Programming Project #05 – Reinforcement Learning

Jacob M. Lundeen JLUNDEE1@JHU.EDU

Department of Data Science
Johns Hopkins University
Baltimore, MD 21218

Abstract

This paper describes twelve Reinforcement Learning experiments and their results. These twelve experiments use the Value Iteration, Q-Learning, and State-Action-Reward-State-Action methods to run a race around three racetracks with two crash scenarios. All three methods were able to converge to an optimal policy across all three racetracks and both crash scenarios. Value Iteration was able to converge with far fewer iterations than both Q-Learning and SARSA.

1. Introduction

Reinforcement Learning (RL) is an area of machine learning concerned with how intelligent agents take actions in its environment. The agent takes these agents based on the notion of maximizing the cumulative award. RL is one of the three machine learning paradigms, alongside supervised and unsupervised learning. RL's begins go back as far back as the early 1900s when scientists were observing animal intelligence and the 'Law of Effect', or trail-and-error, was studied. This Law of Effect is what still drives RL research to this day, the agent learns the environment through positive and negative rewards until it learns how to maximize the positive rewards.

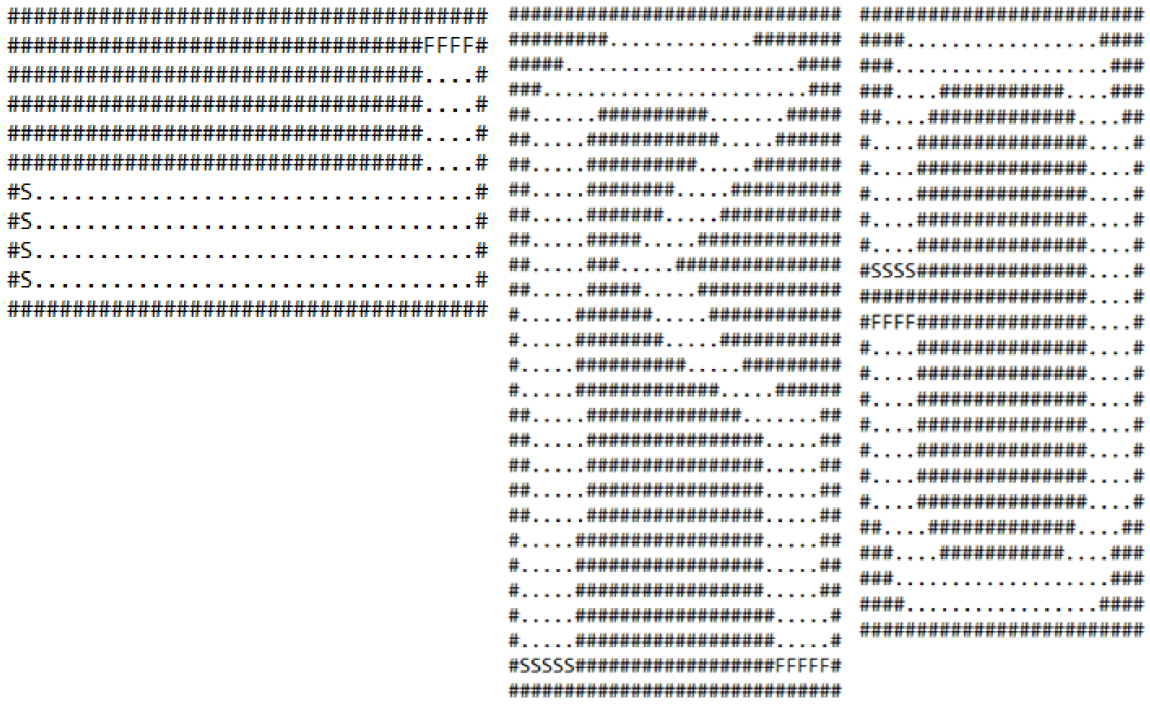
Problem Statement

In this experiment, we will be comparing the performance of three RL methods: Q-Learning (QL), State-Action-Reward-State-Action (SARSA), and Value Iteration (VI). These methods will be used to run a 'race', where the agent is trying to get from the starting line to the finishing line in as few steps as possible without crashing. There are three different racetracks and two methods to handling crashes. My hypotheses are that all three algorithms will be able to find an optimal route; VL will converge the fastest to the optimal route due it only evaluating the current states once; and crash scenario one will converge faster than scenario two.

2. Algorithms and Experimental Methods

Data Sets

There are three files to be used with this experiment: L-Track, R-Track, and O-Track. These 'tracks' are simply alpha-numeric characters with defined meanings used to represent a racetrack. See Figure 1 below:



```
#####################################
################################FFFF#
################################....#
################################....#
################################....#
################################....#
#S..................................#
#S..................................#
#S..................................#
#S..................................#
#####################################
```

```
##############################
#########.............########
#####.....................####
###........................###
##......##########.......#####
##.....############.....######
##.....##########.....########
##.....########.....##########
##.....#######.....###########
##.....#####.....#############
##.....###.....###############
##.....#####.....#############
#.....#######.....############
#.....########.....###########
#.....##########.....#########
#.....#############.....######
##.....##############.......##
##.....################.....##
##.....################.....##
##.....################.....##
##.....################.....##
#.....#################.....##
#.....#################.....##
#.....#################.....##
#.....##################.....#
#.....##################.....#
#SSSSS##################FFFFF#
##############################
```

```
#########################
####.................####
###...................###
###....###########....###
##....#############....##
#....###############....#
#....###############....#
#....###############....#
#....###############....#
#....###############....#
#SSSS###############....#
####################....#
#FFFF###############....#
#....###############....#
#....###############....#
#....###############....#
#....###############....#
#....###############....#
#....###############....#
#....###############....#
##....#############....##
###....###########....###
###...................###
####.................####
#########################
```

Figure 1 - L, R, and O tracks

As we can see, the characters breakdown as such:

- # - Wall
- S – Start
- F – Finish
- . - Track

Experimental Approach

Reinforcement Learning

Reinforcement Learning (RL) is one of the three main areas of Machine Learning alongside supervised and unsupervised learning. In RL, there are two primary functions, the agent, and the environment. The agent is anything that meets the definition of something that perceives its environment and can take autonomous actions to achieve a goal state. Here, the agent is a function. The environment is typically stated in the form of a Markov Decision Process (MDP), due to the general use of dynamic programming techniques (Osinski & Budek, 2021).

The general idea of RL is for the agent to learn an optimal, or nearly optimal, policy that maximizes an arbitrary reward function. The agent interacts with its environment in discrete time steps, where at each time step there is a state $\mathbf{s_t}$, and a reward, $\mathbf{r_t}$. The agent chooses an action a_t, from all possible actions, which is sent the environment and the environment moves to a new state $\mathbf{s_{t+1}}$ and reward $\mathbf{r_{t+1}}$ that is associated with the transition $\mathbf{T(s_t, a_t, s_{t+1})}$. The eventual goal is for the agent to learn a policy, $\boldsymbol{\pi}: \boldsymbol{A}\ \boldsymbol{x}\ \boldsymbol{S} \rightarrow [\mathbf{0}, \mathbf{1}], \boldsymbol{\pi}(\boldsymbol{a}, \boldsymbol{s}) = \boldsymbol{Pr}(\boldsymbol{a_t} = \boldsymbol{a}\ |\boldsymbol{s_t} = \boldsymbol{s})$ which maximizes the expected cumulative reward.

Value Iteration

VI is an iterative algorithm used to determine the optimal policy for model-based learning. VI has been shown to converge to the correct *V** values using the optimal value function. The pseudocode is on p.569 of the textbook (Alpaydin, 2020). These values converge if the maximum difference between two iterations is less then an arbitrary threshold δ. It is possible that the policy itself converges to the optimal before the values converge to their optimal values. The value function, ***V*(s)***, is found with the following equation:

$$V_{i+1}(s) := \max\{\sum_{s'} P_a\,(s'|s)\big(R_a(s, s') + \gamma V_i(s')\big)\}$$

where ***i*** is the iteration number. VI starts at ***i = 0*** and $\boldsymbol{V_0}$ as a guess.

Q-Learning

QL is a model-free RL algorithm to learn the value of an action in a particular state (Osinski & Budek, 2021). Essentially, real world environments are not usually completely known to the agent, which causes issues with algorithms like VI. QL works by adding the maximum reward attainable from future states to the reward for achieving its current state, which effectively influences the current action by the future reward. QL has a function that calculates the quality of a state-action pair: $Q : S\ x\ A \rightarrow \mathbb{R}$. The core algorithm uses a simple iteration update, using the weighted average of the old value and the new information:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \cdot maxQ(s_{t+1}, a) - Q(s_t, a_t))$$

State-Action-Reward-State-Action

SARSA, originally known as "Modified Connectionist Q-Learning", is a variation of QL. The primary difference is that the main function in SARSA updates the Q-Value by using the current state, the action chosen, the reward the agent gets, the state the agent enters after taking that action, and the next action the agent chooses in the new state. The algorithm follows:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Crash Scenarios

When the racecar is moving around the track, it will inevitably crash into a wall. To handle this, two scenarios are used:

- Crash Scenario One – The car starts over at the nearest open spot to the crash location and the velocity is set to (0,0).

- Crash Scenario Two – The car starts over at the original starting position with its velocity set to (0,0).

Tracks L and O will only utilize crash scenario one. The R track will utilize both crash scenarios (at separate times).

Determinism

There is a 20% chance that, any time the racecar attempts to accelerate, it will fail, and the velocity will remain unchanged.

Parameter Tuning

The hyperparameters for this experiment (learning rate and discount factor) were tuned using a greedy search method of nested for loops.

3. Results

First, the table and charts for VI:

Table 1 - Value Iteration Results

Value Iteration				
Num Training Iterations	L-Track	R-Track		O-Track
		Crash 1	Crash 2	
5	500.00	500.00	500.00	3.70
10	10.60	500.00	500.00	6.40
15	12.00	500.00	500.00	6.60
20	10.60	70.30	500.00	4.70
25	10.70	22.50	36.50	6.00
30	11.00	24.00	28.90	4.70
35	11.00	22.90	36.90	5.10
40	11.00	23.70	41.10	5.50
45	11.40	23.80	36.20	3.60

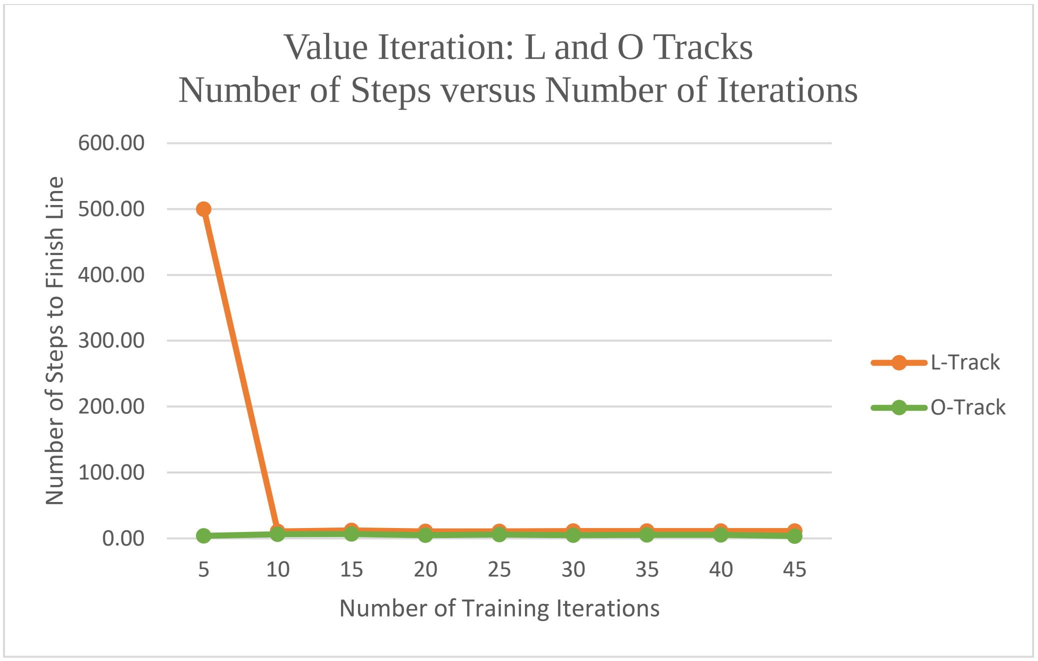


Figure 2 - Value Iteration Learning Curves: L and O Tracks

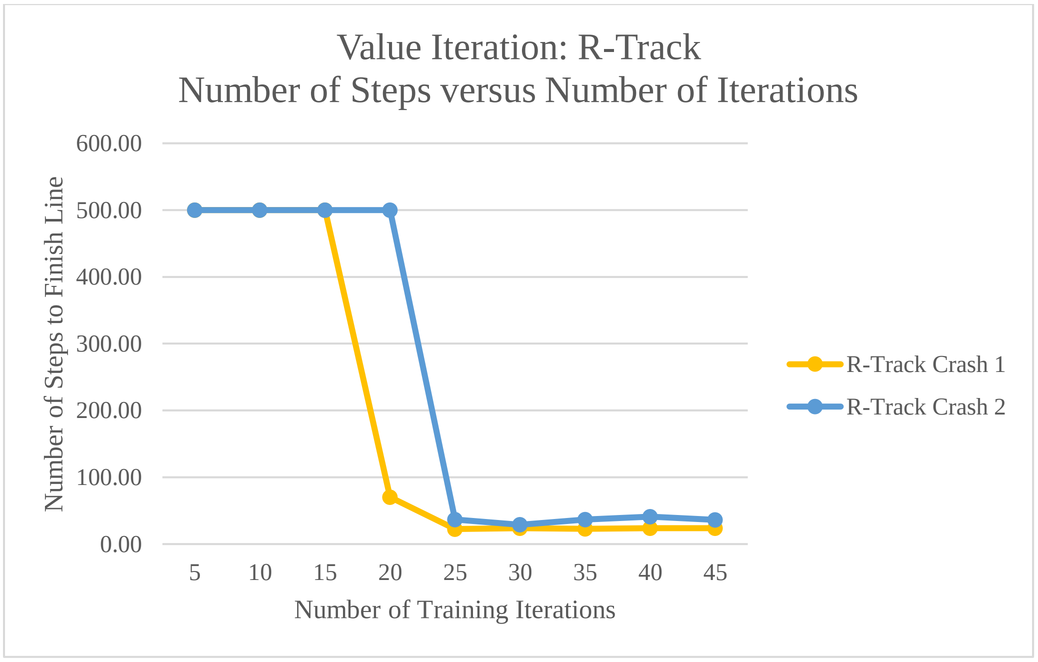


Figure 3 - Value Iteration Learning Curves: R-Track

Next, the table and charts for QL:

Num Training Iterations	**L-Track**	**R-Track**		**O-Track**
		Crash 1	Crash 2	
10000	500.00	500.00	500.00	6.10
1010000	11.00	144.30	500.00	9.10
2010000	11.40	29.00	500.00	5.20
3010000	11.30	28.40	500.00	5.40
4010000	11.20	27.40	47.50	5.70
5010000	11.80	31.00	37.40	104.60
6010000	11.30	26.40	45.50	7.00
7010000	10.90	28.50	41.60	7.60
8010000	11.00	27.60	37.70	54.50
9010000	11.30	28.90	34.10	6.50

Table 2 - Q-Learning Results

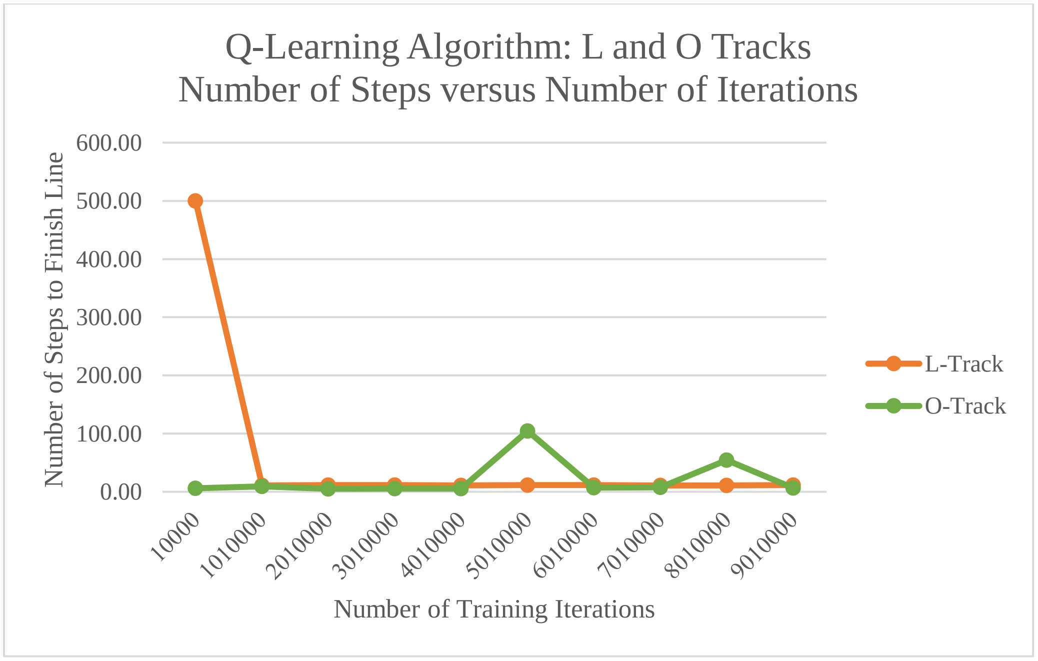


Figure 4 - Q-Learning Learning Curves: L and O Tracks

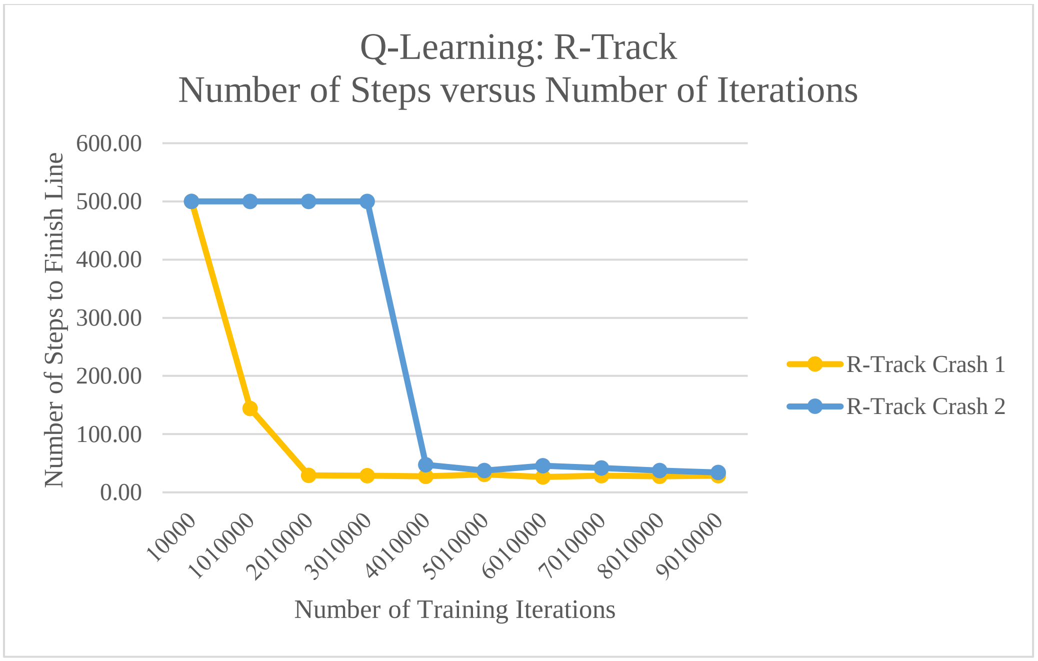


Figure 5 - Q-Learning Learning Curves: R-Track

Finally, the SARSA results:

Table 3 - SARSA-Learning R-Track Results

Num Training Iterations	**R-Track**	
	Crash 1	**Crash 2**
10000	500.00	500.00
1010000	194.50	500.00
2010000	124.40	500.00
3010000	29.20	500.00
4010000	29.50	500.00
5010000	28.30	455.60
6010000	28.60	69.60
7010000	28.60	36.30
8010000	30.70	39.60
9010000	29.80	43.20

Table 4 - SARSA-Learning L&O Track Results

Number of Iterations	**L-Track**	**R-Track**
10000	255.40	320.20
110000	210.80	451.20
210000	14.40	57.20
310000	13.60	13.00
410000	11.10	9.30
510000	12.70	5.90
610000	12.00	8.60
710000	11.60	6.00
810000	12.40	7.90
910000	12.30	7.10

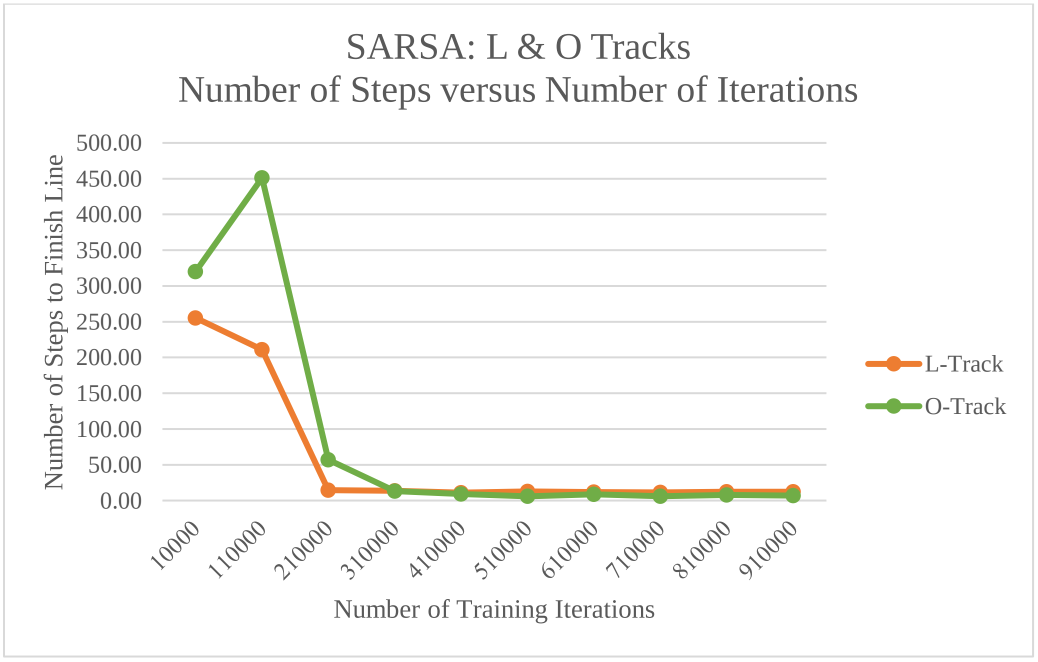


Figure 6 – SARSA Learning Curves: L and O Tracks

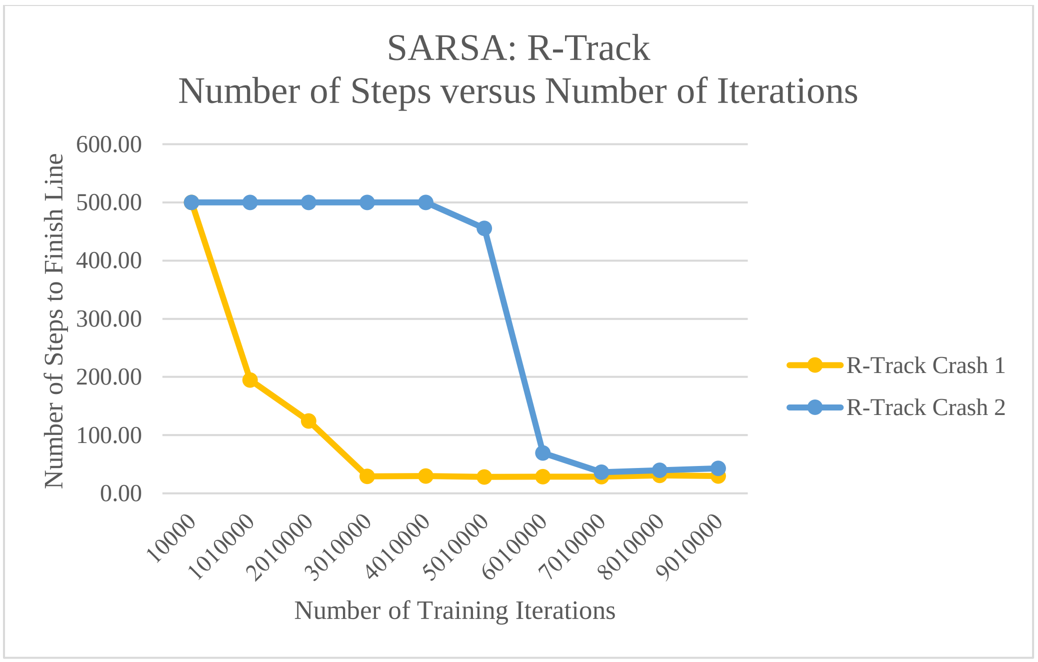


Figure 7 – SARSA Learning Curves: R-Track

4. Discussion

Hypothesis One

My hypothesis was correct. All three RL methods were able to learn an optimal policy to the finish line. All three methods took a few iterations to converge for the L-track and R-track, but all three had no issue with the O-track. What was unexpected was the number of training iterations QL and SARSA took compared to VI. While VI began to converge within 20 iterations, QL and SARSA took at least 10,000 iterations.

Hypothesis Two

Here again, my hypothesis was correct. As stated previously, VI began to converge within twenty iterations across all three tracks. Both QL and SARSA were able to converge but took more time to do so.

Hypothesis Three

Lastly, this hypothesis was also correct. Across all three methods crash scenario one converged faster than crash scenario two. This is to be expected because in crash scenario two, the agent must start over from the beginning, making the agent retrace its steps up to that point. However, once either scenario did converge, the number of steps to finish the race were statistically no different.

5. Conclusion

RL is an incredibly powerful machine learning algorithm that utilizes an agent to learn its environment through repetition to find an optimal policy to traverse the environment. In this experiment VI was able to find the optimal policy with a significantly smaller number of steps (less than 50 versus 10,000 or more).

References

Alpaydin, E. (2020). *Introduction to Machine Learning, fourth edition (Adaptive Computation and Machine Learning series)* (fourth edition). The MIT Press.

Osiński, B., & Budek, K. (2021, January 5). *What is reinforcement learning? The complete guide*. Deepsense.Ai. https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

Sears-Collins, A. S. C. (2019, August 23). *Value Iteration vs. Q-Learning Algorithm in Python Step-By-Step*. Automatic Addison. https://automaticaddison.com/value-iteration-vs-q-learning-algorithm-in-python-step-by-step/#vlcode