Submitted 5/5/2024

Racetrack performance with machine learning

Christopher Lee clee349@jhu.edu

Johns Hopkins University

Abstract

Reinforcement learning allows a policy with the optimal expected reward to be learned. Value iteration can estimate the value of all states, if the models of resulting rewards and states from each state-action pair is known. These values can be used find the optimal policy through policy iteration. If the models of the resulting rewards and states is not known, the optimal policy can still be reached through Q-learning and SARSA. While these methods converge to the optimal policy, they can be computationally expensive. We investigated the performance of value iteration, Q-learning, and SARSA on the racetrack problem, with four tracks. For Q-learning and SARSA, the current action was chosen based on random sampling based on the softmax of the Q-values of each action, with a decaying temperature scaling factor. We found that using a high discount factor is important to reinforcement learning performance on the racetrack problem.

Problem statement

Reinforcement learning allows learning of a policy that maximizes the expected net reward from a system where the reward and next state depend on the previous state and action. Reinforcement learning is different from supervised and unsupervised learning. While the reinforcement learning agent does not receive labeled information, it receives feedback signals that it can use to improve performance of its policy model. Q-learning and SARSA are two model-free reinforcement learning methods, that have been shown to converge to the optimal policy under the assumptions that all state-action pairs are visited infinitely many times, and the learning policy becomes greedy in the limit. The algorithms can be very expensive, therefore, understanding how to improve the speed at which the algorithms reach convergence can be valuable.

The discount factor is an adjustable parameter that may affect the performance of value iteration, Q-learning, and SARSA. The discount factor scales the expected value of the next state (value iteration) or the value of the temporal difference error (q-learning and SARSA), relative to the reward of the current action. In the racetrack problem, the reward of the current action is equal in all states, except for a subset of states near the finish line. Therefore, a high discount factor is important to provide meaningful information to states farther from the finish line. Eligibility traces can also improve the performance of Q-learning and SARSA by updating more states during each trial episode.

In this study, we examine how varying discount and the use of eligibility traces affect the speed of value iteration, q-learning, and SARSA. I hypothesize that a larger (closer to 1) discount and eligibility traces with higher decay coefficients (closer to 1) will improve the speed of the algorithms.

Experimental approach

The reinforcement learning algorithms for learning the racetrack problem are implemented in a Python class named ‘RaceTrack’. Upon initialization, the track text file is opened, and the content

is read and stored in the object. The parameters of the state space are: x (x position), y (y position), v_x (x velocity), v_y (y velocity). The number of states vary among tracks: in different tracks, the state spaces will include different sets of x and y values. The state space of each track includes the starting line and open racetrack positions, but not the finish line and wall positions. The state spaces of all tracks include the same sets of v_x and v_y values: {-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5} for both v_x and v_y.

There are 9 available actions in each of the state spaces. Each action is represented as a 2-component vector, where the first component is the x acceleration and the second component is the y acceleration. The 9 actions are: {[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 0], [1, 1], [1, -1], [1, 0], [1, 1]}.

The class contains three main algorithms: ‘value_iteration’, ‘q_learning’, and ‘sarsa’. There are two additional methods: ‘next_state’ and ‘bresenham’. The ‘next_state’ method finds the reward and next state, given an input state and action. The method checks if the action will result in a collision or finish line crossing, by using the Bresenham line algorithm to determine all positions the race car enters, and determines the reward and next position according to the rules of the racetrack problem.

The q learning and SARSA algorithms include a few tunable parameters, including discount, learning rate and temperature. The discount scales the value of the temporal difference error, while the learning rate scales the value of the reward and discounted temporal difference error. In both algorithms, the current action is randomly chosen based on the softmax of the current Q-values of each state. The Q-values are also scaled by the temperature, in such a way that when the temperature is high, each action is chosen more uniformly, allowing more state-action pairs to be explored. Conversely, when the temperature is low, the actions with higher Q-values are more likely to be chosen, and the action choice is more greedy. The learning rate and temperature are iteratively decayed.

We determined the convergence criterion of value iteration from the difference between value arrays from consecutive iterations. A criterion parameter, epsilon, was compared to the maximum difference. If the maximum difference was smaller than epsilon, the convergence criterion was met. Determining convergence of Q-learning and SARSA was less straightforward, because for each episode, a random subsample of the state space was visited. In addition, while racetrack performance gradually improved over many iterations, the number of moves on each iteration varied largely, and could increase on successive iterations. Because we observed improvements in performance over long periods of iterations, we believe the observed variation of movements on individual iterations reflected the non-determinism of the action selection and acceleration, rather than the learning. However, we observed that the degree of variance decreased as the performance approached the minimum number of moves needed to solve the racetrack. Therefore, we determined the convergence criterion of Q-learning and SARSA by grouping the episodes into epochs of 100 episodes, and finding the variance of the number of moves for each epoch. If the variance was smaller than epsilon, the convergence criterion was met.

Results

Value iteration

Value iteration reached the convergence criterion for all 4 tracks. To show an example these values, the values of the states at velocity [0, 0] of the L track are presented in an image map, where black indicates more negative values, and white indicates values closer to 0. Values are close to zero near the finish line, and more negative farther from the finish line.

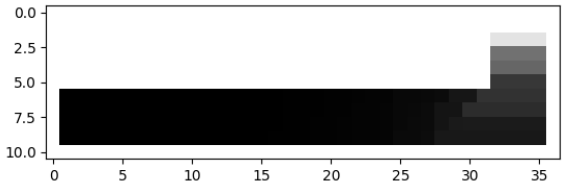
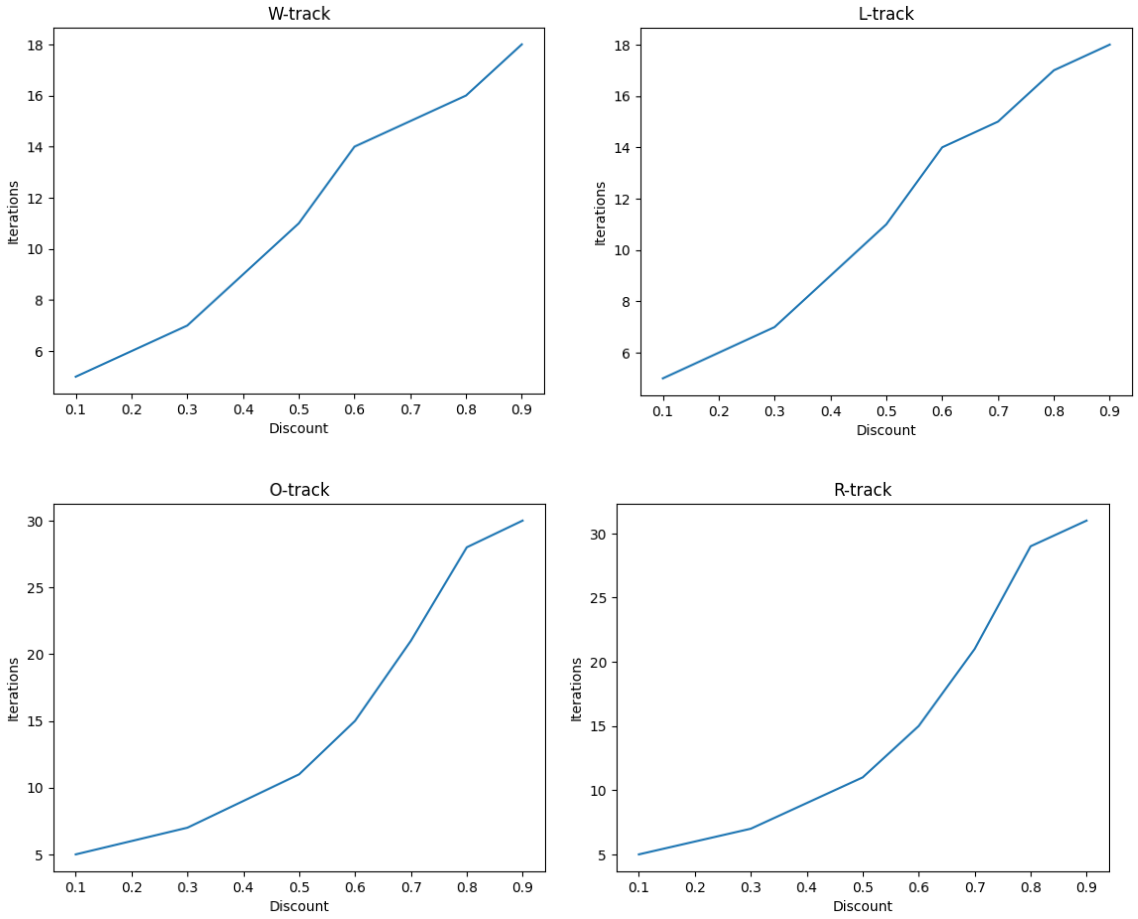


Fig 1. Value Iteration State Values for L track. black indicates more negative values, and white indicates values closer to 0

Fig. 1 was generated with a discount of 0.5. Differences in the image map are perceivable on the vertical segment of the 'L' and around the bend. If the discount value is increased, the perceivable differences extend farther away from the finish line. However, this requires more iterations to complete.

We examined the number of iterations required to reach convergence for the four tracks, and the R-track with restarting at the start line after crashes. Generally, convergence requires more iterations as discount increases. Using discounts of 0.1, 0.2, 0.3, 0.4 and 0.5 requires 5, 6, 7, 9 and 11 in all five conditions, respectively. Beyond a discount of 0.5, the iterations required are similar between the O and R tracks, and between the W and L-tracks, but the O and R-tracks required more iterations than the L and W- tracks. Restarting at the starting line after the crash further increased the iterations required to reach convergence.



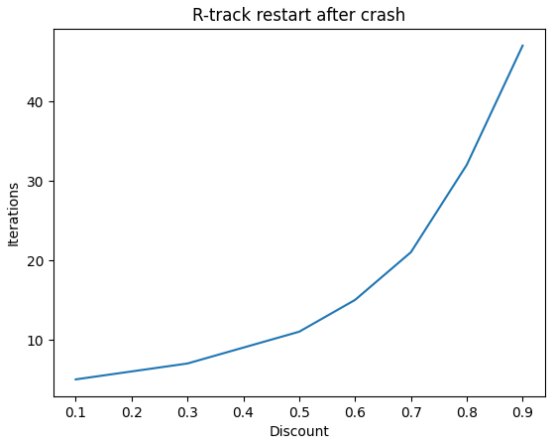


Fig 2. Effect of discount on number of iterations required to reach convergence

Model-free learning

We found that the speed of q-learning and SARSA improved with higher discount for discounts from 0.4-0.9. Furthermore, speed of the algorithms increased supralinearly as discount decreased.

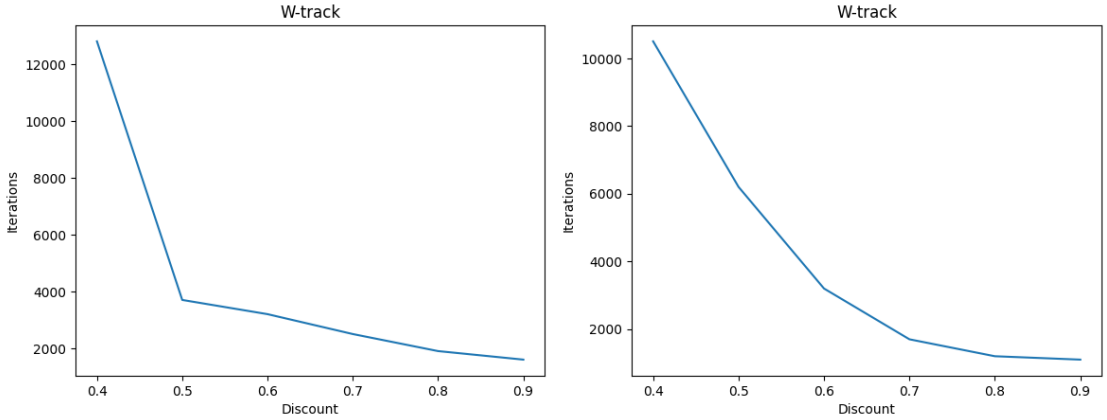


Fig 3. Effect of discount on number of iterations required to reach convergence for q-learning (left) and SARSA (right).

The use of eligibility traces produced modes reductions in number of iterations to convergence, which was unfortunately more than offset by the time required to compute each iteration. Number of iterations are presented for Q-learning on the W-track with a convergence criterion variance of 10000. Three decay rates (0, 0.1, 0.2) were tested, where a decay of 0 ignores the eligibility traces.

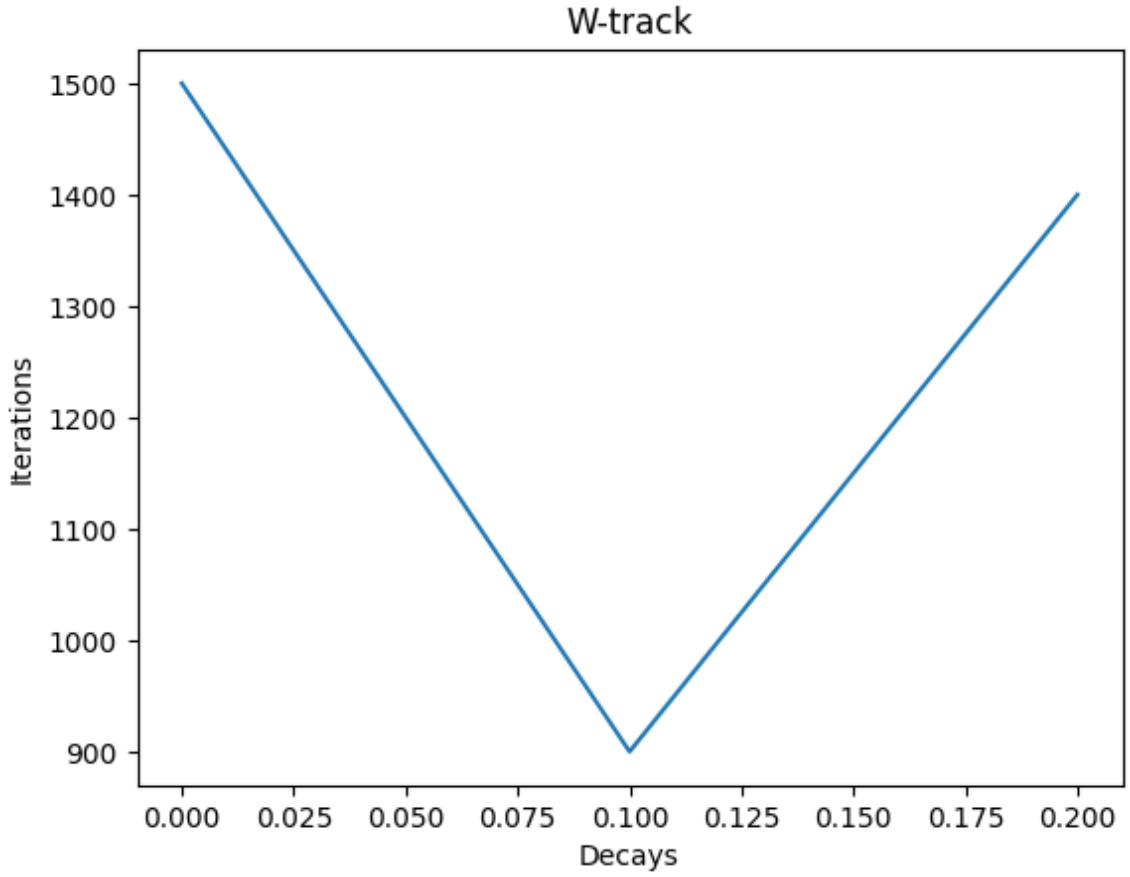


Fig. 4 Effect of eligibility traces on Q-learning for W-track.

For the remaining results, we used a discount of 0.9, and did not use eligibility traces. Adjusting the decay functions of the learning rate and temperatures also strongly affected the rate at which the performance improved. Depicted in Fig. 3, the learning rate $\eta$was iteratively changed as:

$$\eta_{t+1} = \frac{1}{\frac{1}{\eta_t} + .0001}$$

The temperature T was iteratively changed as:

$$T_{t+1} = \frac{1}{\frac{1}{T_t} + .001}$$

The action selection method

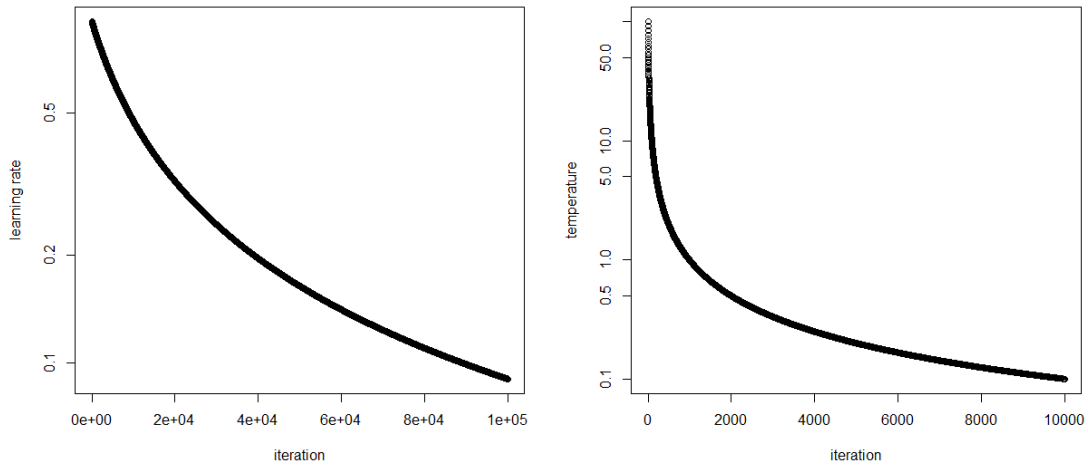


Fig. 4 Learning rate and temperature decay across iterations.

The performance of q-learning improves over iterations, using a discount of 0.9, and an initial learning rate of 0.9. The number of episodes needed to reach a epoch variance of 10 were: 23000 for W, 17500 for L, 30200 for O, 40200 for R. The finish line could be reached in 11 moves for W, in 12 moves of L, in 27 moves for O, and in 25 moves for R.

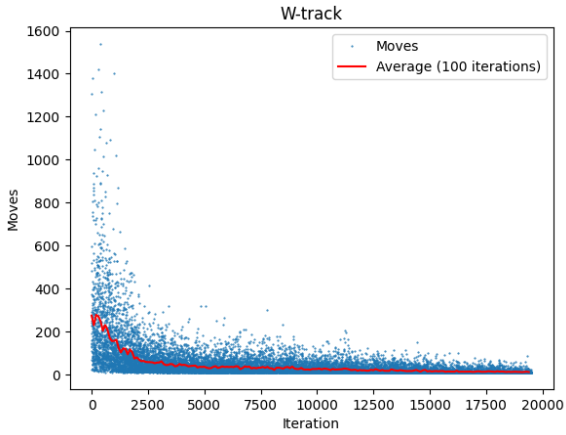


Fig 5. Q-learning performance for W track

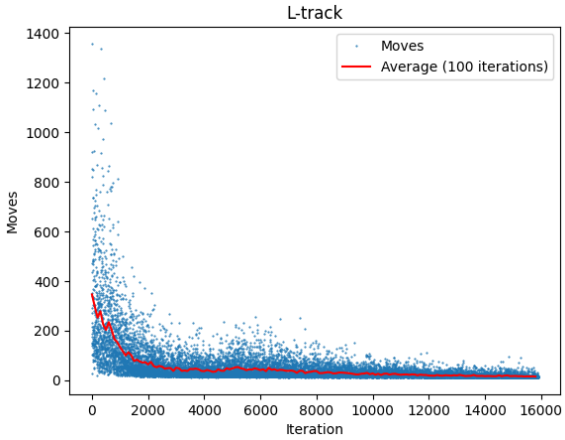


Fig 6. Q-learning performance for L track

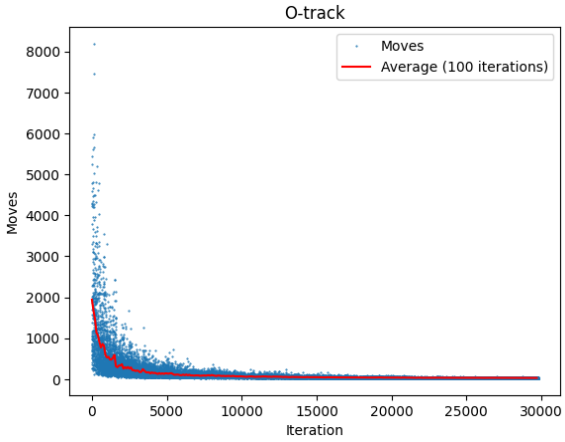


Fig 7. Q-learning performance for O track

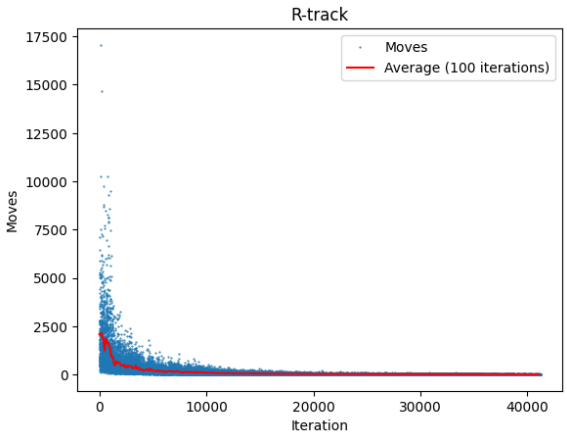


Fig 8. Q-learning performance for R track

We were unable to reach convergence with Q-learning for R track with restarting after crash. The W- and L- tracks were easier to solve. Below we show the performance of Q-learning on the L-track, restarting at the starting line after crashes, with a convergence variance criterion of 1E6. The initial episodes could take up to 300,000 moves to complete.

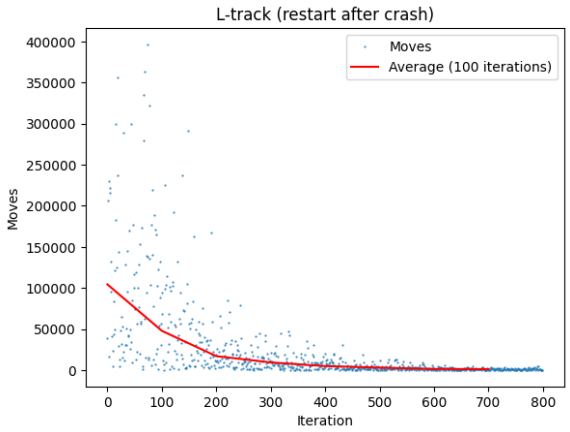


Fig 9. Q-learning performance for L track restarting after crashes

SARSA performed similarly to q-learning improves over iterations, using a discount of 0.9, and an initial learning rate of 0.9. The number of episodes needed to reach a epoch variance of 10 were: 23000 for W, 17500 for L, 30200 for O, 40200 for R. The finish line could be reached in 9 moves for W, in 11 moves of L, in 26 moves for O, and in 27 moves for R.

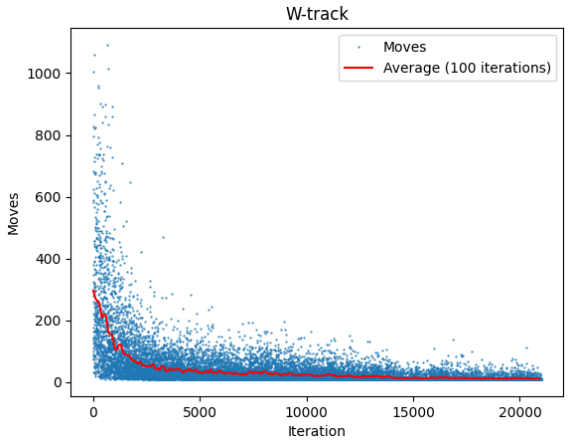


Fig 10. SARSA performance for W track

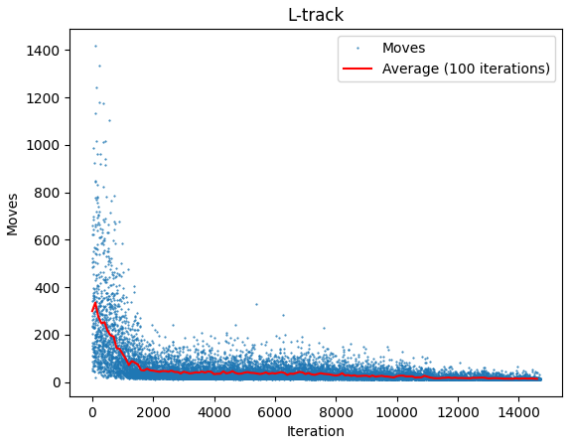


Fig 11. SARSA performance for L track

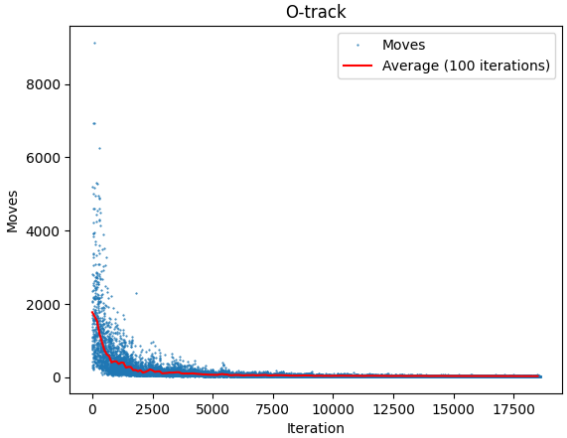


Fig 12. SARSA performance for O track

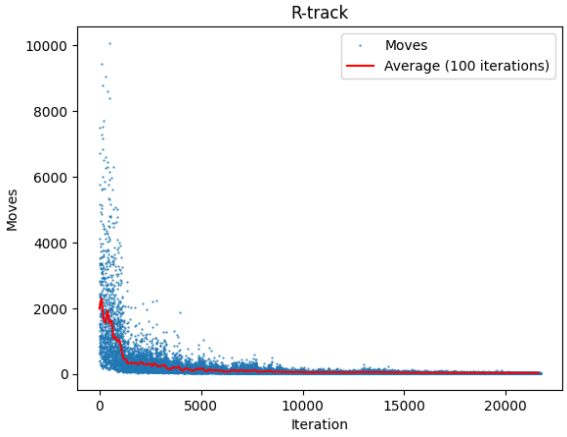


Fig 13. SARSA performance for R track

Discussion

Among the tracks, the W and L-tracks were easier to solve, while the O and R-tracks were more difficult, due to the distance between the starting and finish line, as well as the number of actions required. Restarting the racecar at the starting line after a crash increased the difficulty of the problem. This made the O and R-tracks very difficult, as the actions required to navigate the track were very specific and the probability of finding these actions during exploration was very rare. We were unable to solve the R-track with restarting in reasonable time, but were able to solve the W and L-tracks with restarting.

Using a higher discount factor resulted in faster convergence for q-learning and SARSA for the racetrack problem. We did not find that a higher discount would improve value iteration. However, this is likely because the convergence criterion for value iteration was based on the magnitude of the changes applied to the value array. To clarify this point, using a lower discount required fewer iterations to reach convergence in value iteration. However, the resulting value array was less informative and would have likely produced worse performance than the value array generated with a higher discount.

Contrary to our expectations, the use of eligibility traces did not speed up q-learning and SARSA. With the use of eligibility traces, all state-action pairs previously visited during the trial episode are updated with each move. This is most useful when information about the finishing line (or more generally: high-value states and state-action pairs) is passed to these state-action pairs. However, in the racetrack problem, the value of the finishing line is 0. Therefore, with the use of eligibility traces, q-learning and SARSA update the previously visited state-action pairs with a value of 0 for crossing the finish line, and a negative value for all other results. Then, being on a path that leads to the finish line leads to no change in the last move, which is the same result that all non-visited state-action pairs receive, which is not helpful for learning. We expect that if the finish line had a positive reward, the eligibility traces would have improved performance.

Conclusion

The racetrack problem can be learned iteratively with value iteration, q-learning and SARSA. The O- and R- tracks were especially difficult to solve with a model-free reinforcement learning when crashing resulted in restarting at the starting line. We found that a high discount factor was ideal for the racetrack problem, because the distance from the finish line was more useful information than the reward from each state-action pair. Eligibility traces did not improve the speed of the model free algorithms.