How Close is Close Enough? Finding Optimal Policies in PAC-style Reinforcement Learning

Emma Brunskill
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
emma@csail.mit.edu

Abstract

There has been recent interest in providing formal guarantees on the finite-sample performance of reinforcement learning algorithms. Existing approaches focus on ensuring the algorithm selects actions whose values are ϵ-close to the optimal actions' values. However, in robotics and many other domains, the primary objective is to follow the optimal policy, rather than estimating the precise value of that policy. Here we present preliminary work on an iterative algorithm that dynamically finds the number of samples needed to ensure the optimal policy is followed.

In reinforcement learning (RL) an agent must learn how to act in an unknown environment in order to maximize some function of the reward it receives. To do this the agent must balance between exploration actions that provide it with a better model of the world dynamics and rewards, and exploitation actions to maximize the agent's reward given its current knowledge of the world models. The exploration-exploitation tradeoff is one of the central challenges in reinforcement learning.

In our work we are interested in RL algorithms that have some precise notion of optimality. For example, in the limit of infinite training data, under appropriate conditions, Q-learning [7] is guaranteed to converge to the optimal policy; however no guarantees are made about performance with finite training samples. In contrast, Bayesian RL approaches often set the problem as a partially observable Markov decision process (POMDP) where the model parameters are part of the hidden state (e.g. [6]). A compelling aspect of this approach is that if an optimal solution to the POMDP is computed, such algorithms would yield a policy that optimally trades exploration and exploitation at every step. However, this behavior is not typically achieved since generally only approximate solutions to the POMDP are calculated, and there are usually no bounds given on the performance of these approximate solutions. Instead, probably approximately correct (PAC) RL (e.g. [1]) takes a middle ground approach. It guarantees that for a given ϵ and δ, with probability at least $1 - \delta$, an algorithm will choose an action whose value is ϵ-close to the value of the optimal action, on all but N steps, where N is a function of ϵ, δ, and the problem parameters. In doing so it makes no statement on the optimality of actions taken during those N steps, but ensures that the long term performance is close to optimal.

One challenge in using a PAC-style reinforcement learning algorithm, such as R-max [1], is how to set ϵ. This choice of how close to the optimal value the algorithm's chosen actions will be, has a direct relation to the number of steps needed (N). But in many domains, such as robotics, the final value of the actions performed is not important: instead the objective is to learn to choose the correct action, that is to find the optimal policy. In such scenarios a more suitable goal is to determine the number of samples required to guarantee the agent will follow the optimal policy with probability at least 1-δ. Focusing on this objective may also require a smaller number of samples than is required for highly accurate value estimates, as we will see shortly. In our domains of interest requiring fewer samples is of particular importance, since it is hard and/or expensive to get real training examples, and building a simulator of the domain is nontrivial.

As a step towards this objective, we created an iterative RL algorithm that dynamically sets the number of samples N and stops iterating when the policy has converged to the optimal policy[1]. In the experiment presented below the policy converges significantly before the value function, supporting our hypothesis. As input the algorithm takes a user-specified δ. The number of samples per state-action tuple, N_{sa}, is first set to a low value, and the agent acts using the R-max algorithm, breaking ties randomly between actions with identical Q-values, until all state-action pairs have been tried at least N_{sa} times. At this point we compute the error in the resulting Q-values, which are all now computed using the estimated dynamics model. The maximum difference Δ_Q between the Q-values of two finite state and action MDPs is bounded above by a function of the discount factor γ and the L_1 distance between their dynamics models T_1 and T_2: $\Delta_Q \leq \gamma L_1(T_1, T_2)/(1-\gamma)^2$. To compute an upper bound on the L_1 distance between the estimated and true dynamics models, we estimate confidence bounds on the model parameters. Hoeffding's inequality is distribution independent and can be applied but since the dynamics models here are multinomials, we use the multinomial-specific confidence intervals developed by Goodman [4]. We then use this computed potential error in the Q-values (Δ_Q) to see if the optimal policy for the current estimated Q-values changes. If it doesn't change, then the policy is fixed and iteration stops, leaving the algorithm to continue taking only exploitation actions. If the policy does change, that indicates the uncertainty in the model parameter estimates is still sufficiently large that the optimal policy could be different than the current estimated policy. In this case $N = |S| \cdot |A| \cdot N_{sa}$ is increased and the next iteration commences. The iterations can also halt when Δ_Q reaches a minimum threshold: this can be used if the user has an idea of the minimal difference in values he/she cares about, and to prevent the algorithm from attempting to distinguish between two actions with extremely close values.

We ran our algorithm on the 9-state, 2-action loop MDP (see [2] for a description) with γ set to 0.95 and δ set to 0.5. After the iteration where N was set to 10^5 samples for each state-action pair, the change in Q-values Δ_Q due to model estimation was sufficiently small that the policy was unaffected, and so iteration halted. At this point Δ_Q, or ϵ, was approximately 0.15. In contrast, if ϵ has been chosen in advance to be 0.01, then the number of samples required according to PAC bounds would be over 10^6, an order of magnitude more than needed in the iterative approach. In advance it is often hard to know the accuracy needed to ensure the optimal policy will be followed, and so this approach provides a way to slowly grow the number of samples until this is achieved.

Though this algorithm ensures with probability at least $1-\delta$ that the optimal policy will be followed[2] the number of samples required is still prohibitively large for many domains. In practice a far smaller number of samples than predicted by PAC theory have been found to give good results on real problems (e.g. [5]). In another experiment with the approach proposed in this paper we found that though the difference between the true Q-values would predict that over 10^6 samples would be required, the policy extracted from the estimated Q values stopped changing after $N = 100$. We hope to draw from related work on percentile optimization [3] and robust decision making to better understand this gulf between theory and practice, and develop reinforcement learning algorithms that learn faster and still maintain finite-sample performance guarantees.

References

[1] R. Brafman and M. Tennenholtz. R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

[2] R. Dearden, N. Friedman, and S. Russell. Bayesian Q-learning. In *AAAI*, 1998.

[3] E. Delage and S. Mannor. Percentile optimization in uncertain Markov decision processes with application to efficient exploration. In *ICML*, 2007.

[4] L. Goodman. On simultaneous confidence intervals for multinomial proportions. *Technometrics*, 7:247–254, 1965.

[5] B. Leffler, M. Littman, and T. Edmunds. Efficient reinforcement learning with relocatable action models. In *AAAI*, 2007.

[6] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *ICML*, 2006.

[7] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[1] We assume that rewards R are bounded, and given that, w.l.o.g. that all $R \in [0, 1]$.

[2] Or if the algorithm halts because a minimum Δ_Q is reached, that the policy followed will be Δ_Q-optimal.