MS&E338 Aligning Superintelligence Lecture Notes 00

Foundations for Reinforcement Learning

Authors: Benjamin Van Roy *March 28, 2024*

1 Preamble

The term *superintelligence* refers to an agent that can achieve specified goals more effectively than humans. The subject of reinforcement learning addresses the design of agents that learn to achieve specified goals. This class will build on the foundations of reinforcement learning. These lecture notes offer a concise treatment of material that will be used in the course.

2 The Agent-Environment Interface

In reinforcement learning, an agent interacts with an environment through an interface of the sort illustrated in Figure 1. At each time $t = 0, 1, 2, \ldots$, the agent executes an action A_t, and the environment produces an observation O_{t+1} in response. This form of sequential interaction encompasses an enormous range of activities. Dialogue systems serve as an example.

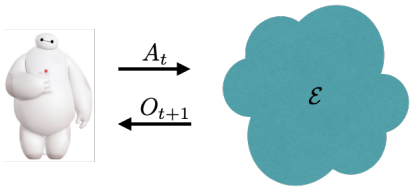

Figure 1: The agent-environment interface.

Example 1. (dialogue) *Consider an agent that engages in sequential correspondence, with information conveyed via messages exchanged with another party. The agent first transmits a message A_0 and receives a response O_1. Such exchanges continue. This interaction could be mediated, for example, by a text interface. For example, the first message A_0 could be "How can I help you?," with a response O_1 of "How does one replace a light bulb?" Subsequent messages transmitted by the agent could seek clarification regarding the task at hand and guide the process through completion.*

We consider a general framing in which the agent and environment interface through finite action and observation sets $\mathcal{A}$ and $\mathcal{O}$. At time t, immediately before selecting A_t, the agent's previous experience is comprised of a history $H_t = (A_0, O_1, A_1, \ldots, A_{t-1}, O_t)$ of actions and observations. We denote by $\mathcal{H}$ the set of all such finite-length histories. This includes the initial history $H_0 = ()$, which is of length zero.

2.1 Environments

Formally, an environment is a tuple $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho)$, identified by a finite set of actions $\mathcal{A}$, a finite set of observations $\mathcal{O}$, and a function ρ that for each history $h \in \mathcal{H}$, action $a \in \mathcal{A}$, and observation $o \in \mathcal{O}$, assigns a probability $\rho(o|h, a)$. Intuitively, at time t, the environment produces the next observation O_{t+1}

as though it is drawn from the probability mass function $\rho(\cdot|H_t, A_t)$. This tuple $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho)$ thus provides all information required to simulate the environment.

A dialogue of the kind presented in Example 1 can be framed in these terms. With a constrained number of characters per text message, the set $\mathcal{A} = \mathcal{O}$ of possibilities is finite. The function ρ assigns probabilities $\rho(\cdot|h, a)$ conditioned on previous messages, expressing the manner in which previous exchanges influence what may come next. For example, if a is "What is the wattage?" then $\rho(o|h, a)$ ought to be relatively large for messages o that communicate common wattage ratings.

Environments and agents can each be viewed as dynamic systems. An environment receives actions and produces observations. An agent receives observations and produces actions. But while the environment is given, the agent is designed to achieve particular goals. The subject of reinforcement learning addresses this design problem.

2.2 Agents

While the word *agent* may conjure an image of a physical entity such as a robot, we will only treat the design of algorithms that could be embedded in such an entity. In particular, we will not deal with design of physical mechanisms but only of what might be thought of as a *mind*. From the environment's perspective, this algorithm samples each action A_t from a probability mass function $\pi_{\text{agent}}(\cdot|H_t)$. We refer to such a function π_{agent} as a *policy*; more broadly, a policy π is a function that assigns a probability $\pi(a|h)$ to each action $a \in \mathcal{A}$ at each history $h \in \mathcal{H}$.

The subject of reinforcement learning tends to focus on the design of agents that function effectively across broad classes of environments rather than specific ones. It is in this sense that an agent learns. In particular, a reinforcement learning agent specializes over time as it learns about the specific environment with which it interacts. Let us consider arcade games as an example to illustrate this notion.

Example 2. (arcade games) *Any arcade game can be framed in terms of an environment $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho)$. Consider, for example, a classic arcade interface, as illustrated in Figure 2. A gamer exerts control via a joystick and a couple buttons while viewing through a screen an evolving pixelated image frame. To cast this in terms of our notation, take the set $\mathcal{A}$ of actions to be all combinations of joystick actuations and button presses. If the joystick can sense moves in four directions, there would be twenty actions consisting of combinations of five joystick states – up, down, left, right, stationary – and two states per button. The set of observations $\mathcal{O}$ is made up of possible frame renderings, which is large but finite. For simplicity, suppose the gamer's cognitive and motor skills allow it to observe the screen once per second and apply an action immediately after each observation. In this case, each time step lasts one second. Given a history $h \in \mathcal{H}$ and action $a \in \mathcal{A}$ there is some probability $\rho(o|h, a)$ of observing $o \in \mathcal{O}$ as the next image.*

One might consider designing an agent that plays a specific arcade game, like Tetris, or that can be applied more broadly, for example, to any arcade game played through a common interface. An effective reinforcement learning agent ought to develop skills over time, without advance knowledge of the specific game. In particular, it ought to select more effective actions after many trials, adapting its action probabilities $\pi_{\text{agent}}(\cdot|H_t)$ as it learns.

Figure 2: A classic arcade interface.

2.3 Interface versus Realization

An agent is designed for a specific *interface*, characterized by action and observation sets $\mathcal{A}$ and $\mathcal{O}$. The tuple $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho)$ defines an environment from the agent's perspective, with respect to this interface. For a fixed physical environment, changing the agent's interface, for example, by increasing the agent's capacity to observe and ingest data, would in general change this mathematical representation. Consider, for example, the same physical environment as observed through a monochrome versus color camera. An agent would have to interface through a different observation set $\mathcal{O}$, with a different observation probability function ρ, depending on the camera.

The tuple $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho)$ characterizes all the agent needs to know about an environment in order to reason about how its actions influence what it observes. However, it is important to note that this representation does not capture how the environment is *realized*. For example, with the arcade game example, many different software implementations can result in the same game dynamics, and thus the same function ρ. The specific implementation is not relevant from the gamer's perspective.

Analogously, while the agent may be implemented as a complex software module, from the perspective of an environment, which interacts with it only through the agent-environment interface, the agent can be viewed as a policy π_{agent}. In particular, given the observation probability function ρ and the policy π_{agent}, one can simulate agent environment interactions, and alternative implementations that generate observations and actions consistent with ρ and π_{agent} would not alter the distribution of histories.

3 Sources of Uncertainty

When designing an agent, the designer faces uncertainty about what the agent will experience after it is deployed. It can be helpful to attribute this uncertainty to multiple sources:

- *Epistemic uncertainty* is about the environment $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho)$. While the sets of actions $\mathcal{A}$ and observations $\mathcal{O}$ are known, the observation probabilities $\rho(o|h, a)$ can be random variables. The agent can then learn about these probabilities in order to improve its ability to make effective decisions.

- *Aleatoric uncertainty* concerns the unpredictably of observations that persists even when ρ is known. In particular, given a history $h \in \mathcal{H}$ and an action $a \in \mathcal{A}$, while $\rho(\cdot|h, a)$ assigns probabilities to possible immediate observations, the realization is randomly drawn.

- *Algorithmic uncertainty* may be introduced through computations carried out by the agent. For example, the agent could apply a randomized algorithm to selects actions in a manner that depends on internally generated random numbers.

To illustrate, let us consider how to attribute uncertainty arising in interactions between a simple agent and a simple coin tossing environment.

Example 3. (coin tossing) *Consider an environment involving two possibly biased coins, with probabilities p_1 and p_2, respectively, of landing heads. Uncertainty about these biases are expressed by a uniform distribution over the unit square. Observations and actions are binary, with $\mathcal{O} = \{0, 1\}$ and $\mathcal{A} = \{1, 2\}$. Heads and tails are indicated by zeros and ones, respectively. Over each time step, the agent executes an action A_t and observes $O_{t+1} \in \mathcal{O}$, according to conditional probabilities $\mathbb{P}(O_{t+1} = 1|p, A_t,) = p_{A_t}$ and $\mathbb{P}(O_{t+1} = 0|p, A_t) = 1 - p_{A_t}$.*

Suppose the agent implements Thompson sampling. Given the prior uncertainty about coin biases p, this algorithm at each time t and for each action a generates a sample $\tilde{p}_{t,a}$ from a beta distribution with parameters $(N_{t,a,1} + 1, N_{t,a,0} + 1)$, where $N_{t,a,o}$ is the number of those tosses of coin a that through time t that landed o. Then, the agent executes the action A_t that maximizes $\tilde{p}_{t,A_t}$.

The agent uses a randomized algorithm – Thompson sampling – and there is therefore algorithmic uncertainty about the choice of A_t given H_t. Where there is a choice is in how to designate other uncertainty as epistemic versus aleatoric.

We can model this as a formal environment $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho)$, with $\rho(1|h, a) = p_a$ and $\rho(0|h, a) = 1 - p_a$. These observation probabilities $\rho(o|h, a)$ depend, aside from o, h, and a, on the coin biases p. Hence, this environment model designates uncertainty about coin biases as epistemic and uncertainty about toss outcomes conditioned on coin biases as aleatoric.

We could alternatively designate uncertainty about both biases and toss outcomes as aleatoric. To do so, let

$$\rho'(o|H_t, A_t) = \frac{N_{t,H_t,A_t,o} + 1}{N_{t,H_t,A_t,0} + N_{t,H_t,A_t,1} + 2}.$$

Note that this ratio is equal to the posterior predictive $\mathbb{P}(O_{t+1} = o|H_t, A_t)$. This probability integrates over uncertainty about p. There is no uncertainty about the observation probability function ρ', defined in this way. Hence, the environment $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho')$ models all uncertainty as aleatoric, though from the agent's perspective, it is equivalent to the environment $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho)$, which exhibits epistemic uncertainty.

4 State

In this section, we introduce the concept of state and discuss the relation of our formulation to Markov decision processes.

4.1 Situational State

The history is a large and growing data structure. To simplify the agent's interpretation of history, it is often useful to construct a *situational state* that provides a summary of the history intended to serve the agent's purposes when it comes to reasoning about decisions.

The situational state is an element of the agent's design rather than the environment. In particular, the agent applies an update function f to produce each situational state

$$S_{t+1} = f(S_t, A_t, O_{t+1}).$$

We will denote the range of f by $\mathcal{S}$. This process begins with a fixed initial state $S_0 \in \mathcal{S}$.

Note that the number of situational states could be infinite, and in fact, we can take the situational state to be the history itself. In this case, $\mathcal{S} = \mathcal{H}$ and the update function serves to concatenate the most recent action and observation. In particular,

$$S_{t+1} = f(S_t, A_t, O_{t+1}) = S_t \cup (A_t, O_{t+1}) = H_t \cup (A_t, O_{t+1}) = H_{t+1},$$

where we use $\cup$ here to denote concatenation. More generally, the situational state can be designed to simplify the agent's interpretation of history, or at least information from history that ought to drive decisions. The following example illustrates.

Example 4. (queueing) *Consider a service station, as illustrated in Figure 3. At each time, the customer at the head of the queue departs if the station completes their service and any number of customers can arrive to enqueue. An agent observes arrivals and departures and selects the current service mode – fast or slow. Each customer pays* \$1 *upon arrival. No cost is incurred when the slow mode is applied or when there is no customer being served. The fast mode of service incurs a cost of* \$0.50 *per timestep.*

This can be modeled as an environment $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho)$ with $\mathcal{A} = \{\text{fast}, \text{slow}\}$ and $\mathcal{O} = \mathbb{Z}_+^2$, with each observation indicating the numbers of arrivals and departures. The observation probability function ρ can exhibit complex dependencies on history, reflecting, for example, the reputational impact of past wait times on current arrival rates. Regardless, an agent may be able to simplify its task by choosing the service mode based on the current queue length rather than all history. With this approach, the agent is treating the queue length as a situational state. This situational state can be updated according to

$$S_{t+1} = f(S_t, A_t, O_{t+1}) = A_t + \text{arrivals}(O_{t+1}) - \text{departures}(O_{t+1}).$$

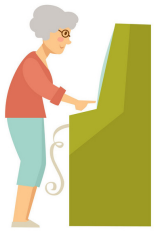

Figure 3: A service station serving a customer.

4.2 State-Modulated Policies

Typical agent designs make use of state-modulated policies. Such a policy π_t generates an action A_t in a manner that depends on the history H_t only through a situational state S_t. For a state-modulated policy, we can write action probabilities as $\pi_t(A_t|S_t) = \pi_t(A_t|H_t)$, replacing the notation for history with state.

In agent typical designs that apply state-modulated policies, the policy π_t itself changes over time and depends on the history H_t. In this case, π_t is a random variable. Thus, the probability $\mathbb{P}(A_t = a|S_t) = \mathbb{E}[\pi_t(a|S_t)|S_t]$ can differ from $\mathbb{P}(A_t = a|H_t, S_t) = \pi_t(a|S_t)$. It is also worth noting that π_t differs from the agent policy π_{agent}, which does not change over time. These policies represent alternative characterizations of the agent's decision process. Their relation can be expressed by $\pi_{\text{agent}}(a|H_t) = \pi_t(a|S_t)$.

The situational state and notion of state-modulated policy serve to simplify the process of agent design. A situational state can be a much simpler object, expressing much less information than the history. Ideally, it is a dramatic compression of history that preserves information that is salient for decision making. To illustrate, consider our queueing example (Example 4). In this environment, arrival and service rates can depend on history in complicated ways. However, instead of designing a complex agent that accounts for those intricacies, we can take the queue length to be the situational state and restrict attention to state-modulated policies. With a reinforcement learning algorithm – for example, Q-learning – that adapts the policy over time, we might arrive at effective though perhaps not optimal behavior.

4.3 Markov Decision Processes

Given an environment $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho)$, we say that an update function f induces a Markov decision process if the environment dynamics are such that $S_{t+1} \perp H_t|\mathcal{E}, S_t, A_t$. That is, conditioned on the current state and action, as well as the environment, the next state is independent of the history. When this is the case, state dynamics are characterized by transition probabilities, defined by

$$P_{ass'} = \sum_{o \in \mathcal{O}: s' = f(s,a,o)} \rho(o|h, a),$$

where h is a history that leads to the situational state s'. In particular, if $h = (a_0, o_1, a_2, \ldots, o_t)$ then $s_0 = S_0$ and and $s' = s_t$, which is generated by an iteartion $s_{k+1} = f(s_k, a_k, o_{k+1})$ for $k = 0, \ldots, t-1$. Note that, because f induces a Markov decision process, conditioned on S_t and A_t, $P_{A_t S_t S_{t+1}}$ is independent of the history H_t. Hence, the choice of history h among those that generate s does not impact the definition of $P_{ass'}$ for state-action-state triples that occur with nonzero probability.

A Markov decision process (MDP) is identified by a tuple $(\mathcal{A}, \mathcal{S}, S_0, P)$. The object P supplies a transition probability matrix P_a for each action $a \in \mathcal{A}$. For each state-modulated policy π, the transition matrix P determines a distribution over state-action trajectories. In particular, each action is distributed according to $\mathbb{P}_\pi(A_t|S_t) = \pi(a|S_t)$, and each next state is distributed according to $\mathbb{P}_\pi(S_{t+1} = s|S_t, A_t) = P_{A_t S_t s}$.

It is interesting to consider a special case where the situational state S_t is taken to be the history H_t. In this case, the update function is

$$f(H_t, A_t, O_{t+1}) = H_t \cup (A_t, O_{t+1}).$$

For any environment $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho)$, $H_{t+1} \perp H_t | \mathcal{E}, H_t, A_t$, and therefore, this update function induces an MDP. The state space of this MDP is the set of histories, and the update function simply appends the latest action and observation. For this MDP, any policy is state-modulated because a policy cannot depend on more than the history itself.

With some abuse of notation, for each state-modulated policy π, we let

$$P_{\pi ss'} = \sum_{a \in \mathcal{A}} \pi(a|s) P_{ass'}.$$

This represents the probability that executing policy π at state s results in a transition to state s'. Note that P_a and P_π are random matrices since they depend on $\mathcal{E}$.

We refer to P_a and P_π, each a matrix of probabilities $P_{ass'}$ or $P_{\pi ss'}$, as transition matrices. Working with transition matrices facilitates the study of and reinforcement learning, deriving benefit from the tools of matrix algebra. For example, probabilities of transitions that occur over multiple time periods can be obtained via matrix multiplication. In particular, for all $s, s' \in \mathcal{S}$, $\mathbb{P}_\pi(S_{t+k} | S_t, \mathcal{E}) = (P_\pi^k)_{S_t S_{t+k}}$. Using matrix notation, we also can write state occurrence probabilities as $\mathbb{P}_\pi(S_t | \mathcal{E}) = (P_\pi^t)_{S_0 S_t}$. Further, for any function $g : \mathcal{S} \mapsto \mathbb{R}$, we have $\mathbb{E}_\pi[g(S_t)|\mathcal{E}] = P_\pi^t g(S_0)$, where in this expression g is interpreted as a vector in $\mathbb{R}^{\mathcal{S}}$.

5 Reward and Return

In reinforcement learning, an agent is designed with respect to particular goals or objectives, encoded in terms of rewards. We consider reward functions $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ that assign rewards to triples according to $R_{t+1} = r(S_t, A_t, S_{t+1})$.

In principle, we could instead take the reward function to be an arbitrary function of history. However, computing a reward at each time that depends on the entirety of a growing history becomes impractical. As such, rewards are typically defined to depend on history only through a simpler situational state.

While we could consider other objectives, we assume in this course that agents are designed to maximize the expected discounted return:

$$\mathbb{E}_{\pi_{\text{agent}}} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \right],$$

for some discount factor $\gamma \in [0, 1)$. Note that this expectation integrates over all sources of uncertainty: epistemic, aleatoric, and algorithmic. We will discuss later in the course axiomatic reasoning that motivates this objective.

6 Value and Policy

Value functions and policies that can be derived from them serve as useful tools for analysis and computation in reinforcement learning. In this section, we define these terms and present some key results pertaining to them.

6.1 Value Functions

We will denote immediate expected reward upon executing $a \in \mathcal{A}$ at $s \in \mathcal{S}$ by

$$\overline{r}_{as} = \sum_{s' \in \mathcal{S}} P_{ass'} r(s, a, s').$$

Further, denote the immediate expected reward under policy π by $\overline{r}_{\pi s} = \sum_{a \in \mathcal{A}} \pi(a|s) \overline{r}_{as}$. We will often view $\overline{r}_\pi$ as a column vector in $\mathbb{R}^{\mathcal{S}}$ so that the expected discounted return can be written as

$$\overline{V}_\pi = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \middle| \mathcal{E} \right] = \sum_{t=0}^{\infty} \gamma^t P_\pi^t \overline{r}_\pi(S_0).$$

We will often work with a notion of expected discounted return starting at a state s:

$$V_\pi(s) = \sum_{t=0}^{\infty} \gamma^t P_\pi^t \overline{r}_\pi(s).$$

If the reward function is bounded, this quantity is well-defined and uniformly bounded over states and policies. We refer to V_π as the *value function* of policy π. Note that $\overline{V}_\pi = V_\pi(S_0)$.

The *optimal* value function V_* is defined by

$$V_*(s) = \sup_\pi V_\pi(s).$$

This is the maximal expected discounted return starting at state s. Note that

$$\overline{V}_* = \sup_\pi \overline{V}_\pi = \sup_\pi V_\pi(S_0) = V_*(S_0).$$

We will also often work with *action value functions*, defined by

$$Q_\pi(s,a) = \overline{r}_{as} + \gamma P_a V_\pi(s) \quad \text{and} \quad Q_*(s,a) = \sup_\pi Q_\pi(s,a),$$

for $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Note that $Q_\pi(s,a)$ is the expected sum of rewards starting at state s, conditioned on immediately executing action a and subsequently following policy π. From their definitions, it is easy to verify that value functions can be derived from action value functions according to

$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q_\pi(s,a) \quad \text{and} \quad V_*(s) = \max_{a \in \mathcal{A}} Q_*(s,a).$$

Given V_* an agent can select an optimal action by planning over a single time step. In particular, it suffices to maximize the sum of immediate reward and the optimal value at the next timestep. For any state s and action a, this sum is the optimal action value $Q_*(s,a)$.

6.2 From Value to Policy

The optimal value $V_*(s)$ is the maximal expected reward starting at state s. The optimal action value $Q_*(s,a)$ is the maximal expected reward if the immediate action is required to be a. Hence, the difference $V_*(s) - Q_*(s,a)$ can be interpreted as the expected loss from selecting action a. The following result expresses the shortfall $V_*(s) - V_\pi(s)$ as an expected sum of such losses.

Theorem 1. (shortfall decomposition) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, discount factors $\gamma \in [0,1)$, state-modulated policies π, and states $s \in \mathcal{S}$,*

$$V_*(s) - V_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \sum_{s' \in \mathcal{S}} (P_\pi^t)_{ss'} \left(V_*(s') - \sum_{a \in \mathcal{A}} \pi(a|s') Q_*(s',a) \right).$$

Proof. We have

$$V_\pi = \sum_{t=0}^{\infty} \gamma^t P_\pi^t \overline{r}_\pi = \sum_{t=0}^{\infty} \gamma^t P_\pi^t (\overline{r}_\pi + \gamma P_\pi V_* - \gamma P_\pi V_*).$$

It follows that

$$\begin{aligned} V_* - V_\pi &= V_* - \sum_{t=0}^{\infty} \gamma^t P_\pi^t (\overline{r}_\pi + \gamma P_\pi V_* - \gamma P_\pi V_*) \\ &= V_* + \sum_{t=0}^{\infty} \gamma^{t+1} P_\pi^{t+1} V_* - \sum_{t=0}^{\infty} \gamma^t P_\pi^t (\overline{r}_\pi + \gamma P_\pi V_*) \\ &= \sum_{t=0}^{\infty} \gamma^t P_\pi^t (V_* - (\overline{r}_\pi + \gamma P_\pi V_*)), \end{aligned}$$

and

$$\begin{aligned} V_*(s) - V_\pi(s) &= \sum_{t=0}^{\infty} \sum_{s' \in \mathcal{S}} \gamma^t (P_\pi^t)_{ss'} (V_*(s') - (\overline{r}_\pi + P_\pi V_*)(s')) \\ &= \sum_{t=0}^{\infty} \sum_{s' \in \mathcal{S}} \gamma^t (P_\pi^t)_{ss'} \left(V_*(s') - \sum_{a \in \mathcal{A}} \pi(a|s') Q_*(s', a) \right). \end{aligned}$$

The result follows. □

Note that specializing this result to the case of $s = S_0$ yields an expression for the lifetime shortfall $\overline{V}_* - \overline{V}_\pi$.

Definition 2. *A state-modulated policy π is* **greedy with respect to** $Q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ *if $\sum_{a \in \mathcal{A}} \pi(a|s) Q(s,a) = \max_{a \in \mathcal{A}} Q(s,a)$ for all $s \in \mathcal{S}$, and the policy is* **greedy with respect to** $V \in \mathbb{R}^{\mathcal{S}}$ *if it is greedy with respect to $Q(s,a) = \overline{r}_{as} + P_a V(s)$ for all $s \in \mathcal{S}$.*

In other words, a policy is greedy with respect to Q if, at every state s, it selects only actions that maximize the value prescribed by Q. Finally, for any $s \in \mathcal{S}$, we say that π is greedy with respect to Q at s if $\pi(\cdot|hs$ selects only actions that maximize $Q(s, \cdot)$.

Recall that a policy π_* is *optimal* if $\overline{V}_{\pi_*} = \overline{V}_*$. We say that $s \in \mathcal{S}$ is *supported by* a policy π if, under policy π, it occurs with positive probability; in other words, $\sum_{t=0}^{\infty} (P_\pi^t)_{S_0 s} > 0$. The shortfall decomposition implies the following.

Corollary 3. (optimality of greedy policy) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, and discount factors $\gamma \in [0, 1)$, a policy π_* is optimal if and only if it is greedy with respect to the optimal action value function Q_* at all supported histories.*

There is also a stronger notion of optimality.

Definition 4. *A policy π_* is* **uniformly optimal** *if $V_{\pi_*} = V_*$.*

Note that any uniformly optimal policy is also optimal. A uniformly optimal policy attains optimal expected future reward starting at any state, including those not supported by the policy. On the other hand, to be *optimal*, it suffices for a policy to maximize expected reward starting at S_0. In particular, an optimal policy does not necessarily optimize expected reward starting at a state that it has no chance of reaching. The following analog to Corollary 3 pertains to uniform optimality and follows from Theorem 1.

Corollary 5. (uniform optimality of greedy policy) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, and discount factors $\gamma \in [0, 1)$, a policy π_* is uniformly optimal if and only if it is greedy with respect to the optimal action value function Q_*.*

Access to the action value function can greatly simplify an agent's reasoning, as it effectively reduces a multi-step decision problem to one with a single step. The action value function Q_* summarizes all

information about the future that is relevant to the immediate decision, allowing the agent to simply select among actions the one that maximizes $Q_*(S_t, \cdot)$.

In MDPs the model complex environments, there are typically so many states that representing Q_* exactly is infeasible. As such, one must resort to approximations. Given an approximation $\tilde{Q}$, a simple approach to decision making involves selecting the action A_t that maximizes $\tilde{Q}(S_t, \cdot)$. This results in a policy $\tilde{\pi}$, which is greedy with respect to $\tilde{Q}$, and Theorem 1 yields insight into its shortfall. In particular,

$$\begin{aligned}\overline{V}_* - \overline{V}_{\tilde{\pi}} &= \sum_{t=0}^{\infty} \gamma^t \sum_{s' \in \mathcal{S}} (P_\pi^t)_{S_0 s'} \left(V_*(s') - \sum_{a \in \mathcal{A}} \pi(a|s') Q_*(s', a) \right) \\ &= \sum_{t=0}^{\infty} \gamma^t \sum_{s' \in \mathcal{S}} (P_\pi^t)_{S_0 s'} \left(V_*(s') - \tilde{V}(s') \right) + \sum_{t=0}^{\infty} \gamma^t \sum_{s' \in \mathcal{S}} (P_\pi^t)_{S_0 s'} \sum_{a \in \mathcal{A}} \pi(a|s') \left(\tilde{Q}(s', a) - Q_*(s', a) \right),\end{aligned}$$

where $\tilde{V}(S_t) = \max_{a \in \mathcal{A}} \tilde{Q}(S_t, a) = \tilde{Q}(S_t, A_t)$. Hence, the shortfall can be expressed in terms of errors in approximating V_* and errors in approximating Q_*. Note that adding any function $g \in \mathbb{R}^{\mathcal{S}}$ to produce a modified approximation $\tilde{Q}(s, a) + g(s)$ does not impact this expression. As such, only relative values of $\tilde{Q}(S_t, \cdot)$ across actions influence the shortfall.

7 Bellman Equations and Bellman Error

Bellman equations offer useful characterizations of value functions. They equate the value at a state with the expected sum of immediate reward and value at the next history. That this relation should hold is intuitive – value represents an expected sum of future rewards and this sum can be decomposed into what is accrued immediately versus later. In this section, we introduce Bellman equations and also study properties of the Bellman error, which measures the extent to which Bellman equations are violated.

7.1 Bellman Equations

Bellman equations admit unique solutions and therefore offer sharp characterizations of optimal value functions. The following theorem formalizes this general technical result.

Theorem 6. (optimal Bellman equations) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, and discount factors $\gamma \in [0, 1)$, among bounded functions $(V, Q) \in \mathbb{R}^{\mathcal{S}} \times \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$, the pair (V_*, Q_*) uniquely solves*

$$Q(s, a) = \overline{r}_{as} + \sum_{s' \in \mathcal{S}} \gamma P_{ass'} V(s') \quad \textit{and} \quad V(s) = \max_{a' \in \mathcal{A}} Q(s, a'),$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

Proof. It is easy to verify from definitions that $V_*(s) = \max_{a' \in \mathcal{A}} Q_*(s, a')$. Let π_* be a uniformly optimal policy. Then,

$$\begin{aligned} Q_*(s, a) &= \sup_\pi Q_\pi(s, a) \\ &= \overline{r}_{as} + \sup_\pi \sum_{s' \in \mathcal{S}} \gamma P_{ass'} V_\pi(s') \\ &= \overline{r}_{as} + \sum_{s' \in \mathcal{S}} \gamma P_{ass'} V_{\pi_*}(s') \\ &= \overline{r}_{as} + \sum_{s' \in \mathcal{S}} \gamma P_{ass'} V_*(s') \\ &= \overline{r}_{as} + \gamma P_a V_*(s). \end{aligned}$$

It follows that (V_*, Q_*) solve the Bellman equation.

We will now establish that (V_*, Q_*) is the unique pair of bounded functions that solves the Bellman equations. Assume for contradiction that there is another pair $(\overline{V}, \overline{Q})$ of bounded functions that solve the Bellman equations. Let $\overline{\pi}$ be greedy with respect to $\overline{Q}$. Then,

$$\overline{V} = \overline{r}_{\overline{\pi}} + \gamma P_{\overline{\pi}} \overline{V} \geq \overline{r}_{\pi_*} + \gamma P_{\pi_*} \overline{V},$$

and iterating these relations T times gives us

$$\overline{V} = \sum_{t=0}^{T-1} \gamma^t P_{\overline{\pi}}^t \overline{r}_{\overline{\pi}} + \gamma^T P_{\overline{\pi}}^T \overline{V} \geq \sum_{t=0}^{T-1} \gamma^t P_{\pi_*}^t \overline{r}_{\pi_*} + \gamma^T P_{\pi_*}^T \overline{V}.$$

Let $z = \|\overline{V}\|_\infty$, which is finite because $\overline{V}$ is bounded, so that $z\mathbf{1} \geq \overline{V} \geq -z\mathbf{1}$. It follows that

$$\sum_{t=0}^{T-1} \gamma^t P_{\overline{\pi}}^t \overline{r}_{\overline{\pi}} + z\gamma^T P_{\overline{\pi}}^T \mathbf{1} \geq \overline{V} \geq \sum_{t=0}^{T-1} \gamma^t P_{\overline{\pi}}^t \overline{r}_{\overline{\pi}} - z\gamma^T P_{\overline{\pi}}^T \mathbf{1}.$$

We have

$$\lim_{T\to\infty} \|\gamma^T P_{\overline{\pi}}^T \mathbf{1}\|_\infty = \lim_{T\to\infty} \gamma^T = 0,$$

and therefore,

$$\lim_{T\to\infty} \left\| \overline{V} - \sum_{t=0}^{T-1} \gamma^t P_{\overline{\pi}}^t \overline{r}_{\overline{\pi}} \right\|_\infty \leq z \lim_{T\to\infty} \|\gamma^T P_{\overline{\pi}}^T \mathbf{1}\|_\infty = 0.$$

We can conclude that $\overline{V} = \sum_{t=0}^\infty \gamma^t P_{\overline{\pi}}^t \overline{r}_{\overline{\pi}} = V_{\overline{\pi}}$, and via an analogous argument, $\overline{V} \geq \sum_{t=0}^\infty \gamma^t P_{\pi_*}^t \overline{r}_{\pi_*} = V_{\pi_*}$. This implies that $\overline{\pi}$ is uniformly optimal, and therefore, $\overline{V} = V_*$. Further, for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$,

$$\overline{Q}(s,a) = \overline{r}_{as} + \sum_{s' \in \mathcal{S}} \gamma P_{ass'} \overline{V}(s') = \overline{r}_{as} + \sum_{s' \in \mathcal{S}} \gamma P_{ass'} V_*(s') = Q_*(s,a),$$

and therefore (V_*, Q_*) is the unique solution among bounded functions to the Bellman equations. □

A similar result characterizes policy value functions (V_π, Q_π) in terms of another set of Bellman equations.

Theorem 7. (fixed-policy Bellman equations) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, discount factors $\gamma \in [0,1)$, and state-modulated policies π, among bounded functions $(V, Q) \in \mathbb{R}^{\mathcal{S}} \times \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, the pair (V_π, Q_π) uniquely solves*

$$Q(s,a) = \overline{r}_{as} + \sum_{s' \in \mathcal{S}} \gamma P_{ass'} V(s') \quad \textit{and} \quad V(s) = \sum_{a' \in \mathcal{A}} \pi_s(a') Q(s,a').$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

We omit the proof, as this result can be established using arguments entirely analogous to those from the proof of Theorem 6.

7.2 Bellman Operators

By Theorem 6,

$$V_*(s) = \max_{a' \in \mathcal{A}} \left(\overline{r}_{as} + \sum_{s' \in \mathcal{S}} \gamma P_{ass'} V_*(s') \right) \quad \forall s \in \mathcal{S},$$

and

$$Q_*(s,a) = \overline{r}_{as} + \sum_{s' \in \mathcal{S}} \gamma P_{ass'} \max_{a' \in \mathcal{A}} Q_*(s',a') \quad \forall s \in \mathcal{S}, a \in \mathcal{A}.$$

Bellman operators offer shorthand notation for expressions of this sort. In particular, for any $V \in \mathbb{R}^{\mathcal{S}}$, let

$$(FV)(s) = \max_{a' \in \mathcal{A}} \left(\overline{r}_{as} + \sum_{s' \in \mathcal{S}} \gamma P_{ass'} V(s') \right) \quad \forall s \in \mathcal{S},$$

and, for any $Q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$, let

$$(FQ)(s,a) = \overline{r}_{as} + \sum_{s' \in \mathcal{S}} \gamma P_{ass'} \max_{a' \in \mathcal{A}} Q(s',a') \quad \forall s \in \mathcal{S}, a \in \mathcal{A}.$$

Note that $(FV)(s)$ is the maximum over immediate actions of expected future reward if expected reward after the current time step is given by V. Similarly, $(FQ)(s,a)$ is the expected immediate reward generated by executing action a plus the maximum over the next action of subsequent expected reward if that is given by Q. These operators facilitate concise statement of Bellman equations. In particular, Theorem 6 can equivalently be stated as follows: $V_* \in \mathbb{R}^{\mathcal{S}}$ and $Q_* \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ are the unique solutions among bounded functions to $V = FV$ and $Q = FQ$.

We additionally define Bellman operators associated with each policy π by

$$(F_\pi V)(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\overline{r}_{as} + \sum_{s' \in \mathcal{S}} \gamma P_{ass'} V(s') \right) \quad \forall s \in \mathcal{S}$$

and

$$(F_\pi Q)(s,a) = \overline{r}_{as} + \sum_{s' \in \overline{\mathcal{H}}} \gamma P_{ass'} \sum_{a' \in \mathcal{A}} \pi(a'|s) Q(s',a') \quad \forall s \in \mathcal{S}, a \in \mathcal{A}.$$

With this notation, Theorem 7 can equivalently be stated as follows: $V_\pi \in \mathbb{R}^{\mathcal{S}}$ and $Q_\pi \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ are the unique solutions among bounded functions to $V = F_\pi V$ and $Q = F_\pi Q$.

7.3 Bellman Error

If a bounded function Q solves the Bellman equation $Q = F_\pi Q$ then $Q = Q_\pi$. The following result establishes that this holds in an approximate sense when Q approximately solves the Bellman equation.

Theorem 8. (policy Bellman error) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, discount factors $\gamma \in [0,1)$, states $s \in \mathcal{S}$, policies π, and functions $Q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$,*

$$V(s) - V_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \sum_{s' \in \mathcal{S}} (P_\pi^t)_{ss'} \sum_{a \in \mathcal{A}} \pi(a|s')(Q(s',a) - F_\pi Q(s',a)),$$

where $V(s') = \sum_{a \in \mathcal{A}} \pi(a|s') Q(s',a)$ for all $s' \in \mathcal{S}$.

Proof. We have

$$\begin{aligned}
\sum_{t=0}^{\infty} \gamma^t P_\pi^t V &= V + \sum_{t=1}^{\infty} \gamma^t P_\pi^t V \\
&= V + \sum_{t=0}^{\infty} \gamma^{t+1} P_\pi^t P_\pi V \\
&= V - V_\pi + \sum_{t=0}^{\infty} \gamma^t P_\pi^t (\overline{r}_\pi + \gamma P_\pi V) \\
&= V - V_\pi + \sum_{t=0}^{\infty} \gamma^t P_\pi^t F_\pi V.
\end{aligned}$$

and it follows that,

$$\begin{aligned} V(s) - V_\pi(s) &= \sum_{t=0}^{\infty} \gamma^t P_\pi^t (V - F_\pi V)(s) \\ &= \sum_{t=0}^{\infty} \gamma^t \sum_{s' \in \mathcal{S}} (P_\pi^t)_{ss'} \sum_{a \in \mathcal{A}} \pi(a|s')(Q(s',a) - F_\pi Q(s',a)). \end{aligned}$$

□

This result implies one that is perhaps easier to parse: for all times t,

$$V(S_t) - V_\pi(S_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k (Q(S_{t+k}, A_{t+k}) - F_\pi Q(S_{t+k}, A_{t+k})) \Big| \mathcal{E}, S_t \right].$$

The difference $Q - F_\pi Q$ is referred to as the *Bellman error* of Q for policy π since it captures the manner in which the Bellman equation is violated. When this Bellman error is zero, Q solves the Bellman equation $Q = F_\pi Q$. In this case, the right-hand-side of the theorem's equation becomes zero, and as such, the theorem indicates that $V = V_\pi$. But the theorem also bears implication on cases where Q differs from $F_\pi Q$, expressing $V(S_0) - V_\pi(S_0)$ as an expected discounted sum of Bellman errors experienced over the time.

Note that Theorem 8 implies a similar result involving value instead of action value function Bellman errors. In particular,

$$V - V_\pi = \sum_{t=0}^{\infty} \gamma^t P_\pi^t (V - F_\pi V).$$

This is because, there is always some $Q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ such that $V(s') = \sum_{a \in \mathcal{A}} \pi(a|s') Q(s',a)$ for all $s' \in \mathcal{S}$, and for such a function Q, we have $\sum_{a \in \mathcal{A}} \pi(a|s') F_\pi Q(s',a) = F_\pi V(s')$.

Specializing Theorem 8 by taking π to be a greedy policy with respect to Q yields the following corollary.

Corollary 9. (greedy Bellman error) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, discount factors $\gamma \in [0,1)$, states $s \in \mathcal{S}$, policies π, and functions $Q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$, if π is greedy with respect to Q then*

$$V(s) - V_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \sum_{s' \in \mathcal{S}} (P_\pi^t)_{ss'} \sum_{a \in \mathcal{A}} \pi(a|s')(Q(s',a) - FQ(s',a)),$$

where $V(s') = \max_{a \in \mathcal{A}} Q(s',a)$ for all $s' \in \mathcal{S}$.

The difference $Q - FQ$ represents the *greedy Bellman error*. When this is equal to zero, $Q = Q_*$ and π is optimal. When the difference is nonzero, the expected discounted sum of the corollary gives $V(s) - V_\pi(s)$, which is the error of $V(s)$ as an estimate of expected discounted return under policy π. It also immediately follows that

$$V - V_\pi = \sum_{t=0}^{\infty} \gamma^t P_\pi^t (V - FV).$$

Shortfall can also be bounded by Bellman error when the action value function is optimistic.

Definition 10. *A value function $V \in \mathbb{R}^{\mathcal{S}}$ or action value function $Q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ is* **optimistic** *if $V \geq V_*$ or $Q \geq Q_*$, respectively. More specifically, a value function $V \in \mathbb{R}^{\mathcal{S}}$ or action value function $Q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ is* **optimistic at** *s or (s,a) if $V(s) \geq V_*(s)$ or $Q(s,a) \geq Q_*(s,a)$, respectively.*

If V is optimistic at s then the shortfall of a greedy policy can be bounded by Bellman error because $V_*(s) - V_\pi(s) \leq V(s) - V_\pi(s)$. This yields the following corollary.

Corollary 11. (optimistic Bellman error) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, discount factors $\gamma \in [0,1)$, states $s \in \mathcal{S}$, policies π, and functions $Q \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ and $V \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ with $V(s') = \max_{a\in\mathcal{A}} Q(s',a)$ for all $s' \in \mathcal{S}$, if V is optimistic and π is greedy with respect to Q then*

$$V_*(s) - V_\pi(s) \leq \sum_{t=0}^{\infty} \gamma^t \sum_{s'\in\mathcal{S}} (P_\pi^t)_{ss'} \sum_{a\in\mathcal{A}} \pi(a|s')(Q(s',a) - FQ(s',a)).$$

This result implies that, under the same conditions, for all times t,

$$V_*(S_t) - V_\pi(S_t) \leq \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k (Q(S_{t+k}, A_{t+k}) - F_\pi Q(S_{t+k}, A_{t+k})) \Big| \mathcal{E}, S_t \right].$$

8 Planning Algorithms and Their Analysis

Planning algorithms aim to produce policies that attain high return. In this section, we present a couple basic planning algorithms and concepts that are useful to their analysis.

8.1 Value Iteration

Value iteration produces a sequence $(Q_k \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}} : k \in \mathbb{Z}_+)$ of approximations converging on Q_*. Initialized with any Q_0, iterates are generated according to $Q_{k+1} = FQ_k$. The algorithm's progression can alternatively be studied through a sequence $(V_k \in \mathbb{R}^{\mathcal{S}} : k \in \mathbb{Z}_+)$, with each iterate defined by $V_k(s) = \max_{a\in\mathcal{A}} Q_k(s,a)$. This sequence similarly evolves according to $V_{k+1} = FV_k$. Action value functions can be written as $Q_{k+1}(s,a) = \overline{r}_{as} + P_a V_k(s)$. As established by the following theorem, for each $k \in \mathbb{Z}_+$ and $s \in \mathcal{S}$, the value $V_k(s)$ can be interpreted as the optimal expected future discounted return if the horizon is truncated after k time steps, with an additional terminal reward given by V_0.

Theorem 12. (truncated horizon) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, discount factors $\gamma \in [0,1)$, and bounded value functions $V_0 \in \mathbb{R}^{\mathcal{S}}$, if $V_{k+1} = FV_k$ for all $k \in \mathbb{Z}_+$ then, for all $s \in \mathcal{S}$,*

$$V_k(s) = \sup_\pi \left(\sum_{t=0}^{k-1} \gamma^t P_\pi^t \overline{r}_\pi + \gamma^k P_\pi^k V_0 \right)(s).$$

Proof. We will establish the result via induction on k. The result trivially holds for $k = 0$. Assume that the result holds for some k. Then,

$$\begin{aligned}
V_{k+1}(s) &= FV_k(s) \\
&= \max_{a\in\mathcal{A}} \left(\overline{r}_{as} + \sum_{s'\in\mathcal{S}} \gamma P_{ass'} V_k(s') \right) \\
&= \max_{a\in\mathcal{A}} \left(\overline{r}_{as} + \sum_{s'\in\mathcal{S}} \gamma P_{ass'} \sup_\pi \left(\sum_{t=0}^{k-1} \gamma^t P_\pi^t \overline{r}_\pi + \gamma^k P_\pi^k V_0 \right)(s') \right) \\
&= \sup_\pi \left(\overline{r}_{\pi s} + \sum_{s'\in\mathcal{S}} \gamma P_{ass'} \left(\sum_{t=0}^{k-1} \gamma^t P_\pi^t \overline{r}_\pi + \gamma^k P_\pi^k V_0 \right)(s') \right) \\
&= \sup_\pi \left(\sum_{t=0}^{k} \gamma^t P_\pi^t \overline{r}_\pi + \gamma^{k+1} P_\pi^{k+1} V_0 \right)(s).
\end{aligned}$$

□

This result implies that, for all times t,

$$V_k(S_t) \leq \sup_\pi \mathbb{E}_\pi \left[\sum_{n=0}^{k-1} \gamma^n R_{t+n} + \gamma^k V_0(S_{t+k}) \Big| \mathcal{E}, S_t \right].$$

8.2 The Contraction Property

For any functions $V \in \mathbb{R}^{\mathcal{S}}$ and $Q \in \mathbb{R}^{\mathcal{S}\tilde{\mathcal{A}}}$, we denote maximum norms by

$$\|V\|_\infty = \sup_{s \in \mathcal{S}} |V(s)| \quad \text{and} \quad \|Q\|_\infty = \sup_{s \in \mathcal{S}} \max_{a \in \mathcal{A}} |Q(s,a)|.$$

Bellman operators are contraction mappings with respect to the maximum norm. We say an operator G is a *contraction mapping* with respect to a norm $\|\cdot\|$ and contraction factor $\gamma \in [0,1)$ if $\|Gg - Gg'\| \leq \gamma \|g - g'\|$ for all functions g and g'.

Theorem 13. (contraction) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, discount factors $\gamma \in [0,1)$, and policies π, F and F_π are contraction mappings with respect to norm $\|\cdot\|_\infty$ and contraction factor γ.*

Proof. We will establish the result for the optimal Bellman operator F on value functions. Analogous arguments apply to operators on action value functions as well as policy-specific operators F_π.

For all $V, \overline{V} \in \mathbb{R}^{\mathcal{S}}$,

$$\begin{aligned}
\|FV - F\overline{V}\|_\infty &= \sup_{s \in \mathcal{S}} \left| \max_{a \in \mathcal{A}} \left(\overline{r}_{as} + \gamma \sum_{s' \in \mathcal{S}} P_{ass'} V(s') \right) - \max_{a \in \mathcal{A}} \left(\overline{r}_{as} + \gamma \sum_{s' \in \mathcal{S}} P_{ass'} \overline{V}(s') \right) \right| \\
&\leq \gamma \sup_{s \in \mathcal{S}} \max_{a \in \mathcal{A}} \left| \sum_{s' \in \mathcal{S}} P_{ass'} (V(s') - \overline{V}(s')) \right| \\
&\leq \gamma \sup_{s \in \mathcal{S}} \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{ass'} \left| V(s') - \overline{V}(s') \right| \\
&\leq \gamma \sup_{s \in \mathcal{S}} \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{ass'} \|V - \overline{V}\|_\infty \\
&= \gamma \|V - \overline{V}\|_\infty.
\end{aligned}$$

The first inequality follows from the fact that, for all scalar functions $f, \overline{f} \in \mathbb{R}^{\mathcal{A}}$ of actions, $|\max_{a \in \mathcal{A}} f(a) - \max_{a \in \mathcal{A}} \overline{f}(a)| \leq \max_{a \in \mathcal{A}} |f(a) - \overline{f}(a)|$. The third inequality follows from Jensen's. □

For $(V_k : k \in \mathbb{Z}_+)$ generated by value iteration, the contraction property implies that

$$\|V_* - V_k\|_\infty = \|F^k V_* - F^k V_0\|_\infty = \gamma^k \|V_* - V_0\|_\infty.$$

Similarly, for $(Q_k : k \in \mathbb{Z}_+)$ generated by value iteration,

$$\|Q_* - Q_k\|_\infty = \|F^k Q_* - F^k Q_0\|_\infty \leq \gamma^k \|Q_* - Q_0\|_\infty.$$

8.3 Policy Iteration

The policy iteration algorithm produces a sequence $(\pi_k : k \in \mathbb{Z}_+)$ of policies. Beginning with π_0, each subsequent iterate is a policy π_{k+1} that is greedy with respect to V_{π_k}. In other words, $F_{\pi_{k+1}} V_{\pi_k} = F V_{\pi_k}$. The policy π_{k+1} can be thought of as one that executes an optimal immediate action assuming that subsequent actions will be selected by π_k. As we will establish, unless π_k is optimal, π_{k+1} represents an improvement, and therefore, the policy iteration algorithm generates a sequence of improving policies.

Our analysis of policy iteration will rely on monotonicity of Bellman operators. We say an operator G is *monotonic* if $Gg \leq Gg'$ for all functions g and g' such that $g \leq g'$. Note that such inequalities denote pointwise relations over elements of the domain of g.

Theorem 14. (monotonicity) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, discount factors $\gamma \in [0, 1)$, and policies π, F and F_π are monotonic.*

Proof. If $V \leq \overline{V}$ then

$$F_\pi V(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\overline{r}_{as} + \gamma \sum_{s' \in \mathcal{S}} P_{ass'} V(s') \right) \leq \sum_{a \in \mathcal{A}} \pi(a|s) \left(\overline{r}_{as} + \gamma \sum_{s' \in \mathcal{S}} P_{ass'} \overline{V}(s') \right) = F_\pi \overline{V}(s),$$

where the inequality follows from the fact that $\pi(a|s) \geq 0$ and $P_{ass'} \geq 0$. Similarly,

$$FV(s) = \max_{a \in \mathcal{A}} \left(\overline{r}_{as} + \gamma \sum_{s' \in \mathcal{S}} P_{ass'} V(s') \right) \leq \max_{a \in \mathcal{A}} \left(\overline{r}_{as} + \gamma \sum_{s' \in \mathcal{S}} P_{ass'} \overline{V}(s') \right) = F\overline{V}(s).$$

Analogous arguments establish monotonicity for the versions of F and F_π that apply to action value functions. □

Policy iteration can be thought of as a process of sequential policy improvement. In particular, the greedy policy π_{k+1} with respect to V_{π_k} should perform at least as well as π_k and in fact be preferable if π_k is not optimal. This is established by the following result.

Theorem 15. (policy improvement) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, discount factors $\gamma \in [0, 1)$, and policies π and π', if $F_{\pi'} V_\pi = F V_\pi$ then $V_{\pi'} \geq V_\pi$. Further, if $V_\pi \neq V_*$ then $V_{\pi'} \neq V_\pi$.*

Proof. Bellman equations (Theorem 7) assert that $V_\pi = F_\pi V_\pi$. It follows that

$$V_\pi = F_\pi V_\pi \leq F V_\pi = F_{\pi'} V_\pi.$$

Monotonicity of $F_{\pi'}$ (Theorem 14) and the fact that $V_\pi \leq F_{\pi'} V_\pi$ imply that $F_{\pi'}^k V_\pi \leq F_{\pi'}^{k+1} V_\pi$ for all $k \in \mathbb{Z}_+$. Combining this with the fact that $\lim_{k \to \infty} F_{\pi'}^k V_\pi = V_{\pi'}$, we have

$$V_\pi \leq F_{\pi'} V_\pi \leq F_{\pi'}^2 V_\pi \leq F_{\pi'}^3 V_\pi \leq \cdots \leq V_{\pi'}.$$

Note that $\lim_{k \to \infty} F_{\pi'}^k V_\pi = V_{\pi'}$ because $F_{\pi'}$ is a contraction mapping with fixed point $V_{\pi'}$. Further, if $V_\pi \neq V_*$ then $F_{\pi'} V_\pi = F V_\pi \neq V_\pi$, which together with the above inequalities implies that $V_{\pi'} \neq V_\pi$. □

This policy improvement theorem implies that policy iterates converge to optimality in the following sense.

Theorem 16. (policy iteration) *For all MDPs $(\mathcal{A}, \mathcal{S}, S_0, P)$, bounded reward functions r, discount factors $\gamma \in [0, 1)$, if π_{k+1} is greedy with respect to V_{π_k} for all $k \in \mathbb{Z}_+$ then, for all $s \in \mathcal{S}$, $\lim_{k \to \infty} V_{\pi_k}(s) = V_*(s)$.*

Proof. By policy improvement (Theorem 15), $(V_{\pi_k} : k \in \mathbb{Z})$ is a nondecreasing sequence of functions that can only converge on V_*, which is a bounded function. This implies pointwise convergence to V_*. That $\lim_{k \to \infty} V_{\pi_k}(s) = V_*(s)$ then follows from the dominated convergence theorem. □

A Probabilistic Framework

This appendix defines the probabilistic framework and the notation we will use in his class. We will model all uncertain quantities as random variables. This is in contrast with much of the reinforcement learning literature, which treats observations and actions as random variables but not the environment. We take the environment to be a random variable as a means for expressing epistemic uncertainty.

A.1 Probability Space

We build on the foundations of probability, based on the Kolmogorov axioms, defining all random quantities with respect to a probability space $(\Omega, \mathbb{F}, \mathbb{P})$. Statements we make have precise meaning within the framework. However, we leave out measure-theoretic formalities for the sake of readability. A mathematically-oriented reader ought to be able to fill in these gaps.

A.2 Events and Random Variables

The probability of an event $F \in \mathbb{F}$ is denoted by $\mathbb{P}(F)$. For any events $F, G \in \mathbb{F}$ with $\mathbb{P}(G) > 0$, the probability of F conditioned on G is denoted by $\mathbb{P}(F|G)$. When a random variable Z takes values in $\mathbb{R}^K$ and has a density with respect to the Lebesgue measure, though $\mathbb{P}(Z = z) = 0$ for all z, conditional probabilities $\mathbb{P}(F|Z = z)$ are well-defined and denoted by $\mathbb{P}(F|Z = z)$. Consider a function $f(z) = \mathbb{P}(F|Z = z)$. Given another random variable Y with the same range as Z, we use the assignment symbol $\leftarrow$ to denote $f(Y)$ by $\mathbb{P}(F|Z \leftarrow Y)$. Note that, in general, $\mathbb{P}(F|Z \leftarrow Y)$ differs from $\mathbb{P}(F|Z = Y)$. The latter conditions on the event that $Z = Y$, while the former represents a change of measure for the variable Z. Note that $\mathbb{P}(F|Z) = \mathbb{P}(F|Z \leftarrow Z)$.

For each possible realization z of a random variable Z, the probability $\mathbb{P}(Z = z)$ that $Z = z$ is a function of z. Note that, if Z takes on values in a continuous space, this probability will typically be zero, but when the space is discrete, the values sum to one. We denote the value of this function evaluated at Z by $\mathbb{P}(Z)$. Note that $\mathbb{P}(Z)$ is itself a random variable because it depends on Z. For random variables Y and Z and possible realizations y and z, the probability $\mathbb{P}(Y = y|Z = z)$ that $Y = y$ conditioned on $Z = z$ is a function of (y, z). Evaluating this function at (Y, Z) yields a random variable, which we denote by $\mathbb{P}(Y|Z)$.

Formally, a random variable X is a measureable function mapping a sample $\omega \in \Omega$ to a realization $X(\omega)$. However, when the intended interpretation is clear from context, we will sometimes say $X \in \mathcal{X}$ to indicate that $\mathcal{X}$ is the range of X. In other words, this serves as shorthand for indicating that realizations of the random variable X take values in $\mathcal{X}$.

A.3 Densities

When a continuous-valued random variable Z has a density, we denote this probability density function by $\mathbf{p}_Z$. In this case, $\mathbb{P}(Z \in \mathcal{Z}) = \int_z \mathbf{p}_Z(z)dz$. A conditional density of X conditioned on Z is analogously written as $\mathbf{p}_{X|Z}(x|z)$. Note that, with this notation, $\mathbf{p}_{X|Z}(x|Z)$ represents a random density, which depends on the realization of Z.

A.4 Equal Distributions

Given two random variables X and Y, we use the notation $X \sim Y$ to indicate that these random variables are equal in distribution; that is, $\mathbb{P}(X \in \cdot) = \mathbb{P}(Y \in \cdot)$ are equal. Here, and more generally, we use $\cdot$ to denote a generic argument of a function. Hence, $P(\cdot) = \mathbb{P}(X \in \cdot)$ is a function, in this case with a set-valued argument, so that $P(\Theta) = \mathbb{P}(X \in \Theta)$.

We will also use the relation $X \sim Y$ more broadly when the objects on the left and right are not necessarily random variables and the interpretation is clear from context. For example, given a probability measure $P(\cdot) = \mathbb{P}(X \in \cdot)$, we have $X \sim P$. Further, we use this notation together with terms for specifying particular distributions, such as $N(0, I)$, exp(1), or bern(0.5). For example, $X \sim N(0, I)$ and $\mathbb{P}(X \in \cdot) \sim N(0, I)$ each indicate that X is distributed standard Gaussian.

A.5 Reinforcement Learning Notation

Particular random variables appear routinely throughout the book. One is the environment $\mathcal{E} = (\mathcal{A}, \mathcal{O}, \rho)$. While $\mathcal{A}$ and $\mathcal{O}$ are deterministic sets that define the agent-environment interface, the observation probability function ρ is a random variable. This randomness reflects the agent designer's epistemic uncertainty about

the environment. The probability measure $\mathbb{P}(\mathcal{E} \in \cdot)$ can be thought of as assigning *prior probabilities* to sets of possible environments. We often consider probabilities $\mathbb{P}(F|\mathcal{E})$ of events F conditioned on the environment $\mathcal{E}$.

For each policy π, random variables $A_0^\pi, O_1^\pi, A_1^\pi, O_2^\pi, \ldots$ denote the sequence of interactions generated by selecting actions according to π. In particular, with $H_t^\pi = (A_0^\pi, O_1^\pi, \ldots, O_t^\pi)$ denoting the history of interactions through time t, we have $\mathbb{P}(A_t^\pi|H_t^\pi) = \pi(A_t^\pi|H_t^\pi)$ and $\mathbb{P}(O_{t+1}^\pi|H_t^\pi, A_t^\pi, \mathcal{E}) = \rho(O_{t+1}^\pi|H_t^\pi, A_t^\pi)$ almost surely. As shorthand, we generally suppress the superscript π and instead indicate the policy through a subscript of $\mathbb{P}$. For example,

$$\mathbb{P}_\pi(A_t|H_t) = \mathbb{P}(A_t^\pi|H_t^\pi) = \pi(A_t^\pi|H_t^\pi),$$

and

$$\mathbb{P}_\pi(O_{t+1}|H_t, A_t, \mathcal{E}) = \mathbb{P}(O_{t+1}^\pi|H_t^\pi, A_t^\pi, \mathcal{E}) = \rho(O_{t+1}^\pi|H_t^\pi, A_t^\pi).$$

When expressing expectations, we use the same subscripting notation as with probabilities. For example, consider a reward function r that maps history, action, and observation to a scalar reward $R_{t+1}^\pi = r(H_t^\pi, A_t^\pi, O_{t+1}^\pi)$. The expected reward is written as $\mathbb{E}[R_{t+1}^\pi] = \mathbb{E}_\pi[R_{t+1}]$, and the expected reward conditioned on the history H_t and action A_t is $\mathbb{E}[R_{t+1}^\pi|H_t^\pi, A_t^\pi] = \mathbb{E}_\pi[R_{t+1}|H_t, A_t]$.