****

**NAIVE REINFORCEMENT LEARNING ON ATARI GAMES**

**Mentor :-** Dr. Manish Kumar Bajpai

(Assistant Professor, CSE Dept. IIITDM Jabalpur)

Pravesh Vajpayee (2018194)

Kshitij Kumar Singh (2018124)

Raj Agrawal Mohit (2018201)

**ABSTRACT**

The purpose of this project is to code reinforcement learning (RL) algorithms, compare the performance of them on playing Atari games. The three RL algorithms we have selected to study in this project are SARSA, naive Q-learning, and monte-carlo tree search(MCTS) with upper confidence bound (UCB).

**INTRODUCTION AND PROBLEM STATEMENT**

In this project , Three RL algorithms we have selected to compare their performance in this project are SARSA, naive Q-learning, and monte-carlo tree search(MCTS) with upper confidence bound (UCB) on different Atari games, and analyze them in order to identify factors which might influence the performance of these algorithms on different types of environments**.**

**ENVIRONMENTS**

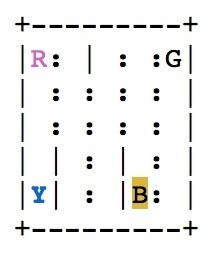
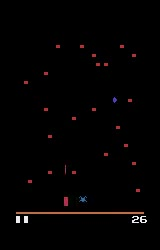
We explore three different environments :

• Taxi-v2 as a deterministic environment with limited observation space.

• Centipede-ram-v0 as a stochastic, non-markovian environment with a relatively big observation space.

**1. Taxi-v2**

Taxi-v2 is a deterministic, single-agent game where the agent’s job is simply to pick up the passenger at one loca-tion and drop him off in another. Agents receive +20 points for a successful dropoff, and lose 1 point for every timestep it takes. There is also a 10 point penalty for illegal pick-up and drop-off actions. [2] There is no randomness that is out of agent’s control, as there are no moving parts in this game other than the agent.

Taxi- V0 Centipede-ram-v0

**2. Centipede-ram-v0**

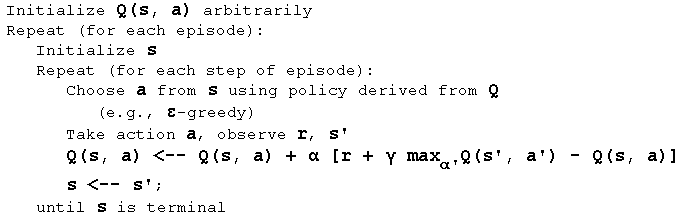
In this game, to maximize the utility, the agent’s goal is to shoot the centipede for a small reward and the spider to gain a bigger reward, while avoiding being hit by them for negative reward. The centipede moves in a predictable downwards fashion at first. However, the spider moves and appears in a random fashion, and additional, shorter centipedes may appear randomly from the agent’s sides. Due to this randomness, it is a stochastic environment**.**

**APPROACH**

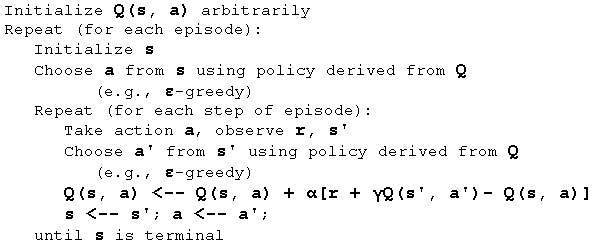
We will be implementing it with 3 RL algorithms i.e. Naive Q Learning , SARSA and Monte Carlo Search.

**Naive Q learning**

Naive Q learning is an algorithm which learns the value of a state (s) - action (a) pair, which we can represent by Q(s, a) , and store it in a table, which we call Q table. Q learning algorithm learns Q(s, a) by updating arg maxa Q(s, a) for any state s the agent is in. Q(s, a) is updated after each action the agent takes, and we assume there is a reward signal after each action. Therefore, the update equation for naive Q-learning is:

**Q(s, a) ← Q(s, a) +α(R(s) + γ arg maxa Q(s 0 , a0 ) − Q(s, a)).**

**SARSA**

Similar to Naive Q-learning, SARSA simulate Bellman update closely, but with a slightly different update equation, which is **Q(s, a) ← Q(s, a) + α(R(s) + γQ(s 0 , a0 ) − Q(s, a))**. With much similarity to naive Q-learning, the difference between these two is the removal of the max operator in SARSA because SARSA is an on-policy algorithm. SARSA keeps updating its Q table which is used for action selection to interact with the environment it is in. However, naive Q-learning is an off-policy algorithm, meaning the action selection algorithm is not necessarily updated, but the Q table is updated with each action taken with respect to the update equation.

**Monte-Carlo Tree Search**

MCTS is a method for making optimal decisions in artificial intelligence problems, typically move planning in combinatorial games.

• Selection: starting at root node R, recursively select optimal child nodes (explained below) until a leaf node L is reached.

• Expansion: if L is not a terminal node (i.e. it does not end the game) then create one or more child nodes and select one C.

• Simulation: run a simulated playout from C until a result is achieved. • Backpropagation: update the current move sequence with the simulation result.

**CONCLUSION**

In conclusion, we will analyze the performance of naive Q-Learning, SARSA, and MCTS on these games i.e. Taxi-V2 and centipede-ram-v0 and few points are expected:

• Naive Q and SARSA will converge on markovian environments, but not so on non-markovian environments.

• Deep Q-Learning using Experience Replay can solve the problem with non-markovian property.

• SARSA is known to converge faster when using epsilon greedy.

In the future, we will extract more insights out of these algorithms by training them on these games and for a greater period of time.

**REFERENCES**

[1] Andrew G.Barto Richard S.Sutton, Reinforcement Learning An Introduction, The MIT Press, 2018.

[2] “Taxi-v2,” 2011.

[3] Peter Norvig Stuart Russel, Artificial Intelligence A Modern Approach, Pearson, 2010.

[4] “Monte carlo tree search,” 2010.

[5] “Implementing deep reinforcement learning models with tensorflow + openai gym,” 2018.

[6] “Playing atari with deep reinforcement learning,” 2013.

[7] https://en.wikipedia.org/wiki/Wikipedia

[8] geeksforgeeks.org

[9] www.cse.unsw.edu.au