Book Review

Self-Learning Control of Finite Markov Chains

by A. S. Poznyak, K. Najim, and E. Gómez-Ramírez

Review by Benjamin Van Roy

This book presents a collection of work on algorithms for learning in Markov decision processes. The problem addressed is very similar in spirit to "the reinforcement learning problem," which has become a central topic in artificial intelligence (see, e.g., [8]). However, the book reflects an entirely different thread of research, and there has been surprisingly little interaction between this and the reinforcement learning thread. A primary objective of this book review is to encourage a cross-fertilization of ideas.

1 Problem Formulation

The authors consider a discrete-time Markov decision process with a finite state space S and a finite action space U. Dynamics are defined by transition probabilities $p_{xy}(u)$, each representing the probability that the next state is $y \in S$ given that the current state is $x \in S$ and an action $u \in U$ is taken. The authors make use of different assumptions about the transition probabilities in various parts of the book. For simplicity, let us assume in our discussion the strongest among these assumptions – that if every decision is selected with equal probability at each time step, the resulting Markov chain is irreducible and aperiodic. At each time step $t = 1, 2, 3, \ldots$, a scalar cost η_t is incurred. It is assumed that $|\eta_t|$ bounded by some known quantity $\overline{\eta}$ and that the cost η_t is conditionally independent of all other information available at time t, given the current state x_t and decision u_t. We denote the conditional expectation by $g(x,u) = E[\eta_t | x_t = x, u_t = u]$.

A control policy is represented in terms of a probability distribution μ over $S \times U$. Each $\mu(x,u)$ represents the "steady-state" probability that the process is in state x and action u is taken. Hence, given a current state x, the policy represented in this form generates an action randomly, selecting each $u \in U$ with probability $\mu(x,u)/\sum_u \mu(x,u)$. If the transition probabilities and cost function are known, standard dynamic programming algorithms can be used to compute a policy μ^* that minimizes average cost. For instance, an optimal policy μ^* can be obtained by solving the linear program:

$$\begin{array}{lll} \text{minimize} & \sum_{x\in S, u\in U} g(x,u)\mu(x,u) & \\ \text{subject to} & \sum_{u\in U} \mu(y,u) = \sum_{x\in S, u\in U} \mu(x,u) p_{xy}(u), & \forall y \in S, \\ & \mu(x,u) \geq 0, & \forall x \in S, u \in U, \\ & \sum_{x\in S, u\in U} \mu(x,u) = 1. & \end{array} \tag{1}$$

Let us denote the minimal average cost – i.e., the optimal value of this linear program – by g^*.

The authors focus on situations where the costs $g(x,u)$ and transition probabilities $p_{xy}(u)$ are unknown. Each algorithm presented generates a sequence of actions $u_1, u_2, u_3, \ldots$ over time as the system evolves. At each time $t = 1, 2, 3, \ldots$, a policy μ_t is generated based on past experience $(x_1, u_1, \eta_1, x_2, u_2, \eta_2, \ldots, x_{t-1}, u_{t-1}, \eta_{t-1}, x_t)$, without further knowledge of the cost function or transition probabilities, and this policy is used to generate the next decision u_t. The goal is to have the average cost $\overline{g}_t = \sum_{\tau=1}^{t} \eta_t / t$ converge to g^* quickly.

Note that various algorithms studied in the reinforcement learning literature fit into this framework. For example, for certain on-line versions of Q-learning and SARSA with "decaying exploration," policies generated are guaranteed to converge to optimal ones [7] (a technical difference in this work is that optimality is defined in terms of a discounted cost metric). Similar results can be established for actor-critic algorithms based on existing reinforcement learning theory [2, 6]. Model-based reinforcement learning algorithms (see, e.g., [4]) also converge in this way, as does the recently proposed E^3 algorithm [5].

2 Algorithms

The authors present three algorithms. Each is motivated by a classical approach to constrained optimization and represents a stochastic variant of the optimization method. The algorithms are based on:

1. **Lagrange Multipliers.** The first approach involves a regularized Lagrangian function:

$$\begin{aligned} L_\delta(\mu, \lambda) &= \sum_{x \in S, u \in U} g(x,u)\mu(x,u) - \sum_{y \in S} \lambda(y) \left(\sum_{u \in U} \mu(y,u) - \sum_{x \in S, u \in U} \mu(x,u) p_{xy}(u) \right) \\ &\quad + \frac{\delta}{2} \left(\sum_{x \in S, u \in U} \mu^2(x,u) - \sum_{y \in S} \lambda^2(y) \right). \end{aligned}$$

   For any $\delta > 0$, this function is strictly convex in μ and strictly concave in λ. Hence, for any $\delta > 0$, there is a unique saddle point $(\mu^*_\delta, \lambda^*_\delta) \in \Delta \times \Re^{|S|}$, where Δ denotes the unit simplex. Furthermore, μ^*_δ converges to an optimal policy as δ goes to 0. The authors define a Lyapunov function

$$W_t(\mu, \lambda) = \sum_{x \in S, u \in U} (\mu(x,u) - \mu^*_{\delta_t}(x,u))^2 + \sum_{y \in S} (\lambda(y) - \lambda^*_{\delta_t}(y))^2,$$

   for a decreasing sequence δ_t, and design a stochastic algorithm that generates a sequence (μ_t, λ_t) aimed at reducing $W_t(\mu_t, \lambda_t)$ over time.

2. **Penalty Functions.** The second approach makes use of a regularized penalty function:

$$\begin{aligned} P_{\beta,\delta}(\mu) &= \beta \sum_{x \in S, u \in U} g(x,u)\mu(x,u) + \frac{1}{2} \sum_{y \in S} \left(\sum_{u \in U} \mu(y,u) - \sum_{x \in S, u \in U} \mu(x,u) p_{xy}(u) \right)^2 \\ &\quad + \frac{\delta}{2} \sum_{x \in S, u \in U} \mu^2(x,u). \end{aligned}$$

   For $\beta, \delta > 0$, there is a unique minimum $\mu^*_{\beta,\delta}$ in Δ, and as β and δ approach 0, $\mu^*_{\beta,\delta}$ becomes an optimal policy. In this case, the authors define a Lyapunov function

$$W_t(\mu) = \sum_{x \in S, u \in U} (\mu(x,u) - \mu^*_{\beta_t,\delta_t}(x,u))^2,$$

   for decreasing sequences β_t and δ_t, and employ a stochastic gradient algorithm that aims at reducing $W_t(\mu_t)$ over time.

3. **Gradient Projection.** Let M denote the set of policies (i.e., the set of functions μ satisfying the constraints of the linear program (1)). Furthermore, let M_ϵ be the subset of M consisting of elements μ with $\mu(x,u) \geq \epsilon$ for all x, u. A gradient projection algorithm is given by

$$\mu_{t+1} = \Pi_{M_{\epsilon_t}}(\mu_t - \gamma_t g),$$

where γ_t is a small step size, $\Pi_{M_{\epsilon_t}}$ is the operator that projects onto the set M_{ϵ_t}, and ϵ_t is a diminishing sequence (also, recall that g and μ_t are functions mapping $S \times U$ to $\Re$). Such a gradient projection algorithm requires knowledge of the transition probabilities $p_{xy}(u)$ and the cost function g. The authors define a stochastic version of this algorithm in which the transition probabilities and the cost function are estimated based on experience. The estimates of transition probabilities are used at each point in time to generate an estimate $\hat{M}_t$ of M_{ϵ_t}. The estimated cost function $\hat{g}_t$ and $\hat{M}_t$ are then used to update the policy, based on a projected gradient step:

$$\mu_{t+1} = \Pi_{\hat{M}_t}(\mu_t - \gamma_t \hat{g}).$$

Reinforcement learning algorithms are typically classified as *model-based* or *model-free*. Model-based algorithms iteratively estimate transition probabilities and the cost function and derive control policies based on these evolving estimates. Model-free algorithms do not estimate transition probabilities or the cost function; instead, they maintain and update another data structure, which generally represents either a policy or a value function or both. Given this terminology, the first two approaches developed in the book would be considered model-free, whereas the third is model-based.

Another division distinguishes classes of model-free algorithms. There are three types:

1. **Value function methods.** These algorithms maintain and update an estimate of a dynamic programming value function, and use it to generate actions.

2. **Policy search methods.** Such algorithms maintain and update a representation of the policy.

3. **Actor-critic methods.** These maintain and update both a value function and a policy. The policy dictates actions, while the value function assists in updating the policy.

Clearly, the algorithm based on a penalty function falls in the second category. The algorithm based on Lagrange multipliers, on the other hand, is an actor-critic method, as we now explain. The dual of the linear program (1) is given by

$$\begin{array}{ll} \text{maximize} & \tilde{g} \\ \text{subject to} & g(x,u) - \tilde{g} + \sum_{y\in S} p_{xy}(u) J(y) \geq J(x), \qquad \forall x \in S, u \in U, \end{array}$$

where the maximization is over variables $\tilde{g} \in \Re$ and $J(x) \in \Re$ for all $x \in S$. Optimal dual variables have natural interpretations: $\tilde{g}$ is the optimal average cost and J is a differential value function. Note that the dual variables $J(x)$ correspond to Lagrange multipliers associated with the constraints

$$\sum_{u\in U} \mu(y,u) = \sum_{x\in S, u\in U} \mu(x,u) p_{xy}(u), \quad \forall y \in S,$$

from the primal linear program. Hence, the Lagrange multipliers $\lambda(x)$ introduced in the first algorithm can be interpreted as differential values, and their role in updating the policy makes the algorithm an actor-critic method.

3 Convergence Analysis

The authors offer convergence proofs for each of their algorithms, showing that the average cost converges to the optimal, under various technical conditions. They establish both almost sure convergence and convergence of expected squared error. The proofs are built on stochastic approximation theory, much like that employed in convergence proofs for reinforcement learning algorithms (see [1], for examples of such convergence proofs from the reinforcement learning literature).

The authors also establish bounds on convergence rates. Let us discuss how the authors think about convergence rates. Define the convergence rate of $\overline{g}_t$ as the largest scalar ρ^* such that $\limsup_{t\to\infty} |\overline{g}_t - g^*|t^\rho < \infty$, with probability one, for all $\rho > \rho^*$. When an optimal policy μ^* is employed to generate actions u_t at every time step, the convergence rate is $1/2$. This is the best such rate that can be achieved and provides a baseline for comparison against learning algorithms.

The authors place a common lower bound of $1/3$ on the convergence rates of both of their model-free algorithms. For the model-based algorithm, the convergence rate is shown to be at least $1/5$. Though the authors do not do this explicitly, it may be possible to show that these bounds are tight, in which case we would conclude that the model-free methods of the book offer a better convergence rate than the model-based one.

There has been a discussion going on for years in reinforcement learning circles about whether there are fundamental advantages to model-free or model-based approaches. Though the bounds from this book may not offer any conclusive insight into this issue, it is intriguing that the particular model-free approaches presented in the book share the same convergence rate bound, while the model-based approach presents an inferior one.

Convergence rate analysis of the type presented in this book has not been a common practice in the reinforcement learning literature. It would be interesting to study such convergence rates for reinforcement learning algorithms, such as versions of Q-learning and SARSA discussed in [7], actor-critic algorithms studied in [2, 6], and the E^3 algorithm [5], and to see how they compare against those associated with algorithms of this book.

Recently, there has been a fair amount of interest in reinforcement learning in a different measure of efficiency, introduced by Kearns and Singh [5] (a similar notion was also introduced by Fiechter [3]). In this context, efficiency calls for $\overline{g}_t$ to be within ϵ of g^* with probability $1-\delta$ at a time t that is polynomial in $1/\epsilon$, $1/\delta$, $|S|$, $|U|$, $\overline{\eta}$, and the worst-case "mixing time" of the process. The E^3 algorithm presented in [5] is shown to be efficient in this sense. Both the metric of efficiency and the design of the E^3 algorithm bring to the fore a trade-off between "exploration" and "exploitation." In particular, to be efficient, an algorithm must balance decisions made to learn about unknown parameters with those made to minimize cost based on current knowledge. It would be interesting to see how algorithms from the book fare with respect to this metric. There is no explicit emphasis in their design on striking this trade-off effectively.

4 Additional Features of the Book

There are a couple notable features presented by the book in addition to what we have discussed. One is that the authors offer versions of their first two algorithms that can be applied when the ultimate policy is required to meet linear constraints. In particular, given multiple cost processes $\eta_t^1, \ldots, \eta_t^K$, with associated conditional expectations $g^1(x_t, u_t), \ldots, g^K(x_t, u_t)$ and running averages $\overline{g}_t^1, \ldots, \overline{g}_t^K$, a sequence μ_t can be generated to maximize $\limsup_{t\to\infty} \overline{g}_t^1$ subject to $\limsup_{t\to\infty} \overline{g}_t^k \leq 0$ for $k = 2, \ldots, K$. To the best of my knowledge, such problems have not been explored in the reinforcement learning literature.

It is worth noting that there are also a number of topics addressed in the reinforcement learning literature that are beyond the scope of this book. Among these are approaches to dealing with intractable state spaces, intractable action spaces, and partially observable state.

Another notable feature that the book offers is Matlab code that implements the presented algorithms. There are also some simple numerical case studies, conducted using the Matlab code, involving problems with four states and three possible actions.

5 Closing Remarks

It appears that there are some opportunities for cross-fertilization between the thread of research represented by this book and the reinforcement learning literature. One is to compare algorithms based on both the convergence rate metric used in the book and the efficiency metric used in [5]. Another interesting area of potential is in the study of actor-critic methods. In particular, it would be interesting to see whether there are relationships between the method designed around Lagrange multipliers and other actor-critic algorithms that have been studied in the reinforcement learning literature. There are likely to be additional opportunities as well.

This book review has focused on technical content rather than commentary on the style of presentation. In closing, it is worth mentioning that the book is not an easy read. The proofs are quite technical and sometimes not self contained – the authors occasionally refer to lemmas from previous papers. There is little written to offer intuition or perspective to the reader. Nevertheless the book is valuable in collecting together a number of interesting results, and as the methods differ from those studied by the reinforcement learning community, it can serve as a source of inspiration and new ideas to that community.

References

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming.* Athena Scientific, 1995.

[2] V.S. Borkar and V.R. Konda, "Actor-Critic Algorithm as Multi-Time Scale Stochastic Approximation Algorithm." *Sadhana, Indian Academy of Sciences, Proceedings in Engineering Sciences,* Vol. 22, pp. 525-543, 1997.

[3] Fiechter, C. N. "Efficient Reinforcement Learning." *COLT94: Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pp. 88-97, ACM Press, 1994.

[4] V. Gullapalli and A. G. Barto, "Convergence of Indirect Adaptive Asynchronous Value Iteration Algorithms." *Advances in Neural Information Processing Systems 6*, pp. 695-702, Morgan Kauffman, 1994.

[5] M. Kearns and S. Singh, "Near-Optimal Reinforcement Learning in Polynomial Time," Vol. 49, pp. 209-232, 2002.

[6] V.R. Konda and V.S. Borkar, "Actor-Critic type learning Algorithms for Markov Decision Processes." *SIAM Journal on Control and Optimization*, Vol. 38, pp. 94-133, 1999.

[7] S. Singh, T. Jaakkola, M.L. Littman, and C. Szpesvari, "Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms." *Machine Learning*, Vol 38, pages 287-308, 2000.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.

[9] C. J. C. H. Watkins and P. Dayan, “Q-Learning.” *Machine Learning*, Vol. 8, pp. 279-292, 1992.