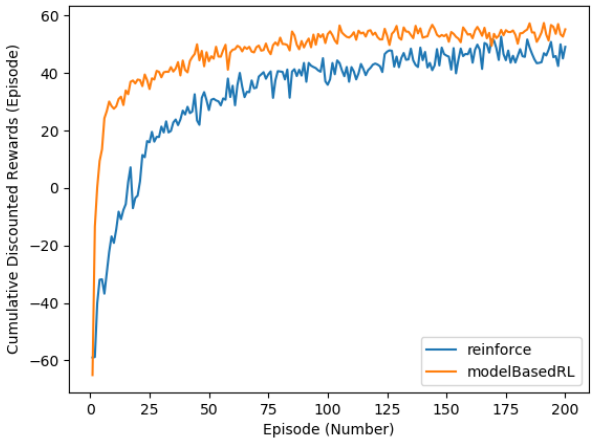
CS 885 Assignment Two

Name: Connor Raymond Stewart

ID: 20673233

Question One

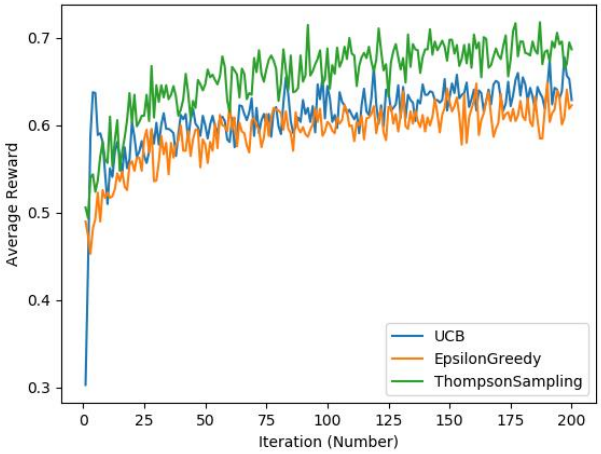
The graphs produced with the question constraints (200 episodes, 100 trials, 100 steps) are below. The reinforce program has an alpha set to 0.01 and the model-based RL with epsilon set to 0.05. We see the following Matplotlib diagram produced for the cumulative discounted rewards per episode when each episode starts in state zero:



:

As can be seen, model-based RL has the highest average reward. However, both values are similar. Since the epsilon value in model-based RL is set to 0.05 and is set to [i / (number iterations)] in reinforce, model-based RL tends to have a lower epsilon value throughout the execution of the program then reinforce, and therefore it needs more time to explore the model then the reinforce algorithm. Furthermore, model-based RL achieves a higher reward than the reinforce algorithm. The reinforce algorithm has lower rewards than the model-based RL algorithm because the selection of actions is random (and therefore stochastic), and the reinforce algorithm has a high variance due to it having a fixed learning rate (alpha). Increasing the variance in the data skews the data to higher rewards, and stochastic action selection means optimal actions cannot always be guaranteed selection.

The graphs produced with the question constraints (200 iterations, 1000 trials) are provided below. For the Epsilon-greedy bandit algorithm, epsilon is set to one divided by the number of iterations, and for the Thompson sampling algorithm, k=1 and prior consists of Beta distributions with all hyperparameters set to 1. We see the following Matplotlib diagram produced as output for the reward earned at each iteration:

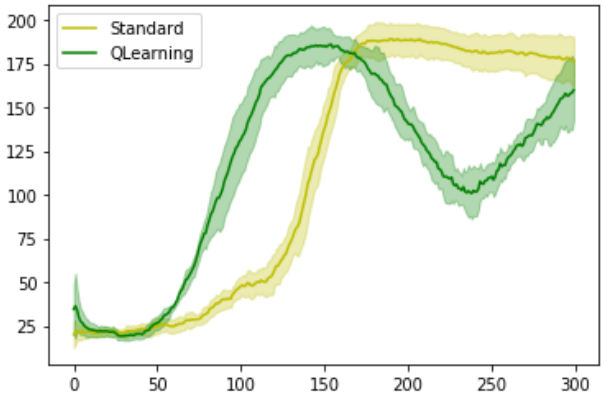


Please note that in the above diagram, Iteration (x-axis) refers to the Episode number.

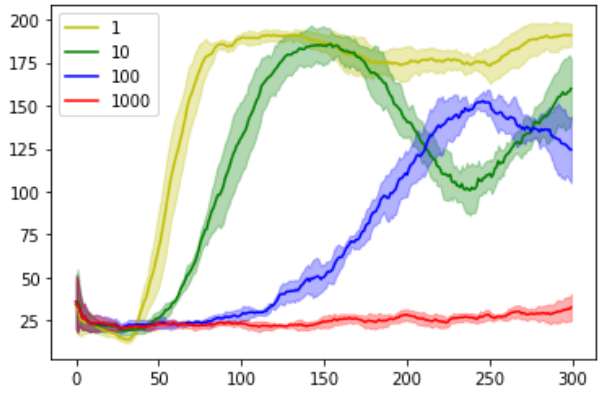
It is easy to see from the above graph that ThompsonSampling has the highest average rewards. Since k is set to one, much exploration occurs during the initial iterations, increasing the rewards near the end of the run. It is essential to let exploration occur since exploration decreases from the increased complexity of the distribution as the program runs. The UBC algorithm performs slightly better than the greedy epsilon algorithm. Since the epsilon greedy method is based on random probabilities, it is stochastic and cannot guarantee the best possible solution. The epsilon greedy method is simple, easy to implement, and fast but has a lower average reward. The UCB method is more accurate since it assumes that the environment is as optimal as possible, meaning that the chance of the program getting lucky and striking optimal rewards is higher than the epsilon greedy method. Playing for luck gives the UCB method a higher probability than the epsilon greedy method since it does not take advantage of its randomness like the UCB method. Ultimately, both the UCB and the epsilon greedy methods have lower rewards than the ThompsonSampling method since they are random, unlike the k exploration reduces non-optimal rewards from exploration early in program execution.

Question Two

DQN vs Soft Q-Learning:



Soft Q-Learning on Cartpole with four temperature values:



Explanation:

The DQN program works to find the maximum possible action if a randomly generated value is less then the lambda, otherwise it randomly selects an action from the set of possible actions. A DQN is a reinforcement learning algorithm that allows us to find the maximum rewarding action per a given state. The QLearning policy works by using epsilon exploration and Boltzmann exploration to sample random actions with probability epsilons and preforming Boltzmann's exploration otherwise. With the QLearning policy we can see that the pole is able to temporarily resume to its previous state (as can be seen by the trough in the graph) and swing. The QLearning policy is slower then the standard policy as it executes extra steps due to using Boltzmann's exploration. It is more accurate then standard DQN since it factors in Boltzmann's exploration to find optimal samples, whereas the standard DQN simply finds the maximum arguments. Therefore, the standard DQN is faster but less accurate then the QLearning model.

The optimal temperature value appears to be 1 or 10. Temperature values of one closely resemble the default DQN networks results with the pole drop sharply increasing in performance followed by little change. The temperature of ten shows a drop but has a sharp recovery near the end of the experiment. At temperature 100, the program is very slow to optimize for the drop, however it does see a growth near the end, however the

preformence is still very poor. At temperature 1000, the pole drop program fails to balance anything, and the performance is very poor up to the end. A temperature that is too high overcomplicates the model and makes sampling for better values very difficult for the program. If the temperature is too low, then the program cannot try and find new solutions, meaning the range of possible samples is poor and the program can only optimize so much. The correct temperature balance is then in the middle of the two extremes.