

PAC Model-Free Reinforcement Learning

Alexander L. Strehl, Lihong Li, Eric Wiewiora,
John Langford, Michael L. Littman

strehl@cs.rutgers.edu, lihong@cs.rutgers.edu, ewiewior@cs.ucsd.edu,
jl@hunch.net, mlittman@cs.rutgers.edu

In the reinforcement-learning (RL) problem (Sutton and Barto, 1998), an agent acts in an unknown or incompletely known environment with the goal of maximizing an external reward signal. One of the fundamental obstacles in RL is the exploration-exploitation dilemma: whether to act to gain new information (explore) or to act consistently with past experience to maximize reward (exploit). This paper models the RL problem as a Markov Decision Process (MDP) environment with finite state and action spaces. Learning takes place from a single continuous thread of experience—no resets nor parallel sampling is used. This abstract is adapted from a paper presented at ICML (Strehl et al., 2006).

When evaluating RL algorithms, there are three essential traits to consider: *space complexity*, *computational complexity*, and *sample complexity*. We define a *timestep* to be a single interaction with the environment. Space complexity measures the amount of memory required to implement the algorithm while computational complexity measures the amount of operations needed to execute the algorithm, per timestep. Sample complexity measures the amount of timesteps for which the algorithm does not behave near optimally or, in other words, the amount of experience it takes to learn to behave well.

We will call algorithms whose sample complexity can be bounded by a polynomial in the environment size and approximation parameters, with high probability, *PAC-MDP* (*Probably Approximately Correct in Markov Decision Processes*). All algorithms known to be PAC-MDP to date involve the maintenance and solution (often by value iteration or mathematical programming) of an internal MDP model. Such algorithms, including R_{\max} (Brafman and Tennenholtz, 2002), and E^3 (Kearns and Singh, 2002), are called *model-based algorithms* and have relatively high space and computational complexities. Another class of algorithms, including most forms of Q-learning, make no effort to learn a model and can be called *model free*.

The hardness of learning an arbitrary MDP as measured by sample complexity

is still relatively unexplored. For simplicity, we let $\tilde{O}(\cdot)$ ($\tilde{\Omega}(\cdot)$) represent $O(\cdot)$ ($\Omega(\cdot)$) where logarithmic factors are ignored. When we consider only the dependence on S (the number of states) and A (the number of actions), the lower bound of Kakade (2003) says that with high probability, the sample complexity of any algorithm will be $\tilde{\Omega}(SA)$. However, the best upper bound known provides an algorithm whose sample complexity is $\tilde{O}(S^2A)$ with high probability. In other words, there are algorithms whose sample complexity is known to be no greater than approximately the number of bits required to specify an MDP to fixed precision. However, there has been no argument proving that learning to act near-optimally takes as long as approximating the dynamics of an MDP. We solve this open problem, first posed by Kakade (2003), by showing that a new algorithm, *Delayed Q-learning*, has sample complexity $\tilde{O}(SA)$, with high probability. Delayed Q-learning is model free and its per-experience computation cost is $O(\ln(A))$, which is much less than that of previous PAC algorithms.

Delayed Q-learning maintains Q-value estimates $Q(s, a)$ for each state-action pair (s, a) , and always acts greedily with respect to these estimates. After (s, a) has been experienced m times (m is a parameter to the algorithm), an update to $Q(s, a)$ may be attempted by the algorithm. It succeeds only if the new estimate is significantly smaller than the current one. The update itself is an average of m standard Bellman backups (as used in value iteration) plus a small bonus. The averaging step is important as it mitigates some of the effects of randomness and, when combined with the bonus, achieves optimism ($Q(s, a) \geq Q^*(s, a)$) with high probability. Using some internal logic, the algorithm ensures that only a finite number of attempted updates will occur. With a careful selection of the parameter m , polynomial in the standard inputs, we can prove that Delayed Q-learning will act near-optimally on all but $\tilde{O}(s, a)$ timesteps, with high probability.

References

- Brafman, R. I., Tenenbholz, M., 2002. R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3, 213–231.
- Kakade, S. M., 2003. On the sample complexity of reinforcement learning. Ph.D. thesis, Gatsby Computational Neuroscience Unit, University College London.
- Kearns, M. J., Singh, S. P., 2002. Near-optimal reinforcement learning in polynomial time. *Machine Learning* 49 (2–3), 209–232.
- Strehl, A. L., Li, L., Wiewiora, E., Langford, J., Littman, M. L., 2006. Pac model-free reinforcement learning. In: *ICML-06: Proceedings of the 23rd international conference on Machine learning*. pp. 881–888.
- Sutton, R. S., Barto, A. G., 1998. *Reinforcement Learning: An Introduction*. The MIT Press.