HOME WORK 1 - CS 687

DANIEL SAM PETE THIYAGU

Part 1

Question 1. $M = (\mathcal{S}, \mathcal{A}, P, R, d_0, \gamma)$ and a fixed policy,

Question 1a. $Pr(S_{t=3} = s \ or \ S_{t=3} = s^{'}) = Pr(S_{t=3} = s) + Pr(S_{t=3} = s^{'})$, Assuming s and $s^{'}$ are distinct states
$Pr(S_{t=3} = s) = \sum_{s^* \in S, a^* \in A} Pr(S_{t=3} = s | S_{t=2} = s^*, A_{t=2} = a^*) * Pr(S_{t=2} = s^*, A_{t=2} = a^*)\}$

$Pr(S_{t=3} = s) = \sum_{s^* \in S, a^* \in A} Pr(S_{t=3} = s | S_{t=2} = s^*, A_{t=2} = a^*) * Pr(A_{t=2} = a^* | S_{t=2} = s^*) * Pr(S_{t=2} = s^*)$

$Pr(S_{t=3} = s) = \sum_{s^* \in S, a^* \in A} Pr(S_{t=3} = s | S_{t=2} = s^*, A_{t=2} = a^*) * Pr(A_{t=2} = a^* | S_{t=2} = s^*) * \sum_{s^{**} \in S, a^{**} \in A} Pr(S_{t=2} = s^* | S_{t=1} = s^{**}, A_{t=1} = a^{**}) * Pr(S_{t=1} = s^{**}, A_{t=1} = a^{**})$

$Pr(S_{t=3} = s) = \sum_{s^* \in S, a^* \in A} Pr(S_{t=3} = s | S_{t=2} = s^*, A_{t=2} = a^*) * Pr(A_{t=2} = a^* | S_{t=2} = s^*) * \sum_{s^{**} \in S, a^{**} \in A} Pr(S_{t=2} = s^* | S_{t=1} = s^{**}, A_{t=1} = a^{**}) * Pr(A_{t=1} = a^{**} | S_{t=1} = s^{**}) * Pr(S_{t=1} = s^{**})$

$Pr(S_{t=3} = s) = \sum_{s^* \in S, a^* \in A} Pr(S_{t=3} = s | S_{t=2} = s^*, A_{t=2} = a^*) * Pr(A_{t=2} = a^* | S_{t=2} = s^*) * \sum_{s^{**} \in S, a^{**} \in A} Pr(S_{t=2} = s^* | S_{t=1} = s^{**}, A_{t=1} = a^{**}) * Pr(A_{t=1} = a^{**} | S_{t=1} = s^{**}) * \sum_{s^{***} \in S, a^{***} \in A} Pr(S_{t=1} = s^{**} | S_{t=0} = s^{***}, A_{t=0} = a^{***}) * Pr(S_{t=0} = s^{***}, A_{t=0} = a^{***})$

$Pr(S_{t=3} = s) = \sum_{s^* \in S, a^* \in A} Pr(S_{t=3} = s | S_{t=2} = s^*, A_{t=2} = a^*) * Pr(A_{t=2} = a^* | S_{t=2} = s^*) * \sum_{s^{**} \in S, a^{**} \in A} Pr(S_{t=2} = s^* | S_{t=1} = s^{**}, A_{t=1} = a^{**}) * Pr(A_{t=1} = a^{**} | S_{t=1} = s^{**}) * \sum_{s^{***} \in S, a^{***} \in A} Pr(S_{t=1} = s^{**} | S_{t=0} = s^{***}, A_{t=0} = a^{***}) * Pr(A_{t=0} = a^{***} | S_{t=0} = s^{***}) * Pr(S_{t=0} = s^{***})$

$Pr(S_{t=3} = s) = \sum_{s^* \in S, a^* \in A} Pr(s^*, a^*, s) * \pi(s^*, a^*) * \sum_{s^{**} \in S, a^{**} \in A}$
$Pr(s^{**}, a^{**}, s^*) * \pi(s^{**}, a^{**}) * \sum_{s^{***} \in S, a^{***} \in A} Pr(s^{***}, a^{***}, s^{**}) * \pi(s^{***}, a^{***}) * d_0(s^{***})$

Similarly $Pr(S_{t=3} = s^{'})) = \sum_{s^* \in S, a^* \in A} Pr(s^*, a^*, s^{'}) * \pi(s^*, a^*) * \sum_{s^{**} \in S, a^{**} \in A} Pr(s^{**}, a^{**}, s^*) * \pi(s^{**}, a^{**}) * \sum_{s^{***} \in S, a^{***} \in A} Pr(s^{***}, a^{***}, s^{**}) * \pi(s^{***}, a^{***}) * d_0(s^{***})$

We add the above two to get the required

Question 1b. $P(A_{t=16} = a^{'} | A_{t=15} = a, S_{t=14} = s) = \sum_{s^* \in S} P(A_{t=16} = a^{'} | A_{t=15} = a, S_{t=16} = s^*, S_{t=14} = s) * P(S_{t=16} = s^* | A_{t=15} = a, S_{t=14} = s)$

We know that $P(A_{t=16} = a^{'} | A_{t=15} = a, S_{t=16} = s^*, S_{t=14} = s) = P(A_{t=16} = a^{'} | S_{t=16} = s^*)$

So $P(A_{t=16} = a^{'} | A_{t=15} = a, S_{t=14} = s) = \sum_{s^* \in S} P(A_{t=16} = a^{'} | S_{t=16} = s^*) * P(S_{t=16} = s^* | A_{t=15} = a, S_{t=14} = s)$

$P(A_{t=16} = a^{'} | A_{t=15} = a, S_{t=14} = s) = \sum_{s^* \in S} P(A_{t=16} = a^{'} | S_{t=16} = s^*) * \sum_{s^{**} \in S} P(S_{t=16} = s^* | A_{t=15} = a, S_{t=15} = s^{**}, S_{t=14} = s) * P(S_{t=15} = s^{**} | S_{t=14} = s, A_{t=15} = a)$

$P(A_{t=16} = a^{'}|A_{t=15} = a, S_{t=14} = s) = \sum_{s^* \in S} P(A_{t=16} = a^{'}|S_{t=16} = s^*) * \sum_{s^{**} \in S} P(S_{t=16} = s^*|A_{t=15} = a, S_{t=15} = s^{**}, S_{t=14} = s) * P(S_{t=15} = s^{**}|S_{t=14} = s)$

$P(A_{t=16} = a^{'}|A_{t=15} = a, S_{t=14} = s) = \sum_{s^* \in S} P(A_{t=16} = a^{'}|S_{t=16} = s^*) * \sum_{s^{**} \in S} P(S_{t=16} = s^*|A_{t=15} = a, S_{t=15} = s^{**}) * \sum_{a^* \in A} P(S_{t=15} = s^{**}|A_{t=14} = a^*, S_{t=14} = s) * P(A_{t=14} = a^*|S_{t=14} = s)$

$P(A_{t=16} = a^{'}|A_{t=15} = a, S_{t=14} = s) = \sum_{s^* \in S} P(A_{t=16} = a^{'}|S_{t=16} = s^*) * \sum_{s^{**} \in S} P(S_{t=16} = s^*|A_{t=15} = a, S_{t=15} = s^{**}) * \sum_{a^* \in A} P(S_{t=15} = s^{**}|A_{t=14} = a^*, S_{t=14} = s) * \pi(s, a^*)$

$P(A_{t=16} = a^{'}|A_{t=15} = a, S_{t=14} = s) = \sum_{s^* \in S} \pi(s^*, a^{'}) * \sum_{s^{**} \in S} P(s^{**}, a, s^*) * \sum_{a^* \in A} P(s, a^*, s^{**}) * \pi(s, a^*)$

Question 1c. Expected Reward at time t=6 would just depend on the previous state. $Exp(Reward_6|S_5) = \sum_{s_6 \in S} Pr(s_6|S_5) * \sum_{a_6 \in A} \pi(s_6, a_6) * Pr(s_7|s_6, a_6) * R(s_6, a_6, s_7)$

We can say that $Pr(s_6|S_5) = \sum_{a_5 \in A} \pi(s_5, a_5) * Pr(s_6|s_5, a_5)$

$Exp(Reward_6|S_5) = \sum_{s_6 \in S} \sum_{a_5 \in A} \pi(s_5, a_5) * Pr(s_6|s_5, a_5) * \sum_{a_6 \in A} \pi(s_6, a_6) * Pr(s_7|s_6, a_6) * R(s_6, a_6, s_7)$

$Exp(Reward_6|S_5) = \sum_{s_6 \in S} \sum_{a_5 \in A} \pi(s_5, a_5) * Pr(s_5, a_5, s_6) * \sum_{a_6 \in A} \pi(s_6, a_6) * Pr(s_6, a_6, s_7) * R(s_6, a_6, s_7)$

Question 1d. $Pr(s_0 = s|s_1 = s^{'}) = \frac{Pr(s_0=s, s_1=s^{'})}{Pr(s_1=s^{'})}$

$Pr(s_0 = s|s_1 = s^{'}) = \frac{Pr(s_0=s, s_1=s^{'})}{Pr(s_1=s^{'})}$

$Pr(s_0 = s|s_1 = s^{'}) = \frac{Pr(s_1=s^{'}|s_0=s) * Pr(s_0=s)}{Pr(s_1=s^{'})}$

$Pr(s_0 = s|s_1 = s^{'}) = (\frac{\sum_{a_0 \in A} \pi(s_0, a_0) * P(s_0, a_0, s_1)) * Pr(s_0=s)}{Pr(s_1=s^{'})}$

$Pr(s_0 = s|s_1 = s^{'}) = (\frac{\sum_{a_0 \in A} \pi(s_0, a_0) * P(s_0, a_0, s_1)) * d_0(s_0=s)}{Pr(s_1=s^{'})}$

We know that $Pr(s_1 = s^{'}) = \sum_{a^* \in A, s^* \in S} Pr(s_1 = s^{'}|s_0 = s^*, a_0 = a^*) * \pi(a^*|s^*) d_0(s^*)$

So

$Pr(s_0 = s|s_1 = s^{'}) = (\frac{\sum_{a_0 \in A} \pi(s_0, a_0) * P(s_0, a_0, s_1)) * d_0(s_0=s)}{\sum_{a^* \in A, s^* \in S} Pr(s^*, a^*, s^{'}) * \pi(a^*|s^*) d_0(s^*)}$

Question 1e. Given the present it doesn't depend on the past.

$Pr(A_{t=5} = a|s_{t=0} = s, s_{t=5} = s^{'}, A_{t=6} = a^{'}) = Pr(A_{t=5} = a|s_{t=5} = s^{'}, A_{t=6} = a^{'})$

$Pr(A_5 = a|s_5 = s^{'}, A_6 = a^{'}) = \frac{Pr(a_5, s_5|a_6)}{Pr(s_5|a_6)}$

$Pr(A_5 = a|s_5 = s^{'}, A_6 = a^{'}) = \frac{Pr(a_5, s_5|a_6)}{Pr(s_5|a_6)}$

$Pr(A_5 = a|s_5 = s^{'}, A_6 = a^{'}) = \frac{Pr(a_5, s_5|a_6)}{Pr(s_5|a_6)}$

Let us just take

$Pr(s_5|a_6) = \frac{Pr(s_5, a_6)}{Pr(a_6)}$

$Pr(s_5|a_6) = \frac{Pr(a_6|s_5) Pr(s_5)}{Pr(a_6)}$

$Pr(s_5|a_6) = \frac{\sum_{s_6 \in S} Pr(a_6|s_6, s_5) Pr(s_6|s_5) Pr(s_5)}{Pr(a_6)}$

$Pr(s_5|a_6) = \frac{\sum_{s_6 \in S} Pr(a_6|s_6) \sum_{a_5 \in A} Pr(s_6|a_5, s_5) Pr(a_5|s_5) Pr(s_5)}{Pr(a_6)}$

$Pr(s_5|a_6) = \frac{\sum_{s_6 \in S} \pi(s_6, a_6) \sum_{a_5 \in A} P(s_5, a_5, s_6) \pi(s_5, a_5) Pr(s_5)}{Pr(a_6)}$

Let us take

$Pr(a_5, s_5|a_6) = \frac{Pr(a_5, s_5, a_6)}{Pr(a_6)}$

$Pr(a_5, s_5|a_6) = \frac{Pr(a_6|s_5, a_5) Pr(a_5, s_5)}{Pr(a_6)}$

$Pr(a_5, s_5|a_6) = \frac{Pr(a_6|s_5, a_5) Pr(a_5, s_5)}{Pr(a_6)}$

$Pr(a_5, s_5|a_6) = \frac{\sum_{s_6} Pr(a_6|s_6) Pr(s_6|s_5, a_5) Pr(a_5, s_5)}{Pr(a_6)}$

$Pr(a_5, s_5|a_6) = \frac{\sum_{s_6} Pr(a_6|s_6) Pr(s_6|s_5, a_5) Pr(a_5|s_5) Pr(s_5)}{Pr(a_6)}$

So in final,

$$Pr(A_5 = a|s_5 = s^{'}, A_6 = a^{'}) = \frac{\frac{\sum_{s_6} Pr(a_6|s_6)Pr(s_6|s_5,a_5)Pr(a_5|s_5)Pr(s_5)}{Pr(a_6)}}{\frac{\sum_{s_6 \in S} \pi(s_6,a_6) \sum_{a_5 \in A} P(s_5,a_5,s_6)\pi(s_5,a_5)Pr(s_5)}{Pr(a_6)}}$$

$$Pr(A_5 = a|s_5 = s^{'}, A_6 = a^{'}) = \frac{\sum_{s_6} Pr(a_6|s_6)Pr(s_6|s_5,a_5)Pr(a_5|s_5)}{\sum_{s_6 \in S} \pi(s_6,a_6) \sum_{a_5 \in A} P(s_5,a_5,s_6)\pi(s_5,a_5)}$$

$$Pr(A_5 = a|s_5 = s^{'}, A_6 = a^{'}) = \frac{\sum_{s_6} Pr(a^{'}|s_6)Pr(s_6|s^{'},a_5)Pr(a|s^{'})}{\sum_{s_6 \in S} \pi(s_6,a^{'}) \sum_{a_5 \in A} P(s^{'},a_5,s_6)\pi(s^{'},a_5)}$$

$$Pr(A_5 = a|s_5 = s^{'}, A_6 = a^{'}) = \frac{\sum_{s_6} \pi(s_6,a^{'})P(s^{'},a_5,s_6)\pi(s^{'},a)}{\sum_{s_6 \in S} \pi(s_6,a^{'}) \sum_{a_5 \in A} P(s^{'},a_5,s_6)\pi(s^{'},a_5)}$$

Question 2. Given there are $|S|$ and $|A|$

In each state you can take $|A|$ actions.

So the number of deterministic policies are $|A|^{|S|}$

Question 3. A variant where the initial angle is chosen uniformly randomly in $[-\pi, \pi]$ and the initial velocity is zero would take a bit more time to converge since the agent might behave differently in the second episode for a fixed policy from the first episode, because of the stochasticity. Estimation of whether a policy is good takes time, since it has to be robust to the different start states.

If it were to start at an angle $\theta \in (-\pi/4, +\pi/4)$ Then the policy will not learn much, it will just learn to be at Top. It might not do well when the start will be different in the next episode.

If it were to start at 0, then on exploring, agent, gets to see that it gains rewards on moving up and when $\theta \in (-\pi/4, +\pi/4)$. So this would require less episodes to solve.

In the deterministic start state, the agents policy or avg reward obtained in a series of episodes would almost remain in the same range for a given policy.

Question 4. **Gridworld** An agent needs 1 episode atleast to converge, since episode only terminates when it reaches the terminal state and receives 10, so it knows that the end goal and has an idea of what states would be bad. It would need more than 1 episodes to find a good policy and converge as well. To get to near optimal policies it needs to have explored through such rewards in the episodes before termination. State space is discrete which makes it easier. My estimate is that it would take 100 episode.

Pendulum

An agent needs to have one episode where it gets the maximum rewards when $\theta \in (-\pi/4, +\pi/4)$ and if time up is 10 seconds or more it is a successful episode. It would need more episodes to find a good policy than the grid world. Since this case can terminate an episode when time reaches 20. To get to near optimal policies it needs to have explored through such rewards in the episodes before termination. State space is continuous which makes it harder.My estimate is that it would take 1000 episode.

Question 5. I have taken the game of Pac man where i have one pac man and one ghost and if the pacman eats the ghost, the ghost can never re-enter. The partial observability lies in the fact that the pacman can not see the ghost unless the ghost is within a radius of distance 2 from pacman. Powerup lasts for 5 seconds

Action Set Action Set = {left, right, down, up, no action}

Reward Reward is 100 if he is on the same position as the ghost while on a powerup, ie pacman eats ghost.

Reward is -100 if he is on the same position as the ghost without a powerup, ie ghost eats pacman.

Reward is 1 if he is on a position with a small point where the ghost is not present.

Reward is 0 if he is on a position with no points.

Reward is 10 if he is on a position with a powerup where the ghost is not present.

Probability - Transition Matrix

$\Pi(s, a)$ = Probability of taking action a in state s = $\{0, 1\}$

It is deterministic in nature.

$p(s^{'}|s, a) \in \{0, 1\}$

$\Pi(s, a)$ = Probability of taking action a in state s = $\{0, 1\}$

State Set

$StateSet = \{PacManPositionothegrid(x, y), Ghost1positionontheGrid(x, y), currentPositionHasPowerUp, currentP$

If Ghost is seen by the Pacman, then the ghost position is the correct position.

If Ghost is not seen by the Pacman, then the position is $(undefined_x, undefined_y)$ stating that the position of the ghost is unknown to pacman.

currentPositionHasPowerUp: is whether the position has a powerup.

currentPositionHasSmallPoint: is whether the position has a small point with reward of 1.

PowerUptime: if pacman gets powerup, PowerUptime is 10 seconds. It gets reduced every time a second is decreased till it reaches zero. Initially when no powerup is present, PowerUptime is 0.

Terminal state Terminal state is when you stay alive for 100 seconds.

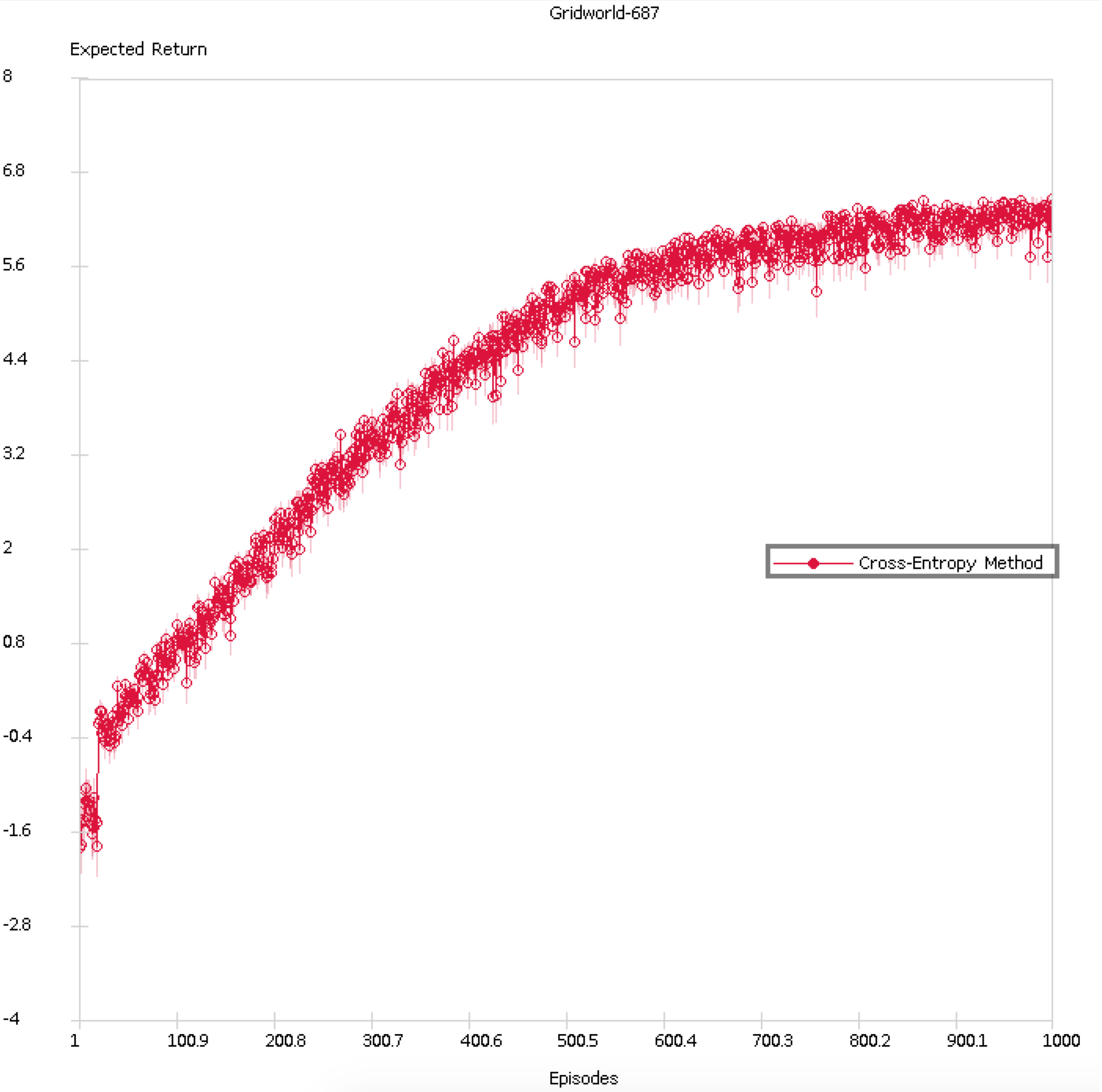
Start state d_0 is the initial state it is always at the left end of the grid map.

Gamma For values of Gamma less than 1 , it should learn, since it has to learn the objective that it has to avoid the Ghost when it sees it and the pacman doesn't have the powerup, to stay alive. Since the major objective is to stay alive for 100 seconds, it just has to stay till the end to get to a near optimal policy. So it has to take nearby incentives and not look too far in the distant future.

1. Part 2

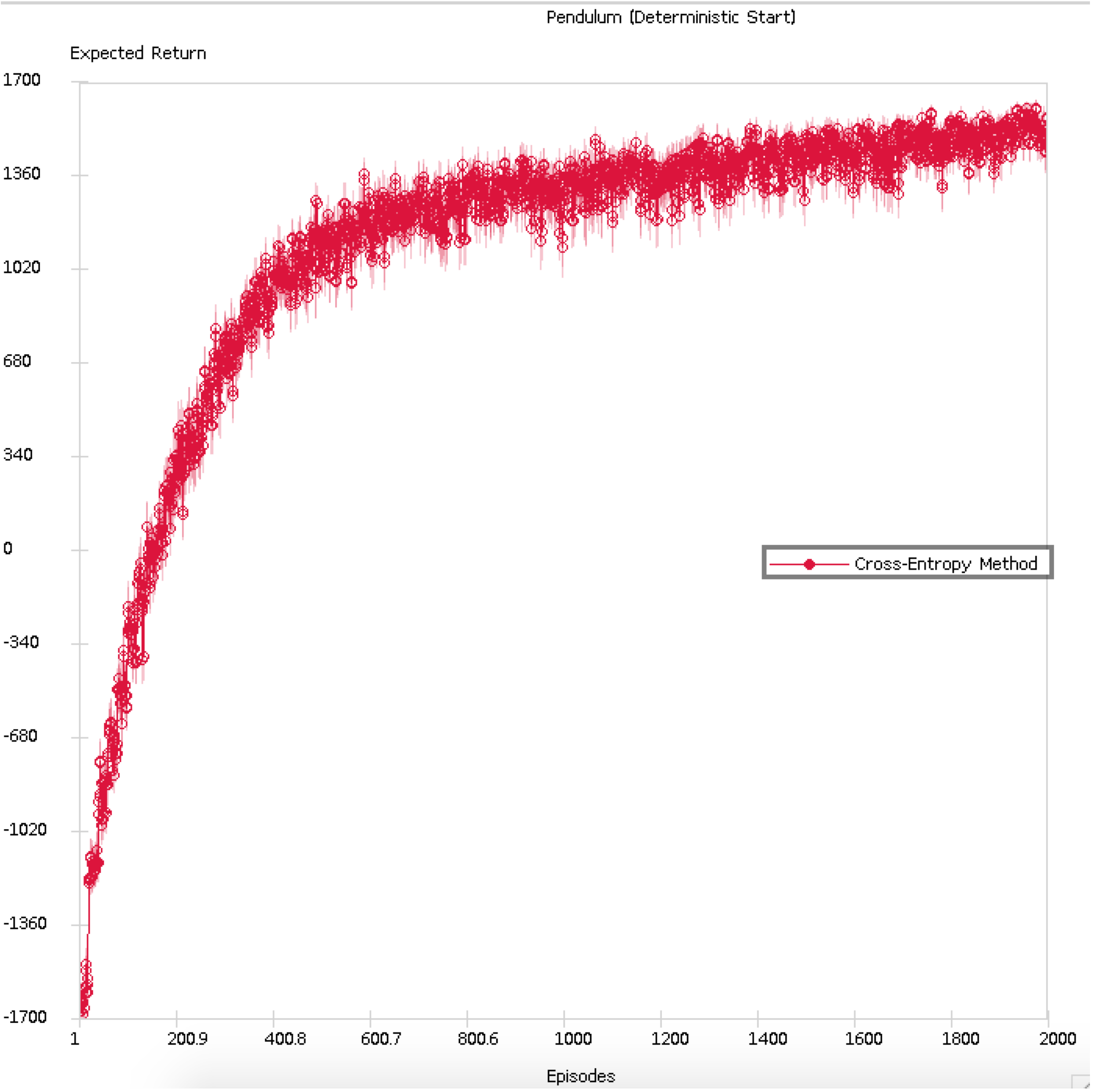
```
theta.setZero();
Sigma = MatrixXd::Zero(numParams, numParams);
MatrixXd Update = MatrixXd::Identity(numParams, numParams);
double epsilon = 10;
for(int i=0;i<numElite;i++){
    theta += thetas[i];
}
theta = theta/numElite;

for(int i=0;i<numElite;i++) {
    Sigma += (thetas[i] - theta) * ((thetas[i] - theta).transpose());
}
Sigma += Update * epsilon;
Sigma = Sigma/(numElite+epsilon);
```

Grid World. :



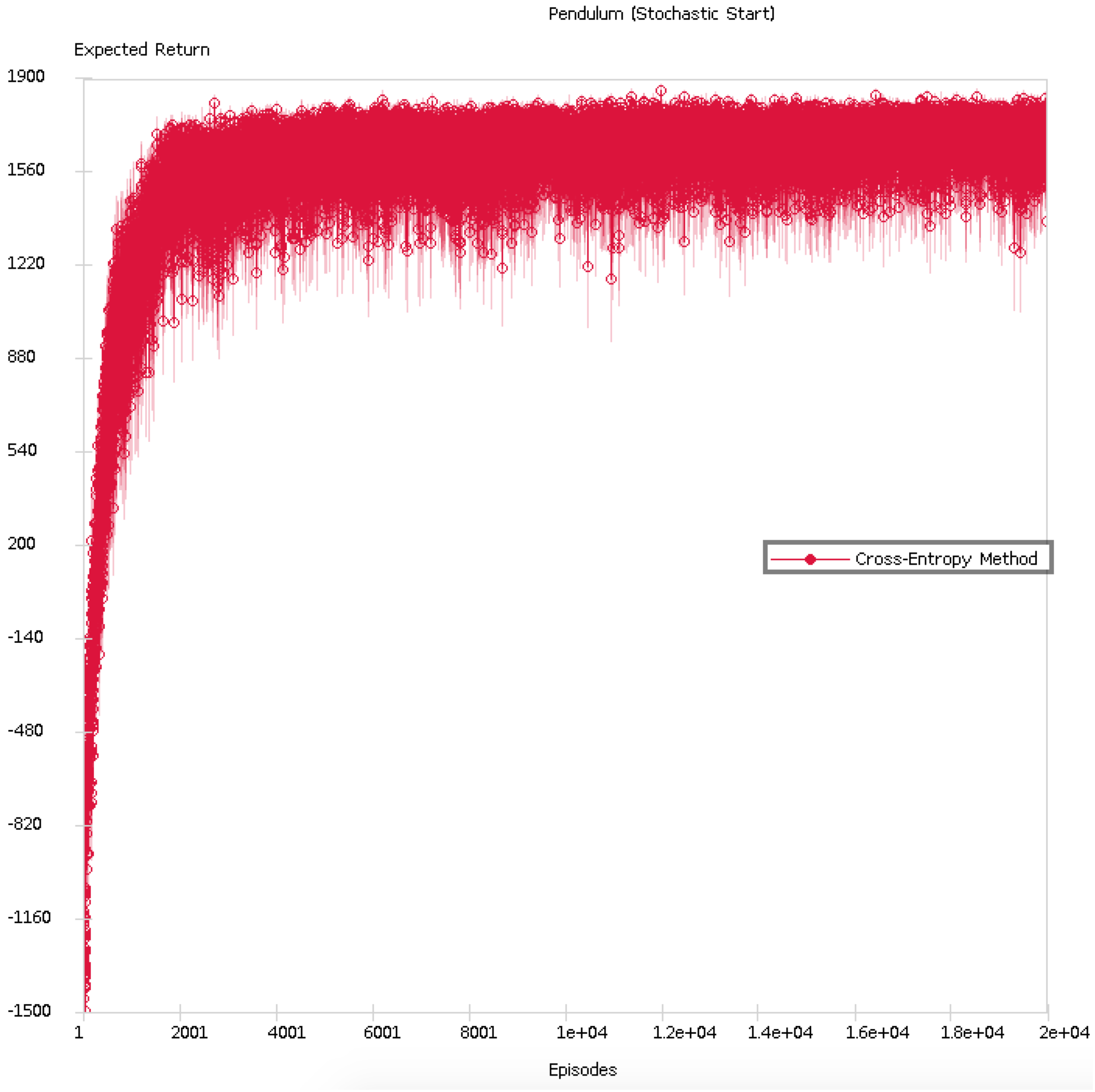
Hyper Parameters:
Enter number of trials (integer): 1000
population (CEM Hyperparameter, integer): 10
numElite (CEM Hyperparameter, integer): 3
episodesPerPolicy (CEM Hyperparameter, integer): 2
sigma (CEM Hyperparameter, real): 10

Pendulum Deterministic Start. :



Hyper Parameters:
Enter number of trials (integer): 1000
population (CEM Hyperparameter, integer): 10
numElite (CEM Hyperparameter, integer): 3
episodesPerPolicy (CEM Hyperparameter, integer): 2
sigma (CEM Hyperparameter, real): 5

Pendulum Stochastic Start. :



number of trials (integer): 30
population (CEM Hyperparameter, integer): 10
numElite (CEM Hyperparameter, integer): 3
episodesPerPolicy (CEM Hyperparameter, integer): 6
sigma (CEM Hyperparameter, real): 10

Experience Info. :

Wall Time to get Working. : Clion build on Mac took almost 4-5 hours for me to setup without any issues. I spent some time understanding the skeleton code supporting the CEM update.

Agent Lifetime. : I would have run the agent atleast 30 times in each domains with small number of trials. For the full trial limit times of 1000, 100, 30 for each domain, i would have run it for 10-20 times for each domain.

Brute Force. : I feel brute force at different ranges of the hyperparameters would prove effective, But Brute Force over random policies would be like a hill climbing method. It might take a lot of time to get to near effective policies given the number of policies available is infinite.

I think it was easier to have used CEM Update instead of randomly searching policies since there are an infinite number of policies.

Gridworld and Pendulum. : The answer in Q3 and Q4 don't accurately match my answer. GridWorld converged faster than the pendulum. The reason for both the domains to take more than my estimate was probably because , the in pendulum, the episodes needed to understand the dynamics of momentum could be more than 1000. Also sometimes in pendulum, it might take some time to search through the big state space in pendulum. Gridworld might take more time to get to a good policy because a combination of good policies , might also result in getting bad decisions at certain states. You can't quite estimate the value of taking the mean of the policies to prove to be good in all the states.

Hard Domain. : Stochastic Start Pendulum is difficult, because parameters were difficult to find and they varied, as the expected reward across different episodes were different for a fixed policy. We need to estimate the policy for atleast a few episodes to be sure that it is a good policy.

Applications. : Cross Entropy Method is good for applications where we need to sample from a huge space of continuous-domain policies. It might not work well for deterministic policies, and other algorithms might work better for that. When start state is fixed you get a good convergence with minimum deviation in error. When there is some stochasity with start then the deviation of the max Expected Rewards is higher, we need to average multiple episodes per policy.