Performance evaluation of Q-learning and SARSA(λ) on Taxi ride problem

Monika Ahirwar CS17M025
Madhura Pande CS17S031
Course Instructor: Dr. L.A. Prashanth

Department of Computer Science and Engineering
Indian Institute of Technology, Madras

- Reinforcement Learning(RL) algorithms are widely used to solve problems where an agent needs to interact with an unknown environment and form strategies to maximize the reward.
- Q-Learning and SARSA (State Action Reward State Action) are two such algorithms which can work with real world environments and help the agent learn smarter strategies.

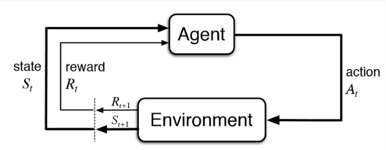

Figure 1: A typical RL setting

The Problem

- We have a 5x5 grid world inhabited by a taxi agent. There are 4 locations (marked R,G,B,Y) from where passengers can be picked up and dropped as well.
- There are rewards and penalties for taking various actions, as imposed by the environment. The agent has to learn the best way to maximize reward.

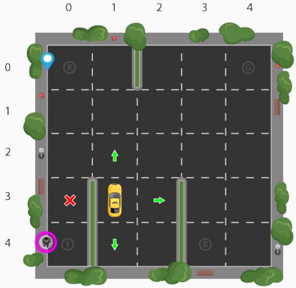

Figure 2: Taxi Environment

The Problem : Underlying MDP!

- States - We need to keep track of taxi's location, passenger's location and intended destination. State space spans across 500 states (5x5x5x4).
- Actions - Six actions are as follows :-
  - Move North
  - Move South
  - Move East
  - Move West
  - Pickup passenger
  - Drop passenger

The Problem : Underlying MDP!

- Rewards
  - There is a -1 reward for each step taken.
  - Agent gets $+20$ for a successful drop-off.
  - -10 for an illegal drop-off, if agent drops the passenger at some random location.
  - Hitting a wall, is same as taking a step incurring a penalty of 1 point.
- Episode - We consider the series of actions taken from the point when passenger is picked, till he is dropped as one Episode.

Our Aim

- We aim to build agents who take policies based on random actions; by following SARSA(λ) and Q-Learning algorithms and compare their performances.
- We also propose a little enhancement in standard Q-Learning method, to better handle exploration-exploitation dilemma conditioned on this environment. We call this agent Smart Q-Learning Agent.

Our Aim

- We aim to optimize the following values and compare it within various agents.
  - Average number of penalties per episode
  - Average number of timesteps per episode
  - Average reward per episode
  - Episodes taken for learning phase

State-Action-Reward-State-Action aka SARSA(λ)

- TD (λ) learns from experience, without a model of any kind
- TD learns from incomplete episodes by bootstrapping
- The drawback is that it evaluates only state values but we need control as well
- In SARSA (λ) is applying TD(λ) prediction method to state - action pairs rather than to states

State-Action-Reward-State-Action aka SARSA(λ)

- First choose $A^{'}$ from $S^{'}$ using policy derived from Q (ϵ - greedy) and update δ
  $\delta = R + \gamma Q(S^{'}, A^{'}) - Q(S, A)$
- Eligibility trace is traced: $E(S, A) = E(S, A) + 1$
- For all s-a pairs update Q and E as follows:
  $Q(s, a) = Q(s, a) + \alpha\delta E(s, a)$
  $E(s, a) = \gamma\lambda E(s, a)$

Performance of SARSA(λ)

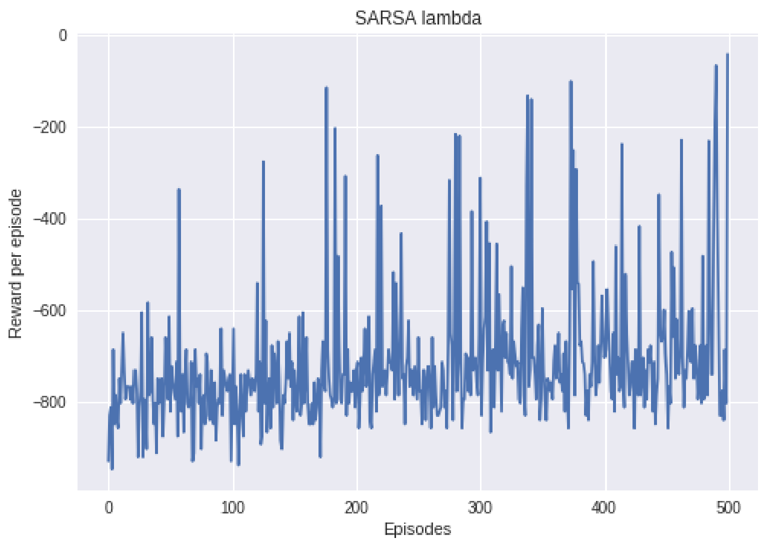

Figure 3: SARSA in 500 iterations

Performance of SARSA(λ)

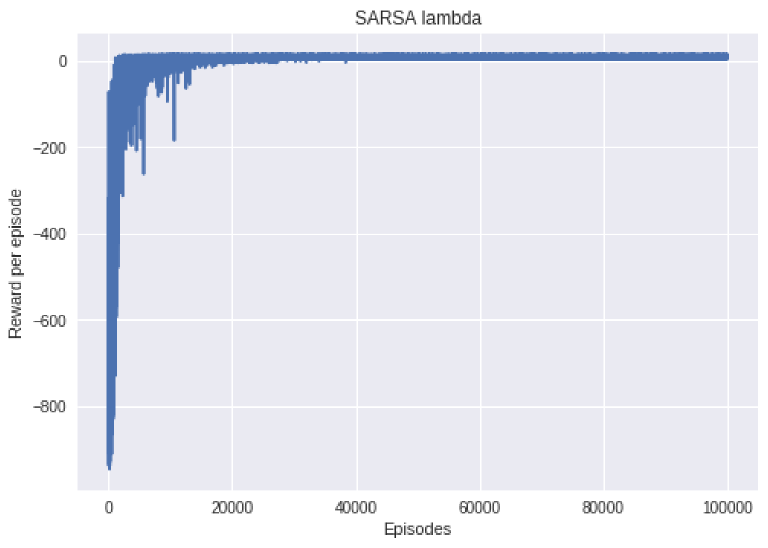

Figure 4: SARSA in 100000 iterations

Observations: SARSA(λ)

- Optimal λ is tuned to be as 0.9
- Changing ϵ of ϵ-greedy policy has significant effect on average rewards
- Decay factor of 0.99975 worked best
- After 100000, average time step per episode is 15.27 and average rewards per episode is 8.37

Q-Learning

- Simulation variant of Value Iteration.
- The update equation.
  $Q(s,a) = Q(s,a) + \alpha(R + \gamma \max_a^{'} Q(s^{'}, a^{'}) - Q(s,a))$
- Q-Learning is an off-policy algorithm and uses ϵ-greedy strategy to choose actions.
- We simulate a large number of episodes so that agent learns about the environment, and keeps the information in Q-Table.
- Each entry of the Q-Table tells the "quality" of action in a particular state.

Smart Q-Learning

- The hyper parameters in the update equation were tuned to maximize reward via multiple experiments.
- Parameter ϵ was observed to influence the results most, as it controls the exploration-exploitation tradeoff, especially in the training phase.
- Grid search technique was used select the best set of parameters.

Smart Q-Learning: Exploration-Exploitation Dilemma

- Q-Learning agent faces the vexed Exploration-Exploitation Dilemma, when trying to learn Q-values from the environment.
- ϵ-greedy strategy is used widely for this.
- We decay epsilon after every episode if the episodic reward is greater than the average reward, biasing the agent more towards exploitation.
- This has drastic effect on convergence rate of the algorithm, thereby on the number of episodes needed to train the agent.

Q-Learning: Results

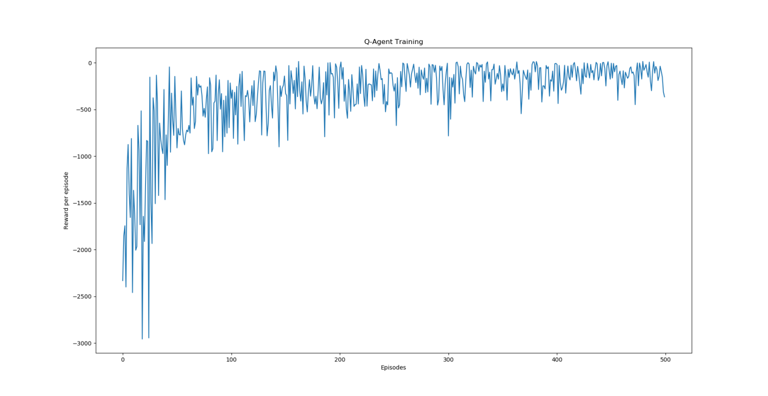

Figure 5: Reward v/s Episode for Q-Learning agent for first 500 episodes of training

Q-Learning: Results

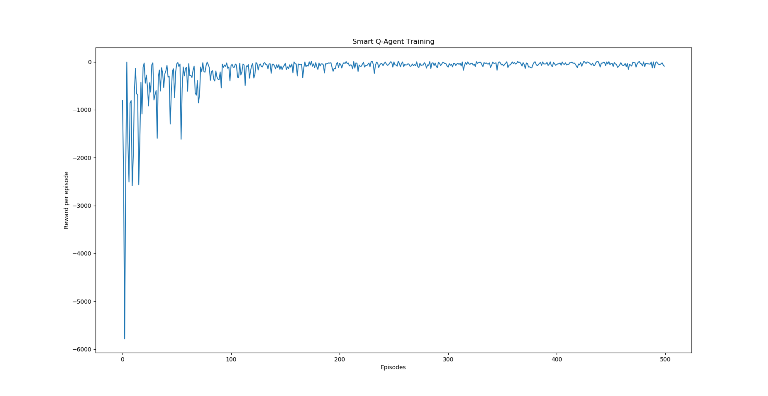

Figure 6: Reward v/s Episode for smart Q-Learning agent for first 500 episodes of training

Q-Learning: Results

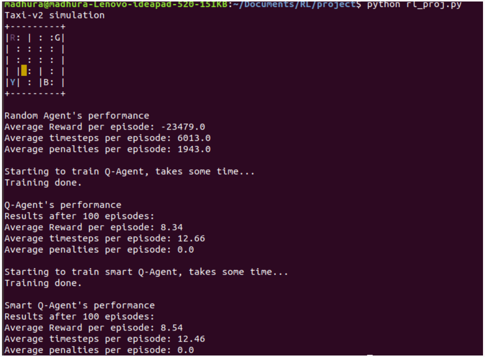

Figure 7: Results

- We observed with Smart Q-Learning agent, the Q-Table stabilized within very less iterations as compared to its usual counterpart.
- The average reward/score obtained per episode is 8.4, averaged over 100 episodes.

Future Work

- Hierarchical Reinforcement Algorithms can be applied to this problem because of its inherent structure. This problem originally was introduced for hierarchical RL setting by Dietterich[2000] [2].
- There is still scope of even finer tuning of parameters to get higher reward value.

References

- An Introduction. Second edition, in progress. Richard S. Sutton and Andrew G. Barto c 2014, 2015. A Bradford Book. The MIT Press. Cambridge, Massachusetts.
- T. G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. Journal of Artificial Intelligence Research, 13:227 303, 2000.
- D. P. Bertsekas and J. N. Tsitsiklis. Neuro-Dynamic Programming. Athena Scientific, 1996.
- Implementation References: https://gym.openai.com/envs/Taxi-v2/, https://gym.openai.com/evaluations/eval_9xUnOhbTkWuZyHDD9NpuQ/, https://github.com/VakhrameevaLiza, https://www.learndatasci.com/tutorials/

Thank You