Trading off Sampling and Taking Expectations in the n-step Q(σ) Algorithm

Kristopher De Asis, Zichen Zhang
Department of Computing Science, University of Alberta
{kldeasis, zichen2}@ualberta.ca

1 Introduction

The n-step Q(σ) algorithm is a proposed algorithm that allows a reinforcement learning agent to either learn from actions it has committed to (sampling) or to learn from all of its potential actions in a particular state weighted by the probability of taking each action (taking expectations) [1]. This behavior is controlled by the parameter $\sigma_t \in [0, 1]$, where at time step t, $\sigma = 1$ denotes that an agent is to take a sample, and $\sigma = 0$ denotes that an agent is to take an expectation. If an agent were to take a sample at every time step, it is equivalent to the n-step SARSA algorithm. If it were only to take expectations (no sampling), it is equivalent to the n-step Tree Backup algorithm. In this paper, we explore the effects of varying the degree of sampling.

To vary the degree of sampling, we propose looking at an n-step Q(σ) agent that randomly selects whether it is to take a sample or an expectation. The ratio of samples and expectations can be tuned by setting the agent's probability of taking a sample $P(\sigma = 1)$. Varying this ratio can be interpreted as interpolating between the n-step SARSA and Tree-Backup algorithms.

2 Experimental Setup

We ran our experiments on a 1-dimensional 19-state random walk with a terminal state on each end. We followed the same setup as in the Example 7.1 in the textbook [1]: terminating on one end receives a reward of -1, terminating on the other end receives a reward of 1, and all other transitions have a reward of 0. The structure of the Markov decision process (MDP) can be seen in Figure 1.

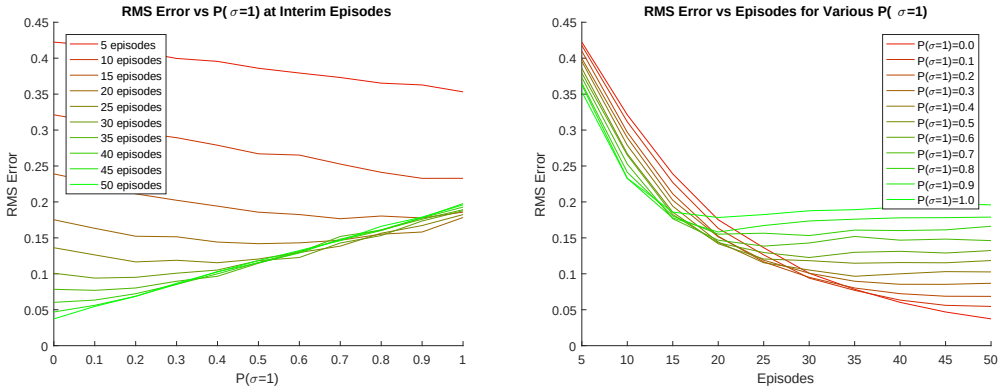
Figure 1: Random walk MDP

The agent behaves on this under the equiprobable random policy with a discount rate of $\gamma = 1$, and learns the action-value function for this behavior policy (on-policy). We then compute the RMS error between its learned action-value function $Q(s, a)$ and the true values under this policy after varying numbers of elapsed episodes.

3 Experiments and Results

To examine the effects of varying $P(\sigma = 1)$, we fix the multi-step parameter n, as well as the step size α. In this experiment, we arbitrarily selected $n = 3$ and $\alpha = 0.5$, and the results are averaged over 250 runs. Figure 2 compares the RMS error of different $P(\sigma = 1)$ settings after varying numbers

of episodes. Based on these plots, it can be seen that different settings for $P(\sigma = 1)$ achieve a lower RMS error depending on how many episodes have elapsed. For earlier episodes, always sampling performs better, and for later episodes, always taking expectations performs better.



(a) RMS Error vs. $P(\sigma = 1)$

(b) RMS Error vs. Episodes

Figure 2: Comparing the RMS error of fixed settings of P(σ=1) at various interim episodes

If we plot the best setting of $P(\sigma = 1)$ over the number of elapsed episodes, we can observe that it has an exponentially decaying trend in Figure 3.

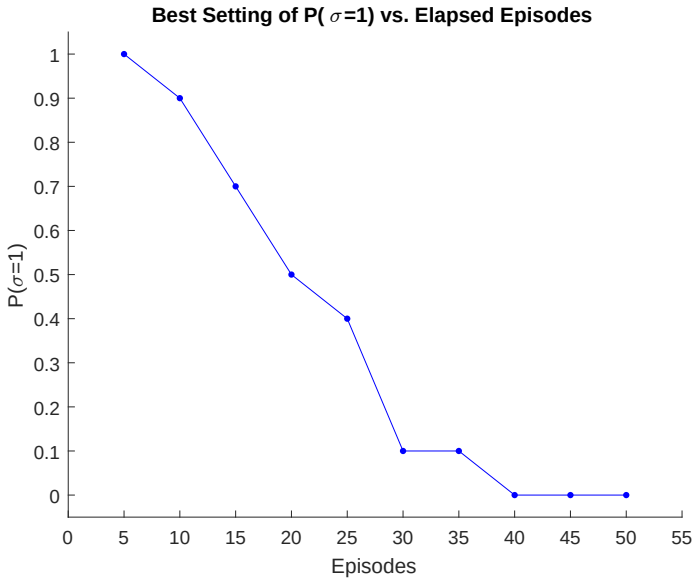
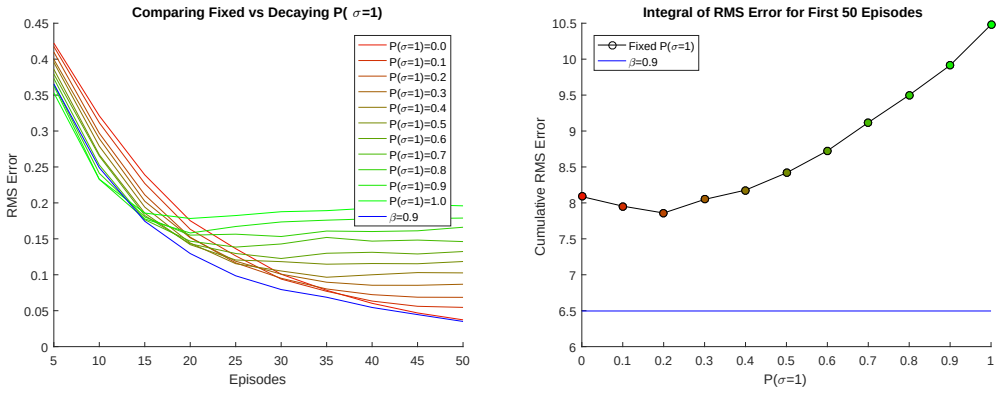


Figure 3: Best $P(\sigma = 1)$ setting for given numbers of elapsed episodes

Based on this, we tested an agent that is initialized with $P(\sigma = 1) = 1$, and after each episode, $P(\sigma = 1)$ gets multiplied by a decay factor $\beta \in [0, 1)$. Figure 4 compares the RMS error of this approach using a decay factor $\beta = 0.9$ to the fixed settings of $P(\sigma = 1)$. It can be seen that adjusting the degree of sampling per episode can converge faster to an accurate estimate of the action-value function than fixed degrees of sampling. Note that the RMS error was computed at every 5th episode, and the 'Cumulative RMS Error' is the approximate integral with respect to episodes using a Riemann sum.



(a) RMS Error at various interim episodes

(b) Integral of RMS Error for first 50 episodes

Figure 4: Comparison of the RMS error of fixed $P(\sigma = 1)$ and decaying P(σ=1)

4 Discussions and Conclusions

In this paper we examined the effects of trading off sampling and taking expectations in the Q(σ) algorithm. We proposed a simple method of setting the degree of sampling by having an agent select whether it is to sample with probability $P(\sigma = 1)$. By examining the effects of different settings of $P(\sigma = 1)$ on the RMS error at varying numbers of elapsed episodes, we showed that different degrees of sampling perform best based on how many episodes have elapsed. We then showed that an exponentially decaying $P(\sigma = 1)$ could converge quicker to a better estimate of the action-value function.

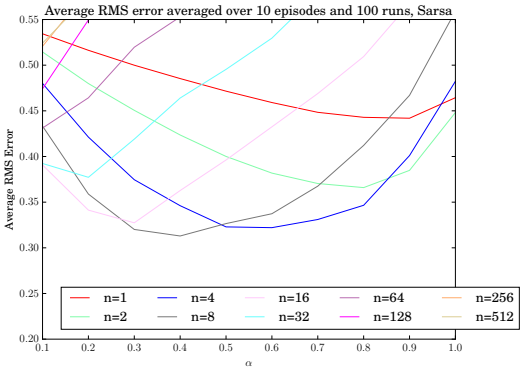
An interpretation of this result is that the SARSA algorithm (full sampling) performs better on earlier episodes, and the Tree-Backup algorithm (no sampling) performs better on later episodes. An intuition behind this is that while both algorithms perform updates based on a current estimate of the value function (bootstrapping), taking an expectation depends on the estimates for all possible actions in the current state while sampling depends only on the estimate of the action that the agent committed to. Due to this, taking an expectation could be relying on several estimated values to be accurate, while sampling only relies on one estimated value to be accurate. In earlier episodes, the estimated action-value function is likely to be inaccurate that updating based on sampling tends to give a more accurate update than taking an expectation. In later episodes, taking an expectation is better because the estimated action-value function is relatively more accurate. Based on this intuition, the optimal degree of sampling depends on the accuracy of the current estimated value function, which depends on how many updates have been performed (or the number of elapsed episodes).

References

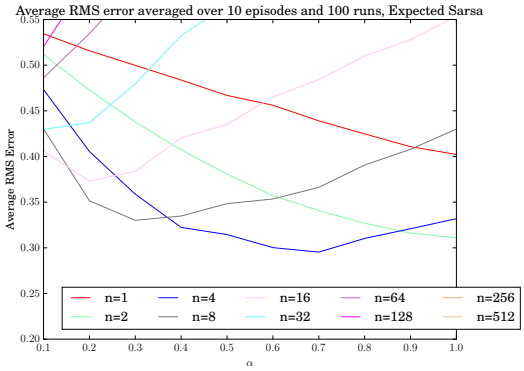
[1] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. 2nd (in progress) edition, 2016. 1, 4

Appendix

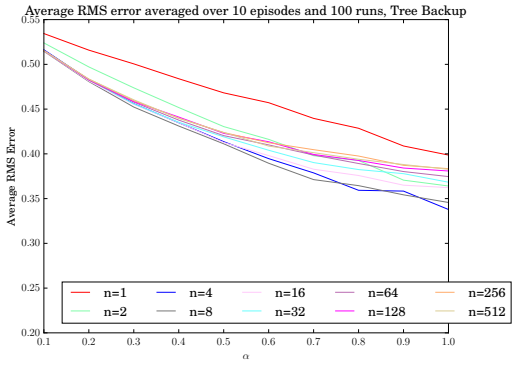
As a sanity check to ensure that we implemented things correctly, also to compare with the performance plots for the random walk MDP given in the textbook [1], we looked into the performance of different multi-step reinforcement learning algorithms with various n and α. These results are included in the appendix as an exploration study as to not obtrude the main findings of this paper.



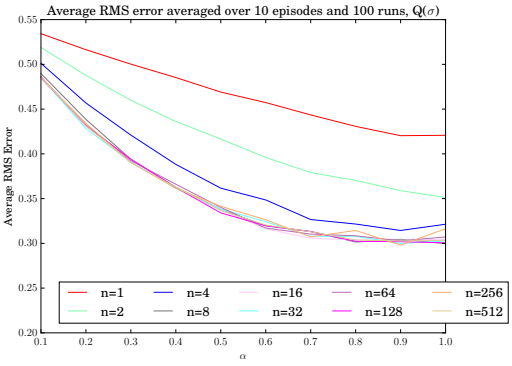
(a) SARSA



(b) Expected SARSA

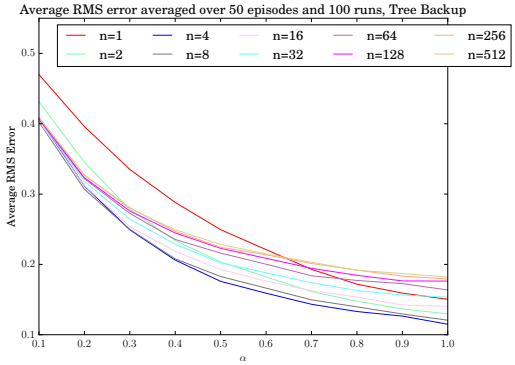


(c) Tree-Backup

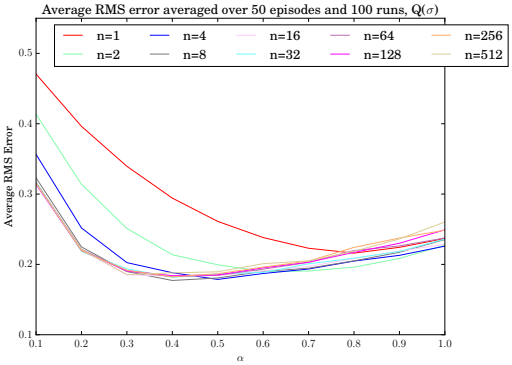


(d) Q(σ)

Figure 5: Average RMS error as a function of α and n, for 10 episodes



(a) Tree-Backup for 50 episodes



(b) Q(σ) for 50 episodes

Figure 6: Average RMS error as a function of α and n, for 50 episodes

To understand the effect of α and n in multi-step reinforcement learning algorithms, we plotted the average RMS error of the action values as a function of various α and n. Fig. 5 shows the result of four methods: SARSA, Expected SARSA, Tree-Backup, and Q(σ) with $P(\sigma = 1) = 0.5$ (Equiprobable chance of sampling or

taking an expectation). Following the settings in the textbook, the RMS error shown in the figure is an average of 100 runs for 10 episodes. We also look at the RMS error after 50 episodes averaged over 100 runs for the Tree-Backup and Q(σ) algorithms in Figure 6. They show the same trends as in the book, but the U-shaped curves are not evident until later episodes for the Tree-Backup and Q(σ) algorithms. A reason for this is the degree of sampling of each algorithm. Seeing a U-shaped curve is indicative that certain settings of α will have reached the steady-state error by that number of episodes, and from the results in Figure 4a, taking more samples (and fewer expectations) results in reaching the steady-state error quicker. It can be seen that full sampling (SARSA) sees the U-shaped curves earliest, and taking a single expectation at the end (Expected SARSA) sees these curves a few episodes later. Taking expectations 50% of the time (Q(σ) with $P(\sigma = 1) = 0.5$) starts to have U-shaped curves at around 50 episodes, and only taking expectations (Tree-Backup) has yet to reach the steady-state error by this number of episodes.