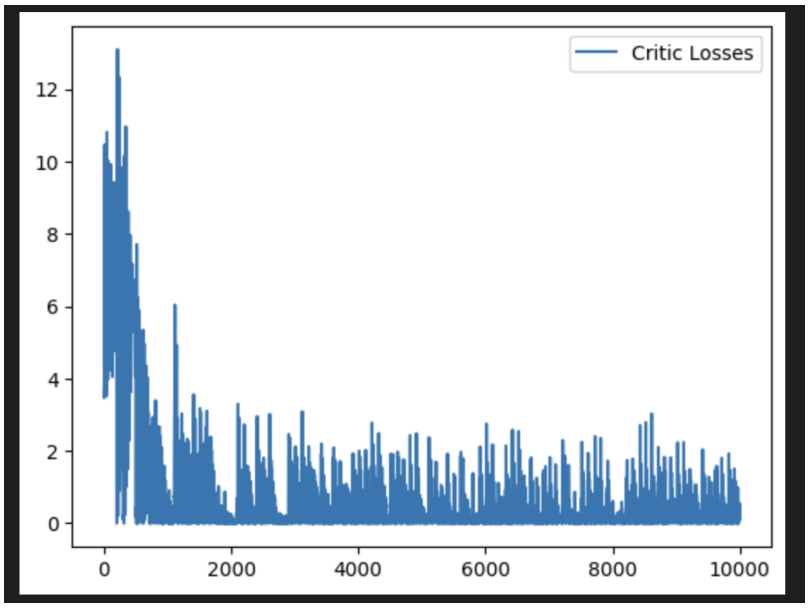
### a



b


Actor Losses
60
40
20
0
−20
−40
−60
−80
0
2000
4000
6000
8000
10000

c


Critic Losses
12
10
8
6
4
2
0
0
2000
4000
6000
8000
10000

### 3

#### a

Increasing the entropy weighting term seems to increase the value of the actor loss function, but (at least for my implementation) changing it does not seem to affect the total episode rewards in any noticeable way.

#### b

It seems that increasing the episode horizon actually has a negative impact on training while decreasing the episode horizon only affects the variance of the total episode rewards. To me, this suggests that something is wrong with my implementation but I can't figure out what it is.

#### c

Changing gamma seems to have little affect on the total episode rewards. Again, this suggests an issue with my implementation which I can't figure out. A smaller gamma does make the loss functions take longer to converge.

As stated before, I think my implementation is unfortunately off, so changing the hyperparameters doesn't have much of an effect on the episode rewards, but the best set of hyperparameters I have found is: gamma = 0.99, entropy weighting = 0.1, horizon = 50

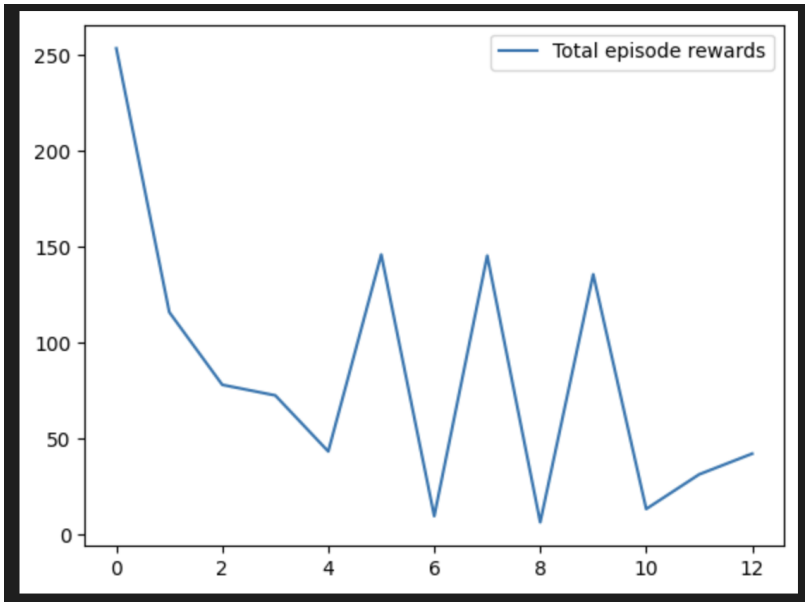
## Problem 3

### 1

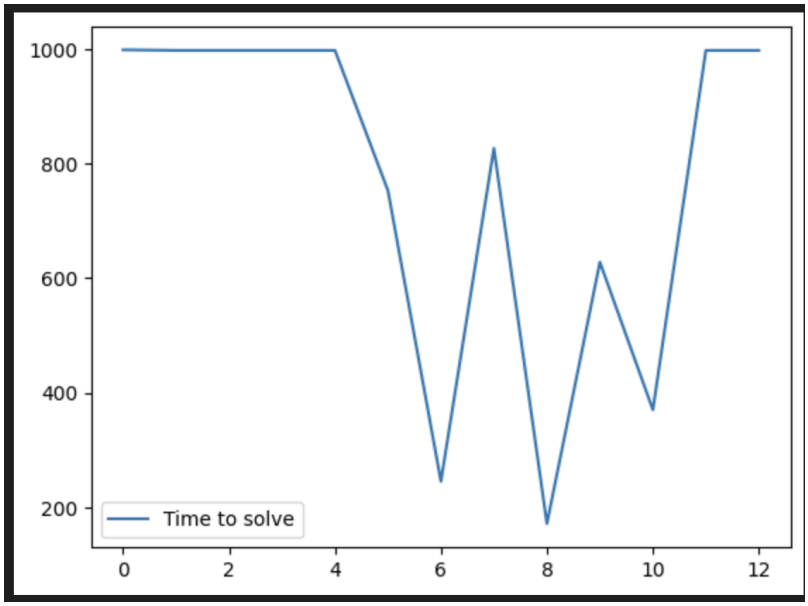
Again, there are an infinite number of states and actions since the state and action spaces are both continuous.

## 2

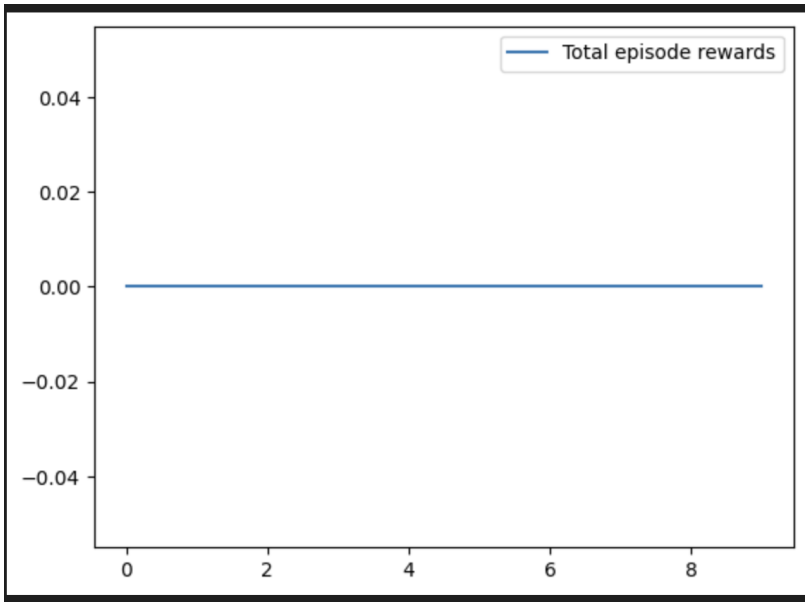
a


Total episode rewards
250
200
150
100
50
0
0
2
4
6
8
10
12

b


1000
800
600
400
200
Time to solve
0
2
4
6
8
10
12

**c**




There is clearly a difference. In this case, the agent does not explore enough and never achieves the goal, so the reward is always 0.

Note: If I had given myself more time, I would've tried to implement a replay buffer to see if this fixed the problem.

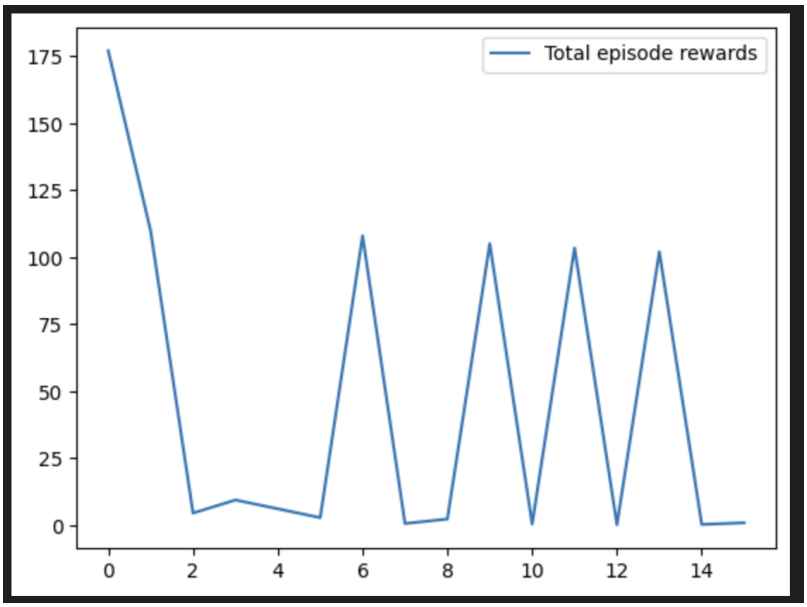
## 3

**a**

It seems from a few examples I tried that a smaller gamma results in a longer time to solve, but the agent successfully solves the problem more often. On the other hand, a larger gamma results in a faster time to solve when it does solve the problem, but it successfully does so less often.

**b**

Using 1/N(s) instead of 1/sqrt(N(s)) resulted in the following plot:




The total episode rewards start at a lower value and decrease more quickly, so after only 2 episodes we can see that when the agent doesn't reach the goal, it gets essentially negligible reward. However, when it does reach the goal, the spike in reward is more stark.

## 4

How big is $10^{350}$?

There are an estimated 7.5 sextillion ($7.5*10^{21}$) grains of sand on earth, let's just assume this is an underestimate and that there are actually 10 sextillion ($10^{22}$) grains of sand on earth. Let's call one generation 100 years. If one member of your family in each generation found one of these grains of sand, you would have to repeat this process $10^{326}$ times to reach $10^{350}$ years.

Now let's say the stock market has crashed a little bit and Warren Buffet's net worth is down to \$100 billion ($\$10^{11}$). If every time your family finds all the grains of sand on earth, Warren Buffet loses 1 penny, then he would have to lose his entire net worth $10^{313}$ times to reach $10^{350}$.

Now let's say there are $10^{120}$ unique game states in chess. If every time Warren Buffet loses his net worth, we find a new unique game state in chess, we would have to exhaust the total number of unique game states $10^{193}$ times to reach $10^{350}$.

Now let's say that there are $10^{80}$ atoms in the universe. If every time we exhaust the total number of unique game states of chess, we give a name to a new atom in the universe, we would have to name every atom $10^{113}$ times to reach $10^{350}$.

Now let's say that there are $10^{51}$ nucleons (protons or neutrons) on Earth. If every time we finished naming every atom in the universe, we came up with

a different name for a new nucleon on Earth, we would have to name every nucleon $10^{62}$ times to reach $10^{350}$.

Now let's say that there are $10^{38}$ DNA base pairs within the entire biomass of Earth. If every time we finished naming every nucleon on Earth, we found a new DNA base pair, we would have to find every base pair on $10^{24}$ times to reach $10^{350}$.

Now let's say that there are $10^{14}$ neural connections in the human brain. If every time we found every base pair on Earth, you fired one neural connection in your brain, you would have to fire every neural connection in your brain 10 billion times to reach $10^{350}$.

Now let's say that there are are 10 billion people on Earth. If every time you fired every neural connection in your brain, you gave a new person on Earth \$1, it would take exactly $10^{350}$ years to give everyone on Earth \$1.