Reinforcement Learning and Monte-Carlo Methods

Nan Jiang

September 15, 2021

In the previous note we considered policy evaluation and optimization when the full specification of the MDP is given, and the major challenge is computational. In the *learning* setting, the MDP specification, especially the transition function P and sometimes the reward function R, is not known to the agent. Instead, the agent can take actions in the environment and observe the state transitions, and may find approximate solutions to policy evaluation or optimization after accumulating some amounts of interaction experience, or *data*. While computation remains an important aspect in the learning setting, this course will largely focus on *sample efficiency*, that is, the problem of achieving a learning goal using as little data as possible.

There are 3 major challenges in reinforcement learning, which many discussions in this course will center around:

– Temporal credit assignment In RL, the value obtained by the agent is the result of decisions made over multiple steps, and a fundamental question is how we should credit the outcome to each of the preceding decisions that lead to it. The challenge is addressed by applying principles of dynamic programming. It is often mentioned to contrast other machine learning paradigms, such as supervised learning, where the agent directly observes supervisory signals and the problem lacks a long-term nature.

– Generalization In many challenging RL problems, the state (and action) space is large and the amount of data available is relatively limited. An agent will not succeed in its learning objective unless it *generalizes* what is learned about one state to other states. This challenge is encountered in other machine learning paradigms as well, and is often addressed by *function approximation* techniques [1].

– Exploration (and exploitation) In RL, the characteristics of the data are largely determined by how the agent chooses actions during data collection. Taking actions to collect a dataset that provides a comprehensive description of the environment, i.e., performing good *exploration*, is a highly nontrivial problem. Sometimes when assessed by some online measures (more on this in Section 2), the agent needs to balance between pursuing the value obtainable with current knowledge (exploitation) and sacrificing performance temporarily to learn more (exploration). In either case, the challenge is often addressed by the principle of *optimism in face of uncertainty* [2].

In the remainder of this note, we first introduce the data collection protocols, and describe different performance measures for RL algorithms. Once the protocols and performance measures are set up properly, we move on and introduce basic algorithms and fundamental solution concepts.

1 Data collection protocols

In this note we consider the following flexible protocol for data collection that subsumes a number of settings considered in the literature. In particular, the dataset D consists of $|D|$ sample trajectories, each of which has length H'; here we use the notation H' for the length of the data trajectories to avoid confusion with the horizon H specified in the problem definition (either explicitly in the finite-horizon formulation, or implicitly as the effective horizon $H = O(1/(1-\gamma))$ in the discounted setting). In the i-th trajectory, the agent starts from some initial state $s_1^{(i)}$, takes actions according to a certain policy (potentially random and non-stationary), and records the H'-step truncated trajectory $(s_1^{(i)}, a_1^{(i)}, r_1^{(i)}, s_2^{(i)}, \ldots, s_{H'+1}^{(i)})$ in the dataset.

To fully specify the characteristics of the dataset, we need to determine (1) the value of H', (2) how $s_1^{(i)}$ is chosen, and (3) how the actions are chosen. Below we discuss a few combinations of interest, and for brevity we will drop the superscript $(\cdot)^{(i)}$ temporarily.

(a) $H' = 1$**, round-robin** s **and** a

When $H' = 1$, the agent only observes a transition tuple (s, a, r, s'). (We drop the time steps in subscripts for brevity as there is only 1 time step.) The state s and action a are chosen cyclically through the state-action space $\mathcal{S} \times \mathcal{A}$, which ensures that every state-action pair receives the same number of samples, often denoted as n. The total number of transitions $|D| = n \cdot |\mathcal{S} \times \mathcal{A}|$. This setting simplifies the data collection procedure and guarantees that all states and actions are uniformly covered in the dataset, which often simplifies the analysis of model-based RL methods [e.g. 3, 4].

(b) $H' = 1$**,** s **and** a **sampled from a distribution**

When the state (and action) space is very large and the budget of data size N is limited, the previous protocol **(a)** becomes inappropriate as we cannot even set n to 1. A useful variant of **(a)** is to assume that (s, a) is sampled i.i.d. from some distribution $\mu \in \Delta(\mathcal{S} \times \mathcal{A})$. If the RL algorithm to be applied incorporates some generalization schemes, we may still hope the algorithm can succeed when there are (s, a) pairs that we have not seen even once in the data, and we will see such analyses later in the course.

(c) $H' > 1$**,** $s_1 \sim d_0$**,** a_t **chosen using a fixed policy**

The previous two protocols assume that the dataset covers all states and actions automatically, which can be unrealistic sometimes. A more realistic protocol is that the initial state s_1 is sampled i.i.d. from the initial distribution d_0. For discounted problems, the trajectory length H' is often set to the effective horizon; for undiscounted finite-horizon problems, H' is naturally set to H, the horizon as in the problem specification.

One might wonder what happens if some states are not reachable from the support of d_0, as this implies that we are not getting any data for those states. However, if the initial states are *always* sampled from d_0 when the agent is deployed, the unreachable states may be treated as if they did not exist and are not our concern.

(d) $H' = 1$**,** $s_1 \sim d_0$**,** a_t **chosen by the algorithm**

All the previous protocols fix the choice of actions during data collection, and consequently the char-

acteristics of the dataset is out of the agent's control (i.e., they correspond to the *batch* setting of RL). In protocol **(d)**, we give the algorithm full control over these actions. Intuitively (and informally), now the agent needs to take smart actions to ensure that it collects a "good" dataset that provides a comprehensive description of the environment, which is indeed the exploration challenge. While exploration is not the focus of this thesis, we introduce this protocol for completeness.

(e) $H' = 1$**,** s**,** a **both chosen by the algorithm**
This is often called the *generative model/simulator* setting, which is a very strong protocol as it can emulate any of the above protocols. Another unique advantage of this protocol is the ability to revisit exactly the same state multiple times, which can be difficult if not impossible in previous protocols like **(b), (c), (d)** especially when the size of the state space is large, and there are algorithms that crucially rely on such functionalities [5, 6].

There are, of course, other protocols that are not in the incomplete list here. For example, in some settings the data is a single long trajectory, which is often combined with some ergodicity-type assumptions to prevent the agent to get stuck in some subset of the state space [7, 8, 9].

2 Performance measures

Now that we have protocols for data collection, and an algorithm outputs some results based on the collected data, we need to specify the measure that we use to assess the quality of these results.

(i) Worst-case loss
Consider policy optimization. Suppose the agent computes a policy π based on the collected data. The worst-case loss measures the quality of the policy by

$$\left\|V^\star - V^\pi\right\|_\infty . \tag{1}$$

Note that $V^\star - V^\pi$ is element-wise non-negative, and the infinite norm takes the largest gap. This measure considers the suboptimality of the proposed policy π for the worst start state s_1. It is mostly used with data collection protocols **(a)** and **(b)** where there are some coverage guarantees over the entire state space.

Since data is random, so is the output policy π and the worst-case loss. To turn the random variable into deterministic quantities, we can either talk about the expected loss, or adopt the Probably Approximately Correct (PAC) framework [10] and derive upper bound on the loss that is satisfied with high probability.

(ii) Loss under initial distribution
Demanding the algorithm to guarantee a small worst-case loss under protocols **(c)** and **(d)** may be vacuous, especially if there are states that are very hard or impossible to reach from the initial distribution d_0. A more mild and natural performance measure is the loss under the initial distribution d_0:

$$J(\pi^\star) - J(\pi) = d_0^\top V^\star - d_0^\top V^\pi. \tag{2}$$

This measure is mostly used with protocol **(c)** and **(d)** when the initial distribution d_0 is a crucial part of the problem definition. Interestingly, when this measure is combined with **(d)** and the PAC framework, the resulting setting is the theoretical framework for studying exploration in finite-horizon reinforcement learning problems [11, 12, 13].

(iii) Worst-case error
Consider policy evaluation, where the agent computes a value function V that approximates V^π for a given π based on data. Similar to **(i)**, we can define the worst-case prediction error as

$$\|V^\pi - V\|_\infty. \tag{3}$$

(iv) Error under initial distribution
Similarly we can define the analogy of **(ii)** for policy evaluation. Let the prediction error with respect to the initial distribution d_0 be:

$$\left|d_0^\top V^\pi - d_0^\top V\right|. \tag{4}$$

Essentially, the algorithm only needs to output a scalar $\hat{v}$ in the place of $d_0^\top V$ to approximate $J(\pi) := d_0^\top V^\pi$, and this is a standard statistical estimation problem. While $\hat{v}$ depends on the data and is random, we can talk about the properties of $\hat{v}$ as an estimator, such as bias, variance, Mean Squared Error (MSE), high-probability guarantees, and so on.

(v) Online measures
All previous measures implicitly assume a clear separation between data collection and deployment. With online performance measures, such a distinction disappears: the agent is evaluated at the same time as it collects data. As an example, consider the following measure that counts the number of "mistakes" made by the algorithm: the agent determines a policy which it uses for every next trajectory, and an error is counted if the policy is more than ϵ sub-optimal. This measure is sometimes referred to as a version of sample complexity for reinforcement learning [14, 15]. Another well-known example is regret, which is also used in online learning literature [16].

3 Monte-Carlo methods

In the remainder of this note we will survey a number of RL methods that are highly relevant to this thesis. We start with Monte-Carlo methods. In the RL context, the term "Monte-Carlo" refers to developing estimates of values without using bootstrapped targets, i.e., not invoking Bellman equations for policy evaluation or optimization where both sides of the equations contain unknowns. Instead, Monte-Carlo methods use the random return from the trajectory to form direct estimates.

As a basic example, consider policy evaluation. Given policy π, we are interested in computing V^π. Monte-Carlo policy evaluation performs the following simple steps: $\forall s \in \mathcal{S}$,

- Collect multiple trajectories by starting from $s_1 = s$ and following π for H steps.

- Compute the H-step discounted return $\sum_{t=1}^{H} \gamma^{t-1} r_t^{(i)}$ and take the average as an estimate of $V^\pi(s)$.

The expected value of the estimator is $V^{\pi,H}(s)$, and H is often set large enough to ensure that $V^{\pi,H}(s)$ approximates $V^{\pi}(s)$ well. In fact, using the analysis in the previous note, we can easily get $\|V^{\pi,H} - V^{\pi}\|_{\infty} \leq \gamma^H R_{\max}/(1-\gamma)$. Hence, the expression for effective horizon in value iteration applies to the policy evaluation setting as well.

The behavior of the algorithm is very straight-forward: for each s as the initial state, we get i.i.d. sample returns with mean $V^{\pi,H}(s)$ and range $[0, R_{\max}/(1-\gamma)]$. By using standard statistical bounds such as Hoeffding's inequality [17], we can obtain an error bound on $|V^{\pi,H}(s) - V^{\pi}(s)|$ as a function of the sample size that holds with high probability, and apply union bound to guarantee accurate estimation in all states simultaneously, i.e., low worst-case prediction error. The same type of analysis will be carried out throughout the thesis so we omit the details here.

In general, getting low worst-case error requires the number of total trajectories to scale at least linearly with the size of the state space $|\mathcal{S}|$.[1] Sometimes we are only interested in a scalar value that characterizes the value of a policy, $J(\pi) := d_0^{\top} V^{\pi}$ (recall performance measure **(iv)**). While we could estimate V^{π} for all states to a good accuracy and compute $d_0^{\top} V^{\pi}$ based on it, there is a simpler and much more effective procedure for doing this:

- Collect trajectories by starting from $s \sim d_0$ and following π for H steps.

- Compute $\sum_{t=1}^{H} \gamma^{t-1} r_t^{(i)}$ and take the average as an estimate of $J(\pi) := d_0^{\top} V^{\pi}$.

A most notable property of this algorithm is that its accuracy guarantee is completely independent of the size of the state space, which is an elegant property that is often exhibited in Monte-Carlo methods [5]. This algorithm is particularly useful when we want to assess the quality of a policy or compare among multiple policies, hence it forms the basis for policy validation and hyperparameter tuning in state-of-the-art empirical research of reinforcement learning.

Proposition 1. *Suppose $\hat{v}^{\pi}$ is the Monte-Carlo estimate of $J(\pi)$. Then with probability at least $1-\delta$, we have $|\hat{v}^{\pi} - J(\pi)| \leq \frac{R_{\max}}{1-\gamma}\sqrt{\frac{1}{2n}\log\frac{2}{\delta}}$.*

Proof. Straight-forward application of Hoeffding's inequality. □

On the other hand, Monte-Carlo policy evaluation requires the data collection protocol **(d)**, and crucially the policy used to collect data should be the same as the policy to be evaluated (i.e., the algorithm is **on-policy**). When such assumptions fail, especially when the data collection policy is different from the evaluated policy, we face an **off-policy** policy evaluation problem. While extension of Monte-Carlo methods still enjoy independence of the state space size [19], the dependence on horizon is exponential [20, 21].

References

[1] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[1] In some cases, mostly for episodic problems, it is possible to reuse a trajectory multiple times to form estimates of different states encountered along the trajectory; depending on how multiple occurrences of the same state is handled, the variants are called "first-visit" or "every-visit" Monte-Carlo policy evaluation [18]. We do not introduce these variants as they grow the effective sample size at most by a multiplicative factor of H, which does not affect our discussion here.

[2] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[3] Shie Mannor, Duncan Simester, Peng Sun, and John N Tsitsiklis. Bias and variance approximation in value function estimates. *Management Science*, 53(2):308–322, 2007.

[4] Cosmin Paduraru, Robert Kaplow, Doina Precup, and Joelle Pineau. Model-based reinforcement learning with state aggregation. In *8th European Workshop on Reinforcement Learning*, 2008.

[5] Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, 49(2-3):193–208, 2002.

[6] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

[7] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

[8] Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. *Advances in Neural Information Processing Systems*, 19:49, 2007.

[9] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

[10] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[11] Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2818–2826, 2015.

[12] Akshay Krishnamurthy, Alekh Agarwal, and John Langford. PAC reinforcement learning with rich observations. In *Advances in Neural Information Processing Systems*, pages 1840–1848, 2016.

[13] Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E. Schapire. Contextual decision processes with low Bellman rank are PAC-learnable. In *International Conference on Machine Learning*, 2017.

[14] Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, University of College London, 2003.

[15] Alexander L Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L Littman. PAC model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881–888. ACM, 2006.

[16] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

[17] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[18] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, March 1998.

[19] Doina Precup, Richard S Sutton, and Satinder P Singh. Eligibility Traces for Off-Policy Policy Evaluation. In *Proceedings of the 17th International Conference on Machine Learning*, pages 759–766, 2000.

[20] Lihong Li, Rémi Munos, and Csaba Szepesvári. Toward minimax off-policy value estimation. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 2015.

[21] Nan Jiang and Lihong Li. Doubly Robust Off-policy Value Evaluation for Reinforcement Learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 652–661, 2016.