Reinforcement Learning: Theory and Algorithms

Alekh Agarwal Nan Jiang Sham M. Kakade

September 9, 2020

WORKING DRAFT: Text not yet at the level of publication.

V2: This version is changing to use un-normalized values.

Contents

Chapter 0

Notation

The reader might find it helpful to refer back to this notation section.

- For a vector v, we let $(v)^2$, $\sqrt{v}$, and $|v|$ be the component-wise square, square root, and absolute value operations.
- Inequalities between vectors are elementwise, e.g. for vectors v, v', we way $v \leq v'$, if the inequality holds elementwise.
- For a vector v, we refer to the j-th component of this vector by either $v(j)$ or $[v]_j$
- Denote the variance of any real valued f under a distribution $\mathcal{D}$ as:

$$\text{Var}_{\mathcal{D}}(f) := E_{x\sim\mathcal{D}}[f(x)^2] - (E_{x\sim\mathcal{D}}[f(x)])^2$$

- It is helpful to overload notation and let P also refer to a matrix of size $(\mathcal{S}\cdot\mathcal{A}) \times \mathcal{S}$ where the entry $P_{(s,a),s'}$ is equal to $P(s'|s,a)$. We also will define P^π to be the transition matrix on state-action pairs induced by a deterministic policy π. In particular, $P^\pi_{(s,a),(s',a')} = P(s'|s,a)$ if $a' = \pi(s')$ and $P^\pi_{(s,a),(s',a')} = 0$ if $a' \neq \pi(s')$. With this notation,

$$\begin{aligned} Q^\pi &= r + PV^\pi \\ Q^\pi &= r + P^\pi Q^\pi \\ Q^\pi &= (I - \gamma P^\pi)^{-1} r \end{aligned}$$

- For a vector $Q \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$, denote the greedy policy and value as:

$$\begin{aligned} \pi_Q(s) &:= \text{argmax}_{a\in\mathcal{A}}\, Q(s,a) \\ V_Q(s) &:= \max_{a\in\mathcal{A}} Q(s,a).. \end{aligned}$$

- For a vector $Q \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$, the *Bellman optimality operator* $\mathcal{T} : \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ is defined as:

$$\mathcal{T}Q := r + PV_Q\,. \tag{0.1}$$

Chapter 1

Markov Decision Processes and Computational Limits

Reinforcement Learning: Theory and Algorithms **Working Draft**

Markov Decision Processes

Alekh Agarwal, Nan Jiang, Sham M. Kakade *Chapter 1*

1.1 Markov Decision Processes

In reinforcement learning, the interactions between the agent and the environment are often described by a Markov Decision Process (MDP) $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \mu)$, specified by:

- A state space $\mathcal{S}$, which may be finite or infinite.
- An action space $\mathcal{A}$, which also may be discrete or infinite.
- A transition function $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ is the space of probability distributions over $\mathcal{S}$ (i.e., the probability simplex). $P(s'|s, a)$ is the probability of transitioning into state s' upon taking action a in state s. We use $P_{s,a}$ to denote the vector $P(\cdot \mid s, a)$.
- A reward function $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$. $r(s, a)$ is the immediate reward associated with taking action a in state s.
- A discount factor $\gamma \in [0, 1)$, which defines a horizon for the problem.
- An initial state distribution $\mu \in \Delta(\mathcal{S})$, which species how the initial state s_0 is generated.

In many cases, we will assume that the initial state is fixed at s_0, i.e. μ is a distribution supported only on s_0.

1.1.1 Interaction protocol

In a given MDP $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \mu)$, the agent interacts with the environment according to the following protocol: the agent starts at some state $s_0 \sim \mu$; at each time step $t = 0, 1, 2, \ldots$, the agent takes an action $a_t \in \mathcal{A}$, obtains the immediate reward $r_t = r(s_t, a_t)$, and observes the next state s_{t+1} sampled according to $s_{t+1} \sim P(\cdot|s_t, a_t)$. The interaction record at time t,

$$\tau_t = (s_0, a_0, r_1, s_1, \ldots, s_t),$$

is called a *trajectory*, which includes the observed state at time t.

1.1.2 The objective, policies, and values

In the most general setting, a policy specifies a decision-making strategy in which the agent chooses actions adaptively based on the history of observations; precisely, a policy is a mapping from a trajectory to an action, i.e. $\pi : \mathcal{H} \to \mathcal{A}$ where $\mathcal{H}$ is the set of all possibly trajectories. A deterministic, *stationary* policy $\pi : \mathcal{S} \to \mathcal{A}$ specifies a decision-making strategy in which the agent chooses actions adaptively based on the current state, i.e., $a_t = \pi(s_t)$. The agent may also choose actions according to a stochastic policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, and, overloading notation, we write $a_t \sim \pi(\cdot|s_t)$. A deterministic policy is a special case when $\pi(s)$ is a point mass for all $s \in \mathcal{S}$.

For a fixed policy and a starting state $s_0 = s$, we define the value function $V^\pi_M : \mathcal{S} \to \mathbb{R}$ as the discounted sum of

future rewards

$$V_M^\pi(s) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \;\Big|\; \pi, s_0 = s\Big].$$

where expectation is with respect to the randomness of the trajectory, that is, the randomness in state transitions and the stochasticity of π. Here, since $r(s, a)$ is bounded between 0 and 1, we have $0 \le V_M^\pi(s) \le 1/(1-\gamma)$.

Similarly, the action-value (or Q-value) function $Q_M^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as

$$Q_M^\pi(s, a) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \;\Big|\; \pi, s_0 = s, a_0 = a\Big].$$

and $Q_M^\pi(s, a)$ is also bounded by $1/(1-\gamma)$.

Given a state s, the goal of the agent is to find a policy π that maximizes the value, i.e. the optimization problem the agent seeks to solve is:

$$\max_\pi V_M^\pi(s) \tag{1.1}$$

where the max is over all (possibly non-stationary and randomized) policies. As we shall see, there exists a single deterministic and stationary policy which is simultaneously optimal for all starting states s.

We drop the dependence on M and write V^π when it is clear from context.

Example 1.1 (Navigation)**.** Navigation is perhaps the simplest to see example of RL. The state of the agent is their current location. The four actions might be moving 1 step along each of east, west, north or south. The transitions in the simplest setting are deterministic. Taking the north action moves the agent one step north of their location, assuming that the size of a step is standardized. The agent might have a goal state g they are trying to reach, and the reward is 0 until the agent reaches the goal, and 1 upon reaching the goal state. Since the discount factor $\gamma < 1$, there is incentive to reach the goal state earlier in the trajectory. As a result, the optimal behavior in this setting corresponds to finding the shortest path from the initial to the goal state, and the value function of a state, given a policy is γ^d, where d is the number of steps required by the policy to reach the goal state.

Example 1.2 (Conversational agent)**.** This is another fairly natural RL problem. The state of an agent can be the current transcript of the conversation so far, along with any additional information about the world, such as the context for the conversation, characteristics of the other agents or humans in the conversation etc. Actions depend on the domain. In the most basic form, we can think of it as the next statement to make in the conversation. Sometimes, conversational agents are designed for task completion, such as travel assistant or tech support or a virtual office receptionist. In these cases, there might be a predefined set of *slots* which the agent needs to fill before they can find a good solution. For instance, in the travel agent case, these might correspond to the dates, source, destination and mode of travel. The actions might correspond to natural language queries to fill these slots.

In task completion settings, reward is naturally defined as a binary outcome on whether the task was completed or not, such as whether the travel was successfully booked or not. Depending on the domain, we could further refine it based on the quality or the price of the travel package found. In more generic conversational settings, the ultimate reward is whether the conversation was satisfactory to the other agents or humans, or not.

Example 1.3 (Strategic games)**.** This is a popular category of RL applications, where RL has been successful in achieving human level performance in Backgammon, Go, Chess, and various forms of Poker. The usual setting consists of the state being the current game board, actions being the potential next moves and reward being the eventual win/loss outcome or a more detailed score when it is defined in the game. Technically, these are multi-agent RL settings, and, yet, the algorithms used are often non-multi-agent RL algorithms.

1.1.3 Bellman consistency equations for stationary policies

By definition, V^π and Q^π satisfy the following *Bellman consistency equations*: for all $s \in \mathcal{S}, a \in \mathcal{A}$,

$$\begin{aligned} V^\pi(s) &= Q^\pi(s, \pi(s)). \\ Q^\pi(s,a) &= r(s,a) + \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[V^\pi(s')\right], \end{aligned} \tag{1.2}$$

where we are treating π as a deterministic policy.

It is helpful to view V^π as vector of length $\mathcal{S}$ and Q^π and r as vectors of length $\mathcal{S} \cdot \mathcal{A}$. We overload notation and let P also refer to a matrix of size $(\mathcal{S} \cdot \mathcal{A}) \times \mathcal{S}$ where the entry $P_{(s,a),s'}$ is equal to $P(s'|s,a)$.

We also will define P^π to be the transition matrix on state-action pairs induced by a deterministic policy π. In particular,

$$P^\pi_{(s,a),(s',a')} := \begin{cases} P(s'|s,a) & \text{if } a' = \pi(s') \\ 0 & \text{if } a' \neq \pi(s') \end{cases}$$

For a randomized stationary policy, we have

$$P^\pi_{(s,a),(s',a')} = P(s'|s,a)\pi(a'|s').$$

With this notation, it is straightforward to verify:

$$Q^\pi = r + PV^\pi \tag{1.3}$$

$$Q^\pi = r + P^\pi Q^\pi\,. \tag{1.4}$$

The above implies that:

$$Q^\pi = (I - \gamma P^\pi)^{-1} r \tag{1.5}$$

where I is the identity matrix. To see that the $I - \gamma P^\pi$ is invertible, observe that for any non-zero vector $x \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$,

$$\begin{aligned} \|(I - \gamma P^\pi)x\|_\infty &= \|x - \gamma P^\pi x\|_\infty \\ &\geq \|x\|_\infty - \gamma \|P^\pi x\|_\infty && \text{(triangle inequality for norms)} \\ &\geq \|x\|_\infty - \gamma \|x\|_\infty && \text{(each element of } P^\pi x \text{ is an average of } x) \\ &= (1-\gamma)\|x\|_\infty > 0 && (\gamma < 1,\, x \neq 0) \end{aligned}$$

which implies $I - \gamma P^\pi$ is full rank.

1.1.4 Bellman optimality equations

Due to the Markov structure, there exists a single stationary and deterministic policy that simultaneously maximizes $V^\pi(s)$ for all $s \in \mathcal{S}$ and maximizes $Q^\pi(s,a)$ for all $s \in \mathcal{S}, a \in \mathcal{A}$; we denote this *optimal policy* as $\pi^\star_M$ (or $\pi^\star$). This is formalized in the following theorem:

Theorem 1.4. *Let Π be the set of all non-stationary and randomized policies. There exists a single stationary and deterministic policy π such that, for all $s \in \mathcal{S}$,*

$$V^\pi(s) = \max_{\pi' \in \Pi} V^{\pi'}_M(s).$$

We refer to such a π as an optimal policy.

Proof: We will show the deterministic and stationary policy $\pi(s) = \text{argmax}_{a\in\mathcal{A}} \max_{\pi'\in\Pi} Q^{\pi'}(s,a)$ is an optimal policy. By definition of π, there exists a policy $\widetilde{\pi}$, such that $\widetilde{\pi}$ chooses the same action as $\pi(s)$ at time step 0 and that, for all $s \in \mathcal{S}$,

$$V^{\widetilde{\pi}}(s) = \max_{\pi'\in\Pi} V_M^{\pi'}(s).$$

In other words, $\widetilde{\pi}$ is an optimal policy.

Define the policy π_τ to be a policy which acts according to π before time τ and, at time τ, it starts to execute the policy $\widetilde{\pi}$, starting from state s_τ. We now show that π_τ is optimal for all $\tau \geq 0$. By construction $\pi_0 = \widetilde{\pi}$ is optimal. Let us now show that π_τ is optimal given that $\pi_{\tau-1}$ is optimal. Observe that:

$$\begin{aligned}
& V^{\pi_{\tau-1}}(s) \\
= \;& \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_\tau, a_\tau) \;\Big|\; s_0 = s\Big] \\
= \;& \mathbb{E}\left[r(s_0,a_0) + \ldots \gamma^{\tau-1} r(s_{\tau-1}, a_{\tau-1}) + \gamma^\tau \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_{t+\tau}, a_{t+\tau}) | s_\tau = s\right] \;\Big|\; s_0 = s, a_0 = a\right] \\
\leq \;& \mathbb{E}\left[r(s_0,a_0) + \ldots \gamma^{\tau-1} r(s_{\tau-1}, a_{\tau-1}) + \gamma^\tau \max_{\pi'\in\Pi}\left(\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_{t+\tau}, a_{t+\tau}) | \pi', s_\tau = s\right]\right) \;\Big|\; s_0 = s, a_0 = a\right] \\
= \;& \mathbb{E}\Big[r(s_0,a_0) + \ldots \gamma^{\tau-1} r(s_{\tau-1}, a_{\tau-1}) + \gamma^\tau V^{\widetilde{\pi}}(s_\tau) \;\Big|\; s_0 = s\Big] \\
= \;& V^{\pi_\tau}(s)\,.
\end{aligned}$$

Now since $V^{\pi_{\tau-1}}(s)$ was optimal, then we have that $V^{\pi_\tau}(s)$ is also optimal. This completes the proof. ■

This shows that we may restrict ourselves to using stationary and deterministic policies without any loss in performance. The following theorem, also due to [Bellman, 1956], gives a precise characterization of the optimal value function.

Theorem 1.5. *Let $Q^\star$ by defined as the vector $Q^\star(s,a) = \max_{\pi\in\Pi} Q^\pi(s,a)$ where Π is the space of all (non-stationary and randomized) policies. We have that a vector $Q \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is equal to $Q^\star$ if and only if it satisfies:*

$$Q^\star(s,a) = r(s,a) + \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\max_{a'\in\mathcal{A}} Q^\star(s',a')\right]. \tag{1.6}$$

Before we prove this claim, we will provide a few definitions. We use $V^\star$ and $Q^\star$ as a shorthand for $V^{\pi^\star}$ and $Q^{\pi^\star}$, respectively. We let π_Q denote the greedy policy with respect to a vector $Q \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$, i.e

$$\pi_Q(s) := \text{argmax}_{a\in\mathcal{A}}\, Q(s,a)\,.$$

where ties are broken in some arbitrary (and deterministic) manner. With this notation, the optimal policy $\pi^\star$ is obtained by choosing actions greedily (with arbitrary tie-breaking mechanisms) with respect to Q, i.e.

$$\pi^\star = \pi_{Q^\star}\,.$$

Let us also use the notation to turn a vector $Q \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ into a vector of length $|\mathcal{S}|$.

$$V_Q(s) := \max_{a\in\mathcal{A}} Q(s,a).$$

The *Bellman optimality operator* $\mathcal{T}_M : \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$ is defined by the follows: for a vector $Q \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$,

$$\mathcal{T}Q := r + \gamma P V_Q\,. \tag{1.7}$$

This allows us to rewrite Equation 1.6 in the concise form:

$$Q^\star = \mathcal{T}Q^\star$$

i.e. $Q^\star$ is a fixed point of the operator $\mathcal{T}$.

Proof: We first show that $Q^\star$ (the state-action value of an optimal policy) satisfies $Q^\star = \mathcal{T}Q^\star$. For an optimal value function, we have that $V^\star(s) = \max_a Q^\star(s,a)$. For all actions $a \in \mathcal{A}$, we have:

$$\begin{aligned} Q^\star(s,a) &= \max_\pi Q^\pi(s,a) = r(s,a) + \max_\pi \left(\mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\pi(s')]\right) \\ &\overset{(a)}{=} r(s,a) + \mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\star(s')] \\ &= r(s,a) + \mathbb{E}_{s'\sim P(\cdot|s,a)}[\max_{a'} Q^\star(s',a')]. \end{aligned}$$

Here the equality (a) follows from Theorem 1.4 due to that there exists a policy that is optimal for every starting state.

For the converse, suppose $Q = \mathcal{T}Q$ for some Q. For $\pi = \pi_Q$, this implies that $Q = r + \gamma P^{\pi_Q} Q$. This implies:

$$Q = (I - \gamma P^{\pi_Q})^{-1} r = Q^\pi$$

using Equation 1.5 in the last step. In other words, Q is the action value of the policy π_Q. Now observe for any other policy π':

$$\begin{aligned} Q^{\pi'} - Q &= Q^{\pi'} - Q^\pi \\ (I - \gamma P^{\pi'})^{-1} r - (I - \gamma P^\pi)^{-1} r & \\ &= (I - \gamma P^{\pi'})^{-1}((I - \gamma P^\pi) - (I - \gamma P^{\pi'}))Q^\pi \\ &= \gamma (I - \gamma P^{\pi'})^{-1}(P^{\pi'} - P^\pi)Q^\pi \,. \end{aligned}$$

The proof is completed by noting that $(P^{\pi'} - P^\pi)Q^\pi \leq 0$. To see this, observe that:

$$[(P^{\pi'} - P^\pi)Q^\pi]_{s,a} = \mathbb{E}_{s'\sim P(\cdot|s,a)}[Q^\pi(s', \pi'(s')) - Q^\pi(s', \pi(s'))] \leq 0$$

where we use $\pi = \pi_Q$ in the last step. ■

1.2 Computational Complexity

The remainder of this section will be concerned with computing an optimal policy when given knowledge of the MDP $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$. While much of this book is concerned with statistical limits, understanding the computational limits can be informative. We will consider algorithms which give both exact and approximately optimal policies. In particular, we will be interested in polynomial time (and strongly polynomial time) algorithms.

Suppose that (P, r, γ) in our MDP M is specified with rational entries. Let $L(P, r, \gamma)$ denote the total bit-size required to specify M, and assume that basic arithmetic operations $+, -, \times, \div$ take unit time. Here, we may hope for an algorithm which (exactly) returns an optimal policy whose runtime is polynomial in $L(P, r, \gamma)$ and the number of states and actions.

More generally, it may also be helpful to understand which algorithms are *strongly* polynomial. Here, we do not want to explicitly restrict (P, r, γ) to be specified by rationals. An algorithm is said to be strongly polynomial if it returns an optimal policy with runtime that is polynomial in only the number of states and actions (with no dependence on $L(P, r, \gamma)$).

	Value Iteration	Policy Iteration	LP-Algorithms
Poly?	$\lvert\mathcal{S}\rvert^2\lvert\mathcal{A}\rvert\frac{L(P,r,\gamma)\log\frac{1}{1-\gamma}}{1-\gamma}$	$(\lvert\mathcal{S}\rvert^3+\lvert\mathcal{S}\rvert^2\lvert\mathcal{A}\rvert)\frac{L(P,r,\gamma)\log\frac{1}{1-\gamma}}{1-\gamma}$	$\lvert\mathcal{S}\rvert^3\lvert\mathcal{A}\rvert L(P,r,\gamma)$
Strongly Poly?	✗	$(\lvert\mathcal{S}\rvert^3+\lvert\mathcal{S}\rvert^2\lvert\mathcal{A}\rvert)\cdot\min\left\{\frac{\lvert\mathcal{A}\rvert^{\lvert\mathcal{S}\rvert}}{\lvert\mathcal{S}\rvert}, \frac{\lvert\mathcal{S}\rvert^2\lvert\mathcal{A}\rvert\log\frac{\lvert\mathcal{S}\rvert^2}{1-\gamma}}{1-\gamma}\right\}$	$\lvert\mathcal{S}\rvert^4\lvert\mathcal{A}\rvert^4\log\frac{\lvert\mathcal{S}\rvert}{1-\gamma}$

Table 1.1: Computational complexities of various approaches. Polynomial time algorithms depend on the bit complexity, $L(P,r,\gamma)$, while strongly polynomial algorithms do not. Note that only for a fixed value of γ are value and policy iteration polynomial time algorithms; otherwise, they are not polynomial time algorithms. Similarly, only for a fixed value of γ is policy iteration a strongly polynomial time algorithm. In contrast, the LP-approach leads to both polynomial time and strongly polynomial time algorithms; for the latter, the approach is an interior point algorithm. See text for further discussion, and Section 1.5 for references. Here, $|\mathcal{S}|^2|\mathcal{A}|$ is the assumed runtime per iteration of value iteration, and $|\mathcal{S}|^3+|\mathcal{S}|^2|\mathcal{A}|$ is the assumed runtime per iteration of policy iteration (note that for this complexity we would directly update the values V rather than Q values, as described in the text); these runtimes are consistent with assuming cubic complexity for linear system solving.

1.3 Iterative Methods

Planning refers to the problem of computing $\pi^\star_M$ given the MDP specification $M=(\mathcal{S},\mathcal{A},P,r,\gamma)$. This section reviews classical planning algorithms that compute $Q^\star$.

1.3.1 Q-Value Iteration

A simple algorithm is to iteratively apply the fixed point mapping: starting at some Q, we iteratively apply $\mathcal{T}$:

$$Q \leftarrow \mathcal{T}Q\,,$$

This is algorithm is referred to as *Q-value iteration*.

Lemma 1.6. *(contraction) For any two vectors $Q, Q' \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$,*

$$\|\mathcal{T}Q-\mathcal{T}Q'\|_\infty \le \gamma\|Q-Q'\|_\infty$$

Proof: First, let us show that for all $s\in\mathcal{S}$, $|V_Q(s)-V_{Q'}(s)|\le\max_{a\in\mathcal{A}}|Q(s,a)-Q'(s,a)|$. Assume $V_Q(s)>V_{Q'}(s)$ (the other direction is symmetric), and let a be the greedy action for Q at s. Then

$$|V_Q(s)-V_{Q'}(s)| = Q(s,a)-\max_{a'\in\mathcal{A}}Q'(s,a') \le Q(s,a)-Q'(s,a) \le \max_{a\in\mathcal{A}}|Q(s,a)-Q'(s,a)|.$$

Using this,

$$\begin{aligned}
\|\mathcal{T}Q-\mathcal{T}Q'\|_\infty &= \gamma\|PV_Q-PV_{Q'}\|_\infty \\
&= \gamma\|P(V_Q-V_{Q'})\|_\infty \\
&\le \gamma\|V_Q-V_{Q'}\|_\infty \\
&= \gamma\max_s|V_Q(s)-V_{Q'}(s)| \\
&\le \gamma\max_s\max_a|Q(s,a)-Q'(s,a)| \\
&= \gamma\|Q-Q'\|_\infty
\end{aligned}$$

where the first inequality uses that each element of $P(V_Q - V_{Q'})$ is a convex average of $V_Q - V_{Q'}$ and the second inequality uses our claim above. ■

The following result bounds the sub-optimality of the greedy policy itself, based on the error in Q-value function.

Lemma 1.7. *For any vector $Q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$,*

$$V^{\pi_Q} \geq V^\star - \frac{2\|Q - Q^\star\|_\infty}{1-\gamma}\mathbb{1}.$$

where $\mathbb{1}$ denotes the vector of all ones.

Proof: Fix state s and let $a = \pi_Q(s)$. We have:

$$\begin{aligned}
V^\star(s) - V^{\pi_Q}(s) =& Q^\star(s, \pi^\star(s)) - Q^{\pi_Q}(s, a) \\
=& Q^\star(s, \pi^\star(s)) - Q^\star(s, a) + Q^\star(s, a) - Q^{\pi_Q}(s, a) \\
=& Q^\star(s, \pi^\star(s)) - Q^\star(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[V^\star(s') - V^{\pi_Q}(s')] \\
\leq& Q^\star(s, \pi^\star(s)) - Q(s, \pi^\star(s)) + Q(s, a) - Q^\star(s, a) \\
&+ \gamma \mathbb{E}_{s' \sim P(s,a)}[V^\star(s') - V^{\pi_Q}(s')] \\
\leq& 2\|Q - Q^\star\|_\infty + \gamma \|V^\star - V^{\pi_Q}\|_\infty.
\end{aligned}$$

where the first inequality uses $Q(s, \pi^\star(s)) \leq Q(s, \pi_Q(s)) = Q(s, a)$ due to the definition of π_Q. ■

Theorem 1.8. *(Q-value iteration convergence). Set $Q^{(0)} = 0$. For $k = 0, 1, \ldots$, suppose:*

$$Q^{(k+1)} = \mathcal{T} Q^{(k)}$$

Let $\pi^{(k)} = \pi_{Q^{(k)}}$. For $k \geq \frac{\log \frac{2}{(1-\gamma)^2 \epsilon}}{1-\gamma}$,

$$V^{\pi^{(k)}} \geq V^\star - \epsilon \mathbb{1}\,.$$

Proof: Since $\|Q^\star\|_\infty \leq 1/(1-\gamma)$, $Q^{(k)} = \mathcal{T}^k Q^{(0)}$ and $Q^\star = \mathcal{T} Q^\star$, Lemma 1.6 gives

$$\|Q^{(k)} - Q^\star\|_\infty = \|\mathcal{T}^k Q^{(0)} - \mathcal{T}^k Q^\star\|_\infty \leq \gamma^k \|Q^{(0)} - Q^\star\|_\infty = (1 - (1-\gamma))^k \|Q^\star\|_\infty \leq \exp(-(1-\gamma)k)\,.$$

The proof is completed with our choice of γ and using Lemma 1.7. ■

Iteration complexity for an exact solution. With regards to computing an exact optimal policy, when the gap between the current objective value and the optimal objective value is smaller than $2^{-L(P,r,\gamma)}$, then the greedy policy will be optimal. This leads to claimed complexity in Table 1.1. Value iteration is not strongly polynomial algorithm due to that, in finite time, it may never return the optimal policy.

1.3.2 Policy Iteration

The policy iteration algorithm starts from an arbitrary policy π_0, and repeat the following iterative procedure: for $k = 0, 1, 2, \ldots$

1. *Policy evaluation.* Compute Q^{π_k}
2. *Policy improvement.* Update the policy:

$$\pi_{k+1} = \pi_{Q^{\pi_k}}$$

In each iteration, we compute the Q-value function of π_k, using the analytical form given in Equation 1.5, and update the policy to be greedy with respect to this new Q-value. The first step is often called *policy evaluation*, and the second step is often called *policy improvement*.

Lemma 1.9. *We have that:*

1. $Q^{\pi_{k+1}} \geq \mathcal{T}Q^{\pi_k} \geq Q^{\pi_k}$
2. $\|Q^{\pi_{k+1}} - Q^\star\|_\infty \leq \gamma \|Q^{\pi_k} - Q^\star\|_\infty$

Proof: We start with the first part. Note that the policies produced in policy iteration are always deterministic, so $V^{\pi_k}(s) = Q^{\pi_k}(s, \pi_k(s))$ for all iterations k and states s. Hence,

$$\begin{aligned}
\mathcal{T}Q^{\pi_k}(s,a) &= r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[\max_{a'} Q^{\pi_k}(s',a')] \\
&\geq r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[Q^{\pi_k}(s', \pi_k(s'))] \\
&= Q^{\pi_k}(s,a).
\end{aligned}$$

Using this,

$$\begin{aligned}
Q^{\pi_{k+1}}(s,a) &= r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[Q^{\pi_{k+1}}(s', \pi_{k+1}(s'))] \\
&\geq r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[Q^{\pi_k}(s', \pi_{k+1}(s'))] \\
&= r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[\max_{a'} Q^{\pi_k}(s',a')] \\
&= \mathcal{T}Q^{\pi_k}(s,a)
\end{aligned}$$

which proves the first claim.

For the second claim,

$$\|Q^\star - Q^{\pi_{k+1}}\|_\infty \geq \|Q^\star - \mathcal{T}Q^{\pi_k}\|_\infty = \|\mathcal{T}Q^\star - \mathcal{T}Q^{\pi_{k+1}}\|_\infty \leq \gamma \|Q^\star - Q^{\pi_k}\|_\infty$$

where we have used that $Q^\star \geq Q^{\pi_{k+1}} \geq Q^{\pi_k}$ in second step and the contraction property of $\mathcal{T}(\cdot)$ (see Lemma 1.6 in the last step. ■

With this lemma, a convergence rate for the policy iteration algorithm immediately follows.

Theorem 1.10. *(policy iteration convergence). Let π_0 be any initial policy. For $k \geq \frac{\log \frac{1}{(1-\gamma)\epsilon}}{1-\gamma}$, the k-th policy in policy iteration has the following performance bound:*

$$Q^{\pi^{(k)}} \geq Q^\star - \epsilon \mathbb{1}\,.$$

Iteration complexity for an exact solution. With regards to computing an exact optimal policy, it clear from the previous results that policy iteration is no worse than value iteration. However, with regards to obtaining an exact solution MDP that is independent of the bit complexity, $L(P, r, \gamma)$, improvements are possible (and where we assume basic arithmetic operations on real numbers are order one cost). Naively, the number of iterations of policy iterations is bounded by the number of policies, namely $|\mathcal{A}|^{|\mathcal{S}|}$; here, a small improvement is possible, where the number of iterations of policy iteration can be bounded by $\frac{|\mathcal{A}|^{|\mathcal{S}|}}{|\mathcal{S}|}$. Remarkably, for a fixed value of γ, policy iteration can be show to be a strongly polynomial time algorithm, where policy iteration finds an exact policy in at most $\frac{|\mathcal{S}|^2 |\mathcal{A}| \log \frac{|\mathcal{S}|^2}{1-\gamma}}{1-\gamma}$ iterations. See Table 1.1 for a summary, and Section 1.5 for references.

1.4 The Linear Programming Approach

It is helpful to understand an alternative approach to finding an optimal policy for a known MDP. With regards to computation, consider the setting where our MDP $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \mu)$ is known and P, r, and γ are all specified by rational numbers. Here, from a computational perspective, the previous iterative algorithms are, strictly speaking, not polynomial time algorithms, due to that they depend polynomially on $1/(1-\gamma)$, which is not polynomial in the description length of the MDP . In particular, note that any rational value of $1-\gamma$ may be specified with only $O(\log \frac{1}{1-\gamma})$ bits of precision. In this context, we may hope for a fully polynomial time algorithm, when given knowledge of the MDP, which would have a computation time which would depend polynomially on the description length of the MDP M, when the parameters are specified as rational numbers. We now see that the LP approach provides a polynomial time algorithm.

1.4.1 The Primal LP and A Polynomial Time Algorithm

Consider the following optimization problem with variables $V \in \mathbb{R}^{|\mathcal{S}|}$:

$$\begin{aligned} \max \quad & \sum_s \mu(s) V(s) \\ \text{subject to} \quad & V(s) \geq r(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \quad \forall a \in \mathcal{A},\ s \in \mathcal{S} \end{aligned}$$

Here, the optimal value function $V^\star(s)$ is the unique solution to this linear program. With regards to computation time, linear programming approaches only depend on the description length of the coefficients in the program, due to that this determines the computational complexity of basic additions and multiplications. Thus, this approach will only depend on the bit length description of the MDP, when the MDP is specified by rational numbers.

Computational complexity for an exact solution. Table 1.1 shows the runtime complexity for the LP approach, where we assume a standard runtime for solving a linear program. The strongly polynomial algorithm is an interior point algorithm. See Section 1.5 for references.

Policy iteration and the simplex algorithm. It turns out that the policy iteration algorithm is actually the simplex method with block pivot. While the simplex method, in general, is not a strongly polynomial time algorithm, the policy iteration algorithm is a strongly polynomial time algorithm, provided we keep the discount factor fixed. See [Ye, 2011].

1.4.2 The Dual LP and the State-Action Polytope

For a fixed (possibly stochastic) policy π, let us define the state-action visitation distribution ν_μ^π as:

$$\nu_\mu^\pi(s,a) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \Pr^\pi(s_t = s, a_t = a)$$

where $\Pr^\pi(s_t = s, a_t = a)$ is the state-action visitation probability, where we execute π in M starting at state $s_0 \sim \mu$.

It is straightforward to verify that ν_μ^π satisfies, for all states $s \in \mathcal{S}$:

$$\sum_a \nu_\mu^\pi(s,a) = (1-\gamma)\mu(s) + \gamma \sum_{s',a'} P(s|s',a') \nu_\mu^\pi(s',a').$$

Let us define the state-action polytope as follows:

$$\mathcal{K} := \{\nu | \, \nu \geq 0 \text{ and } \sum_a \nu(s,a) = (1-\gamma)\mu(s) + \gamma \sum_{s',a'} P(s|s',a')\nu(s',a')\}$$

We now see that this set precisely characterizes all state-action visitation distributions.

Lemma 1.11. *We have that $\mathcal{K}$ is equal to the set of all feasible state-action distributions, i.e. $\nu \in \mathcal{K}$ if and only if there exists a stationary (and possibly randomized) policy π such that $\nu_\mu^\pi = \nu$.*

With respect the variables $\nu \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, the dual LP formulation is as follows:

$$\begin{aligned} \max \quad & \frac{1}{1-\gamma} \sum_{s,a} \nu(s,a) r(s,a) \\ \text{subject to} \quad & \nu \in \mathcal{K} \end{aligned}$$

Note that $\mathcal{K}$ is itself a polytope, and one can verify that this is indeed the dual of the aforementioned LP. This approach provides an alternative approach to finding an optimal solution.

If $\nu^\star$ is the solution to this LP, then we have that:

$$\pi^\star(a|s) = \frac{\nu^\star(s,a)}{\sum_{a'} \nu^\star(s,a')}.$$

An alternative optimal policy is $\text{argmax}_a \, \nu^\star(s,a)$ (and these policies are identical if the optimal policy is unique).

1.5 Bibliographic Remarks

We refer the reader to [Puterman, 1994] for a more detailed treatment of dynamic programming and MDPs. [Puterman, 1994] also contains a thorough treatment of the dual LP, along with a proof of Lemma 1.11

With regards to the computational complexity of policy iteration, [Ye, 2011] showed that policy iteration is a strongly polynomial time algorithm for a fixed discount rate [1]. Also, see [Ye, 2011] for a good summary of the computational complexities of various approaches. [Mansour and Singh, 2013] showed that the number of iterations of policy iteration can be bounded by $\frac{|\mathcal{A}|^{|\mathcal{S}|}}{|\mathcal{S}|}$.

With regards to a strongly polynomial algorithm, the CIPA algorithm [Ye, 2005] is an interior point algorithm with the claimed runtime in Table 1.1.

Lemma 1.7 is due to Singh and Yee [1994].

[1] The stated strongly polynomial runtime in Table 1.1 for policy iteration differs from that in [Ye, 2011] due to we assume that the runtime per iteration of policy iteration is $|\mathcal{S}|^3 + |\mathcal{S}|^2|\mathcal{A}|$.

Bibliography

Richard Bellman. Dynamic programming and Lagrange multipliers. *Proceedings of the National Academy of Sciences*, 42(10):767–769, 1956.

Yishay Mansour and Satinder Singh. On the complexity of policy iteration. *UAI*, 01 2013.

Martin Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

Satinder Singh and Richard Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233, 1994.

Yinyu Ye. A new complexity result on solving the markov decision problem. *Math. Oper. Res.*, 30:733–749, 08 2005.

Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. *Math. Oper. Res.*, 36(4):593–603, 2011.

Appendix A

Concentration

Lemma A.1. *(Hoeffding's inequality) Suppose $X_1, X_2, \ldots X_n$ are a sequence of independent, identically distributed (i.i.d.) random variables with mean μ. Let $\bar{X}_n = n^{-1}\sum_{i=1}^n X_i$. Suppose that $X_i \in [b_-, b_+]$ with probability 1, then*

$$P(\bar{X}_n \geq \mu + \epsilon) \leq e^{-2n\epsilon^2/(b_+ - b_-)^2}.$$

Similarly,

$$P(\bar{X}_n \leq \mu - \epsilon) \leq e^{-2n\epsilon^2/(b_+ - b_-)^2}.$$

The Chernoff bound implies that with probability $1 - \delta$:

$$\bar{X}_n - EX \leq (b_+ - b_-)\sqrt{\ln(1/\delta)/(2n)}\,.$$

Lemma A.2. *(Bernstein's inequality) Suppose $X_1, \ldots, X_n$ are independent random variables. Let $\bar{X}_n = n^{-1}\sum_{i=1}^n X_i$, $\mu = \mathbb{E}\bar{X}_n$, and Var$(X_i)$ denote the variance of X_i. If $X_i - EX_i \leq b$ for all i, then*

$$P(\bar{X}_n \geq \mu + \epsilon) \leq \exp\left[-\frac{n^2\epsilon^2}{2\sum_{i=1}^n \mathit{Var}(X_i) + 2nb\epsilon/3}\right].$$

If all the variances are equal, the Bernstein inequality implies that, with probability at least $1 - \delta$,

$$\bar{X}_n - EX \leq \sqrt{2\mathrm{Var}(X)\ln(1/\delta)/n} + \frac{2b\ln(1/\delta)}{3n}.$$