CPEN 502 Assignment Part 2 – Reinforcement Learning (LUT)

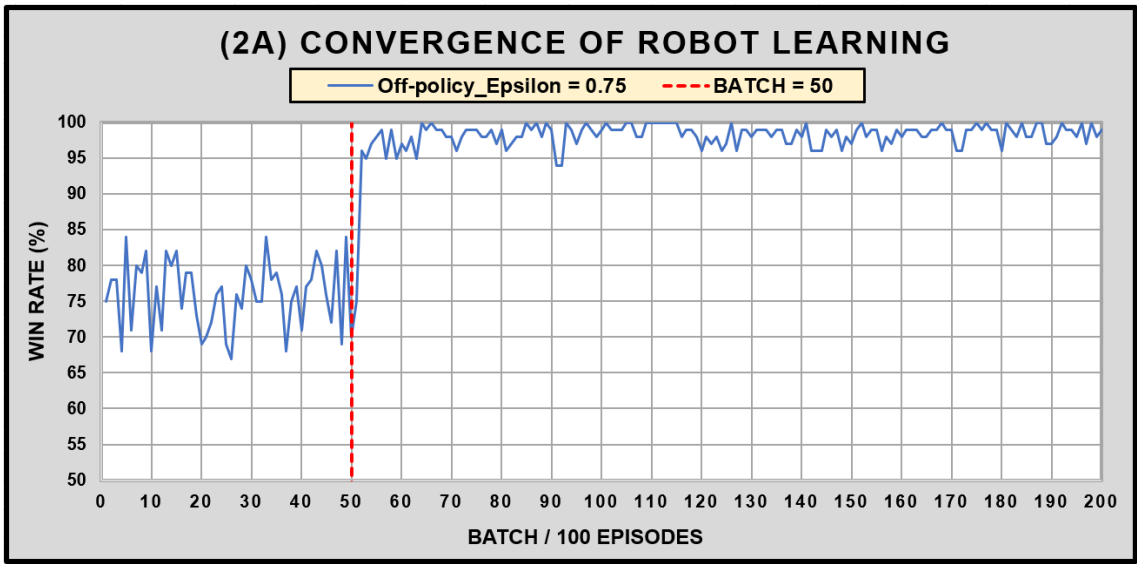
Student Name: Chao-Wu Chu
Student Number: 85406312

(2) Once you have your robot working, measure its learning performance as follows:

As for part 1, your submission should be a brief document clearly showing the graphs requested about. Please number your graphs as above and also include in your report an appendix section containing your source code.

(a) Draw a graph of a parameter that reflects a measure of progress of learning and comment on the convergence of learning of your robot.

Answer:



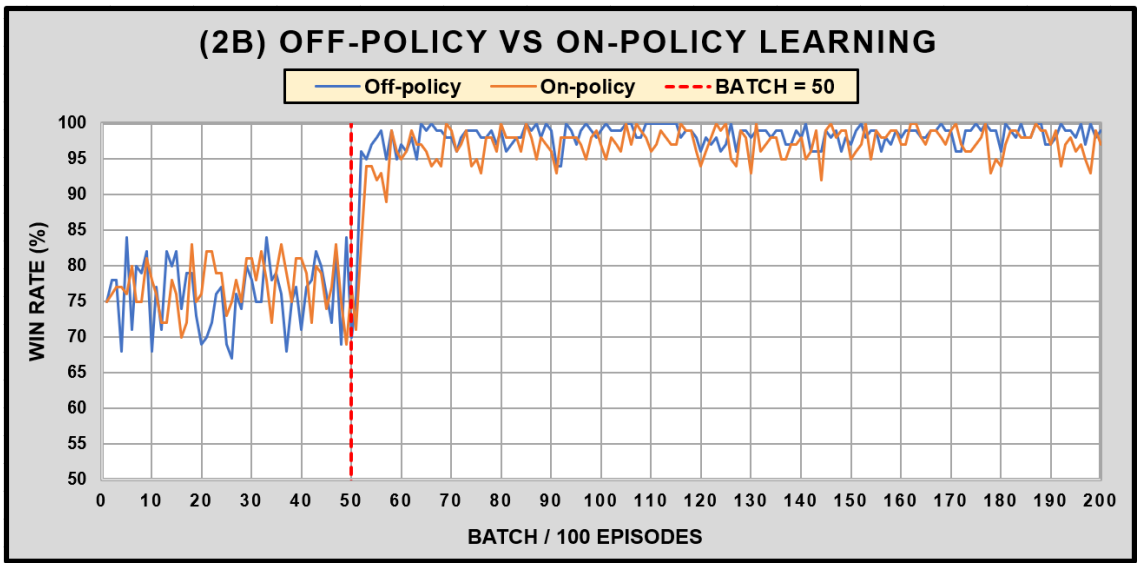
In this training set, I chose the Tracker robot as the opponent for my robot. To prevent problems caused by learning too quickly or slowly, I set the learning rate (α) to 0.5 to maintain a moderate overall learning speed. Additionally, I set the discount rate (γ) to 0.8, hoping that when updating the Q-value, the robot would primarily focus on the expected Q-value of future actions, rather than overly leaning towards rewards, to minimize distortion in the overall Q-learning process.

I structured the training in batches, with 100 episodes per batch (round). After every 100 episodes, I recorded the win rate. Initially, I set the epsilon (ε) value to 0.75, aiming to allow my robot more opportunities to explore new action combinations and discover the Tracker robot's weaknesses. After continuing exploration for 50 batches (5000 episodes), I set epsilon to 0 and proceeded with 150 batches (15000 episodes) of evaluation. The training and evaluation were conducted over a total of 200 batches (20000 episodes).

From the results, it is evident that my robot's initial win rate was around 75%. After undergoing Q-learning, the robot's win rate improved to approximately 97% during the evaluation phase and stabilized at this value. My robot indeed learned how to consistently defeat the Tracker robot.

(b) Using your robot, show a graph comparing the performance of your robot using on-policy learning vs off-policy learning.

Answer:



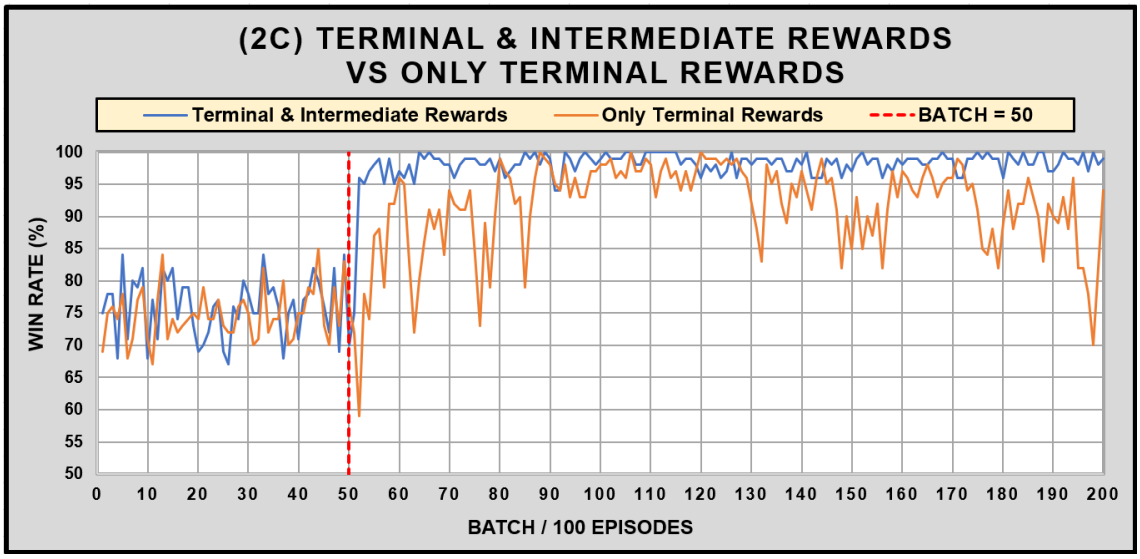
In both the off-policy and on-policy training sets, the opponent remained the Tracker robot, with all other hyperparameters set identically to those in Question 1.

From the graph, it's observable that, after training, while the on-policy's converged value post 50 batches during the evaluation phase (where epsilon = 0) was slightly lower than that of off-policy, overall, my robot showed no significant difference when following off-policy or on-policy strategies. Furthermore, in both off-policy and on-policy training, the top 10 most visited states and action combinations shared up to 8 similarities. Based on this, the action patterns of the robot, after updating the Q-value following the two policies, could have around 70-80% similarity. I have come up with three potential reasons for this observation. These reasons might also provide directions for adjustments if one needs to discern the differences between off-policy and on-policy.

First, a possibility is that the environment is not complex enough. Although I consider the Tracker robot to be a fairly strong opponent within the sample robots, it might still not be challenging enough to allow my robot to find distinctly different learning paths in on-policy and off-policy. A more complex opponent or battlefield (perhaps increasing the number of robot tanks on the field) might allow for distinct action patterns to emerge in the exploration phase of on-policy and off-policy. The second possibility is that the state-action space is too simple, leading both on-policy and off-policy to find overly similar combinations, impacting the win rate similarly. Redefining the state-action space into a more complex set could potentially reveal differences. Finally, the influence of rewards is also a factor. As known from the Q-learning formula, Q-value updates can be significantly influenced by large rewards. However, the expected Q-value of future actions is exactly where off-policy and on-policy differ importantly. By adjusting the values and distribution of rewards, I believe we can achieve more distinct outcomes in off-policy and on-policy learning.

(c) Implement a version of your robot that assumes only terminal rewards and show & compare its behaviour with one having intermediate rewards.

Answer:



In these two training sets, the opponent was still the Tracker robot, with all other hyperparameters set as in Question 1.

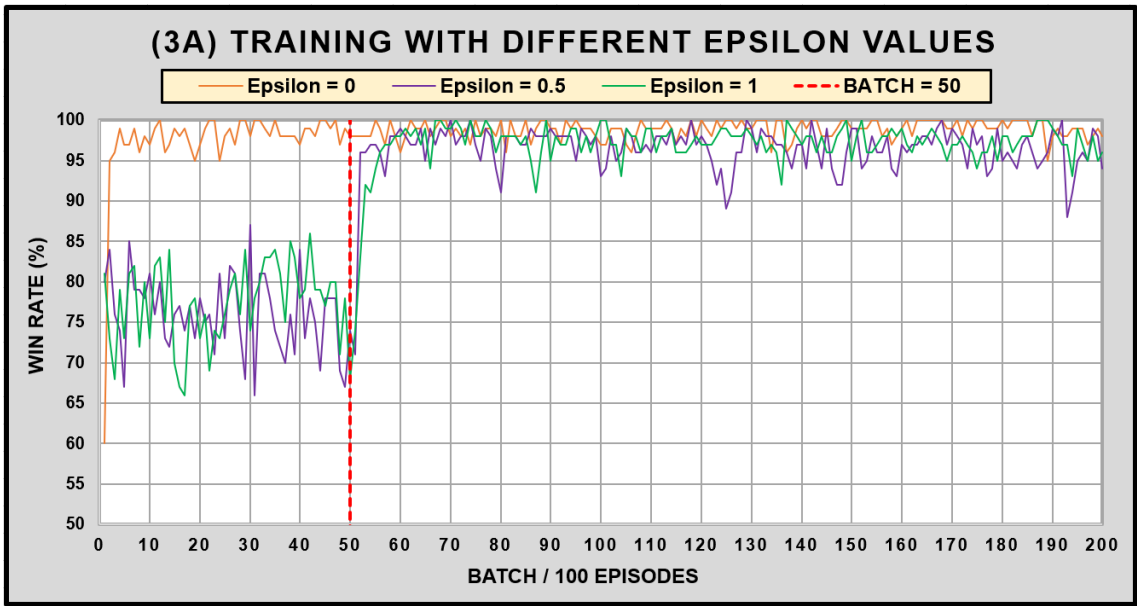
From the graph, it is clear that the training results with only terminal rewards were significantly inferior to those with both terminal and intermediate rewards. With only terminal rewards, the win rate during the evaluation phase (after 50 batches) was very unstable, mostly fluctuating between 80-85%. In contrast, training that included both terminal and intermediate rewards resulted in a more stable win rate, consistently maintaining over 95% during the evaluation phase.

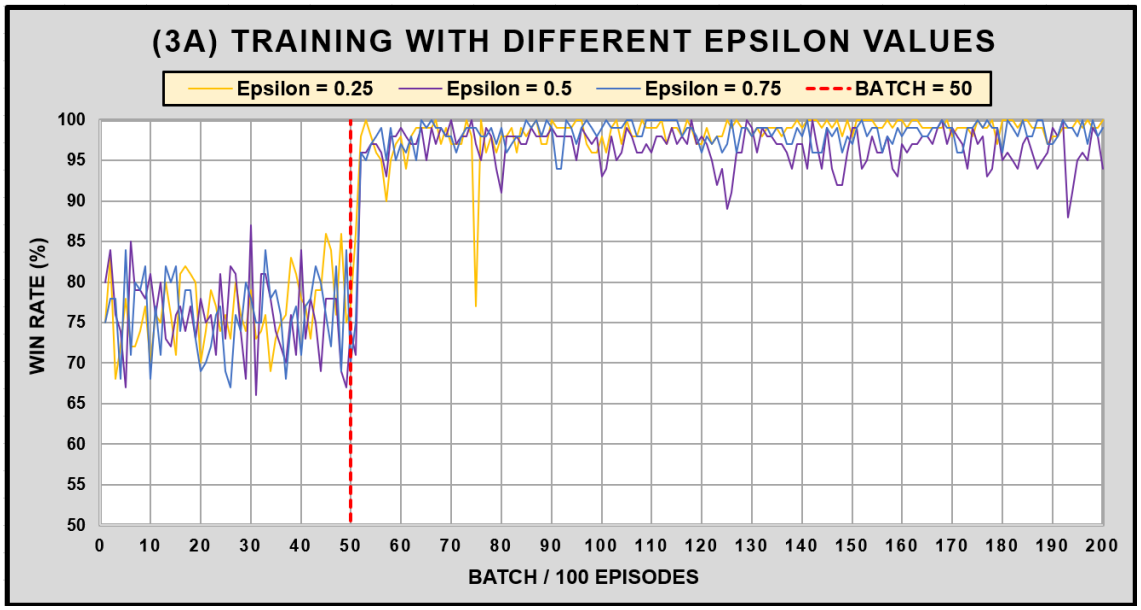
I allocated intermediate rewards to three scenarios: when my robot hit the opponent, when my robot was hit by the opponent, and when my robot collided with a boundary. Achieving the first condition earned a positive intermediate reward, while the latter two conditions resulted in negative intermediate rewards. By defining these positive and negative events and assigning intermediate rewards, my robot could more rapidly identify key actions after each Q-value update, ultimately finding a stable method to consistently defeat the opponent.

(3) This part is about exploration. While training via RL, the next move is selected randomly with probability e and greedily with probability 1 – e.

(a) Compare training performance using different values of e including no exploration at all. Provide graphs of the measured performance of your tank vs e.

Answer:





From the results, it is evident that during the training phase, when epsilon equals 0, the win rate can directly reach close to the convergence value of around 95%. Observing my robot's strategy against the Tracker robot, I found that making every effort to shoot at the Tracker robot from the beginning is a key strategy for defeating it. Notably, the only positive intermediate reward is assigned to hitting the opponent. I believe part of the reason is that the rewards encourage my robot to frequently engage in shooting, which coincidentally exploits a major weakness in the Tracker robot's behavior pattern. Being hit early on tends to lead to the Tracker robot's defeat in that episode. This suggests that the rewards and state-action space are very effective against the Tracker robot, allowing us to find the best solution for winning even without exploration.

However, when epsilon has other values, the exploration phase during training shows very similar performance. In the evaluation phase, the graphs and convergence values are almost identical. This could be attributed to an overly simplistic environment (including the opponent's behavior pattern), making it difficult for my robot to find better strategies through exploration. Therefore, the impact of exploration, regardless of the size of epsilon, is much less significant than the rewards and the state-action setup's effectiveness against the Tracker robot. Even with epsilon set to 1 (where all actions are selected randomly), it is still challenging to find exploration strategies that significantly influence the outcome.

In summary, it is not easy to pinpoint specific issues in a round of Q-learning, as many factors can influence the outcome, and they are often interrelated. However, we can still find directions for adjusting parameters from certain clues, such as the visit counts in the state-action table, the robot's behavior patterns, the distribution of rewards, and so on.