Determining optimal policies in combat robotics using reinforcement learning

Frederick Hinder

Mentored by Julie Foresta, Ian Michel-Tyler, Jesse Milzman, Long Quang, Dr. Sean Gart, Virginia To

SMA Science and Mathematics Academy

ARL

Introduction

Reinforcement learning (RL) is a field of artificial intelligence where optimal policies for decision processes are approximated using methods of trial and error (Sutton & Barto, 2018). Monte Carlo RL methods leverage the law of large numbers to approximate the value (Q-value) of taking a given action in a given state (configuration of an environment). After each iteration of training (episode), a Monte Carlo RL agent will update Q-values of visited state-action pairs by averaging the current Q-value with the returns (sum of the rewards received after taking an action and the discounted future returns).

In combat robotics competitions, competitors aim to disable an opponent's robot while maintaining their own operational status. Accurate driving and strategic maneuvering provide a competitive edge, suggesting that RL could develop superior policies compared to existing policies. The purpose of this project was to generate a policy for movement in combat robotics that outperforms simple control policies.

Methods and Materials

This project was coded in Python and used the Gymnasium library from OpenAI. The combat robotics environment tracked the position (in inches) and the direction (in radians) of both the agent and the adversary. Neither robot was able to move beyond the dimensions of the arena which was 96″ × 96″.

Video recordings of the Jolt 3 lb combat robotics kit driving at various velocities were made. These recordings were analyzed with Kinovea, a software for measuring motion in video, to create a set of actions (action space) for the combat robotics environment. Two control policies based on simple objectives were implemented as adversaries for an agent. One policy was aggressive, selecting actions that minimized the distance to the agent. The other policy was defensive policy, selecting actions that minimized the angle between its spinner's direction and the agent's direction.

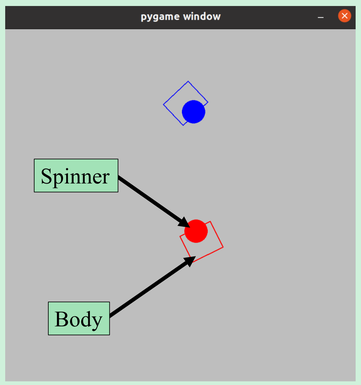


Figure 1 (above): A user interface with the agent (blue) and adversary (red) in the 96″ × 96″ arena. The dimensions of the robots were based on that of the Jolt! 3 lb combat robotics kit.

Three RL agents (software that interacts with an environment) were created that used Monte Carlo methods to estimate the values of state-action pairs against an adversary with an aggressive policy. While learning, each agent received discretized representations of the locations and directions of itself and the adversary, which is necessary for learning

Methods and Materials (continued)

with a table of Q-values (Q-table) (ten Hagen, 2001). Location was quantized into intervals of 12″, 6″, or 4″ and direction into 8 intervals of 45°. Another set of three RL agents was created to learn against a defensive policy using the same parameters. The cumulative performance of each agent is shown in Graph 1.

Figure 2 (right): The pseudocode for the Monte Carlo methods used in this project. S is the state space, $A(s)$ is the action space, $Q(s, a)$ is the Q-table, and S_T is the set of terminal states. Episodes terminated when the spinner of one or both robots overlapped the body of the other or after 3 minutes (standard match length) of time in the combat robotics environment. At the termination of an episode, the agent received 1 point if only its spinner contacted the body of the other robot, 0 if both spinners contacted both bodies, and −1 otherwise. Each training session of 1,000,000 episodes took around 1.5 hours.

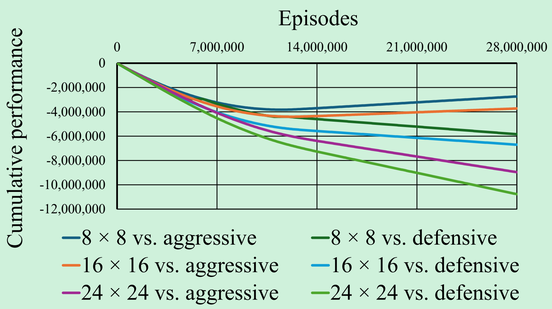
Initialize, for all $s \in S$, $a \in A(s)$:
  $Q(s, a) \leftarrow 0$
  $Counts(s, a) \leftarrow 0$
  $\varepsilon \leftarrow 1$

Repeat for 28,000,000 episodes:
  $Done \leftarrow$ false
  $State \leftarrow$ a random member of S that is not terminal
  While $Done$ = false:
    $Action \leftarrow \arg\max_a Q(State, a)$
    With probability ε:
      $Action \leftarrow$ a random member of $A(State)$
    Step environment by $Action$
    $Done \leftarrow (State \in S_T)$
  $R \leftarrow$ reward received in the terminal state
  For each pair (s, a) appearing in the episode:

$$Q(s, a) \leftarrow \frac{Counts(s, a) \cdot Q(s, a) + R}{Counts(s, a) + 1}$$

  $Counts(s, a) \leftarrow Counts(s, a) + 1$
  $R \leftarrow 0.95 \cdot R$
$\varepsilon \leftarrow \max\left(\varepsilon - \frac{1}{14{,}000{,}000}, 0.1\right)$

Cumulative performance during training



Graph 1 (left): A plot of the cumulative performance of each agent. A positive slope indicates an agent outperforming its adversary while a negative implies the opposite. During training, the 8 × 8 vs. defensive agent did not properly save its performance twice.

Results

One thousand episodes were generated for each control policy to test its performance with an adversary of the same policy. Each policy generated by the agents was tested in episodes with the same initial state as their corresponding control policy. The performances of the agent policies were compared to the performance of their corresponding control policy with a paired t-test to determine whether they could outperform the control policy. The statistics for each test are shown in Table 1 and the sample distribution of differences for each test are shown in Graph 2.

	M	SD	$t(999)$	p
Aggressive (control)	0.000	0.683		
8 × 8 vs. aggressive	0.189	0.815	6.12	< .001
16 × 16 vs. aggressive	0.185	0.823	5.89	< .001
24 × 24 vs. aggressive	0.028	0.858	0.85	.198
Defensive (control)	−0.016	0.692		
8 × 8 vs. defensive	−0.023	0.792	−0.21	.584
16 × 16 vs. defensive	0.013	0.818	0.91	.182
24 × 24 vs. defensive	−0.092	0.833	−2.33	.990

Table 1 (right): A table of the statistics for each paired t-test. The last two columns are the results of the one-tailed paired t-test for the differences between the RL policy's performance and the corresponding control policy's performance. Using the alpha level of .05, the results for the 8 × 8 vs. aggressive and 16 × 16 vs. aggressive agents suggest that the policies generated by these agents perform significantly better than the aggressive control policy.

Paired differences of performance
(*agent policy − control policy*)

Graph 2 (right): Differences in performance between each agent policy and corresponding control policy. Positive differences are instances of the agent policy outperforming the control policy.

Conclusions

The successful learning with the 8 × 8 and 16 × 16 discretizations trained against an aggressive policy indicate that Monte Carlo RL methods can sometimes improve driving policies for combat robotics and similar environments. Despite the effectiveness of the policies produced, these methods require the discretization of continuous state spaces which do not necessarily converge to optimal policies (ten Hagen, 2001). Future studies could explore applying RL methods, such as neural networks, that can learn in environments with continuous state spaces to combat robotics.

References

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press. https://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

ten Hagen, S. H. G. (2001). *Continuous state space Q-learning for control of nonlinear systems* (Doctoral dissertation, Universiteit van Amsterdam [Host]). https://pure.uva.nl/ws/files/3749585/18781_Thesis.pdf