Using Eligibility Traces to Find the Best Memoryless Policy in Partially Observable Markov Decision Processes

Alexandros Tzikas ALEXTZIK@STANFORD.EDU
Stanford University, Stanford, CA 94305

1 *Introduction*

Sarsa(λ) is a well-known reinforcement-learning, memoryless algorithm in Markov decision processes (MDPs). It is a modification to the traditional Sarsa algorithm, aiming to speed-up learning and assign rewards to state-action pairs in a more efficient manner, allowing for the current reward to influence the Q value of state-action pairs that were visited previously, but led to the reward. In this paper, the authors apply Sarsa(λ) to partially observable Markov decision processes (POMDPs) by considering observation-action pairs and empirically show that the algorithm can find the best, or at least a good, memoryless policy in various POMDP problems, performing comparably well with other methods. This contrasts the performance of other algorithms, such as Q-learning and the original Sarsa algorithm, which have been unsuccessful in POMDP settings [1,2]. The authors use the same Sarsa(λ) implementation as in MDPs, but modify the Q table to map observations to actions in the POMDP framework. In other words, the algorithm only uses information that is available to the agent. For problems where the baseline Sarsa(λ) algorithm performs poorly, the authors propose a variant that maps observation histories to actions, where an observation history is the current observation augmented with 1 or 2 previous observations. The augmented Sarsa(λ) is able to find a comparable policy to other far more computationally expensive algorithms in the problems studied.

[1] S. P. Singh, T. S. Jaakkola, and M. I. Jordan, "Learning Without State-Estimation in Partially Observable Markovian Decision Processes," in *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, 1994.

[2] M. L. Littman, "Memoryless Policies: Theoretical Limitations and Practical Results," in *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, 1994.

2 *Approach*

A POMDP can be represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathbf{T}, \mathbf{O}, R, \gamma)$, where $\mathcal{S}$ is the set of environment states, $\mathcal{A}$ is the set of possible actions by the agent and $\mathcal{O}$ is the set of the agent's observations. For Sarsa(λ), we assume finite $\mathcal{S}, \mathcal{A}$, and $\mathcal{O}$. Furthermore, $\mathbf{T}$ is the transition model, where $\mathbf{T}(s' \mid s, a) = \Pr(S_{t+1} = s' \mid S_t = s, A_t = a)$ is the probability of transitioning to state s', given that the agent is currently at s and has taken action a. Similarly $\mathbf{O}(o \mid s, a) = \Pr(O_{t+1} = o \mid S_{t+1} = s, A_t = a)$ is the probability of observing observation o given that the agent has transitioned to state s through action a. Finally, $R = R(s, a)$ represents the reward received when taking action a at s and γ is the discount factor for future rewards.

In POMDPs, memoryless Sarsa(λ) uses the currently observed tuple $(o_t, a_t, r_t, o_{t+1}, a_{t+1})$ to learn the optimal Q^* table for the observation-action pairs. $Q^\pi(o, a)$ represents the expected cumulative discounted reward after performing action a at o and then following policy π. Sarsa(λ) follows a behavior policy and updates the Q table associated with it. By using an ϵ-greedy policy with decreasing ϵ throughout the simulation, it is able to move toward the optimal policy and Q table. The basic difference between Sarsa and Sarsa(λ) is that Sarsa(λ) employs eligibility traces to propagate the current reward to observation-action pairs visited previously, but which led to the current reward. The current eligibility trace $\eta_t(o, a)$ associated with observation-action pair (o, a) indicates the magnitude of the update of $Q(o, a)$, based on the current reward.

The implementation of memoryless Sarsa(λ) for the POMDP settings, following the description outlined in the paper, is Algorithm 2.1. At first, the Q table $Q(\cdot)$ and the eligibility traces $\eta(\cdot)$ are initialized to 0. Also, an initial observation o_t is received. Then, until convergence of the Q table, we perform a step forward, receiving the current reward r_t, observing the next observation o_{t+1} and choosing the next action a_{t+1}, using the behavior policy. We calculate the temporal difference error between the current estimate $Q(o_t, a_t)$ and its updated estimate using the received reward r_t and the current $Q(o_{t+1}, a_{t+1})$. This temporal difference is an example of both Monte-Carlo sampling and bootstrapping, as mentioned by Sutton et al. [3], since it uses received samples and current estimates. The next step is to update the eligibility traces, setting the eligibility trace of the current pair (o_t, a_t) to 1 and scaling the rest through the exponential decay. Finally, note that the Q value of all (o, a) pairs is updated using the temporal difference error, although for each pair the update is scaled by its eligibility trace. This is in contrast to Q-learning and traditional Sarsa, where only the current observation-action pair is updated. In the algorithm, α refers to the learning rate, λ to the eligibility trace exponential decay. In the paper, $\alpha = 0.01$ and $\lambda = 0.9$. It is important to note that the eligibility traces are reinitialized to 0 at the end of each episode. During deployment, the learned memoryless policy is simply choosing $\text{argmax}_a Q(o_t, a)$, where o_t is the current observation. In other words, we use the greedy policy with respect to $Q(\cdot)$.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Massachusetts Institute of Technology, 2018.

The major differences between the implementation of this paper and the Sarsa(λ) implementation by Kochenderfer et al. [4] for MDPs are the following:

[4] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for Decision Making*. Massachusetts Institute of Technology, 2022.

- In the paper, the eligibility traces' maximum value is 1, whereas in the implementation by Kochenderfer et al. the visitation count is used as an eligibility trace
- In the paper, eligibility traces are first updated, prior to the Q table, opposite to the approach by Kochenderfer et al.

```
1:  $t \leftarrow 0$
2:  Initialize $Q_t$ and $\eta_{t-1}$ to 0 for all $(o, a)$ pairs
3:  Observe initial observation $o_t$
4:  Choose action $a_t$ based on $\epsilon$-greedy policy
5:  **while** $Q$ not converged **do**
6:      Obtain $r_t$ and $o_{t+1}$ from reward and observation model
7:      Choose $a_{t+1}$ based on $\epsilon$-greedy policy
8:      $\eta_t(o_t, a_t) = 1$
9:      **for** $o \neq o_t$ or $a \neq a_t$ **do**
10:         $\eta_t(o, a) = \gamma \lambda \eta_{t-1}(o, a)$
11:     $\delta_t = r_t + \gamma * Q_t(o_{t+1}, a_{t+1}) - Q_t(o_t, a_t)$
12:     **for** $o \in \mathcal{O}$ and $a \in \mathcal{A}$ **do**
13:         $Q_{t+1}(o, a) = Q_t(o, a) + \alpha * \delta_t * \eta_t(o, a)$
14:     $t \leftarrow t + 1$
15: **return** $Q$
```

Algorithm 2.1: Memoryless Sarsa(λ) for POMDPs

The authors of the paper also propose an augmented Sarsa(λ) implementation for POMDPs, where the current observation is augmented with the immediate previous 1 or 2 observations. This variant is shown in Algorithm 2.2. The only difference between Algorithm 2.2 and

Algorithm 2.1 is that in Algorithm 2.2 observation tuples $\bar{o}$ are mapped to actions. A major difficulty with this approach emerges in the beginning of each episode, when there is only one observation. It is important to assign the initial observation tuple in such a way as to not hinder the algorithm's performance or suggest the existence of false correlations between observations and actions. However, the paper does not describe the approach used.

```
1: $t \leftarrow 0$
2: Initialize $Q_t$ and $\eta_{t-1}$ to 0 for all $(\bar{o}, a)$ pairs, where $\bar{o} = \{o_1, \dots, o_{K+1}\}$
3: Observe initial observation $\bar{o}_t = \{o_{t-K}, \dots, o_t\}$
4: Choose action $a_t$ based on $\epsilon$-greedy policy
5: while $Q$ not converged do
6:     Obtain $r_t$ and $\bar{o}_{t+1}$ from reward and observation model
7:     Choose $a_{t+1}$ based on $\epsilon$-greedy policy
8:     $\eta_t(\bar{o}_t, a_t) = 1$
9:     for $\bar{o} \neq \bar{o}_t$ or $a \neq a_t$ do
10:        $\eta_t(\bar{o}, a) = \gamma\lambda\eta_{t-1}(\bar{o}, a)$
11:    $\delta_t = r_t + \gamma * Q_t(\bar{o}_{t+1}, a_{t+1}) - Q(\bar{o}_t, a_t)$
12:    for $\bar{o} \in O^K$ and $a \in A$ do
13:        $Q_{t+1}(\bar{o}, a) = Q_t(\bar{o}, a) + \alpha * \delta_t * \eta_t(\bar{o}, a)$
14:    $t \leftarrow t + 1$
15: return $Q$
```

Algorithm 2.2: Memory-based Sarsa(λ) for POMDPs

3 Implementation

Memoryless Sarsa(λ) was implemented, as indicated in the paper. Python and object-oriented programming are used. The algorithm was tested in Parr and Russell's 4×3 maze problem, which is described in the paper. The simulation environment, based on the problem, was also implemented. The results appear in Figure 1 and completely agree with the results presented in the paper. The code implementation can be found here: `https://github.com/alextzik/reinforcement_learning-2021/tree/main/SARSA_lambda`.

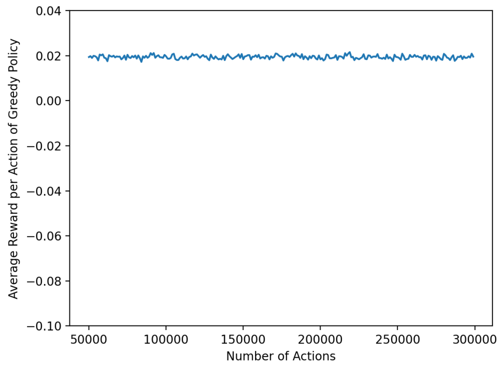

Figure 1. Memoryless Sarsa(λ) in Parr and Russell's grid world.

An implementation of the augmented Sarsa(λ) with 1 observation was also created (found in the aforemention Github repository) and the results are shown in Figure 2. The results

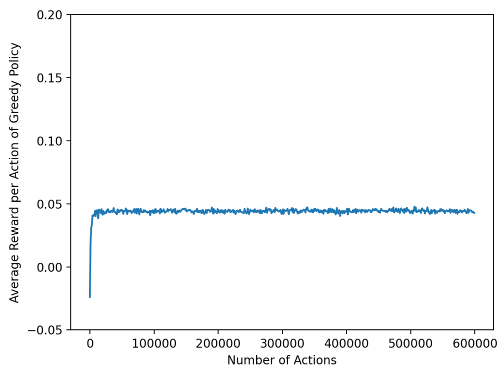

Figure 2. Sarsa(λ) augmented with 1 observation in Parr and Russell's grid world.

for this augmented version do not agree with those in the paper, although an improvement still exists compared to memoryless Sarsa(λ). One possible reason is the initialization of the observation tuple in the beginning of the episode or the re-initialization of the eligibility traces at the beginning of each episode. The paper does not provide details on the former and changes might have occurred in the latter.

4 *Discussion*

Although Sarsa(λ) is an algorithm designed for MDP settings, in this paper it is shown to perform well in POMDP problems as well. In particular, it is able to output a good policy for problems with good memoryless policies (i.e., policies that map immediate observations to actions and yield near-optimal return). In general, the algorithm is simple and easy to implement. However, when actually implementing the algorithm, some hidden difficulties arise. Details about the POMDP problems considered are missing from the paper, making it difficult to reproduce the results. For example, where the agent re-initializes at the end of each episode in Parr and Russell's grid world is not mentioned. The transition model in the same grid world is not adequately described and the metrics used, especially the average reward per step, is not easily understandable. It could refer to the mean of the rewards in each of the 101 trials or to the reward per step over the steps in all trials. The major difficulty however, arises when we try to implement the augmented Sarsa(λ) with 1 previous observation. The paper does not mention what the initial observation tuple is, which is a crucial element of the algorithm. This may be the main reason for the disagreement between ours and the paper's results.

It is important to note that, although the results of the paper seem promising, they should be received with caution. Firstly, it is not known when a real POMDP problem has a good memoryless policy or what characteristics the problem must possess for a good memoryless policy to exist. It is thus not easy to check if our problem falls in this category and thus if Sarsa(λ) is suitable. It is also discouraging that even small problems can have bad memoryless policies. For these reasons, although Sarsa(λ) finds a good (or best) memoryless policy, it is questionable if it can tackle real-world problems. Secondly, although the authors propose the augmented Sarsa(λ) for problems with bad memoryless policies, the difference between this approach and other state estimation algorithms is unclear. In addition, mapping observation tuples to actions requires more memory and is more computational costly. We notice a memory occupancy of $O(|\mathcal{O}|^K|A|)$, which is exponential in $\mathcal{O}$. Computation time scales in the same way, because all observation tuple-action pairs are updated in each iteration. Finally, the number of previous observations that need to be added to the history for Sarsa(λ) to perform well might not be small for real-world problems, further limiting the algorithm's applicability.

References

1. M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for Decision Making*. Massachusetts Institute of Technology, 2022.
2. M. L. Littman, ''Memoryless Policies: Theoretical Limitations and Practical Results,'' in *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, 1994.
3. S. P. Singh, T. S. Jaakkola, and M. I. Jordan, ''Learning Without State-Estimation in Partially Observable Markovian Decision Processes,'' in *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, 1994.
4. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Massachusetts Institute of Technology, 2018.