# HW4

Aaron Elliot

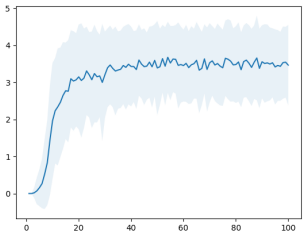
November 2018

## Question 1

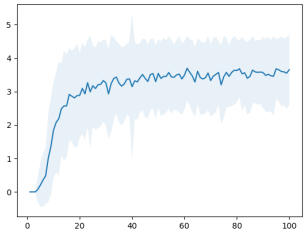
In this question, we applied Q Learning and SARSA to CartPole and Gridworld. I found that over 100 trials of 100 episodes, both algorithms on converged to optimal or decent policies for GridWorld and CartPole respectively.

We had control of four hyper-parameters: how we normalize the state, the policy, $\alpha$, and basis dimension for the Fourier basis. I tuned these hyper-parameters manually. Testing both 3 and 5 dimensions for the Fourier basis on CartPole, I found both training and learning to be faster with 3. Further, I normalized the state to $[0, 1]$. As well, I found that $\alpha$ on the order of $1e-3$ gave the best results. Lastly, for policy I chose epsilon greedy, where $\varepsilon = \frac{1}{n}$, where $n$ is the number of episodes seen in this trial. I considered scaling down $\alpha$ with the number of episodes, but my method converged well with a constant $\alpha$.

The learning curves for SARSA and Q-Learning on the GridWorld and Cart-Pole domains can be seen in figures 1, 2. For these figures, the y-axis is the discounted return, and the x-axis is the number of episodes trained on. The blue line is the mean discounted return observed over 100 trials, and the grey region represents one-standard deviation away from the mean.

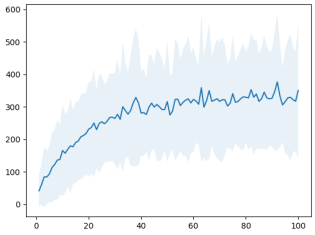



(a) SARSA-Tabular

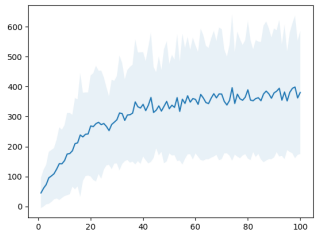



(b) Q-Learning Tabular

Figure 1: GridWorld Domain




(a) Sarsa with Fourier Basis




(b) Q-Learning with Fourier Basis

Figure 2: CartPole Domain

# Question 2

As we did in question 1, we are testing Q-Learning and SARSA on the Cartpole domain. Except, instead of using the Fourier basis, we are using the polynomial basis.

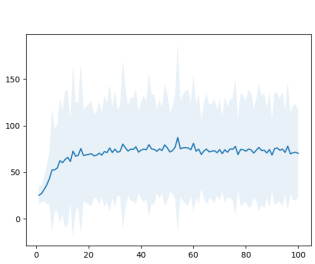
Just as above, we can tune how we normalize, the policy, $\alpha$, and basis dimension for the Polynomial basis. I tested a large range of values for the hyper-parameters. However, I received the best results when I normalized the state to $[-1, 1]$, with 5 dimensional polynomial basis. However, even with my best choice of hyper-parameters the mean discounted return only reached just above 50. As such, I came to the conclusion that this is the best we can expect from this choice of basis. As the literature would predict, our weights converge close to $w_\infty$, where $w_\infty$ is the optimal weights for our basis (Tsitsiklis and Van Roy, 1997, Theorem 1).

The learning curves for this question can be seen in figure 3. In each of these, the y-axis is the discounted return, and the x-axis is the number of episodes trained on. The blue line is the mean discounted return observed over 100 trials, and the grey region represents one-standard deviation away from the mean.

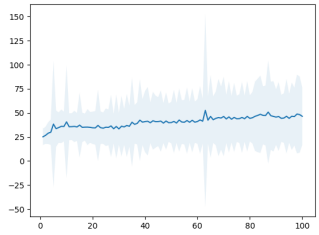
# Question 3

Reflecting back on CEM. I found CEM harder to tune than Q or SARSA. I guess this is because CEM had significantly more hyper parameters to tune and at the time I was naive to tuning them. Further, CEM took vastly more episodes to learn. My implementation of CEM took $\approx 600,000$ episodes, while Q and SARSA require less than 100.

I would expect CEM to perform well in any MDP where the genetic algo-




(a) SARSA with polynomial basis Basis

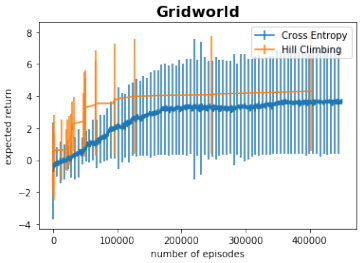



(b) Q-Learning with Polynomial

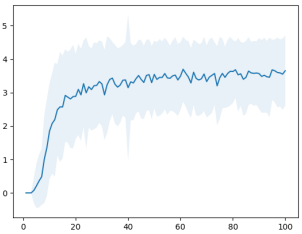
Figure 3: CartPole Domain

rithm like search is most efficient; e.g. where there is only a small region of weights with good outcomes and finding that region is hard without "jumping" to it. If the MDP results in a more of a smooth surface for $J$, then I'd except SARSA and Q-Learning to be superior.

As keeping my laptop from my siblings in my mom's tiny house is not reasonable, I cannot keep my laptop running CEM for 10 hours over the holiday. As such, I think it is reasonable to simply include the graphic from HW2 along-side the new graphic. This can be seen in figures 5 and 4.




(a) CEM & Best-First

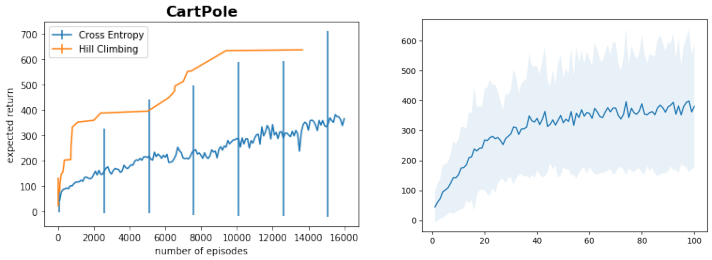



(b) Q-Learning Tabular

Figure 4: GridWorld Domain

# Question 4

Implementing softmax action selection for the Gridworld domain was quite simple. I found this change naturally fit the hyper-parameters already used in Question 1. As softmax picks uniformly random for very small $\sigma$, and is greedy for large $\sigma$, having $\sigma$ scale with $n$ made sense. So, I tested $\sigma$ as proportional to $n$, where $n$ is the number of episodes seen so far. I tested, $\sigma = n, .5n, .1n, .2n$; I

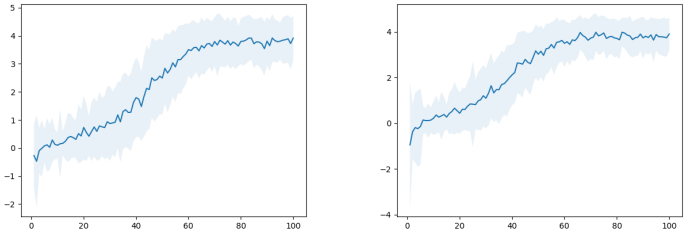



(a) CEM & Best-First (b) Q-Learning with Fourier Basis

Figure 5: CartPole Domain

found that $\sigma = .2n$ had the best results.

In question 1, we used an epsilon greedy policy. Figure 1 uses a epsilon greedy policy versus figure 6 uses a softmax policy. Comparing the two reveals that a softmax policy led to slower learning at first, but does approach the optimal policy faster in the last stretch. I think this is due to how we scaled $\sigma$ with $n$.



(a) SARSA with softmax policy (b) Q-Learning with softmax

Figure 6: GridWorld Domain

# Question 5

Not attempted