Deep Reinforcement Learning for Repeated Slotted Multiple Access

USC Viterbi School of Engineering

Victor Hui, Prof. Bhaskar Krishnamachari, Prof. Michael Neely
University of Southern California, Los Angeles, California 90089

Introduction

Multiple access control is a classical computer science problem with many well-established protocols such as Slotted Aloha [1] or CSMA [2]. However, these protocols are intended for randomly arrived data rather than the *repeated game* format that is crux of this research. Rather than data link capture, the format of these experiments are more in common with the *Iterated Prisoner's Dilemma* [3] studied under Game Theory.

This investigation builds on top of Professor Michael Neely's research [4] to explore the open question of how to minimize the expected time to capture the channel for a n-user situation. Over the past few semesters, Prof. Neely conducted a 2-user multiple access game at USC between student designed algorithms, of which an algorithm called 4-state was a consistent winner. In his paper, he showcased the algorithm to have an optimal expected score when competing against independent versions of itself.

Through this research, our goal is to see if Deep Reinforcement Learning can produce an optimal policy that is equal to or more effective than 4-state at optimal channel capture against the same student algorithms.

Background

Reinforcement Learning (RL)

Reinforcement Learning is one of the pillars of Machine Learning where the agent learns from mistakes. A generic Reinforcement Learning model consists of an action-reward feedback loop where the agent's utility is defined by the reward function, as the agent learn to act so as to maximize expected rewards.

Deep Reinforcement Learning (DRL)

Deep reinforcement learning combines reinforcement learning techniques with an artificial neural network. The neural network helps manage large state problems by using implicit function representation instead of an explicit table such as in traditional Q-learning. In this project, we employed a Deep Q Network such as in [5] as it has shown to be very effective when applied to a large state space.

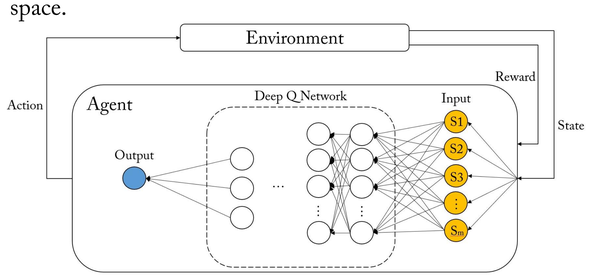

Figure 1. Action Reward Feed-back Loop w/DQN

Methods

Setting up the problem

We treat the classical multiple access scenario with success, collision, or idle on every slot. We assume that the system delivers the perfect feedback on the number of users who transmitted at the end of each slot. Both the Agent and the Opponent keep a record of transmission history.

		Opponent Action: Transmit	Opponent Action: No Transmit
Agent Action	Transmit	Neither Score	Agent Scores
	No Transmit	Opponent Scores	Neither Score

Figure 2. Scoring System

At any given time slot, if one transmits while the other refrains, the transmitter score increases by 1 point. If neither transmits or both transmit, score does not increase.

Round – A pair of transactions, lasting 1 time slot each
Game – 100 rounds of repeated channel capture
State Space – 2x99 sized array containing both Agent and Opponent history

Reward Structure

In Reinforcement Learning, the reward is the feedback from the environment. We have investigated both the effects of changing the reward function itself and changing when the reward function is applied.

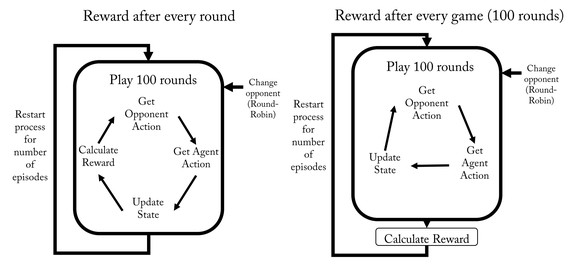

Figure 3. Reward after every round vs game

Opponents

As a preliminary test of concept, five opponent algorithms (3 deterministic and 2 non-deterministic) were chosen.
EvenOdd: transmits 0 on even rounds, 1 otherwise.
TitforTat0: transmits 0 on first round, tit-for-tat after.
TitforTat1: transmits 1 on first round, tit-for-tat after.
3-State: Start in state 1, randomly transmit until the first success, then move to a turn-based policy that oscillated between state 2 (remain idle for one slot) and state 3 (transmit repeatedly until program scores). [4]
4-State: First 3 states are identical to 3-State, however, if the opponent does not score while 4-state is in state 2, the algorithm moves to state 4 and repeatedly transmitted on all remaining slots until collision. 4-state has the property that it cannot lose a game by more than 1 point. [4]

Results

Testing the agent

Reward structure 1: +1 if the agent scored that round, -1 if the agent didn't.

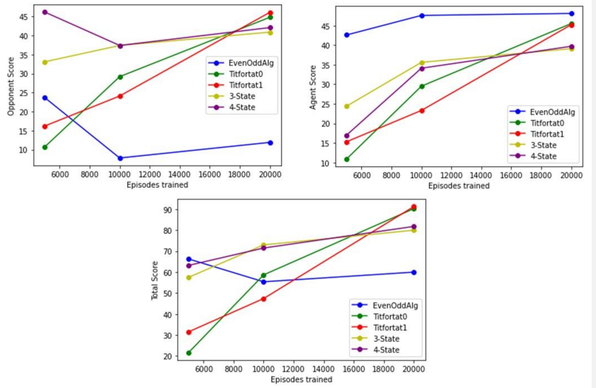

Figure 4. Results of reward structure 1 w/training

- Under this reward structure, the agent gets penalized if neither scored or if opponent scored.
- Figure 4 shows the DRL agent scoring higher as number of episodes trained increases.
- Increase in total score suggests the DRL agent learned to cooperate with opponents. (Algorithms such as 3-State will only hold transmission if it had scored in the previous round)

The unexpected winning policy

Reward structure 2: reward = difference between agent and opponent score, reward is updated after every game.

EvenOdd	
Avg_Total_Score	49.7
Avg_Opponent_Score	1.1
Avg_Agent_Score	48.6
Tscore_History	[48, 49, 49, 49, 53, 50, 48, 51, 49, 51]
Opponent_History	[1, 2, 0, 0, 4, 1, 0, 2, 0, 1]
Agent_History	[47, 47, 49, 49, 49, 49, 48, 49, 49, 50]

TitforTat1	
Avg_Total_Score	4.7
Avg_Opponent_Score	2.4
Avg_Agent_Score	2.3
Tscore_History	[4, 4, 2, 8, 5, 4, 14, 2, 0, 4]
Opponent_History	[2, 2, 1, 4, 3, 2, 7, 1, 0, 2]
Agent_History	[2, 2, 1, 4, 2, 2, 7, 1, 0, 2]

TitforTat0	
Avg_Total_Score	6.8
Avg_Opponent_Score	2.9
Avg_Agent_Score	3.9
Tscore_History	[7, 5, 5, 13, 7, 3, 11, 9, 5, 3]
Opponent_History	[3, 2, 2, 6, 3, 1, 5, 4, 2, 1]
Agent_History	[4, 3, 3, 7, 4, 2, 6, 5, 3, 2]

3-State	
Avg_Total_Score	4.9
Avg_Opponent_Score	2
Avg_Agent_Score	2.9
Tscore_History	[5, 4, 3, 11, 1, 3, 3, 9, 7, 3]
Opponent_History	[2, 2, 1, 5, 0, 1, 1, 4, 3, 1]
Agent_History	[3, 2, 2, 6, 1, 2, 2, 5, 4, 2]

4-State	
Avg_Total_Score	5.9
Avg_Opponent_Score	2.5
Avg_Agent_Score	3.4
Tscore_History	[2, 9, 3, 1, 7, 7, 7, 7, 13, 3]
Opponent_History	[1, 4, 1, 0, 3, 3, 3, 3, 6, 1]
Agent_History	[1, 5, 2, 1, 4, 4, 4, 4, 7, 2]

Figure 5. Results of reward structure 2

- Under this reward structure, agent gets penalized only if opponent scores.
- Figure 5 shows DRL agent with a winning policy against opponent algorithms (draws against Titfortat1)
- Agent behaves by always transmitting.
- Against Titfortat0, 3-State and 4-State, the maximum score the optimal policy can win by is 1.

Discussion

EvenOdd	
Avg_Total_Score	98.9
Avg_Opponent_Score	49.4
Avg_Agent_Score	49.5

TitforTat0	
Avg_Total_Score	99.2
Avg_Opponent_Score	49.5
Avg_Agent_Score	49.7

TitforTat1	
Avg_Total_Score	99.2
Avg_Opponent_Score	49.4
Avg_Agent_Score	49.8

3-State	
Avg_Total_Score	99.2
Avg_Opponent_Score	49.5
Avg_Agent_Score	49.7

4-State	
Avg_Total_Score	99.1
Avg_Opponent_Score	49.6
Avg_Agent_Score	49.5

Figure 6. 4-State against opponent algorithms

Takeaway

[4] explored that 3-State and 4-State algorithms perform exceptionally well against these 5 chosen algorithms (as seen in Figure 6). Figure 4 shows us that with intense training, our DRL agent is also capable of reaching scores of the high 40s against the same algorithms.

If the game is played in a competitive manner, Figure 5 shows us that the DRL agent can produce decisive winning policies against these algorithms.

Future Directions

These are only the first results of an ongoing research project. So far, we have demonstrated the effectiveness of a DRL agent against a relevant subset of opponent strategies shown in [4]. Moving forward, we will expand in three main directions:

1. Increasing the number of opponent strategies:
   Can the DRL agent develop a more complex policy that can deal with a wider range of opponent strategies.
2. Developing the complexity of the reward system:
   The current reward structure is biased, as the agent receives greater reward when playing against a simple deterministic algorithm, it develops a policy more biased towards performing well against such opponents.
3. Validating the DRL agent using opponents that it has not trained against.

References

[1] Anurag Kumar, D. Manjunath, Joy Kuri, 2004, Communication Networking: An analytical Approach

[2] Marcelo M. Carvalho, J.J. Garcia-Luna-Aceves, 2018, Carrier-Sense Multiple Access with Transmission Acquisition

[3] David M. Chess, 1988, Simulating the Evolution of Behavior: the Iterated Prisoners' Dilemma Problem

[4] Michael J Neely, 2021. Repeated Games, Optimal Channel Capture, and Open Problems for Slotted Multiple Access

[5] Mnih et al, 2013, Playing Atari with Deep Reinforcement Learning