Notes on Reinforcement Learning

Paulo Eduardo Rauber

2014

1 Introduction

Reinforcement learning is the study of agents that act in an environment with the goal of maximizing cumulative reward signals. The agent is not told which actions to take, but discovers which actions yield most rewards by trying them. Its actions also may affect not only the immediate rewards but rewards for the next situations. There are several methods to solve the reinforcement learning problem.

One of the challenges in reinforcement learning is the exploration versus exploitation problem, which is the trade-off between obtaining rewards from perceived safe options against exploring other possibilities which may be advantageous.

A reinforcement learning problem can be divided into four subelements: policy, reward function, value function and a model.

A policy specifies how an agent behaves in a given scenario. It can be seen as a mapping from the perceived environmental state to actions.

A reward function defines the goal in a reinforcement learning problem. It maps each state of the environment to a numerical reward, indicating the intrinsic desirability of that state. An agent's sole objective is to maximize the total reward it receives in the long run. Therefore, the reward function should not be modifiable by the agent.

A value function maps a state to an estimate of the total reward an agent can expect to accumulate over the future starting from that state. Even though a given state may have a small reward, it may be a prerequisite for achieving states with higher rewards.

A model is an optional element of a reinforcement learning system. A model describes the behavior of the environment. For example, it can be used to predict the results of actions (both states and rewards). It is possible to have agents that learn by trial and error, build a model of the environment, and then use the model for planning.

2 Evaluative Feedback

The n-armed bandit problem is very useful to introduce a number of basic learning methods. Suppose an agent is faced with a choice among n different actions. After each action, a numerical reward is sampled based on a stationary probability distribution associated to that particular action. The agent's goal is to maximize the total reward after a number of iterations.

In this case, the value of an action is the expected reward that follows it. If an agent knew the value of every action, it would suffice to choose the action with the highest value.

The agent can maintain an estimate of the value of each action. The action with the highest estimated value is called greedy. Choosing the greedy action (exploitation) may guarantee higher rewards in the short run, while choosing non-greedy actions may lead to better outcomes later.

Let $Q_t(a)$ denote the estimated value of action a at iteration t. We also let $Q^*(a)$ denote the value of a. A natural way to estimate the value of action a is to average the rewards already received. Concretely, if k_a denotes the number of times action a was chosen, we can compute $Q_t(a)$ using

$$Q_t(a) = \frac{r_1 + \ldots + r_{k_a}}{k_a}.$$

By the law of large numbers, when $k_a \to \infty$, $Q_t(a) \to Q^*(a)$.

A very simple alternative to always choosing the action with the highest estimated value is the ϵ-greedy selection rule. This rule chooses the greedy action with probability $1 - \epsilon$ and a random action (uniformly) with probability ϵ. Using this rule, it is important that ties between greedy actions be broken at random. After infinite iterations, every action will be sampled infinite times. Therefore, $Q_t(a)$ converges to $Q^*(a)$ and the chance of choosing the optimal action converges to $1 - \epsilon$.

Another possibility is to weight the probability of choosing an action with its current value estimate. Let $\pi_t(a)$ denote the probability of choosing action a at iteration t and n denote the total number of actions. The following equation is called the softmax action selection rule:

$$\pi_t(a) = \frac{e^{\frac{Q_t(a)}{\tau}}}{\sum_{b=1}^{n} e^{\frac{Q_t(b)}{\tau}}}.$$

In this case, the probability of an action being selected is in proportion to its estimated value. The parameter τ is called temperature. When τ is large, the probabilities are more uniform. When τ is near zero, the rule is greedier. In the case of two actions, this function becomes the sigmoid function.

Fixing an action a and letting k denote the number of times it has been chosen, the equation for estimating action values as an average of observed rewards can be rewritten as

$$Q_{k+1} = Q_k + \frac{1}{k+1}(r_{k+1} - Q_k).$$

If the problem is non-stationary, however, it may be better to estimate the value of an action using

$$Q_{k+1} = Q_k + \alpha(r_{k+1} - Q_k),$$

where $0 < \alpha \leq 1$ is called the step-size parameter. In this case, it can be shown that the value of Q_k is given by

$$Q_k = (1-\alpha)^k Q_0 + \sum_{i=1}^{k} \alpha(1-\alpha)^{k-i} r_i.$$

Since it can be shown that $(1-\alpha)^k + \sum_{i=1}^{k} \alpha(1-\alpha)^k = 1$, this update rule is a weighted average of the initial estimate and the observed rewards. The recently observed rewards have more importance than older rewards. In fact, the weight decreases exponentially for older rewards.

If we denote by $\alpha_k(a)$ the step-size parameter used to process the reward given after the k-th selection of action a, the choice of $\alpha_k(a) = \frac{1}{k}$ leads to the sample average method, while $\alpha_k(a) = \alpha$ leads to the constant step-size method.

There are two conditions that guarantee the convergence of the value estimate to the value of an action a:

$$\sum_{k=1}^{\infty} \alpha_k(a) = \infty,$$

and

$$\sum_{k=1}^{\infty} \alpha_k^2(a) < \infty.$$

Therefore, convergence is guaranteed for the sample average method, but not for the constant step size method. However, constant step sizes are advantageous if the problem is non-stationary, meaning that the value of actions may change during the iterations.

There is a simple trick that can be employed to make an agent explore in early iterations. If the initial value estimates are set optimistically, i.e., significantly higher than the value of the optimal action, even a greedy agent will explore considerably in the initial iterations until the value estimates are lowered.

Suppose that an observed variable (the state of the system) contained information about the next reward for a given action. It would be possible to achieve better results than using the strategies described in this chapter. This motivates the full reinforcement learning problem.

3 The Reinforcement Learning Problem

A reinforcement learning agent interacts with the environment at each of a sequence of discrete time steps $t = 0, 1, 3, \ldots$. At each time step t, the agent receives some representation of the environment's state $s_t \in \mathcal{S}$, where $\mathcal{S}$ is the set of possible states. The agent then selects an action $a_t \in \mathcal{A}(s_t)$, where $\mathcal{A}(s_t)$ is the set of available actions in state s_t. One time step later, the agent receives a reward r_{t+1} and a new state s_{t+1}. The agent implements a policy $\pi(s, a)$: the probability that $A_t = a$ given that $S_t = s$.

Modeling a problem as a reinforcement learning problem is challenging. Particularly, the boundary between agent and environment is sometimes not clear. A good strategy may be to abstract as much of the problem as possible and let the agent focus on learning that which is hard to explicitly program. For instance, the action primitives for a robot may be pre-programmed tasks instead of mechanical control. Anything that cannot be controlled directly by the agent should be considered part of the environment, including rewards.

At each time step, the agent receives a reward $r_t \in \mathbb{R}$. The objective of the agent is to maximize the total amount of reward it receives in the long run. The choice of a reward function is critical in a reinforcement learning problem. It is particularly important not to impart previous knowledge into the reward function about how to solve a problem, but only what the agent should achieve. Previous knowledge should be imparted into state value estimates, which will be discussed later.

Let $r_{t+1}, r_{t+2}, r_{t+3}, \ldots$ denote the sequence of rewards received after time step t. A function from a sequence of rewards to a real number is called a return function, and its value for a specific sequence is called return. A reinforcement learning agent aims to maximize the expected return.

In a simple formulation, the return could be given by $u_t = r_{t+1} + r_{t+2} + \ldots + r_T$, where T is the final time step. When T is finite, we call the problem episodic. An episode ends when the agent reaches a terminal state. After that, the time is reset and the environment samples from a distribution of initial states.

However, when $T = \infty$, the simple formulation of u_t may be problematic, since the return could be infinite. This is called a continuous problem. In this case, it is better to consider the discounted return given by

$$u_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{T} \gamma^k r_{t+k+1}, \tag{1}$$

where $0 \leq \gamma < 1$ is called the discount rate. The discount rate determines the present value of future rewards. A reward received k time steps into the future is only worth γ^{k-1} times what it would be worth if it were received immediately. In this formulation, as long as the sequence $\{r_k\}$ is bounded, the (discounted) return is finite. The choice of γ is important to determine how future-oriented is the agent.

In both continuous and episodic problems, the expected return for each state can be estimated by averaging the returns observed in several instances of the problem (which may be approximations in the continuous case).

The choice of a reward function is dependent on the definition of return. For instance, consider an agent which tries to escape a maze. In the episodic case, the reward could be -1 for every move the agent makes until it reaches the exit. In this case, after T time steps, the reward for the agent would be $-T$, and maximizing the expected return would imply in finding the exit early. In the discounted case, the reward could be 1 whenever the agent found the final step and 0 otherwise. In this case, the expected return would be γ^k, and minimizing k would be implied. However, if the same formulation were used in the episodic, undiscounted case, the agent would have no incentive to find the exit early.

An episodic problem can be modeled as a continuous problem by simply creating a state which yields no rewards and transitions only to itself. In fact, if we let $\gamma = 1$, we can use the same formulation given in Eq. 1 for continuous tasks, as long as $T = \infty$ and $\gamma = 1$ are never both true.

In a reinforcement learning problem the state represents whatever information is available to the agent. States may be immediate sensations or highly processed versions of data and are often represented by vectors. States don't need to represent the environment perfectly or even all that could be useful in making marginally better decisions.

However, a state should respect the Markov property as well as possible. A state that summarizes all that is relevant for decision making that was observed in past states is said to have the Markov property. More precisely, given the sequence of states, action and rewards $s_t, a_t, r_t, \ldots r_1, s_0, a_0$ (called history) observed by an agent, the state signal is said to have the Markov property if

$$P(S_{t+1} = s', R_{t+1} = r' \mid s_t, a_t, r_t, \ldots, r_1, s_0, a_0) = P(S_{t+1} = s', R_{t+1} = r' \mid s_t, a_t),$$

for all s', r' and histories. We call the probability distribution function on the right side the one-step dynamics of the reinforcement learning problem. It can be shown that the one-step dynamics can be used to derive the probability of all future states and expected rewards only from the initial state.

A reinforcement learning problem that satisfies the Markov property is called a Markov decision process, often called *MDP*. In particular, if the state and action sets are finite, the problem is called a finite MDP.

A Markov decision process is defined by its state and action sets and by the one-step dynamics of the environment. Given any state s and action a, the probability $\mathcal{P}^a_{ss'}$ of each next state s' is given by

$$\mathcal{P}^a_{ss'} = P(S_{t+1} = s' \mid S_t = s, A_t = a) = \sum_{r'} P(S_{t+1} = s', R_{t+1} = r' \mid S_t = s, A_t = a).$$

Given any state s, action a and next state s', the expected value of the next reward is given by

$$\mathcal{R}^a_{ss'} = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s'] = \frac{1}{\mathcal{P}^a_{ss'}} \sum_{r'} r' P(S_{t+1} = s', R_{t+1} = r' \mid A_t = a, S_t = s).$$

The value $V^\pi(s)$ of a state s is defined as the expected return of starting in state s and following the policy π. The policy assigns a probability $\pi(s, a)$ to taking an action a when in a state s. More precisely,

$$V^\pi(s) = \mathbb{E}_\pi[U_t \mid S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right]. \tag{2}$$

Similarly, the value of taking an action a when in state s and afterwards following the policy π is denoted by $Q^\pi(s, a)$ and defined as

$$Q^\pi(s, a) = \mathbb{E}_\pi[U_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right].$$

The function V^π is called state value function for policy π and the function Q^π the action value function for policy π. These two functions can be estimated, for a given policy, by observing several instances of the reinforcement learning problem.

The state value function for a policy π is also given by

$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^\pi(s')]. \tag{3}$$

In order to prove this result, we will employ the notation described in the machine learning notes by the same author. Firstly, note that Equation 2 can be rewritten as

$$V^\pi(s) = \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s) \sum_{k=0}^{T} \gamma^k r_{t+k+1},$$

where the dependency on the policy π is left implicitly for simplicity of notation. By marginalization,

$$V^\pi(s) = \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} \left[\sum_{a_t} \sum_{s_{t+1}} p(r_{t+1:T+t+1}, a_t, s_{t+1} \mid S_t = s) \right] \sum_{k=0}^{T} \gamma^k r_{t+k+1}.$$

By the chain rule of probability,

$$V^\pi(s) = \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} \sum_{a_t} \sum_{s_{t+1}} p(a_t \mid S_t = s) p(s_{t+1} \mid S_t = s, a_t) p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \sum_{k=0}^{T} \gamma^k r_{t+k+1}.$$

By the distributive property and reordering the three outermost summations,

$$V^\pi(s) = \sum_{a_t} p(a_t \mid S_t = s) \sum_{s_{t+1}} p(s_{t+1} \mid S_t = s, a_t) \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \sum_{k=0}^{T} \gamma^k r_{t+k+1}. \tag{4}$$

Let E denote the limit in the previous equation, such that

$$E = \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \sum_{k=0}^{T} \gamma^k r_{t+k+1}.$$

By isolating the first term in the innermost summation,

$$E = \lim_{T \to \infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \left[r_{t+1} + \sum_{k=1}^{T} \gamma^k r_{t+k+1} \right].$$

By the linearity of expectation, $E = E_1 + E_2$, where

$$E_1 = \lim_{T\to\infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) r_{t+1} = \mathbb{E}_\pi \left[R_{t+1} \mid S_t = s, a_t, s_{t+1} \right] = \mathcal{R}^{a_t}_{ss_{t+1}},$$

and

$$E_2 = \lim_{T\to\infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \sum_{k=1}^{T} \gamma^k r_{t+k+1}.$$

By changing the indices in the innermost summation,

$$E_2 = \lim_{T\to\infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \sum_{k=0}^{T-1} \gamma^{k+1} r_{t+k+2}.$$

By moving a constant factor of γ outside of the innermost summation,

$$E_2 = \gamma \lim_{T\to\infty} \sum_{r_{t+1:T+t+1}} p(r_{t+1:T+t+1} \mid S_t = s, a_t, s_{t+1}) \sum_{k=0}^{T-1} \gamma^{k} r_{t+k+2}.$$

Because $R_{t+2:T+t+1} \perp\!\!\!\perp S_t, A_t, R_{t+1} \mid S_{t+1}$ due to the Markov property,

$$E_2 = \gamma \lim_{T\to\infty} \mathbb{E}_\pi \left[\sum_{k=0}^{T-1} \gamma^k R_{t+k+2} \mid s_{t+1} \right] = \gamma V^\pi(s_{t+1}).$$

Returning to Equation 4,

$$V^\pi(s) = \sum_{a_t} p(a_t \mid S_t = s) \sum_{s_{t+1}} p(s_{t+1} \mid S_t = s, a_t) \left[\mathcal{R}^{a_t}_{ss_{t+1}} + \gamma V^\pi(s_{t+1}) \right].$$

By making the dependency on π explicit and renaming variables,

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s') \right],$$

as we wanted to show.

Using a similar argument, it can also be shown that the action value function is given by

$$Q^\pi(s,a) = \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma \sum_{a'} \pi(s',a') Q^\pi(s',a')].$$

These two equations are called the Bellman equations and can be interpreted intuitively. Note that, given a policy and the one-step dynamics of the problem, each of the equations defines a system of $|\mathcal{S}|$ linear equations and variables. For any given policy, it is possible to find the *unique* expected return for any given state and action at a state.

It is possible to prove, in the case of a continuous discounted return problem, that adding a constant to all the rewards only adds another constant to the value of all states. However, in the case of a episodic task, this is not generally true, due to the variable number of time steps that may occur in each episode.

There are two alternative ways of writing the state and action value functions in terms of each other:

$$V^\pi(s) = \sum_a \pi(s,a) Q^\pi(s,a),$$

and

$$Q^\pi(s,a) = \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V(s')].$$

Solving a reinforcement learning problem consists of finding a policy that has high expected return. Let $\pi \geq \pi'$ if and only if $V^\pi(s) \geq V^{\pi'}(s)$ for all $s \in \mathcal{S}$. The optimal policy π^* is such that $\pi^* \geq \pi$ for any policy π. It is possible to show that this policy exists but is not necessarily unique. We define the optimal state value function V^* and the optimal action value function Q^* as

$$V^*(s) = \max_{\pi} V^{\pi}(s) = \max_{a} \sum_{s'} \mathcal{P}^{a}_{ss'}[\mathcal{R}^{a}_{ss'} + \gamma V^*(s')],$$

and

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) = \sum_{s'} \mathcal{P}^{a}_{ss'}[\mathcal{R}^{a}_{ss'} + \gamma \max_{a'} Q^*(s',a')],$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$. These are called the Bellman optimality equations. Each equation gives a system of nonlinear equations that can be used to find $V^*(s)$ or $Q^*(s,a)$ for any state s and action a. In practice, however, solving nonlinear systems of equations in problems with many states may be unfeasible.

An optimal policy π^* can be found given V^* or Q^*. In the case of V^*, it suffices to choose one of the actions a that maximizes the right hand side of its Bellman optimality equation with uniform probability. In the case of Q^*, it suffices to choose the a such that $Q^*(s,a)$ is maximal.

A large amount of memory may be required to build up approximations of value functions, policies and models. Therefore, in many practical problems, these functions must be approximated by a parameterized function.

4 Dynamic Programming

The term dynamic programming refers to a collection of reinforcement learning algorithms that can be used to compute optimal policies given a perfect model of the environment (one-step dynamics). It is assumed that the set of states and actions are finite.

Policy evaluation is the name given to computing the state value function V^π given an arbitrary policy π. As already stated, it is possible to solve the system of linear equations with variables $V^\pi(s)$ (Eq. 3). However, it is also possible to create a sequence of approximations to V^π by the following rule:

$$V_{k+1}(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V_k(s')],$$

for all $s \in \mathcal{S}$. The initial value for each state can be arbitrary. This sequence is guaranteed to converge to $V^\pi(s)$ for all $s \in \mathcal{S}$, as we show in Appendix 4.1. This method is called iterative policy evaluation.

In a given iteration of policy evaluation, instead of computing the new estimate for each state using the estimate from the previous iteration, it is also valid to consider the latest estimate available for each state. This variation is classified as an in-place algorithm (Alg. 1).

Algorithm 1 Iterative policy evaluation (in-place)

```
Input: policy $\pi$, one-step dynamics functions $\mathcal{P}$ and $\mathcal{R}$, discount factor $\gamma$, tolerance $\theta$.
Output: Value function $V = V^\pi$ when $\theta \to 0$.
for each $s \in \mathcal{S}$ do
    $V(s) \leftarrow 0$
end for
repeat
    $\Delta \leftarrow 0$
    for each $s \in \mathcal{S}$ do
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V(s')]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end for
until $\Delta < \theta$
```

A deterministic policy π is one such that, for all $s \in \mathcal{S}$, $\pi(s,a) = 1$ for some a and $\pi(s,b) = 0$ for all $b \neq a$. Thus, in the case of deterministic policies, it is simpler to consider π as a function from states to actions.

Let π and π' be any pair of deterministic policies such that, for all $s \in \mathcal{S}$, $Q^\pi(s, \pi'(s)) \geq V^\pi(s)$. The policy improvement theorem guarantees that $V^{\pi'}(s) \geq V^\pi(s)$ for all $s \in \mathcal{S}$. This gives a natural way of improving a given policy π to a better policy π' by the following rule:

$$\pi'(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')].$$

A sequence of evaluations and improvements can be applied to an arbitrary initial policy. Once a policy can no longer be improved, it is guaranteed to be optimal by the Bellman optimality equations. This technique for finding an optimal policy is called policy iteration.

A very simple and efficient alternative to policy iteration is called value iteration. This technique successively improves the estimates for the value of each state $s \in \mathcal{S}$, beginning with an arbitrary value $V_0(s)$, by the following rule:

$$V_{k+1}(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V_k(s')].$$

It can be shown that the sequence $\{V_k\}$ converges to V^*. Algorithm 2 describes how an optimal policy can be obtained by value iteration.

Asynchronous algorithms can also be used to update the estimates for V.

There are two main disadvantages of both policy iteration and value iteration: they required expensive sweeps over the state space, which may be prohibitively large, and a complete model of the environment.

Algorithm 2 Value iteration

```
Input: one-step dynamics functions $\mathcal{P}$ and $\mathcal{R}$, discount factor $\gamma$, tolerance $\theta$.
Output: optimal deterministic policy $\pi$ when $\theta \to 0$.
for each $s \in \mathcal{S}$ do
    $V(s) \leftarrow 0$
end for
repeat
    $\Delta \leftarrow 0$
    for each $s \in \mathcal{S}$ do
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V(s')]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end for
until $\Delta < \theta$
for each $s \in \mathcal{S}$ do
    $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V(s')]$
end for
```

4.1 Appendix: convergence of iterative policy evaluation

This appendix presents a proof of the following theorem.

Theorem 4.1 (Convergence of iterative policy evaluation)**.** Consider the Bellman operator $T^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ for the policy π. Given an arbitrary $\mathbf{v}_0 \in \mathbb{R}^{|\mathcal{S}|}$, consider also the sequence $(\mathbf{v}_n)_{n \geq 0}$ where $\mathbf{v}_{k+1} = T^\pi(\mathbf{v}_k)$. Finally, consider the vector $\mathbf{v}^\pi \in \mathbb{R}^{|\mathcal{S}|}$ such that $v^\pi_s = V^\pi(s)$, for any $s \in \mathcal{S}$. For any $n \geq 0$,

$$\mathbf{v}_n \underset{||\cdot||_\infty}{\to} \mathbf{v}^\pi,$$

$$\mathbf{v}^\pi = T^\pi(\mathbf{v}^\pi),$$

$$||\mathbf{v}_n - \mathbf{v}^\pi||_\infty \leq \gamma^n ||\mathbf{v}_0 - \mathbf{v}^\pi||_\infty.$$

Before presenting the proof, we first introduce relevant definitions, lemmas, and theorems.

Definition 4.1 (Norm)**.** Consider a vector space Z over a field F. A function $||\cdot|| : Z \to [0, \infty)$ is a norm if

$$||\mathbf{u} + \mathbf{v}|| \leq ||\mathbf{u}|| + ||\mathbf{v}||,$$

$$||a\mathbf{v}|| = |a|||\mathbf{v}||,$$

$$||\mathbf{v}|| = 0 \implies \mathbf{v} = \mathbf{0},$$

for all $\mathbf{u}$, $\mathbf{v} \in Z$ and $a \in F$.

Definition 4.2 (Euclidean norm)**.** The Euclidean norm $||\cdot||_2 : \mathbb{R}^d \to [0, \infty)$ is given by

$$||\mathbf{v}||_2 = \sqrt{\sum_i v_i^2}.$$

Definition 4.3 (Maximum norm)**.** The maximum norm $||\cdot||_\infty : \mathbb{R}^d \to [0, \infty)$ is given by

$$||\mathbf{v}||_\infty = \max_i |v_i|.$$

Definition 4.4 (Convergence of a sequence)**.** A sequence $(\mathbf{v}_n)_{n\geq 0} = \mathbf{v}_0, \mathbf{v}_1, \ldots$ is said to converge to a vector $\mathbf{v}$ in the norm $||\cdot||$, denoted $\mathbf{v}_n \underset{||\cdot||}{\to} \mathbf{v}$, if

$$\lim_{n\to\infty} ||\mathbf{v}_n - \mathbf{v}|| = 0.$$

In other words, if $\mathbf{v}_n \underset{||\cdot||}{\to} \mathbf{v}$, then for every $\epsilon > 0$ there is an N such that for every $n \geq N$, we have $||\mathbf{v}_n - \mathbf{v}|| < \epsilon$.

Definition 4.5 (Bellman operator)**.** Consider a reinforcement learning task with states $\mathcal{S} = \{1, 2, \dots, |\mathcal{S}|\}$, actions $\mathcal{A} = \{1, 2, \dots, |\mathcal{A}|\}$, and discount factor $\gamma < 1$. For any vector $\mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$ and state $s \in \mathcal{S}$, the Bellman operator $T^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ for the policy π is given by

$$T^\pi(\mathbf{v})_s = \sum_a \pi(s, a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma v_{s'} \right].$$

Definition 4.6 (Cauchy sequence)**.** Consider a normed vector space $(Z, ||\cdot||)$. A sequence $(\mathbf{v}_n)_{n \geq 0} = \mathbf{v}_0, \mathbf{v}_1, \dots$ of vectors in this space is Cauchy if

$$\lim_{n \to \infty} \sup_{m \geq n} ||\mathbf{v}_n - \mathbf{v}_m|| = 0.$$

In other words, if a sequence $(\mathbf{v}_n)_{n \geq 0}$ is Cauchy, then for every $\epsilon > 0$ there is an N such that for every $n \geq N$, we have $\sup_{m \geq n} ||\mathbf{v}_n - \mathbf{v}_m|| < \epsilon$.

Definition 4.7 (Banach space)**.** A Banach space is a normed vector space $(Z, ||\cdot||)$ where if $(\mathbf{v}_n)_{n \geq 0}$ is a Cauchy sequence then $\mathbf{v}_n \underset{||\cdot||}{\to} \mathbf{v}$ for some vector $\mathbf{v}$.

For any d, both $(\mathbb{R}^d, ||\cdot||_2)$ and $(\mathbb{R}^d, ||\cdot||_\infty)$ are Banach spaces, although we omit the corresponding proofs.

Definition 4.8 (L-contraction)**.** Consider a normed vector space $(Z, ||\cdot||)$ and a function $T : Z \to Z$. The function T is L-Lipschitz if

$$||T(\mathbf{u}) - T(\mathbf{v})|| \leq L||\mathbf{u} - \mathbf{v}||,$$

for all $\mathbf{u}, \mathbf{v} \in Z$. If $L < 1$, then T is also an L-contraction.

Lemma 4.1. Consider a normed vector space $(Z, ||\cdot||)$, an L-Lipschitz function $T : Z \to Z$, and a sequence $(\mathbf{v}_n)_{n \geq 0} = \mathbf{v}_0, \mathbf{v}_1, \dots$ of vectors in this space. If $\mathbf{v}_n \underset{||\cdot||}{\to} \mathbf{v}$, then $T(\mathbf{v}_n) \underset{||\cdot||}{\to} T(\mathbf{v})$.

Proof. For any $n \geq 0$, by the definition of an L-Lipschitz function,

$$0 \leq ||T(\mathbf{v}_n) - T(\mathbf{v})|| \leq L||\mathbf{v}_n - \mathbf{v}||.$$

Since $\mathbf{v}_n \underset{||\cdot||}{\to} \mathbf{v}$,

$$\lim_{n \to \infty} L||\mathbf{v}_n - \mathbf{v}|| = L \lim_{n \to \infty} ||\mathbf{v}_n - \mathbf{v}|| = 0.$$

By the squeeze theorem,

$$\lim_{n \to \infty} ||T(\mathbf{v}_n) - T(\mathbf{v})|| = 0.$$

□

The following theorem is central in our proof of Theorem 4.1.

Theorem 4.2 (Banach's fixed point theorem)**.** If $(Z, ||\cdot||)$ is a Banach space and $T : Z \to Z$ is an L-contraction, then T has a unique fixed point $\mathbf{v}$. Furthermore, for any $\mathbf{v}_0 \in Z$, let $\mathbf{v}_{n+1} = T(\mathbf{v}_n)$. For any $n \geq 0$,

$$\mathbf{v}_n \underset{||\cdot||}{\to} \mathbf{v},$$
$$||\mathbf{v}_n - \mathbf{v}|| \leq L^n ||\mathbf{v}_0 - \mathbf{v}||.$$

Proof. We first show that the sequence $(\mathbf{v}_n)_{n \geq 0}$ is Cauchy, which guarantees that $\mathbf{v}_n \underset{||\cdot||}{\to} \mathbf{v}$ for some vector $\mathbf{v}$.

As a first step, we show that $||\mathbf{v}_{n+k} - \mathbf{v}_n|| \leq L^n ||\mathbf{v}_k - \mathbf{v}_0||$ for any $n, k \geq 0$. The case $n = 0$ is trivial. Suppose that the inductive hypothesis is true for some n, and consider the case $n + 1$:

$$\begin{aligned}
||\mathbf{v}_{n+k+1} - \mathbf{v}_{n+1}|| &= ||T(\mathbf{v}_{n+k}) - T(\mathbf{v}_n)|| && \text{(definition of the sequence)} \\
&\leq L||\mathbf{v}_{n+k} - \mathbf{v}_n|| && \text{(definition of } L\text{-contraction)} \\
&\leq L^{n+1} ||\mathbf{v}_k - \mathbf{v}_0||, && \text{(inductive hypothesis)}
\end{aligned}$$

as we wanted to show.

By introducing zeros, note that $||\mathbf{v}_k - \mathbf{v}_0|| = ||\mathbf{v}_k + (-\mathbf{v}_{k-1} + \mathbf{v}_{k-1}) + \ldots + (-v_1 + v_1) - \mathbf{v}_0||$, for any $k \geq 1$. Therefore,

$$\begin{aligned}
||\mathbf{v}_k - \mathbf{v}_0|| &= \left\| \sum_{i=1}^{k} \mathbf{v}_i - \mathbf{v}_{i-1} \right\| && \text{(reorganizing terms)} \\
&\leq \sum_{i=1}^{k} ||\mathbf{v}_i - \mathbf{v}_{i-1}|| && \text{(triangle inequality)} \\
&\leq \sum_{i=1}^{k} L^{i-1} ||\mathbf{v}_1 - \mathbf{v}_0|| && (n = i-1 \text{ and } k = 1 \text{ using the earlier result}) \\
&\leq \frac{||\mathbf{v}_1 - \mathbf{v}_0||}{1-L}. && \left(\lim_{n\to\infty} \sum_{i=0}^{n} L^n = \frac{1}{1-L} \right).
\end{aligned}$$

We are now close to showing that $(\mathbf{v}_n)_{n\geq 0}$ is Cauchy. For any $n, k \geq 0$, combining the previous two results,

$$0 \leq ||\mathbf{v}_{n+k} - \mathbf{v}_n|| \leq L^n ||\mathbf{v}_k - \mathbf{v}_0|| \leq L^n \frac{||\mathbf{v}_1 - \mathbf{v}_0||}{1-L}.$$

Therefore, for any fixed $n \geq 0$,

$$0 \leq \sup_{k\geq 0} ||\mathbf{v}_{n+k} - \mathbf{v}_n|| \leq \sup_{k\geq 0} L^n \frac{||\mathbf{v}_1 - \mathbf{v}_0||}{1-L} = L^n \frac{||\mathbf{v}_1 - \mathbf{v}_0||}{1-L}.$$

Because $0 \leq L < 1$,

$$\lim_{n\to\infty} L^n \frac{||\mathbf{v}_1 - \mathbf{v}_0||}{1-L} = \frac{||\mathbf{v}_1 - \mathbf{v}_0||}{1-L} \lim_{n\to\infty} L^n = 0.$$

Therefore, by the squeeze theorem,

$$\lim_{n\to\infty} \sup_{k\geq 0} ||\mathbf{v}_{n+k} - \mathbf{v}_n|| = 0,$$

which completes the proof that $(\mathbf{v}_n)_{n\geq 0}$ is Cauchy. Let $\mathbf{v}$ denote the vector such that $\mathbf{v}_n \underset{||\cdot||}{\rightarrow} \mathbf{v}$.

Our next step is to show that $\mathbf{v}$ is a fixed point of T. For any n,

$$\begin{aligned}
0 \leq ||T(\mathbf{v}) - \mathbf{v}|| &= ||T(\mathbf{v}) + (-T(\mathbf{v}_n) + T(\mathbf{v}_n)) - \mathbf{v}|| && \text{(introducing zeros)} \\
&\leq ||T(\mathbf{v}) - T(\mathbf{v}_n)|| + ||T(\mathbf{v}_n) - \mathbf{v}|| && \text{(triangle inequality)} \\
&\leq L||\mathbf{v} - \mathbf{v}_n|| + ||\mathbf{v}_{n+1} - \mathbf{v}|| && \text{(definition of the sequence and } L\text{-contraction)}
\end{aligned}$$

Because $\mathbf{v}_n \underset{||\cdot||}{\rightarrow} \mathbf{v}$,

$$\lim_{n\to\infty} L||\mathbf{v} - \mathbf{v}_n|| + ||\mathbf{v}_{n+1} - \mathbf{v}|| = 0.$$

Therefore, by the squeeze theorem

$$||T(\mathbf{v}) - \mathbf{v}|| = \lim_{n\to\infty} ||T(\mathbf{v}) - \mathbf{v}|| = 0.$$

By the definition of a norm, $T(\mathbf{v}) - \mathbf{v} = \mathbf{0}$, which implies $T(\mathbf{v}) = \mathbf{v}$, completing the proof.

Our next step is to show that the fixed point of T is unique. Suppose that $T(\mathbf{u}) = \mathbf{u}$ and $T(\mathbf{v}) = \mathbf{v}$ for some vectors $\mathbf{u}$ and $\mathbf{v}$. In that case,

$$||\mathbf{u} - \mathbf{v}|| = ||T(\mathbf{u}) - T(\mathbf{v})|| \leq L||\mathbf{u} - \mathbf{v}||.$$

If we suppose that $||\mathbf{u} - \mathbf{v}|| > 0$, dividing the inequation by $||\mathbf{u} - \mathbf{v}||$ leads to the conclusion that $L \geq 1$. However, T is an L-contraction, contradicting our supposition. Therefore, $||\mathbf{u} - \mathbf{v}|| \leq 0$, which implies that $\mathbf{u} = \mathbf{v}$.

Our last step is to show that $||\mathbf{v}_n - \mathbf{v}|| \leq L^n ||\mathbf{v}_0 - \mathbf{v}||$, for any n. The case $n = 0$ is trivial. Suppose that the inductive hypothesis is true for some n, and consider the case $n + 1$:

$$\begin{aligned} ||\mathbf{v}_{n+1} - \mathbf{v}|| &= ||T(\mathbf{v}_n) - T(\mathbf{v})|| && \text{(definition of the sequence and fixed point)} \\ &\leq L||\mathbf{v}_n - \mathbf{v}|| \leq L^{n+1}||\mathbf{v}_0 - \mathbf{v}||, && \text{(definition of } L\text{-contraction and inductive hypothesis)} \end{aligned}$$

as we wanted to show.

□

We now present the proof of Theorem 4.1.

Theorem 4.1 (Convergence of iterative policy evaluation)**.** Consider the Bellman operator $T^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ for the policy π. Given an arbitrary $\mathbf{v}_0 \in \mathbb{R}^{|\mathcal{S}|}$, consider also the sequence $(\mathbf{v}_n)_{n \geq 0}$ where $\mathbf{v}_{k+1} = T^\pi(\mathbf{v}_k)$. Finally, consider the vector $\mathbf{v}^\pi \in \mathbb{R}^{|\mathcal{S}|}$ such that $v^\pi_s = V^\pi(s)$, for any $s \in \mathcal{S}$. For any $n \geq 0$,

$$\begin{gathered} \mathbf{v}_n \underset{||\cdot||_\infty}{\to} \mathbf{v}^\pi, \\ \mathbf{v}^\pi = T^\pi(\mathbf{v}^\pi), \\ ||\mathbf{v}_n - \mathbf{v}^\pi||_\infty \leq \gamma^n ||\mathbf{v}_0 - \mathbf{v}^\pi||_\infty. \end{gathered}$$

Proof. As a first step, we show that $\mathbf{v}^\pi$ is a fixed point of T^π. For any $s \in \mathcal{S}$, by the definition of T^π and Eq. 3,

$$T^\pi(\mathbf{v}^\pi)_s = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma v^\pi_{s'}\right] = v^\pi_s.$$

Our next step is to show that the Bellman operator $T^\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ is a γ-contraction. Because $(\mathbb{R}^{|\mathcal{S}|}, ||\cdot||_\infty)$ is a Banach space and $\mathbf{v}^\pi$ is a fixed point of T^π, the desired results follow from Banach's fixed point theorem.

Note that, for any two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$ and every state $s \in \mathcal{S}$,

$$\begin{aligned} T^\pi(\mathbf{u})_s - T^\pi(\mathbf{v})_s &= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma u_{s'}\right] - \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma v_{s'}\right] \\ &= \sum_a \sum_{s'} \pi(s,a)\mathcal{P}^a_{ss'}\mathcal{R}^a_{ss'} + \sum_a \sum_{s'} \pi(s,a)\mathcal{P}^a_{ss'}\gamma u_{s'} - \sum_a \sum_{s'} \pi(s,a)\mathcal{P}^a_{ss'}\mathcal{R}^a_{ss'} - \sum_a \sum_{s'} \pi(s,a)\mathcal{P}^a_{ss'}\gamma v_{s'} \\ &= \sum_a \sum_{s'} \pi(s,a)\mathcal{P}^a_{ss'}\gamma u_{s'} - \sum_a \sum_{s'} \pi(s,a)\mathcal{P}^a_{ss'}\gamma v_{s'} \\ &= \gamma \sum_a \sum_{s'} \pi(s,a)\mathcal{P}^a_{ss'}[u_{s'} - v_{s'}]. \end{aligned}$$

By the definition of maximum norm, for any two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$,

$$\begin{aligned} ||T^\pi(\mathbf{u}) - T^\pi(\mathbf{v})||_\infty &= \gamma \max_s \left| \sum_a \sum_{s'} \pi(s,a)\mathcal{P}^a_{ss'}[u_{s'} - v_{s'}] \right| && \text{(definition of maximum norm)} \\ &\leq \gamma \max_s \sum_a \sum_{s'} \left| \pi(s,a)\mathcal{P}^a_{ss'}[u_{s'} - v_{s'}] \right| && \text{(triangle inequality)} \\ &= \gamma \max_s \sum_a \sum_{s'} \pi(s,a)\mathcal{P}^a_{ss'}|u_{s'} - v_{s'}| && \text{(multiplicativity)} \\ &\leq \gamma \max_s \sum_a \sum_{s'} \pi(s,a)\mathcal{P}^a_{ss'}||\mathbf{u} - \mathbf{v}||_\infty && \text{(definition of maximum norm)} \\ &= \gamma||\mathbf{u} - \mathbf{v}||_\infty \max_s \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} && \text{(distributivity)} \\ &= \gamma||\mathbf{u} - \mathbf{v}||_\infty && \text{(unit measure).} \end{aligned}$$

□

5 Monte Carlo Methods

Monte Carlo methods can be used to solve the reinforcement learning problem without a model (one-step dynamics) for the environment. The agent learns by averaging observed returns. Even if it is possible to construct a model of the environment, it is often easier to let the agent learn in this way.

These methods also work by successive policy evaluation and improvement. The most simple way to estimate the value of a state under π is to average the observed returns after a number of episodes in which the state appears. If only the first occurrence of a state s in an episode is considered to improve the estimate for $V^{\pi}(s)$, then the method is called first-visit. Otherwise, when every occurrence of a state s is considered, the method is called every-visit. In both cases, when the number of visits to s tends to infinity, this kind of estimate tends to $V^{\pi}(s)$.

Since a model is not available for Monte Carlo methods, it is useful to estimate action values instead of state values. Evaluating a policy π by a Monte Carlo method consists on experiencing several episodes and averaging the returns that follow every possible state action pair (s, a) to obtain an estimate for $Q^{\pi}(s, a)$.

As in the previous section, we are interested in improving a policy π. In this case, we do this episode-by-episode. However, unlike in the previous section, the policies must be ϵ-soft, since we need to guarantee that the method improves its estimate for the action values. Algorithm 3 illustrates a Monte Carlo based algorithm to solve the reinforcement learning problem. It is assumed that there is no discounting, since the length of each episode *should* be finite.

Algorithm 3 Monte Carlo control algorithm

```
Input: set of states $\mathcal{S}$, number of episodes $N$, probability of choosing random action $\epsilon$.
Output: deterministic policy $\pi$, optimal when $N \to \infty$.
for each $s \in \mathcal{S}$ do
    for each action $a \in \mathcal{A}(s)$ do
        $Q(s, a) \leftarrow 0$
        $n(s, a) \leftarrow 0$
    end for
end for
for each $i$ in $\{1, \ldots, N\}$ do
    Experience a new episode $e$ following an $\epsilon$-greedy policy based on $Q$.
    for each state-action pair $(s, a)$ in the episode $e$ do
        $u \leftarrow$ return following $(s, a)$ in the episode $e$.
        $n(s, a) \leftarrow n(s, a) + 1$
        $Q(s, a) \leftarrow Q(s, a) + \frac{1}{n(s,a)}[u - Q(s, a)]$
    end for
end for
for each state $s \in \mathcal{S}$ do
    $\pi(s) \leftarrow \arg\max_a Q(s, a)$
end for
```

A method to solve the reinforcement learning problem that uses the value of other states to compute the value of a state s is said to bootstrap. Since Monte Carlo methods do not bootstrap, in contrast with dynamic programming, it is possible to evaluate the value of each state separately, which can be necessary in some situations. It is also possible to evaluate a policy while following another, which is described in detail in [1].

6 Temporal-Difference Learning

Temporal-difference learning methods are also capable of learning from experience without a model of the environment. However, differently from Monte Carlo methods, they bootstrap. Intuitively, this allows them to extract more information from their experiences.

Instead of waiting for the end of an episode to update their state or action values, a TD agent updates at every time step. The simplest TD method is called $TD(0)$, and updates the state value estimates using the following rule:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)], \tag{5}$$

where α is the learning rate and γ is a discount parameter. This rule updates the value of a state based on the reward after one time-step and the estimated value of the next state.

Sarsa is an on-policy method to solve the control problem: finding an optimal policy by interacting with the environment. The idea is to adapt Equation 5 to estimate action values, which involves using the tuple $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$, which gives the method's name. As with other control algorithms, the starting point is an arbitrary ϵ-soft policy, which is used to estimate action values and improve the policy, which is done after each time step. Sarsa is exemplified by algorithm 4, which uses ϵ-greedy policies.

Algorithm 4 Sarsa control algorithm

```
Input: set of states $\mathcal{S}$, number of episodes $N$, learning rate $\alpha$, probability of choosing random action $\epsilon$, discount factor $\gamma$.
Output: deterministic policy $\pi$, optimal when $N \to \infty$ and $\alpha$ decays appropriately.
for each $s \in \mathcal{S}$ do
    for each action $a \in \mathcal{A}(s)$ do
        $Q(s, a) \leftarrow 0$
    end for
end for
for each $i$ in $\{1, \ldots, N\}$ do
    $s \leftarrow$ initial state for episode $i$
    Select action $a$ for state $s$ according to an $\epsilon$-greedy policy based on $Q$.
    while state $s$ is not terminal do
        $r \leftarrow$ observed reward for action $a$ at state $s$
        $s' \leftarrow$ observed next state for action $a$ at state $s$
        Select action $a'$ for state $s'$ according to an $\epsilon$-greedy policy based on $Q$.
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
        $s \leftarrow s'$
        $a \leftarrow a'$
    end while
end for
for each state $s \in \mathcal{S}$ do
    $\pi(s) \leftarrow \arg\max_a Q(s, a)$
end for
```

Another advantage of TD methods over Monte Carlo methods is that episodes don't need to be finite.

An alternative to Sarsa is Q-Learning, an off-policy control method. This means that Q-learning learns the action values for a policy that it does not use to act on the environment. The main difference to Sarsa is the update rule, which is given by

$$Q(s_t, a_a) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)].$$

Instead of choosing a' using the current policy, which may be ϵ-soft for learning purposes, this update rule considers the best possible action according to the current estimate. This avoids the problem of misattributing an inherent low value for a state in which a bad action choice was made. For the same reason, Q-learning may converge faster but have lower efficacy during the learning phase (called online efficacy). Algorithm 5 exemplifies a particular implementation of Q-learning.

Alternatively, the update rule for Q-learning can be given by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \sum_a \pi(s_t, a) Q(s_{t+1}, a) - Q(s_t, a_t)].$$

Algorithm 5 Q-learning control algorithm

```
Input: set of states $\mathcal{S}$, number of episodes $N$, learning rate $\alpha$, probability of choosing random action $\epsilon$, discount factor $\gamma$.
Output: deterministic policy $\pi$, optimal when $N \to \infty$ and $\alpha$ decays appropriately.
for each $s \in \mathcal{S}$ do
    for each action $a \in \mathcal{A}(s)$ do
        $Q(s,a) \leftarrow 0$
    end for
end for
for each $i$ in $\{1, \ldots, N\}$ do
    $s \leftarrow$ initial state for episode $i$
    while state $s$ is not terminal do
        Select action $a$ for state $s$ according to an $\epsilon$-greedy policy based on $Q$.
        $r \leftarrow$ observed reward for action $a$ at state $s$
        $s' \leftarrow$ observed next state for action $a$ at state $s$
        $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a' \in \mathcal{A}(s')} Q(s',a') - Q(s,a)]$
        $s \leftarrow s'$
    end while
end for
for each state $s \in \mathcal{S}$ do
    $\pi(s) \leftarrow \arg\max_a Q(s,a)$
end for
```

In this case, the probability of choosing each action is taken into account, which may improve its online efficacy.

It is often possible to design a more effective learning algorithm when some part of the environment dynamics is known. For instance, if the state following a state action pair is always known, it is possible to base policies on the value of possible successor states, by learning what is called an afterstate value function. In that case, the control algorithm can learn the state value function V instead of Q.

7 Eligibility Traces

The n-step return at time t is defined as

$$u_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n}).$$

This return can be used to update state (action) values using

$$V(s_t) \leftarrow V(s_t) + \alpha[u_t^{(n)} - V(s_t)].$$

When $n = 1$, the update rule above is equivalent to the temporal difference learning rule. When $n \to \infty$, this rule is equivalent to the rule used in Monte Carlo methods.

It is also possible to average $u_t^{(i)}$ for different i. One particularly important weighted average of variable step returns is the λ-return u_t^λ, which is given by

$$u_t^\lambda = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} u_t^{(n)}.$$

In this way, the weights given to each n-step return sum to 1. The 1-step return has the highest weight, which decreases by a factor of λ for each following return. When $\lambda = 0$, the definition is once again equivalent to temporal difference learning with one-step return. When $\lambda \to 1$, the method is equivalent to Monte Carlo. This gives a somewhat simple way to control the balance between state (action) value estimates and looking ahead for rewards. This idea has been shown to be very effective in practice for good choices of λ.

In practice, the update rule based on u_t^λ is efficiently implemented by using eligibility traces. The eligibility trace e_t for a state s at time t is defined as

$$e_t(s) = \begin{cases} 0 & \text{if } t = 0, \\ \gamma\lambda e_{t-1}(s) + 1 & \text{if } s = s_t \text{ and } t \neq 0, \\ \gamma\lambda e_{t-1}(s) & \text{if } s \neq s_t \text{ and } t \neq 0. \end{cases}$$

The following update rule should be applied to *every* state (action) value:

$$V(s) \leftarrow V(s) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]e_t(s). \tag{6}$$

Intuitively, the observed error for the estimate of the value of the current state s_t changes the estimate of s by a factor of $\alpha e_t(s)$, which is proportional to how soon state s was last visited. In this context, λ is called the trace decay factor.

The control algorithms presented in the previous section can be adapted to use eligibility traces as described by Algorithms 6 and 7. Note that the algorithm $Q(\lambda)$ needs to erase its eligibility traces once a non-greedy action is taken because of its off-policy evaluation.

The eligibility traces presented in this section are also called accumulating traces. Replacing traces are an alternative definition that may be better suited to some applications, particularly when the agent is likely to take the same action at a given state several times, causing a detrimental accumulation of its eligibility trace. The replacing trace e_t for a state s at time t is defined as

$$e_t(s) = \begin{cases} 0 & \text{if } t = 0, \\ 1 & \text{if } s = s_t \text{ and } t \neq 0, \\ \gamma\lambda e_{t-1}(s) & \text{if } s \neq s_t \text{ and } t \neq 0. \end{cases}$$

Algorithm 6 Sarsa(λ) control algorithm

Input: set of states $\mathcal{S}$, number of episodes N, learning rate α, probability of choosing random action ϵ, discount factor γ, trace decay λ

Output: deterministic policy π, optimal when $N \to \infty$ and α decays appropriately.

```
for each $s \in \mathcal{S}$ do
  for each action $a \in \mathcal{A}(s)$ do
    $Q(s, a) \leftarrow 0$
    $e(s, a) \leftarrow 0$
  end for
end for
for each $i$ in $\{1, \dots, N\}$ do
  Set the eligibility trace $e$ to zero for every valid state action pair.
  $s \leftarrow$ initial state for episode $i$
  Select action $a$ for state $s$ according to an $\epsilon$-greedy policy based on $Q$.
  while state $s$ is not terminal do
    $r \leftarrow$ observed reward for action $a$ at state $s$
    $s' \leftarrow$ observed next state for action $a$ at state $s$
    Select action $a'$ for state $s'$ according to an $\epsilon$-greedy policy based on $Q$.
    $e(s, a) \leftarrow e(s, a) + 1$
    for each $s'' \in \mathcal{S}$ do
      for each action $a'' \in \mathcal{A}(s'')$ do
        $Q(s'', a'') \leftarrow Q(s'', a'') + \alpha[r + \gamma Q(s', a') - Q(s, a)]e(s'', a'')$
        $e(s'', a'') \leftarrow \gamma\lambda e(s'', a'')$
      end for
    end for
    $s \leftarrow s'$
    $a \leftarrow a'$
  end while
end for
for each state $s \in \mathcal{S}$ do
  $\pi(s) \leftarrow \arg\max_a Q(s, a)$
end for
```

Algorithm 7 Q(λ) control algorithm

Input: set of states $\mathcal{S}$, number of episodes N, learning rate α, probability of choosing random action ϵ, discount factor γ, trace decay λ

Output: deterministic policy π, optimal when $N \to \infty$ and α decays appropriately.

```
for each $s \in \mathcal{S}$ do
  for each action $a \in \mathcal{A}(s)$ do
    $Q(s, a) \leftarrow 0$
    $e(s, a) \leftarrow 0$
  end for
end for
for each $i$ in $\{1, \ldots, N\}$ do
  Set the eligibility trace $e$ to zero for every valid state action pair.
  $s \leftarrow$ initial state for episode $i$
  Select action $a$ for state $s$ according to an $\epsilon$-greedy policy based on $Q$.
  while state $s$ is not terminal do
    $r \leftarrow$ observed reward for action $a$ at state $s$
    $s' \leftarrow$ observed next state for action $a$ at state $s$
    Select action $a'$ for state $s'$ according to an $\epsilon$-greedy policy based on $Q$.
    Select the greedy action $a^*$ for state $s'$ according to $Q$, preferring $a'$ when tied.
    $e(s, a) \leftarrow e(s, a) + 1$
    for each $s'' \in \mathcal{S}$ do
      for each action $a'' \in \mathcal{A}(s'')$ do
        $Q(s'', a'') \leftarrow Q(s'', a'') + \alpha[r + \gamma Q(s', a^*) - Q(s, a)]e(s'', a'')$
        if $a^* = a'$ then
          $e(s'', a'') \leftarrow \gamma\lambda e(s'', a'')$
        else
          $e(s'', a'') \leftarrow 0$
        end if
      end for
    end for
    $s \leftarrow s'$
    $a \leftarrow a'$
  end while
end for
for each state $s \in \mathcal{S}$ do
  $\pi(s) \leftarrow \arg\max_a Q(s, a)$
end for
```

8 Generalization and Function Approximation

In some situations, it is inadvisable to store state or action value estimates in arrays. In large (or infinite) state spaces, some states may be seen very rarely. In these cases, the state or action value estimates must generalize across states. The kind of generalization discussed in this section is based on function approximation, which is studied extensively in machine learning. For more details, see the corresponding notes by the same author.

The state value function V^π can be approximated by a parametric function $V : \mathcal{S} \times \mathbb{R}^m \to \mathbb{R}$. The goal of policy evaluation becomes finding a $\boldsymbol{\theta}$ such that $V^\pi(s) \approx V(s; \boldsymbol{\theta})$ for every $s \in \mathcal{S}$. Changing the parameter vector $\boldsymbol{\theta}$ will typically change the value estimates of several states.

For a given policy π, consider a dataset $\mathcal{D} = \{(s_i, V^\pi(s_i))\}_{i=1}^N$, where $s_i \in \mathcal{S}$ for every i. Any regression method could be used to fit V to this dataset. For simplicity, this section will focus on linear regression by stochastic gradient descent.

The mean squared error $J(\boldsymbol{\theta})$, a common measure of the quality of function approximation, is given by

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} [V^\pi(s_i) - V(s_i; \boldsymbol{\theta})]^2.$$

A parameter vector $\boldsymbol{\theta}^*$ such that $J(\boldsymbol{\theta}^*) \leq J(\boldsymbol{\theta})$ for any other vector $\boldsymbol{\theta} \in \mathbb{R}^m$ is called a global optimum of J. Finding a global optimum is rarely practical for complicated functions. Several methods used in practice are guaranteed only to find local optima, which are global optima in a restricted domain of the function.

Gradient descent is a very common technique for finding local optima in smooth functions. Stochastic gradient descent is a variant that is particularly useful in reinforcement learning. Starting from an arbitrary estimate $\boldsymbol{\theta}_0$, for any $t \geq 0$, the esimate $\boldsymbol{\theta}_{t+1}$ is given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{1}{2}\alpha \nabla_{\boldsymbol{\theta}} [V^\pi(s_t) - V(s_t; \boldsymbol{\theta}_t)]^2,$$

where the pair $(s_t, V^\pi(s_t))$ is drawn at random (with replacement) from $\mathcal{D}$, α is the learning rate, and $\nabla_{\boldsymbol{\theta}}[V^\pi(s_t) - V(s_t; \boldsymbol{\theta}_t)]^2$ is the gradient of $[V^\pi(s_t) - V(s_t; \boldsymbol{\theta}_t)]^2$ with respect to $\boldsymbol{\theta}$. This gradient is the vector of partial derivatives of the error term (squared term) with respect to each component of $\boldsymbol{\theta}$. Intuitively, at each time step, the estimate is changed in the direction that locally minimizes the error (squared term) for state s_t. The direction in which the error is lowered for each component of $\boldsymbol{\theta}_t$, for a particular state s_t, is given by this gradient.

By the chain rule, the equation above can be rewritten as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha[V^\pi(s_t) - V(s_t; \boldsymbol{\theta}_t)] \nabla_{\boldsymbol{\theta}} V(s_t; \boldsymbol{\theta}_t).$$

If α decays with time in the same way as described in Section 2, this method is guaranteed to converge to a local optimum of J.

Naturally, if $V^\pi(s)$ were available for all states $s \in \mathcal{S}$, there would be no need for function approximation. Furthermore, in practice, a dataset will be given by $\mathcal{D} = \{(s_i, v_i)\}_{i=1}^N$, where v_i is an estimate of the value of s_i under policy π, for every i. In that case, given an estimate for the parameter vector $\boldsymbol{\theta}_t$ and a pair (s_t, v_t) drawn at random (with replacement) from $\mathcal{D}$, the parameter vector $\boldsymbol{\theta}_t$ may be updated using

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha[v_t - V(s_t; \boldsymbol{\theta}_t)] \nabla_{\boldsymbol{\theta}} V(s_t; \boldsymbol{\theta}_t).$$

Instead of drawing pairs (s_t, v_t) at random from a fixed dataset, it is also possible to perform updates based on the states (and corresponding returns) observed while interacting with the environment using a policy π. This is the foundation of the algorithms presented in this section.

The update for $TD(\lambda)$ is given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha[u_t^\lambda - V(s_t; \boldsymbol{\theta}_t)] \nabla_{\boldsymbol{\theta}} V(s_t; \boldsymbol{\theta}_t).$$

It can be shown that this is equivalent to the following update rule using eligibility traces:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \delta_t \boldsymbol{e}_t,$$

where

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}; \boldsymbol{\theta}_t) - V(s_t; \boldsymbol{\theta}_t),$$

and

$$\boldsymbol{e}_t = \gamma\lambda\boldsymbol{e}_{t-1} + \nabla_{\boldsymbol{\theta}} V(s_t; \boldsymbol{\theta}_t),$$

and $\boldsymbol{e}_0 = \mathbf{0}$. Note that there is a vector of eligibility traces with the same number of components as a parameter vector.

Algorithm 8 is a complete algorithm for approximating V^π using gradient descent.

Algorithm 8 Gradient descent TD(λ) value estimation algorithm

```
Input: policy $\pi$, number of episodes $N$, learning rate $\alpha$, discount factor $\gamma$, trace decay $\lambda$
Output: parameter vector $\boldsymbol{\theta}$
Initialize $\boldsymbol{\theta}$ arbitrarily
for each $i$ in $\{1, \dots, N\}$ do
    $\boldsymbol{e} \leftarrow \mathbf{0}$
    $s \leftarrow$ initial state for episode $i$
    while state $s$ is not terminal do
        $a \leftarrow \pi(s)$
        $r \leftarrow$ observed reward for action $a$ at state $s$
        $s' \leftarrow$ observed next state for action $a$ at state $s$
        $\delta \leftarrow r + \gamma V(s'; \boldsymbol{\theta}) - V(s; \boldsymbol{\theta})$
        $\boldsymbol{e} \leftarrow \gamma\lambda\boldsymbol{e} + \nabla_{\boldsymbol{\theta}} V(s; \boldsymbol{\theta})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\boldsymbol{e}$
        $s \leftarrow s'$
    end while
end for
```

Linear functions of the parameter vector $\boldsymbol{\theta}$ are widely used for function approximation, mainly due to their stronger theoretical guarantees. In that case, a state is often represented by a *feature vector* $\boldsymbol{\phi}(s)$, which should be useful for generalization across states. Given a parameter vector $\boldsymbol{\theta}$, the value estimate $V(s; \boldsymbol{\theta})$ is then given by

$$V(s; \boldsymbol{\theta}) = \sum_i \phi(s)_i \theta_i.$$

Clearly, $\nabla_{\boldsymbol{\theta}} V(s; \boldsymbol{\theta}) = \boldsymbol{\phi}(s)$. Algorithms 9 and 10 illustrate Sarsa(λ) and Q(λ) based on linear function approximation.

The choice of feature vectors is crucial to the success of any function approximation method. Intuitively, the features should correspond to natural features of the state, which should facilitate generalization. For instance, a robot might represent its state as a combination of its position, velocity, and remaining power. In the case of linear models, it might be necessary to create features that are the combinations of other natural features, since linear models are, in general, incapable of modeling interactions such as feature i being beneficial only in the absence of feature j.

High dimensional state spaces often need to be simplified for use with linear models. There are several techniques to accomplish this. The states can be represented in feature vectors by discretizing the state space into overlapping hyperspheres (coarse coding) or by tiling into hypercubes (tile coding, which may combine several tilings with different offsets). States can also be represented by feature vectors of distance calculations to random points sampled in the state space (similar to coding by radial basis functions).

Algorithm 9 Sarsa(λ) control algorithm for linear function approximation

```
Input: feature vector $\boldsymbol{\phi}(s,a)$ for all state-action pairs $(s,a)$, number of episodes $N$, learning rate $\alpha$, probability of
    choosing random action $\epsilon$, discount factor $\gamma$, trace decay $\lambda$
Output: parameter vector $\boldsymbol{\theta}$
$\boldsymbol{\theta} \leftarrow \mathbf{0}$
for each $i$ in $\{1, \ldots, N\}$ do
    $\boldsymbol{e} \leftarrow \mathbf{0}$
    $s \leftarrow$ initial state for episode $i$
    for each $a \in A(s)$: do
        $Q(a) \leftarrow \sum_i \theta_i \phi(s,a)_i$
    end for
    $a \leftarrow \arg\max_{a \in A(s)} Q(a)$
    With probability $\epsilon$: $a \leftarrow$ random action in $A(s)$
    while state $s$ is not terminal do
        $\boldsymbol{e} \leftarrow \gamma\lambda\boldsymbol{e} + \boldsymbol{\phi}(s,a)$
        $r \leftarrow$ observed reward for action $a$ at state $s$
        $s' \leftarrow$ observed next state for action $a$ at state $s$
        $\delta \leftarrow r - Q(a)$
        for each $a \in A(s)$: do
            $Q(a) \leftarrow \sum_i \theta_i \phi(s',a)_i$
        end for
        $a' \leftarrow \arg\max_{a \in A(s')} Q(a)$
        With probability $\epsilon$: $a' \leftarrow$ random action in $A(s')$
        $\delta \leftarrow \delta + \gamma Q(a')$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\boldsymbol{e}$
        $s \leftarrow s'$
        $a \leftarrow a'$
    end while
end for
```

Algorithm 10 Q(λ) control algorithm for linear function approximation

```
Input: feature vector $\boldsymbol{\phi}(s,a)$ for all state-action pairs $(s,a)$, number of episodes $N$, learning rate $\alpha$, probability of
    choosing random action $\epsilon$, discount factor $\gamma$, trace decay $\lambda$
Output: parameter vector $\boldsymbol{\theta}$
$\boldsymbol{\theta} \leftarrow \mathbf{0}$
for each $i$ in $\{1, \ldots, N\}$ do
    $\boldsymbol{e} \leftarrow \mathbf{0}$
    $s \leftarrow$ initial state for episode $i$
    for each $a \in A(s)$: do
        $Q(a) \leftarrow \sum_i \theta_i \phi(s,a)_i$
    end for
    while state $s$ is not terminal do
        if with probability $1 - \epsilon$: then
            $a \leftarrow \arg\max_a Q(a)$
            $\boldsymbol{e} \leftarrow \gamma\lambda\boldsymbol{e} + \boldsymbol{\phi}(s,a)$
        else
            $a \leftarrow$ random action in $A(s)$
            $\boldsymbol{e} \leftarrow \boldsymbol{\phi}(s,a)$
        end if
        $r \leftarrow$ observed reward for action $a$ at state $s$
        $s' \leftarrow$ observed next state for action $a$ at state $s$
        $\delta \leftarrow r - Q(a)$
        for each $a \in A(s')$: do
            $Q(a) \leftarrow \sum_i \theta_i \phi(s',a)_i$
        end for
        $a' \leftarrow \arg\max_{a \in A(s')} Q(a)$
        $\delta \leftarrow \delta + \gamma Q(a')$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta\boldsymbol{e}$
        $s \leftarrow s'$
    end while
end for
```

9 Planning and Learning

Model free reinforcement learning methods, such as temporal difference learning, can be used together with model based methods, such as dynamic programming. In section 3, the one-step dynamics of the environment was presented as a complete model, i.e., a mechanism that can be used by the agent to predict the consequence of its actions.

In the case of deterministic environments, where the outcome of an action at a given state is always the same, it is sufficient to store these outcomes in a table as the agent learns. In stochastic environments, it is necessary to learn distribution models that represent the probabilities of each possible outcome and expected rewards.

In this context, planning is the process where the agent simulates, using its internal model, the environment and updates its value estimates.

Algorithm 11 illustrates the combination of direct reinforcement learning, using one-step Q-learning, and model learning in a deterministic environment. This method could be enhanced by eligibility traces, but that was omitted for simplicity.

Algorithm 11 Dyna-Q control algorithm for deterministic environments

Input: set of states $\mathcal{S}$, number of episodes N, number of planning steps per episode K, learning rate α, probability of choosing random action ϵ, discount factor γ.

Output: deterministic policy π, optimal when $N \to \infty$ and α decays appropriately.

```
for each $s \in \mathcal{S}$ do
  for each action $a \in \mathcal{A}(s)$ do
    $Q(s, a) \leftarrow 0$
  end for
end for
for each $i$ in $\{1, \dots, N\}$ do
  $s \leftarrow$ initial state for episode $i$
  while state $s$ is not terminal do
    Select action $a$ for state $s$ according to an $\epsilon$-greedy policy based on $Q$.
    $r \leftarrow$ observed reward for action $a$ at state $s$
    $s' \leftarrow$ observed next state for action $a$ at state $s$
    $M(s, a) \leftarrow s', r$
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a' \in A(s')} Q(s', a') - Q(s, a)]$
    $s_{\text{next}} \leftarrow s'$
    for each $k$ in $\{1, \dots, K\}$ do
      $s \leftarrow$ random previously observed state
      $a \leftarrow$ random action previously taken in state $s$
      $s', r \leftarrow M(s, a)$
      $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a' \in A(s')} Q(s', a') - Q(s, a)]$
    end for
    $s \leftarrow s_{\text{next}}$
  end while
end for
for each state $s \in \mathcal{S}$ do
  $\pi(s) \leftarrow \arg\max_{a \in A(s)} Q(s, a)$
end for
```

The main advantage of using model learning is the possibility of updating the value of any state at a given time step. This includes states that are not even visited in a given episode. In practice, this may considerably reduce the necessity of interaction with the environment.

Algorithm 12 presents a variation of the previous algorithm for stochastic environments. The algorithm performs full backup, as in dynamic programming methods, using estimates of $\mathcal{P}^a_{ss'}$ and $\mathcal{R}^a_{ss'}$ learned during interaction with the environment.

However, in the case of non-stationary environments, meaning that the environment changes its behavior with time, a model may become incorrect. In this case, the agent may become stuck behaving suboptimally. Even using ϵ-greedy action selection, some action sequences may become too unlikely. This is a classical instance of the exploitation versus exploration problem. There is a simple heuristic that can overcome this problem in planning agents. If the agent stores the time n since the last execution of action a at state s, the planning step may consider the reward for the state-action combination as $r + \kappa\sqrt{n}$ for a small parameter κ and the reward estimate (given

Algorithm 12 Dyna-Q control algorithm for stochastic environments

```
Input: set of states $\mathcal{S}$, number of episodes $N$, number of planning steps per episode $K$, learning rate $\alpha$, probability
    of choosing random action $\epsilon$, discount factor $\gamma$.
Output: deterministic policy $\pi$, optimal when $N \to \infty$ and $\alpha$ decays appropriately.
for each $s \in \mathcal{S}$ do
    for each action $a \in \mathcal{A}(s)$ do
        $Q(s, a) \leftarrow 0$
        $R(s, s', a) \leftarrow 0$
        $\mathcal{N}(s, a) \leftarrow 0$
        $\mathcal{N}_n(s, a, s') \leftarrow 0$
    end for
end for
for each $i$ in $\{1, \ldots, N\}$ do
    $s \leftarrow$ initial state for episode $i$
    while state $s$ is not terminal do
        Select action $a$ for state $s$ according to an $\epsilon$-greedy policy based on $Q$.
        $r \leftarrow$ observed reward for action $a$ at state $s$
        $s' \leftarrow$ observed next state for action $a$ at state $s$
        $\mathcal{N}(s, a) \leftarrow \mathcal{N}(s, a) + 1$
        $\mathcal{N}_n(s, a, s') \leftarrow \mathcal{N}_n(s, a, s') + 1$
        $R(s, s', a) \leftarrow R(s, s', a) + \frac{r - R(s, s', a)}{\mathcal{N}_n(s, a, s')}$
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a' \in A(s')} Q(s', a') - Q(s, a)]$
        $s_{\text{next}} \leftarrow s'$
        for each $k$ in $\{1, \ldots, K\}$ do
            $s \leftarrow$ random previously observed state
            $a \leftarrow$ random action previously taken in state $s$
            $Q(s, a) \leftarrow \sum_{s'} [\frac{\mathcal{N}_n(s, a, s')}{\mathcal{N}(s, a)}][R(s, a, s') + \gamma \max_{a'} Q(s', a')]$
        end for
        $s \leftarrow s_{\text{next}}$
    end while
end for
for each state $s \in \mathcal{S}$ do
    $\pi(s) \leftarrow \arg\max_{a \in A(s)} Q(s, a)$
end for
```

by the model) r. In this case, a whole sequence of exploratory actions may be undertaken. In practice, however, controlling κ may be challenging.

In algorithm 11, simulated transitions are sampled uniformly. However, this is usually not the best choice, even for the dynamic programming methods presented in section 4. A simple heuristic called prioritized sweeping may be employed to choose transitions for planning. This heuristic prefers to update values for transitions that go into states that had their values considerably changed in the current episode. A method based on this heuristic, for deterministic environments, is presented in algorithm 13.

Algorithm 13 Prioritized sweeping Dyna-Q control algorithm for deterministic environments

Input: set of states $\mathcal{S}$, number of episodes N, number of planning steps per episode K, learning rate α, probability of choosing random action ϵ, discount factor γ, heap threshold h.

Output: deterministic policy π, optimal when $N \to \infty$ and α decays appropriately.

```
$\mathcal{H} \leftarrow$ empty maximum priority queue (max-heap)
for each $s \in \mathcal{S}$ do
    for each action $a \in \mathcal{A}(s)$ do
        $Q(s, a) \leftarrow 0$
    end for
end for
for each $i$ in $\{1, \dots, N\}$ do
    $s \leftarrow$ initial state for episode $i$
    while state $s$ is not terminal do
        Select action $a$ for state $s$ according to an $\epsilon$-greedy policy based on $Q$.
        $r \leftarrow$ observed reward for action $a$ at state $s$
        $s' \leftarrow$ observed next state for action $a$ at state $s$
        $M(s, a) = s', r$
        $p \leftarrow |r + \gamma \max_{a' \in A(s')} Q(s', a') - Q(s, a)|$
        if $p > h$ then
            Insert $(s, a)$ into $\mathcal{H}$ with priority $p$
        end if
        $s_{\text{next}} \leftarrow s'$
        for each $k$ in $\{1, \dots, K\}$ or empty($\mathcal{H}$) do
            $s, a \leftarrow$ first($\mathcal{H}$)
            $s', r \leftarrow M(s, a)$
            $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a' \in A(s')} Q(s', a') - Q(s, a)]$
            for each $\bar{s}, \bar{a}$ predicted to lead to $s$ do
                $\bar{r} \leftarrow$ predicted reward for transition $\bar{s}, \bar{a}$
                $p \leftarrow |\bar{r} + \gamma \max_{a \in A(s)} Q(s, a) - Q(\bar{s}, \bar{a})|$
                if $p > h$ then
                    Insert $(\bar{s}, \bar{a})$ into $\mathcal{H}$ with priority $p$
                end if
            end for
        end for
        $s \leftarrow s_{\text{next}}$
    end while
end for
for each state $s \in \mathcal{S}$ do
    $\pi(s) \leftarrow \arg\max_{a \in A(s)} Q(s, a)$
end for
```

Algorithm 13 maintains a priority queue (max-heap) with every state-action pair whose estimate would change considerably when updated, ordered by the size of the change. When the pair is updated, the change is propagated to transitions that are predicted to lead to it. This idea dramatically increases the performance of agents in some environments.

Another important heuristic is called trajectory sampling. In this case, transitions are selected for planning following the on-policy distribution. In other words, state-action pairs that are frequently visited are more likely to be sampled for updates by planning.

10 Policy Gradient Methods *

For details on the probability notation employed in this section, see the machine learning notes by the same author.

Consider an agent that interacts with its environment in a sequence of episodes, each of which lasts for exactly T time steps (without loss of generality for episodic problems).

At each time step t, suppose that the agent observes a state $s_t \in \text{Val}(S_t)$, receives a reward $r_t \in \text{Val}(R_t)$, and chooses an action $a_t \in \text{Val}(A_t)$. Furthermore, suppose that $\text{Val}(S_t), \text{Val}(R_t)$, and $\text{Val}(A_t)$ are finite for all t.

A policy gradient method represents a policy by a probability distribution over actions given states, and its goal is finding parameters for such a policy that achieve maximum expected return.

Let $\boldsymbol{\tau} = s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T, s_T$ denote a trajectory in a particular episode, such that $\boldsymbol{\tau} \in \text{Val}(\mathbf{T})$. Assuming the Markov property, the probability $p(\boldsymbol{\tau} \mid \boldsymbol{\theta})$ of trajectory $\boldsymbol{\tau}$ given the parameters $\boldsymbol{\theta}$ is given by

$$p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}, r_{t+1} \mid s_t, a_t) p(a_t \mid s_t, \boldsymbol{\theta}),$$

where $p(a_t \mid s_t, \boldsymbol{\theta})$ is the probability of action a_t given state s_t and parameters $\boldsymbol{\theta}$, which is given by the policy.

Given a policy parameterized by $\boldsymbol{\theta}$, consider the expected return $J(\boldsymbol{\theta})$ given by

$$J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T} R_t \mid \boldsymbol{\theta}\right] = \sum_{t=1}^{T} \mathbb{E}\left[R_t \mid \boldsymbol{\theta}\right].$$

Using the law of the unconscious statistician,

$$J(\boldsymbol{\theta}) = \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) \sum_{t=1}^{T} r_t = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} r_t p(\boldsymbol{\tau} \mid \boldsymbol{\theta}).$$

Consider the task of finding a $\boldsymbol{\theta}^*$ such that $J(\boldsymbol{\theta}^*) = \max_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. Assuming $J(\boldsymbol{\theta})$ is differentiable with respect to $\boldsymbol{\theta}$, it is possible to attempt to solve this optimization task by gradient-based methods, such as gradient ascent.

The partial derivative $\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$ of J with respect to θ_j at $\boldsymbol{\theta}$ is given by

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} r_t \frac{\partial}{\partial \theta_j} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}).$$

Suppose that $p(\boldsymbol{\tau} \mid \boldsymbol{\theta})$ is positive for any $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$. The so-called likelihood ratio trick uses the fact that

$$\frac{\partial}{\partial \theta_j} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) \frac{1}{p(\boldsymbol{\tau} \mid \boldsymbol{\theta})} \frac{\partial}{\partial \theta_j} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} \log p(\boldsymbol{\tau} \mid \boldsymbol{\theta}).$$

By using the expression above,

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} \log p(\boldsymbol{\tau} \mid \boldsymbol{\theta}).$$

Because we have already assumed that $p(\boldsymbol{\tau} \mid \boldsymbol{\theta})$ is positive for all $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$,

$$\log p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = \log p(s_0) + \sum_{t=0}^{T-1} \log p(s_{t+1}, r_{t+1} \mid s_t, a_t) + \sum_{t=0}^{T-1} \log p(a_t \mid s_t, \boldsymbol{\theta}).$$

Therefore,

$$\frac{\partial}{\partial \theta_j} \log p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = \sum_{t'=0}^{T-1} \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}).$$

By using the expression above,

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) r_t \left[\sum_{t'=0}^{T-1} \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid s_{t'}, \boldsymbol{\theta})\right].$$

It will be useful to split the innermost summation in the expression above into before and after t, leading to

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) \left[r_t \sum_{t'=0}^{t-1} \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}) + r_t \sum_{t'=t}^{T-1} \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}) \right].$$

Alternatively, the expression above can be written as

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{t'=0}^{t-1} \mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right] + \sum_{t=1}^{T} \sum_{t'=t}^{T-1} \mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right].$$

We will now show that the rightmost nested summations in the expression above can be dismissed.

By representing the random variables involved in a trajectory using a Bayesian network, it can be seen that $A_{t'} \perp\!\!\!\perp R_t \mid S_{t'}, \boldsymbol{\theta}$ for $t' \geq t$. In other words, the action at time step $t' \geq t$ does not depend on the reward at time step t given the state at time step t' and the policy parameters. The analogous statement is not generally true for $t' < t$. For $t' \geq t$, this independence leads to

$$\mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right] = \sum_{r_t} \sum_{a_{t'}} \sum_{s_{t'}} p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}) p(r_t, s_{t'} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}).$$

By reversing the likelihood-ratio trick,

$$\mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right] = \sum_{r_t} \sum_{a_{t'}} \sum_{s_{t'}} p(r_t, s_{t'} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}).$$

By changing the order of summations and pushing constants outside the innermost summation,

$$\mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right] = \sum_{r_t} \sum_{s_{t'}} p(r_t, s_{t'} \mid \boldsymbol{\theta}) r_t \sum_{a_{t'}} \frac{\partial}{\partial \theta_j} p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}).$$

Finally, using the fact that $\frac{\partial}{\partial \theta_j} 1 = 0$,

$$\mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right] = \sum_{r_t} \sum_{s_{t'}} p(r_t, s_{t'} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} \sum_{a_{t'}} p(a_{t'} \mid s_{t'}, \boldsymbol{\theta}) = 0.$$

We may now remove the rightmost nested summations in the previous expression for $\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$, which gives

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{t'=0}^{t-1} \mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right] = \mathbb{E}\left[\sum_{t=1}^{T} R_t \sum_{t'=0}^{t-1} \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid S_{t'}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right].$$

By reordering the summations, the expression above can be conveniently rewritten as

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \frac{\partial}{\partial \theta_j} \log p(A_t \mid S_t, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} R_{t'} \mid \boldsymbol{\theta} \right].$$

Finally, the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ of the expected return J at $\boldsymbol{\theta}$ is given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(A_t \mid S_t, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} R_{t'} \mid \boldsymbol{\theta} \right].$$

This result has a subtle intuitive interpretation. In simple terms, the gradient of the expected return is a sum of expected values of random vectors that correspond to each time step. The expected value for time step t weights a direction that locally increases the probability of each possible decision by its expected (positive or negative) outcome. Therefore, in gradient ascent, positive expected outcomes contribute towards making the probability of a decision higher, while negative expected outcomes contribute towards making the probability of a decision lower.

Consider a sequence $\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_N$ of N trajectories obtained by following the policy parameterized by $\boldsymbol{\theta}$, and let

$$\boldsymbol{\tau}_i = s_{i,0}, a_{i,0}, r_{i,1}, s_{i,1}, a_{i,1}, r_{i,2}, \ldots, s_{i,T-1}, a_{i,T-1}, r_{i,T}, s_{i,T}.$$

A Monte Carlo estimate $\hat{\mathbf{g}}(\boldsymbol{\theta})$ to $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is given by

$$\hat{\mathbf{g}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(a_{i,t} \mid s_{i,t}, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} r_{i,t'},$$

and may be used for gradient ascent. However, in practice, this estimate may require too many trajectories for accuracy. The remainder of this section presents some alternatives.

Firstly, we will show that introducing a so-called baseline function results in an alternative expression for the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. Concretely, consider the expected value

$$\mathbb{E}\left[\sum_{t=0}^{T-1} \frac{\partial}{\partial \theta_j} \log p(A_t \mid S_t, \boldsymbol{\theta}) \left[\left[\sum_{t'=t+1}^{T} R_{t'}\right] - B_t^{\boldsymbol{\theta}}(S_t)\right] \mid \boldsymbol{\theta}\right] = \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) - \sum_{t=0}^{T-1} \mathbb{E}\left[\frac{\partial}{\partial \theta_j} \log p(A_t \mid S_t, \boldsymbol{\theta}) B_t^{\boldsymbol{\theta}}(S_t)\right],$$

where $B_t^{\boldsymbol{\theta}}$ is an arbitrary function that may depend on the policy parameters $\boldsymbol{\theta}$ and the time step t. By definition,

$$\mathbb{E}\left[\frac{\partial}{\partial \theta_j} \log p(A_t \mid S_t, \boldsymbol{\theta}) B_t^{\boldsymbol{\theta}}(S_t)\right] = \sum_{s_t} \sum_{a_t} p(a_t \mid s_t, \boldsymbol{\theta}) p(s_t \mid \boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} \log p(a_t \mid s_t, \boldsymbol{\theta}) B_t^{\boldsymbol{\theta}}(s_t).$$

Reversing the likelihood ratio trick,

$$\mathbb{E}\left[\frac{\partial}{\partial \theta_j} \log p(A_t \mid S_t, \boldsymbol{\theta}) B_t^{\boldsymbol{\theta}}(S_t)\right] = \sum_{s_t} p(s_t \mid \boldsymbol{\theta}) B_t^{\boldsymbol{\theta}}(s_t) \sum_{a_t} \frac{\partial}{\partial \theta_j} p(a_t \mid s_t, \boldsymbol{\theta}) = \sum_{s_t} p(s_t \mid \boldsymbol{\theta}) B_t^{\boldsymbol{\theta}}(s_t) \frac{\partial}{\partial \theta_j} 1 = 0.$$

Therefore, an alternative expression for the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(A_t \mid S_t, \boldsymbol{\theta}) \left[\left[\sum_{t'=t+1}^{T} R_{t'}\right] - B_t^{\boldsymbol{\theta}}(S_t)\right] \mid \boldsymbol{\theta}\right].$$

A common choice of baseline function $B_t^{\boldsymbol{\theta}}$ is an approximation to the (typically unknown) value function $V_t^{\boldsymbol{\theta}}$ given by

$$V_t^{\boldsymbol{\theta}}(s) = \mathbb{E}\left[\sum_{t'=t+1}^{T} R_{t'} \mid S_t = s, \boldsymbol{\theta}\right].$$

Because $V_t^{\boldsymbol{\theta}}(s)$ is the expected return for starting at state s at time step t and following a policy parameterized by $\boldsymbol{\theta}$, the corresponding estimate to $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ intuitively tends to increase the probability of actions that are followed by surprisingly large returns, and is generally more efficient in practice than the alternative presented so far. For details on value function approximation methods, see the previous sections. Note that, in episodic problems, the value of a state often depends on the time step.

Consider once again the partial derivative $\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$ of J with respect to θ_j at $\boldsymbol{\theta}$ given by

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \frac{\partial}{\partial \theta_j} \log p(A_t \mid S_t, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} R_{t'} \mid \boldsymbol{\theta}\right].$$

By definition,

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) &= \sum_{t=0}^{T-1} \sum_{s_t} \sum_{a_t} \sum_{r_{t+1}} \cdots \sum_{r_T} p(s_t, a_t, r_{t+1}, \ldots, r_T \mid \boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} \log p(a_t \mid s_t, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} r_{t'} \\
&= \sum_{t=0}^{T-1} \sum_{s_t} \sum_{a_t} p(s_t, a_t \mid \boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} \log p(a_t \mid s_t, \boldsymbol{\theta}) \sum_{r_{t+1}} \cdots \sum_{r_T} p(r_{t+1}, \ldots, r_T \mid s_t, a_t, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} r_{t'} \\
&= \mathbb{E}\left[\sum_{t=0}^{T-1} \frac{\partial}{\partial \theta_j} \log p(A_t \mid S_t, \boldsymbol{\theta}) Q_t^{\boldsymbol{\theta}}(S_t, A_t) \mid \boldsymbol{\theta}\right],
\end{aligned}$$

where the so-called action value function $Q_t^{\boldsymbol{\theta}}$ is given by

$$Q_t^{\boldsymbol{\theta}}(s, a) = \mathbb{E}\left[\sum_{t'=t+1}^{T} R_{t'} \mid S_t = s, A_t = a, \boldsymbol{\theta}\right].$$

Intuitively, $Q_t^{\boldsymbol{\theta}}(s, a)$ is the expected return for starting at state s at time step t, choosing action a, and then following the policy parameterized by $\boldsymbol{\theta}$.

Therefore, another expression for the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ of the expected return J at $\boldsymbol{\theta}$ is given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(A_t \mid S_t, \boldsymbol{\theta}) Q_t^{\boldsymbol{\theta}}(S_t, A_t) \mid \boldsymbol{\theta}\right].$$

However, using this expression to obtain an estimate to the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ typically requires approximating the unknown action value function $Q_t^{\boldsymbol{\theta}}$. Such approximation will generally introduce bias (convergence to an undesired expected value) in the corresponding estimate to $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. For details on action value function approximation methods, see the previous sections. Note that, in episodic problems, the action value of a state-action pair often depends on the time step.

The advantage function $A_t^{\boldsymbol{\theta}}$ is defined by $A_t^{\boldsymbol{\theta}}(s, a) = Q_t^{\boldsymbol{\theta}}(s, a) - V_t^{\boldsymbol{\theta}}(s)$, for all s and a. By combining the previous results, yet another expression for the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ of the expected return J at $\boldsymbol{\theta}$ is given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(A_t \mid S_t, \boldsymbol{\theta}) A_t^{\boldsymbol{\theta}}(S_t, A_t) \mid \boldsymbol{\theta}\right].$$

Using this expression to obtain an estimate to the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ typically requires approximating the unknown advantage function $A_t^{\boldsymbol{\theta}}$. Such approximation also generally introduces a bias in the corresponding estimate to $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. However, in practice, such estimate is generally more efficient than the alternatives presented so far.

11 Recurrent Policy Gradient Methods *

In contrast to the reinforcement learning methods presented in the previous sections, recurrent policy gradient methods admit partially observable environments [5]. For details on the probability notation employed in this section, see the machine learning notes by the same author. This section also uses a slightly different convention for labeling time steps.

Consider an agent that interacts with its environment in a sequence of episodes, each of which lasts for exactly $T > 1$ time steps (without loss of generality for episodic problems).

At each time step t, suppose that the environment is in hidden state $s_t \in \text{Val}(S_t)$, the agent receives an observation $x_t \in \text{Val}(X_t)$, a reward $r_t \in \text{Val}(R_t)$, and chooses an action $a_t \in \text{Val}(A_t)$. Furthermore, suppose that $\text{Val}(S_t), \text{Val}(X_t), \text{Val}(R_t)$, and $\text{Val}(A_t)$ are finite, for all t.

A recurrent policy gradient method represents a policy by a probability distribution over actions given the history of observations and actions, and its goal is finding parameters for such a policy that achieve maximum expected return.

Let $\boldsymbol{\tau} = s_1, x_1, r_1, a_1, \ldots, s_{T-1}, x_{T-1}, r_{T-1}, a_{T-1}, s_T, x_T, r_T$ denote a trajectory in a particular episode, such that $\boldsymbol{\tau} \in \text{Val}(\mathbf{T})$. Furthermore, let $v_{i:j} = v_i, v_{i+1}, \ldots, v_j$.

We will assume that the probability $p(\boldsymbol{\tau} \mid \boldsymbol{\theta})$ of trajectory $\boldsymbol{\tau}$ given the parameters $\boldsymbol{\theta}$ is given by

$$p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = p(s_1)p(x_1 \mid s_1)p(r_1 \mid s_1) \prod_{t=1}^{T-1} p(a_t \mid x_{1:t}, a_{1:t-1}, \boldsymbol{\theta}) p(s_{t+1} \mid s_t, a_t) p(x_{t+1} \mid s_{t+1}) p(r_{t+1} \mid s_{t+1}),$$

where $p(a_t \mid x_{1:t}, a_{1:t-1}, \boldsymbol{\theta})$ is the probability of action a_t given the history of observations and actions up to time t, which is given by the policy. Intuitively, the hidden state s_t encodes all the information required to generate the observation x_t and the reward r_t. Furthermore, the action a_t may depend on the observed history up to time t (excluding the rewards, which may be encoded into the observations without loss of generality).

Given a policy parameterized by $\boldsymbol{\theta}$, consider the expected return $J(\boldsymbol{\theta})$ given by

$$J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T} R_t \mid \boldsymbol{\theta}\right] = \sum_{t=1}^{T} \mathbb{E}\left[R_t \mid \boldsymbol{\theta}\right].$$

Using the law of the unconscious statistician,

$$J(\boldsymbol{\theta}) = \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) \sum_{t=1}^{T} r_t = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} r_t p(\boldsymbol{\tau} \mid \boldsymbol{\theta}).$$

Consider the task of finding a $\boldsymbol{\theta}^*$ such that $J(\boldsymbol{\theta}^*) = \max_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. Assuming $J(\boldsymbol{\theta})$ is differentiable with respect to $\boldsymbol{\theta}$, it is possible to attempt to solve this optimization task by gradient-based methods, such as gradient ascent.

The partial derivative $\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$ of J with respect to θ_j at $\boldsymbol{\theta}$ is given by

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} r_t \frac{\partial}{\partial \theta_j} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}).$$

Suppose that $p(\boldsymbol{\tau} \mid \boldsymbol{\theta})$ is positive for any $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$. The so-called likelihood ratio trick uses the fact that

$$\frac{\partial}{\partial \theta_j} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) \frac{1}{p(\boldsymbol{\tau} \mid \boldsymbol{\theta})} \frac{\partial}{\partial \theta_j} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} \log p(\boldsymbol{\tau} \mid \boldsymbol{\theta}).$$

By using the expression above,

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} \log p(\boldsymbol{\tau} \mid \boldsymbol{\theta}).$$

Because we have already assumed that $p(\boldsymbol{\tau} \mid \boldsymbol{\theta})$ is positive for all $\boldsymbol{\tau}$ and $\boldsymbol{\theta}$,

$$\log p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = \log \left[p(s_1) p(x_1 \mid s_1) p(r_1 \mid s_1) \right] + \sum_{t=1}^{T-1} \log \left[p(a_t \mid x_{1:t}, a_{1:t-1}, \boldsymbol{\theta}) p(s_{t+1} \mid s_t, a_t) p(x_{t+1} \mid s_{t+1}) p(r_{t+1} \mid s_{t+1}) \right].$$

Therefore,

$$\frac{\partial}{\partial \theta_j} \log p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) = \sum_{t'=1}^{T-1} \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid x_{1:t'}, a_{1:t'-1}, \boldsymbol{\theta}).$$

By using the expression above,

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) r_t \left[\sum_{t'=1}^{T-1} \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid x_{1:t'}, a_{1:t'-1}, \boldsymbol{\theta}) \right].$$

It will be useful to split the innermost summation in the expression above into before and after t, leading to

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau} \mid \boldsymbol{\theta}) \left[r_t \sum_{t'=1}^{t-1} \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid x_{1:t'}, a_{1:t'-1}, \boldsymbol{\theta}) + r_t \sum_{t'=t}^{T-1} \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid x_{1:t'}, a_{1:t'-1}, \boldsymbol{\theta}) \right].$$

Alternatively, the expression above can be written as

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{t'=1}^{t-1} \mathbb{E} \left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid X_{1:t'}, A_{1:t'-1}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right] + \sum_{t=1}^{T} \sum_{t'=t}^{T-1} \mathbb{E} \left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid X_{1:t'}, A_{1:t'-1}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right].$$

We will now show that the rightmost nested summations in the expression above can be dismissed, because $E = \mathbb{E} \left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid X_{1:t'}, A_{1:t'-1}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right] = 0$ for $t' \geq t$.

By representing the random variables involved in a trajectory using a Bayesian network, it can be seen that $A_{t'} \perp\!\!\!\perp R_t \mid X_{1:t'}, A_{1:t'-1}, \boldsymbol{\theta}$ for $t' \geq t$. In other words, the action at time step $t' \geq t$ does not depend on the reward at time step t given the history of observations and actions up to time t' and the policy parameters. The analogous statement is not generally true for $t' < t$. For $t' \geq t$, this independence leads to

$$\begin{aligned} E &= \sum_{r_t} \sum_{x_{1:t'}} \sum_{a_{1:t'-1}} \sum_{a_{t'}} p(r_t, x_{1:t'}, a_{1:t'-1}, a_{t'} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid x_{1:t'}, a_{1:t'-1}, \boldsymbol{\theta}) \\ &= \sum_{r_t} \sum_{x_{1:t'}} \sum_{a_{1:t'-1}} \sum_{a_{t'}} p(a_{t'} \mid x_{1:t'}, a_{1:t'-1}, \boldsymbol{\theta}) p(r_t, x_{1:t'}, a_{1:t'-1} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} \log p(a_{t'} \mid x_{1:t'}, a_{1:t'-1}, \boldsymbol{\theta}). \end{aligned}$$

By reversing the likelihood-ratio trick,

$$E = \sum_{r_t} \sum_{x_{1:t'}} \sum_{a_{1:t'-1}} \sum_{a_{t'}} p(r_t, x_{1:t'}, a_{1:t'-1} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} p(a_{t'} \mid x_{1:t'}, a_{1:t'-1}, \boldsymbol{\theta}).$$

By pushing constants outside the innermost summation and using the fact that $\frac{\partial}{\partial \theta_j} 1 = 0$,

$$E = \sum_{r_t} \sum_{x_{1:t'}} \sum_{a_{1:t'-1}} p(r_t, x_{1:t'}, a_{1:t'-1} \mid \boldsymbol{\theta}) r_t \frac{\partial}{\partial \theta_j} \sum_{a_{t'}} p(a_{t'} \mid x_{1:t'}, a_{1:t'-1}, \boldsymbol{\theta}) = 0.$$

We may now remove the rightmost nested summations in the previous expression for $\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$, which gives

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \sum_{t=1}^{T} \sum_{t'=1}^{t-1} \mathbb{E}\left[R_t \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid X_{1:t'}, A_{1:t'-1}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right] = \mathbb{E}\left[\sum_{t=1}^{T} R_t \sum_{t'=1}^{t-1} \frac{\partial}{\partial \theta_j} \log p(A_{t'} \mid X_{1:t'}, A_{1:t'-1}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \right].$$

By reordering the summations, the expression above can be conveniently rewritten as

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T-1} \frac{\partial}{\partial \theta_j} \log p(A_t \mid X_{1:t}, A_{1:t-1}, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} R_{t'} \mid \boldsymbol{\theta} \right].$$

Finally, the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ of the expected return J at $\boldsymbol{\theta}$ is given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(A_t \mid X_{1:t}, A_{1:t-1}, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} R_{t'} \mid \boldsymbol{\theta} \right].$$

This result has a subtle intuitive interpretation. In simple terms, the gradient of the expected return is a sum of expected values of random vectors that correspond to each time step. The expected value for time step t weights a direction that locally increases the probability of each possible decision by its expected (positive or negative) outcome. Therefore, in gradient ascent, positive expected outcomes contribute towards making the probability of a decision higher, while negative expected outcomes contribute towards making the probability of a decision lower.

Consider a sequence $\boldsymbol{\tau}_1, \dots, \boldsymbol{\tau}_N$ of N trajectories obtained by following the policy parameterized by $\boldsymbol{\theta}$, and let

$$\boldsymbol{\tau}_i = s_{i,1}, x_{i,1}, r_{i,1}, a_{i,1}, \dots, s_{i,T-1}, x_{i,T-1}, r_{i,T-1}, a_{i,T-1}, s_{i,T}, x_{i,T}, r_{i,T}.$$

A Monte Carlo estimate $\hat{\mathbf{g}}(\boldsymbol{\theta})$ to $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ such that $\hat{\mathbf{g}}(\boldsymbol{\theta}) \to \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ when $N \to \infty$ is given by

$$\hat{\mathbf{g}}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(a_{i,t} \mid x_{i,1:t}, a_{i,1:t-1}, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} r_{i,t'},$$

and may be used for gradient ascent. Notice that this estimate does not require the hidden states of the environment. However, in practice, it may require too many trajectories for accuracy. The remainder of this section presents some alternatives.

Firstly, we will show that introducing a so-called baseline function results in an alternative expression for the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. Concretely, consider the expected value

$$\mathbb{E}\left[\sum_{t=1}^{T-1} \frac{\partial}{\partial \theta_j} \log p(A_t \mid X_{1:t}, A_{1:t-1}, \boldsymbol{\theta}) \left[\left[\sum_{t'=t+1}^{T} R_{t'} \right] - B_t^{\boldsymbol{\theta}}(X_{1:t}, A_{1:t-1}) \right] \mid \boldsymbol{\theta} \right] = \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) - C,$$

where $B_t^{\boldsymbol{\theta}}$ is an arbitrary function that may depend on the policy parameters $\boldsymbol{\theta}$ and the time step t, and

$$C = \mathbb{E}\left[\sum_{t=1}^{T-1} \frac{\partial}{\partial \theta_j} \log p(A_t \mid X_{1:t}, A_{1:t-1}, \boldsymbol{\theta}) B_t^{\boldsymbol{\theta}}(X_{1:t}, A_{1:t-1}) \mid \boldsymbol{\theta} \right].$$

By definition,

$$\begin{aligned}C &= \sum_{t=1}^{T-1} \sum_{x_{1:t}} \sum_{a_{1:t-1}} \sum_{a_t} p(x_{1:t}, a_{1:t-1}, a_t \mid \boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} \log p(a_t \mid x_{1:t}, a_{1:t-1}, \boldsymbol{\theta}) B_t^{\boldsymbol{\theta}}(x_{1:t}, a_{1:t-1}) \\ &= \sum_{t=1}^{T-1} \sum_{x_{1:t}} \sum_{a_{1:t-1}} p(x_{1:t}, a_{1:t-1} \mid \boldsymbol{\theta}) B_t^{\boldsymbol{\theta}}(x_{1:t}, a_{1:t-1}) \sum_{a_t} p(a_t \mid x_{1:t}, a_{1:t-1}, \boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} \log p(a_t \mid x_{1:t}, a_{1:t-1}, \boldsymbol{\theta}).\end{aligned}$$

Reversing the likelihood ratio trick and using the fact that $\frac{\partial}{\partial \theta_j} 1 = 0$,

$$C = \sum_{t=1}^{T-1} \sum_{x_{1:t}} \sum_{a_{1:t-1}} p(x_{1:t}, a_{1:t-1} \mid \boldsymbol{\theta}) B_t^{\boldsymbol{\theta}}(x_{1:t}, a_{1:t-1}) \frac{\partial}{\partial \theta_j} \sum_{a_t} p(a_t \mid x_{1:t}, a_{1:t-1}, \boldsymbol{\theta}) = 0.$$

Therefore, an alternative expression for the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(A_t \mid X_{1:t}, A_{1:t-1}, \boldsymbol{\theta}) \left[\left[\sum_{t'=t+1}^{T} R_{t'} \right] - B_t^{\boldsymbol{\theta}}(X_{1:t}, A_{1:t-1}) \right] \mid \boldsymbol{\theta} \right].$$

A possible choice for baseline function $B_t^{\boldsymbol{\theta}}$ is an approximation to the (typically unknown) value function $V_t^{\boldsymbol{\theta}}$ given by

$$V_t^{\boldsymbol{\theta}}(x'_{1:t}, a'_{1:t-1}) = \sum_{t'=t+1}^{T} \mathbb{E}\left[R_{t'} \mid X_{1:t} = x'_{1:t}, A_{1:t-1} = a'_{1:t-1}, \boldsymbol{\theta} \right].$$

Because $V_t^{\boldsymbol{\theta}}$ gives the expected return for following a policy parameterized by $\boldsymbol{\theta}$ after a particular history of observations and actions, the corresponding estimate to $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ intuitively tends to increase the probability of actions that are followed by surprisingly large returns, and may be more efficient in practice than the alternative presented so far. Value function approximation for partially observable environments requires extensions to the methods presented in the previous sections.

Consider once again the partial derivative $\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$ of J with respect to θ_j at $\boldsymbol{\theta}$ given by

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T-1} \frac{\partial}{\partial \theta_j} \log p(A_t \mid X_{1:t}, A_{1:t-1}, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} R_{t'} \mid \boldsymbol{\theta} \right].$$

By definition,

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) &= \sum_{t=1}^{T-1} \sum_{x_{1:t}} \sum_{a_{1:t}} \sum_{r_{t+1:T}} p(x_{1:t}, a_{1:t}, r_{t+1:T} \mid \boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} \log p(a_t \mid x_{1:t}, a_{1:t-1}, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} r_{t'} \\ &= \sum_{t=1}^{T-1} \sum_{x_{1:t}} \sum_{a_{1:t}} p(x_{1:t}, a_{1:t} \mid \boldsymbol{\theta}) \frac{\partial}{\partial \theta_j} \log p(a_t \mid x_{1:t}, a_{1:t-1}, \boldsymbol{\theta}) \sum_{r_{t+1:T}} p(r_{t+1:T} \mid x_{1:t}, a_{1:t}, \boldsymbol{\theta}) \sum_{t'=t+1}^{T} r_{t'} \\ &= \mathbb{E}\left[\sum_{t=1}^{T-1} \frac{\partial}{\partial \theta_j} \log p(A_t \mid X_{1:t}, A_{1:t-1}, \boldsymbol{\theta}) Q_t^{\boldsymbol{\theta}}(X_{1:t}, A_{1:t}) \mid \boldsymbol{\theta} \right],\end{aligned}$$

where the so-called action value function $Q_t^{\boldsymbol{\theta}}$ is given by

$$Q_t^{\boldsymbol{\theta}}(x'_{1:t}, a'_{1:t}) = \sum_{t'=t+1}^{T} \mathbb{E}\left[R_{t'} \mid X_{1:t} = x'_{1:t}, A_{1:t} = a'_{1:t}, \boldsymbol{\theta} \right].$$

Intuitively, $Q_t^{\boldsymbol{\theta}}$ also gives the expected return for following the policy parameterized by $\boldsymbol{\theta}$ after a particular history of observations and actions.

Therefore, another expression for the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ of the expected return J at $\boldsymbol{\theta}$ is given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(A_t \mid X_{1:t}, A_{1:t-1}, \boldsymbol{\theta}) Q_t^{\boldsymbol{\theta}}(X_{1:t}, A_{1:t}) \mid \boldsymbol{\theta} \right].$$

However, using this expression to obtain an estimate to the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ typically requires approximating the unknown action value function $Q_t^{\boldsymbol{\theta}}$. Such approximation will generally introduce bias (convergence to an undesired expected value) in the corresponding estimate to $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. Action value function approximation for partially observable environments requires extensions to the methods presented in the previous sections.

The advantage function $A_t^{\boldsymbol{\theta}}$ is defined by $A_t^{\boldsymbol{\theta}}(x_{1:t}, a_{1:t}) = Q_t^{\boldsymbol{\theta}}(x_{1:t}, a_{1:t}) - V_t^{\boldsymbol{\theta}}(x_{1:t}, a_{1:t-1})$, for all $x_{1:t}$ and $a_{1:t}$. By combining the previous results, yet another expression for the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ of the expected return J at $\boldsymbol{\theta}$ is given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log p(A_t \mid X_{1:t}, A_{1:t-1}, \boldsymbol{\theta}) A_t^{\boldsymbol{\theta}}(X_{1:t}, A_{1:t}) \mid \boldsymbol{\theta}\right].$$

Using this expression to obtain an estimate to the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ typically requires approximating the unknown advantage function $A_t^{\boldsymbol{\theta}}$. Such approximation also generally introduces a bias in the corresponding estimate to $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. However, in practice, such estimate may be more efficient than the alternatives presented so far.

12 Tabular Bayesian Reinforcement Learning*

For an introduction to Bayesian statistics and the probability notation employed in this section, see the machine learning notes by the same author.

Recall that a finite Markov decision process is defined by a finite set of states $\mathcal{S} = \{1, \dots, |\mathcal{S}|\}$, a finite set of actions $\mathcal{A} = \{1, \dots, |\mathcal{A}|\}$, and the one-step dynamics model of an environment. For every state s, action a, and state s', the one-step dynamics model defines the probability $\mathcal{P}_{ss'}^a$ of transitioning to state s' given action a in state s and the expected reward $\mathcal{R}_{ss'}^a$ upon transitioning to state s' given action a in state s.

Consider a finite set of rewards $\mathcal{R}$. Furthermore, suppose that every reward is independent of its corresponding state given the state and action that precede it. If we let $\mathcal{Q}_{s,r}^a$ denote the probability of reward r given action a in state s, then $\mathcal{R}_{ss'}^a = \sum_r r\mathcal{Q}_{s,r}^a$. Therefore, $\mathcal{R}_{ss'}^a$ can be easily derived given $\mathcal{Q}_{s,r}^a$ for every r.

A history $h_t = s_0, a_0, r_1, s_1, \dots, s_{t-1}, a_{t-1}, r_t, s_t$ summarizes an interaction between an agent and an environment up to time step t. Let $\mathcal{H}$ denote an infinite set that contains every history of every length. An adaptive policy $\tilde{\pi} : \mathcal{H} \times \mathcal{A} \to [0, 1]$ defines the probability $\tilde{\pi}(h, a)$ of every action a given every history h.

The Bayesian reinforcement learning method presented in this section represents its knowledge about an environment by a probability distribution over (one-step dynamics) models and uses this knowledge to find an optimal adaptive policy.

Let $\boldsymbol{\theta}^{(s,a)} \in [0, 1]^{|\mathcal{S}|}$ denote a vector such that $\theta_{s'}^{(s,a)} = \mathcal{P}_{ss'}^a$ for every state s, action a, and state s'. Furthermore, let $\boldsymbol{\theta}$ denote a vector that concatenates every vector in $\{\boldsymbol{\theta}^{(s,a)} \mid (s, a) \in \mathcal{S} \times \mathcal{A}\}$.

Similarly, let $\boldsymbol{\mu}^{(s,a)} \in [0, 1]^{|\mathcal{R}|}$ denote a vector such that $\mu_{i(r)}^{(s,a)} = \mathcal{Q}_{s,r}^a$ for every state s, action a, and reward r, where $i : \mathcal{R} \to \{1, \dots, |\mathcal{R}|\}$ maps each reward to a distinct identifier. Finally, let $\boldsymbol{\mu}$ denote a vector that concatenates every vector in $\{\boldsymbol{\mu}^{(s,a)} \mid (s, a) \in \mathcal{S} \times \mathcal{A}\}$.

Considering our assumptions, the probability $p(h_t \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi})$ of history h_t given the transition parameters $\boldsymbol{\theta}$, the reward parameters $\boldsymbol{\mu}$, and the adaptive policy $\tilde{\pi}$ is given by

$$p(h_t \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) = p(s_0) \prod_{t'=1}^{t} p(a_{t'-1} \mid h_{t'-1}, \tilde{\pi}) p(r_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\mu}) p(s_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\theta}).$$

Note that if a history $h_{t+1} = h_t, a_t, r_{t+1}, s_{t+1}$ extends a history h_t by action a_t, reward r_{t+1}, and state s_{t+1},

$$p(h_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) = p(h_t, a_t, r_{t+1}, s_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) = p(h_t \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) p(a_t \mid h_t, \tilde{\pi}) p(r_{t+1} \mid s_t, a_t, \boldsymbol{\mu}) p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}).$$

Assuming that transition parameters, reward parameters, and adaptive policy are chosen independently,

$$p(h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) = p(\boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) p(h_t \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) = p(\boldsymbol{\theta}) p(\boldsymbol{\mu}) p(\tilde{\pi}) p(h_t \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}).$$

Assuming independence between parameters of distinct pairs of states and actions,

$$p(\boldsymbol{\theta}) = \prod_{(s,a)} p(\boldsymbol{\theta}^{(s,a)}) \qquad\qquad p(\boldsymbol{\mu}) = \prod_{(s,a)} p(\boldsymbol{\mu}^{(s,a)}).$$

For every state s and action a, a convenient (conjugate) prior for the transition parameters $\boldsymbol{\theta}^{(s,a)}$ is the Dirichlet joint probability density function. In that case, the prior density $p(\boldsymbol{\theta}^{(s,a)})$ is given by

$$p(\boldsymbol{\theta}^{(s,a)}) = \text{Dirichlet}(\boldsymbol{\theta}^{(s,a)} \mid \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{s'} \left(\theta^{(s,a)}_{s'}\right)^{\alpha_{s'}-1}$$

for every $\boldsymbol{\theta}^{(s,a)}$ in the $|\mathcal{S}|$-dimensional probability simplex, where B is the generalized Beta function. The vector $\boldsymbol{\alpha} \in \mathbb{R}^{|\mathcal{S}|}_{>0}$ contains hyperparameters (pseudo-counts). For instance, we may let $\alpha_{s'} = 1/|\mathcal{S}|$ for every state s'.

Similarly, for every state s and action a, let the prior density $p(\boldsymbol{\mu}^{(s,a)})$ for reward parameters $\boldsymbol{\mu}^{(s,a)}$ be given by

$$p(\boldsymbol{\mu}^{(s,a)}) = \text{Dirichlet}(\boldsymbol{\mu}^{(s,a)} \mid \boldsymbol{\alpha'}) = \frac{1}{B(\boldsymbol{\alpha'})} \prod_{r} \left(\mu^{(s,a)}_{i(r)}\right)^{{\alpha'}_{i(r)}-1}$$

for every $\boldsymbol{\mu}^{(s,a)}$ in the $|\mathcal{R}|$-dimensional probability simplex and hyperparameter vector $\boldsymbol{\alpha'} \in \mathbb{R}^{|\mathcal{R}|}_{>0}$. For instance, we may let ${\alpha'}_{i(r)} = 1/|\mathcal{R}|$ for every reward r.

The corresponding posterior density $p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t, \tilde{\pi})$ for model parameters $(\boldsymbol{\theta}, \boldsymbol{\mu})$ is given by

$$p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t, \tilde{\pi}) = p(h_t, \tilde{\pi})^{-1} p(\boldsymbol{\theta}) p(\boldsymbol{\mu}) p(\tilde{\pi}) p(s_0) \prod_{t'=1}^{t} p(a_{t'-1} \mid h_{t'-1}, \tilde{\pi}) p(r_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\mu}) p(s_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\theta}).$$

By eliminating constants with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$ and reorganizing terms,

$$p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t, \tilde{\pi}) \propto_{\boldsymbol{\theta}, \boldsymbol{\mu}} \left[p(\boldsymbol{\theta}) \prod_{t'=1}^{t} p(s_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\theta}) \right] \left[p(\boldsymbol{\mu}) \prod_{t'=1}^{t} p(r_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\mu}) \right].$$

The first term on the right side of the proportionality statement above is given by

$$p(\boldsymbol{\theta}) \prod_{t'=1}^{t} p(s_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\theta}) = \left[\prod_{(s,a)} \text{Dirichlet}(\boldsymbol{\theta}^{(s,a)} \mid \boldsymbol{\alpha}) \right] \left[\prod_{t'=1}^{t} \theta^{(s_{t'-1}, a_{t'-1})}_{s_{t'}} \right].$$

For every state s and action a, let $\mathbf{M}^{(s,a)} \in \mathbb{N}^{|\mathcal{S}|}$ denote a vector such that $M^{(s,a)}_{s'}$ is the number of times in the history h_t that action a in state s resulted in state s'. The previous equation can then be rewritten as

$$p(\boldsymbol{\theta}) \prod_{t'=1}^{t} p(s_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\theta}) = \left[\prod_{(s,a)} \frac{1}{B(\boldsymbol{\alpha})} \prod_{s'} \left(\theta^{(s,a)}_{s'}\right)^{\alpha_{s'}-1} \right] \left[\prod_{(s,a)} \prod_{s'} \left[\theta^{(s,a)}_{s'}\right]^{M^{(s,a)}_{s'}} \right].$$

By reorganizing products,

$$p(\boldsymbol{\theta}) \prod_{t'=1}^{t} p(s_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\theta}) = \left[\prod_{(s,a)} \frac{1}{B(\boldsymbol{\alpha})} \right] \left[\prod_{(s,a)} \prod_{s'} \left(\theta^{(s,a)}_{s'}\right)^{\alpha_{s'} + M^{(s,a)}_{s'} - 1} \right].$$

From the definition of a Dirichlet probability density function,

$$p(\boldsymbol{\theta}) \prod_{t'=1}^{t} p(s_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\theta}) = \left[\prod_{(s,a)} \frac{1}{B(\boldsymbol{\alpha})} \right] \left[\prod_{(s,a)} \text{Dirichlet}(\boldsymbol{\theta}^{(s,a)} \mid \boldsymbol{\alpha} + \mathbf{M}^{(s,a)}) B(\boldsymbol{\alpha} + \mathbf{M}^{(s,a)}) \right].$$

By eliminating constants with respect to $\boldsymbol{\theta}$,

$$p(\boldsymbol{\theta}) \prod_{t'=1}^{t} p(s_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\theta}) \propto_{\boldsymbol{\theta}} \prod_{(s,a)} \text{Dirichlet}(\boldsymbol{\theta}^{(s,a)} \mid \boldsymbol{\alpha} + \mathbf{M}^{(s,a)}).$$

Analogously,

$$p(\boldsymbol{\mu}) \prod_{t'=1}^{t} p(r_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\mu}) \propto_{\boldsymbol{\mu}} \prod_{(s,a)} \text{Dirichlet}(\boldsymbol{\mu}^{(s,a)} \mid \boldsymbol{\alpha}' + \mathbf{N}^{(s,a)}),$$

where $\mathbf{N}^{(s,a)} \in \mathbb{N}^{|\mathcal{R}|}$ denotes a vector such that $N_{i(r)}^{(s,a)}$ is the number of times in the history h_t that action a in state s resulted in reward r.

Therefore, the posterior density $p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t, \tilde{\pi})$ for model parameters $(\boldsymbol{\theta}, \boldsymbol{\mu})$ is given by

$$p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t, \tilde{\pi}) \propto_{\boldsymbol{\theta},\boldsymbol{\mu}} \left[\prod_{(s,a)} \text{Dirichlet}(\boldsymbol{\theta}^{(s,a)} \mid \boldsymbol{\alpha} + \mathbf{M}^{(s,a)}) \right] \left[\prod_{(s,a)} \text{Dirichlet}(\boldsymbol{\mu}^{(s,a)} \mid \boldsymbol{\alpha}' + \mathbf{N}^{(s,a)}) \right].$$

Because both sides of the proportionality statement above integrated over $\boldsymbol{\theta}$ and $\boldsymbol{\mu}$ result in one,

$$p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t, \tilde{\pi}) = \left[\prod_{(s,a)} \text{Dirichlet}(\boldsymbol{\theta}^{(s,a)} \mid \boldsymbol{\alpha} + \mathbf{M}^{(s,a)}) \right] \left[\prod_{(s,a)} \text{Dirichlet}(\boldsymbol{\mu}^{(s,a)} \mid \boldsymbol{\alpha}' + \mathbf{N}^{(s,a)}) \right].$$

For every $\boldsymbol{\theta}, \boldsymbol{\mu}$, h_t, and $\tilde{\pi}$, note that $p(\boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi} \mid h_t) = p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t, \tilde{\pi}) p(\tilde{\pi} \mid h_t) = f(\boldsymbol{\theta}, \boldsymbol{\mu}, h_t) g(\tilde{\pi}, h_t)$ for some fixed f and g, which implies $p(\boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi} \mid h_t) = p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) p(\tilde{\pi} \mid h_t)$. Consequently, if $p(\tilde{\pi} \mid h_t) > 0$,

$$p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) = \frac{p(\boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi} \mid h_t)}{p(\tilde{\pi} \mid h_t)} = p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t, \tilde{\pi}).$$

Therefore, given the history, the model parameters are independent of the adaptive policy.

For every $\boldsymbol{\theta}, \boldsymbol{\mu}$, and h_t, also note that $p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) = f(\boldsymbol{\theta}, h_t) g(\boldsymbol{\mu}, h_t)$ for some fixed f and g, which implies $p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) = p(\boldsymbol{\theta} \mid h_t) p(\boldsymbol{\mu} \mid h_t)$. Consequently, if $p(\boldsymbol{\mu} \mid h_t) > 0$,

$$p(\boldsymbol{\theta} \mid h_t) = \frac{p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t)}{p(\boldsymbol{\mu} \mid h_t)} \propto_{\boldsymbol{\theta}} \prod_{(s,a)} \text{Dirichlet}(\boldsymbol{\theta}^{(s,a)} \mid \boldsymbol{\alpha} + \mathbf{M}^{(s,a)}).$$

Because both sides of the proportionality statement above integrated over $\boldsymbol{\theta}$ clearly result in one,

$$p(\boldsymbol{\theta} \mid h_t) = \prod_{(s,a)} \text{Dirichlet}(\boldsymbol{\theta}^{(s,a)} \mid \boldsymbol{\alpha} + \mathbf{M}^{(s,a)}).$$

Analogously, the posterior density $p(\boldsymbol{\theta}^{(s,a)} \mid h_t)$ is given by

$$p(\boldsymbol{\theta}^{(s,a)} \mid h_t) = \text{Dirichlet}(\boldsymbol{\theta}^{(s,a)} \mid \boldsymbol{\alpha} + \mathbf{M}^{(s,a)}).$$

Similarly, the posterior density $p(\boldsymbol{\mu}^{(s,a)} \mid h_t)$ is given by

$$p(\boldsymbol{\mu}^{(s,a)} \mid h_t) = \text{Dirichlet}(\boldsymbol{\mu}^{(s,a)} \mid \boldsymbol{\alpha}' + \mathbf{N}^{(s,a)}).$$

The value $V^{\tilde{\pi}}(h_t)$ of an adaptive policy $\tilde{\pi}$ given the history h_t is defined by

$$V^{\tilde{\pi}}(h_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid h_t, \tilde{\pi} \right],$$

where $\gamma \in [0, 1)$ is a hyperparameter that controls the importance of future rewards.

Let $r^* = \max \mathcal{R}$ denote the maximum reward. The value $V^{\tilde{\pi}}(h_t)$ is upper bounded, since

$$V^{\tilde{\pi}}(h_t) \leq \sum_{k=0}^{\infty} \gamma^k r^* = r^* \sum_{k=0}^{\infty} \gamma^k = \frac{r^*}{1 - \gamma}.$$

An adaptive policy $\tilde{\pi}$ is optimal if $V^{\tilde{\pi}}(h) \geq V^{\tilde{\pi}'}(h)$ for every adaptive policy $\tilde{\pi}'$ and history h. The remainder of this section is concerned with finding an optimal adaptive policy.

By the law of total expectation,

$$V^{\tilde{\pi}}(h_t) = \sum_{a_t} \sum_{r_{t+1}} \sum_{s_{t+1}} p(a_t, r_{t+1}, s_{t+1} \mid h_t, \tilde{\pi}) \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid h_t, a_t, r_{t+1}, s_{t+1}, \tilde{\pi} \right].$$

By reordering terms and using the fact that $p(a_t \mid h_t, \tilde{\pi}) = \tilde{\pi}(h_t, a_t)$,

$$V^{\tilde{\pi}}(h_t) = \sum_{a_t} \tilde{\pi}(h_t, a_t) \sum_{r_{t+1}} \sum_{s_{t+1}} p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi}) \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid h_{t+1}, \tilde{\pi} \right],$$

where history $h_{t+1} = h_t, a_t, r_{t+1}, s_{t+1}$ denotes history h_t extended by action a_t, reward r_{t+1}, and state s_{t+1}.

Because the reward r_{t+1} is part of the history h_{t+1},

$$V^{\tilde{\pi}}(h_t) = \sum_{a_t} \tilde{\pi}(h_t, a_t) \sum_{r_{t+1}} \sum_{s_{t+1}} p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi}) \left[r_{t+1} + \gamma \mathbb{E}\left[\sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k+1} \mid h_{t+1}, \tilde{\pi} \right] \right].$$

By the definition of the value $V^{\tilde{\pi}}(h_{t+1})$ of an adaptive policy $\tilde{\pi}$ given the history h_{t+1},

$$V^{\tilde{\pi}}(h_t) = \sum_{a_t} \tilde{\pi}(h_t, a_t) \sum_{r_{t+1}} \sum_{s_{t+1}} p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi}) \left[r_{t+1} + \gamma V^{\tilde{\pi}}(h_{t+1}) \right].$$

By marginalization, the probability $p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi})$ is given by

$$p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi}) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(r_{t+1}, s_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi}) p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid a_t, h_t, \tilde{\pi}) \, d\boldsymbol{\theta} \, d\boldsymbol{\mu}.$$

In what follows, we will examine the two terms inside these integrals.

First, consider the probability $p(r_{t+1}, s_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi})$, which is given by

$$p(r_{t+1}, s_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi}) = \frac{p(r_{t+1}, s_{t+1}, \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi})}{p(\boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi})} = \frac{p(h_{t+1}, \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi})}{p(\boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi})} = \frac{p(h_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) p(\boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi})}{p(\boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi})},$$

where history $h_{t+1} = h_t, a_t, r_{t+1}, s_{t+1}$ extends history h_t by action a_t, reward r_{t+1}, and state s_{t+1}.

By the recursive relationship between probabilities of histories,

$$p(r_{t+1}, s_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi}) = \frac{p(h_t \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) p(a_t \mid h_t, \tilde{\pi}) p(r_{t+1} \mid s_t, a_t, \boldsymbol{\mu}) p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}) p(\boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi})}{p(\boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi})}.$$

Note that $p(r_{t+1}, s_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi}) = f(r_{t+1}, \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi}) g(s_{t+1}, \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi})$ for some fixed f and g, which implies that $p(r_{t+1}, s_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi}) = p(r_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi}) p(s_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi})$. Consequently, if $p(r_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi}) > 0$,

$$p(s_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi}) = \frac{p(r_{t+1}, s_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi})}{p(r_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi})} \propto_{s_{t+1}} p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}).$$

Because both sides of the proportionality statement above summed over s_{t+1} clearly result in one,

$$p(s_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi}) = p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}).$$

Analogously,

$$p(r_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi}) = p(r_{t+1} \mid s_t, a_t, \boldsymbol{\mu}).$$

Consequently,

$$p(r_{t+1}, s_{t+1} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, h_t, a_t, \tilde{\pi}) = p(r_{t+1} \mid s_t, a_t, \boldsymbol{\mu}) p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}).$$

Second, consider the probability $p(\boldsymbol{\theta}, \boldsymbol{\mu}, a_t \mid h_t, \tilde{\pi})$, which is given by

$$p(\boldsymbol{\theta}, \boldsymbol{\mu}, a_t \mid h_t, \tilde{\pi}) = \frac{p(\boldsymbol{\theta}, \boldsymbol{\mu}, a_t, h_t, \tilde{\pi})}{p(h_t, \tilde{\pi})} = p(h_t, \tilde{\pi})^{-1} \sum_{r_{t+1}} \sum_{s_{t+1}} p(\boldsymbol{\theta}, \boldsymbol{\mu}, a_t, r_{t+1}, s_{t+1}, h_t, \tilde{\pi}).$$

By the recursive relationship between probabilities of histories,

$$p(\boldsymbol{\theta}, \boldsymbol{\mu}, a_t \mid h_t, \tilde{\pi}) = p(h_t, \tilde{\pi})^{-1} \sum_{r_{t+1}} \sum_{s_{t+1}} p(h_t \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) p(a_t \mid h_t, \tilde{\pi}) p(r_{t+1} \mid s_t, a_t, \boldsymbol{\mu}) p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}) p(\boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}).$$

By moving constants outside summations,

$$p(\boldsymbol{\theta}, \boldsymbol{\mu}, a_t \mid h_t, \tilde{\pi}) = p(h_t, \tilde{\pi})^{-1} p(h_t \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) p(a_t \mid h_t, \tilde{\pi}) p(\boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) \sum_{r_{t+1}} p(r_{t+1} \mid s_t, a_t, \boldsymbol{\mu}) \sum_{s_{t+1}} p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}).$$

Because each summation clearly results in one,

$$p(\boldsymbol{\theta}, \boldsymbol{\mu}, a_t \mid h_t, \tilde{\pi}) = p(h_t, \tilde{\pi})^{-1} p(h_t \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}) p(a_t \mid h_t, \tilde{\pi}) p(\boldsymbol{\theta}, \boldsymbol{\mu}, \tilde{\pi}).$$

Note that $p(\boldsymbol{\theta}, \boldsymbol{\mu}, a_t \mid h_t, \tilde{\pi}) = f(\boldsymbol{\theta}, \boldsymbol{\mu}, h_t, \tilde{\pi}) g(a_t, h_t, \tilde{\pi})$ for some fixed f and g, which implies $p(\boldsymbol{\theta}, \boldsymbol{\mu}, a_t \mid h_t, \tilde{\pi}) = p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t, \tilde{\pi}) p(a_t \mid h_t, \tilde{\pi})$. Consequently, if $p(a_t \mid h_t, \tilde{\pi}) > 0$ and $p(\tilde{\pi} \mid h_t) > 0$,

$$p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid a_t, h_t, \tilde{\pi}) = \frac{p(\boldsymbol{\theta}, \boldsymbol{\mu}, a_t \mid h_t, \tilde{\pi})}{p(a_t \mid h_t, \tilde{\pi})} = p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t, \tilde{\pi}) = p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) = p(\boldsymbol{\theta} \mid h_t) p(\boldsymbol{\mu} \mid h_t),$$

where the last two equalities follow from previous results.

Finally, the previous results can be combined to show that the probability $p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi})$ is given by

$$p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi}) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(r_{t+1} \mid s_t, a_t, \boldsymbol{\mu}) p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid h_t) p(\boldsymbol{\mu} \mid h_t) \, d\boldsymbol{\theta} \, d\boldsymbol{\mu}.$$

By moving constants outside integrals,

$$p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi}) = \left[\int_{\boldsymbol{\mu}} p(r_{t+1} \mid s_t, a_t, \boldsymbol{\mu}) p(\boldsymbol{\mu} \mid h_t) \, d\boldsymbol{\mu} \right] \left[\int_{\boldsymbol{\theta}} p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid h_t) \, d\boldsymbol{\theta} \right].$$

Note that $p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi}) = f(r_{t+1}, h_t, a_t, \tilde{\pi}) g(s_{t+1}, h_t, a_t, \tilde{\pi})$ for some fixed f and g, which implies $p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi}) = p(r_{t+1} \mid h_t, a_t, \tilde{\pi}) p(s_{t+1} \mid h_t, a_t, \tilde{\pi})$. Consequently, if $p(r_{t+1} \mid h_t, a_t, \tilde{\pi}) > 0$,

$$p(s_{t+1} \mid h_t, a_t, \tilde{\pi}) = \frac{p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi})}{p(r_{t+1} \mid h_t, a_t, \tilde{\pi})} \propto_{s_{t+1}} \int_{\boldsymbol{\theta}} p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid h_t) \, d\boldsymbol{\theta}.$$

Note that the result of summing the term on the right of the proportionality statement above over s_{t+1} is

$$\sum_{s_{t+1}} \int_{\boldsymbol{\theta}} p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid h_t) \, d\boldsymbol{\theta} = \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid h_t) \sum_{s_{t+1}} p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}) \, d\boldsymbol{\theta} = 1.$$

Because both sides of the proportionality statement above summed over s_{t+1} clearly result in one,

$$p(s_{t+1} \mid h_t, a_t, \tilde{\pi}) = \int_{\boldsymbol{\theta}} p(s_{t+1} \mid s_t, a_t, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid h_t) \, d\boldsymbol{\theta}.$$

Analogously,

$$p(r_{t+1} \mid h_t, a_t, \tilde{\pi}) = \int_{\boldsymbol{\mu}} p(r_{t+1} \mid s_t, a_t, \boldsymbol{\mu}) p(\boldsymbol{\mu} \mid h_t) \, d\boldsymbol{\mu}.$$

Considering the definition of the transition parameters $\boldsymbol{\theta}$ and the law of total expectation,

$$p(s_{t+1} \mid h_t, a_t, \tilde{\pi}) = \int_{\boldsymbol{\theta}} \theta_{s_{t+1}}^{(s_t, a_t)} p(\boldsymbol{\theta} \mid h_t) \, d\boldsymbol{\theta} = \int_{\boldsymbol{\theta}^{(s_t, a_t)}} \theta_{s_{t+1}}^{(s_t, a_t)} p(\boldsymbol{\theta}^{(s_t, a_t)} \mid h_t) \, d\boldsymbol{\theta}^{(s_t, a_t)}.$$

By recalling that the posterior over transition parameters is given by a Dirichlet distribution,

$$p(s_{t+1} \mid h_t, a_t, \tilde{\pi}) = \int_{\boldsymbol{\theta}^{(s_t, a_t)}} \theta_{s_{t+1}}^{(s_t, a_t)} \, \text{Dirichlet}(\boldsymbol{\theta}^{(s_t, a_t)} \mid \boldsymbol{\alpha} + \mathbf{M}^{(s_t, a_t)}) \, d\boldsymbol{\theta}^{(s_t, a_t)}.$$

Because the probability $p(s_{t+1} \mid h_t, a_t, \tilde{\pi})$ is the expected value of a (jointly-)Dirichlet variable,

$$p(s_{t+1} \mid h_t, a_t, \tilde{\pi}) = \frac{\alpha_{s_{t+1}} + M^{(s_t,a_t)}_{s_{t+1}}}{\sum_{s'} \alpha_{s'} + M^{(s_t,a_t)}_{s'}},$$

where $M^{(s,a)}_{s'}$ is the number of times in the history h_t that action a in state s resulted in state s'.

Analogously,

$$p(r_{t+1} \mid h_t, a_t, \tilde{\pi}) = \frac{\alpha'_{i(r_{t+1})} + N^{(s_t,a_t)}_{i(r_{t+1})}}{\sum_{r'} \alpha'_{i(r')} + N^{(s_t,a_t)}_{i(r')}},$$

where $i : \mathcal{R} \to \{1, \ldots, |\mathcal{R}|\}$ maps each reward to a distinct identifier, and $N^{(s,a)}_{i(r)}$ is the number of times in the history h_t that action a in state s resulted in reward r.

Therefore, the probability $p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi})$ is given by

$$p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi}) = \left[\frac{\alpha'_{i(r_{t+1})} + N^{(s_t,a_t)}_{i(r_{t+1})}}{\sum_{r'} \alpha'_{i(r')} + N^{(s_t,a_t)}_{i(r')}} \right] \left[\frac{\alpha_{s_{t+1}} + M^{(s_t,a_t)}_{s_{t+1}}}{\sum_{s'} \alpha_{s'} + M^{(s_t,a_t)}_{s'}} \right].$$

Recall that the value $V^{\tilde{\pi}}(h_t)$ of an adaptive policy $\tilde{\pi}$ given the history h_t is

$$V^{\tilde{\pi}}(h_t) = \sum_{a_t} \tilde{\pi}(h_t, a_t) \sum_{r_{t+1}} \sum_{s_{t+1}} p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi}) \left[r_{t+1} + \gamma V^{\tilde{\pi}}(h_{t+1}) \right],$$

where the history $h_{t+1} = h_t, a_t, r_{t+1}, s_{t+1}$ extends a history h_t by action a_t, reward r_{t+1}, and state s_{t+1}. In that case, let $\bar{\mathcal{P}}^{a_t}_{h_t h_{t+1}} = p(r_{t+1}, s_{t+1} \mid h_t, a_t, \tilde{\pi})$ and $\bar{\mathcal{R}}^{a_t}_{h_t h_{t+1}} = r_{t+1}$. Alternatively, if history h' is incompatible with history h and action a, let $\bar{\mathcal{P}}^{a}_{hh'} = \bar{\mathcal{R}}^{a}_{hh'} = 0$. In either case, note that $\bar{\mathcal{P}}^{a}_{hh'}$ is independent of the adaptive policy $\tilde{\pi}$.

The value $V^{\tilde{\pi}}(h)$ of an adaptive policy $\tilde{\pi}$ given the history h can then be rewritten as

$$V^{\tilde{\pi}}(h) = \sum_{a} \tilde{\pi}(h, a) \sum_{h'} \bar{\mathcal{P}}^{a}_{hh'} \left[\bar{\mathcal{R}}^{a}_{hh'} + \gamma V^{\tilde{\pi}}(h') \right].$$

Note that the equation above is the Bellman equation for a policy $\tilde{\pi}$ of an infinite Markov decision process whose state space is the set of histories $\mathcal{H}$. Most importantly, the one-step dynamics model of this so-called Bayes-adaptive Markov decision process is entirely known. Unfortunately, finding an optimal policy for a known infinite Markov decision process is not generally feasible.

Suppose that $V^{\tilde{\pi}}(h_t) = 0$ for every adaptive policy $\tilde{\pi}$ and history h_t whenever $t > T$. In that case, policy evaluation requires computing only a finite number of values. However, because there are $|\mathcal{S}| \left(|\mathcal{A}||\mathcal{R}||\mathcal{S}|\right)^t$ histories with t time steps, the number of histories with up to T time steps is given by

$$\sum_{t=0}^{T} |\mathcal{S}| \left(|\mathcal{A}||\mathcal{R}||\mathcal{S}|\right)^t = |\mathcal{S}| \left(\frac{1 - \left(|\mathcal{A}||\mathcal{R}||\mathcal{S}|\right)^{T+1}}{1 - |\mathcal{A}||\mathcal{R}||\mathcal{S}|} \right).$$

Therefore, although dynamic programming algorithms could be used to find optimal adaptive policies given a (time-limited) Bayes-adaptive Markov decision process, the number of states involved is usually prohibitively large.

13 Tabular Bayesian Reinforcement Learning Algorithms*

The Bayesian reinforcement learning methods presented in this section represent their knowledge about an environment by a probability distribution over models and use this knowledge to search for adequate actions.

A Bayes-adaptive control algorithm (Alg. 14) employs an iterative procedure that starts with an arbitrary adaptive policy (Line 1) and searches for a better adaptive policy after each interaction with the environment (Line 5). If this search were able to find an adaptive policy $\tilde{\pi}$ such that $V^{\tilde{\pi}}(h_t) \geq V^{\tilde{\pi}'}(h_t)$ for every adaptive policy $\tilde{\pi}'$ and time step t, then this control algorithm would be optimal. Unfortunately, that is not generally feasible.

As a concrete example of a Bayes-adaptive control algorithm, this section presents Bayes-adaptive recurrent policy gradients, which is inspired by Bayes-adaptive planning with function approximation [6].

Algorithm 14 Bayes-adaptive control algorithm

```
Input: prior over (one-step dynamics) models, number of interactions $T$, discount factor $\gamma$
$\tilde{\pi} \leftarrow$ arbitrary adaptive policy
$s_0 \leftarrow$ initial state given by the environment
$h_0 \leftarrow s_0$
for each $t \in \{0, \dots, T-1\}$ do
    $\tilde{\pi} \leftarrow$ adaptive policy that potentially improves on $\tilde{\pi}$ given the prior and the history $h_t$
    $a_t \leftarrow$ action drawn from the adaptive policy $\tilde{\pi}$ for the history $h_t$
    $r_{t+1}, s_{t+1} \leftarrow$ reward and state given by the environment for action $a_t$
    $h_{t+1} \leftarrow (h_t, (a_t, r_{t+1}, s_{t+1}))$
end for
```

In order to employ Bayes-adaptive recurrent policy gradients, suppose that the optimal adaptive policy belongs to a parameterized family of functions $\{\tilde{\pi}^{(\boldsymbol{\psi})} \mid \boldsymbol{\psi} \in \Psi\}$, where Ψ is a set of parameters. In that case, $\tilde{\pi}^{(\boldsymbol{\psi})}(h, a)$ is the probability of action a given the history h and the adaptive policy parameters $\boldsymbol{\psi}$. For instance, the adaptive policy $\tilde{\pi}^{(\boldsymbol{\psi})}$ may correspond to a recurrent neural network parameterized by $\boldsymbol{\psi}$.

Whenever convenient, we let the adaptive policy parameters $\boldsymbol{\psi}$ denote the adaptive policy $\tilde{\pi}^{(\boldsymbol{\psi})}$. For instance, the value $V^{\boldsymbol{\psi}}(h_t)$ of the adaptive policy (parameters) $\boldsymbol{\psi}$ given the history h_t is defined by

$$V^{\boldsymbol{\psi}}(h_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid h_t, \boldsymbol{\psi}\right],$$

where $\gamma \in [0, 1)$ is a hyperparameter that controls the importance of future rewards.

The value $V_H^{\boldsymbol{\psi}}(h_t)$ of the adaptive policy $\boldsymbol{\psi}$ given the history h_t for the horizon H is defined by

$$V_H^{\boldsymbol{\psi}}(h_t) = \mathbb{E}\left[\sum_{k=0}^{H-1} \gamma^k R_{t+k+1} \mid h_t, \boldsymbol{\psi}\right],$$

such that $\lim_{H \to \infty} V_H^{\boldsymbol{\psi}}(h_t) = V^{\boldsymbol{\psi}}(h_t)$. By the law of total expectation,

$$V_H^{\boldsymbol{\psi}}(h_t) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t, \boldsymbol{\psi}) \mathbb{E}\left[\sum_{k=0}^{H-1} \gamma^k R_{t+k+1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}\right] \, d\boldsymbol{\theta} \, d\boldsymbol{\mu}.$$

Because the model parameters are independent of the adaptive policy given the history,

$$V_H^{\boldsymbol{\psi}}(h_t) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) \mathbb{E}\left[\sum_{k=0}^{H-1} \gamma^k R_{t+k+1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}\right] \, d\boldsymbol{\theta} \, d\boldsymbol{\mu}.$$

By letting $h_{t+H} = h_t, a_t, r_{t+1}, s_{t+1}, \dots, a_{t+H-1}, r_{t+H}, s_{t+H}$ denote history h_t extended by H time steps,

$$V_H^{\boldsymbol{\psi}}(h_t) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) \sum_{h_{t+H}} p(h_{t+H} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) \sum_{k=0}^{H-1} \gamma^k r_{t+k+1} \, d\boldsymbol{\theta} \, d\boldsymbol{\mu}.$$

The equation above can also be rewritten as

$$V_H^{\boldsymbol{\psi}}(h_t) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) \sum_{k=0}^{H-1} \gamma^k \sum_{h_{t+H}} p(h_{t+H} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) r_{t+k+1} \, d\boldsymbol{\theta} \, d\boldsymbol{\mu}.$$

If we assume that $V_H^{\boldsymbol{\psi}}(h_t)$ is differentiable with respect to ψ_j for all j, it is possible to use gradient-based methods (such as gradient ascent) to attempt to find an optimal adaptive policy for the horizon H. Using the Leibniz integral rule, the partial derivative $\frac{\partial}{\partial \psi_j} V_H^{\boldsymbol{\psi}}(h_t)$ of $V_H^{\boldsymbol{\psi}}(h_t)$ with respect to ψ_j at $\boldsymbol{\psi}$ is given by

$$\frac{\partial}{\partial \psi_j} V_H^{\boldsymbol{\psi}}(h_t) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) \sum_{k=0}^{H-1} \gamma^k \sum_{h_{t+H}} \frac{\partial}{\partial \psi_j} p(h_{t+H} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) r_{t+k+1} \, d\boldsymbol{\theta} \, d\boldsymbol{\mu}.$$

If we further assume that $p(h_{t+H} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) > 0$ for every h_{t+H}, h_t, $\boldsymbol{\theta}, \boldsymbol{\mu}$, and $\boldsymbol{\psi}$, the so-called likelihood ratio trick uses the fact that

$$\frac{\partial}{\partial \psi_j} p(h_{t+H} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) = p(h_{t+H} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) \frac{\partial}{\partial \psi_j} \log p(h_{t+H} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}).$$

By the recursive relationship between probabilities of histories,

$$p(h_{t+H} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) = p(h_t \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) \prod_{t'=t+1}^{t+H} p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}) p(r_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\mu}) p(s_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\theta}).$$

Because history h_{t+H} extends history h_t, note that $p(h_{t+H}, h_t \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) = p(h_{t+H} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})$. Furthermore,

$$p(h_{t+H} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) = \frac{p(h_{t+H} \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})}{p(h_t \mid \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})} = \prod_{t'=t+1}^{t+H} p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}) p(r_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\mu}) p(s_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\theta}).$$

By taking the logarithm of both sides,

$$\log p(h_{t+H} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) = \sum_{t'=t+1}^{t+H} \log p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}) + \log p(r_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\mu}) + \log p(s_{t'} \mid s_{t'-1}, a_{t'-1}, \boldsymbol{\theta}).$$

Because only the action probabilities depend on the adaptive policy parameters,

$$\frac{\partial}{\partial \psi_j} \log p(h_{t+H} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) = \sum_{t'=t+1}^{t+H} \frac{\partial}{\partial \psi_j} \log p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}).$$

Finally, the partial derivative $\frac{\partial}{\partial \psi_j} V_H^{\boldsymbol{\psi}}(h_t)$ may be rewritten using the likelihood ratio trick, which gives

$$\frac{\partial}{\partial \psi_j} V_H^{\boldsymbol{\psi}}(h_t) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) \sum_{k=0}^{H-1} \gamma^k \sum_{h_{t+H}} p(h_{t+H} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) \sum_{t'=t+1}^{t+H} r_{t+k+1} \frac{\partial}{\partial \psi_j} \log p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}) \, d\boldsymbol{\theta} \, d\boldsymbol{\mu}.$$

The expectation over histories on the right side of the equation above can be rewritten such that

$$\frac{\partial}{\partial \psi_j} V_H^{\boldsymbol{\psi}}(h_t) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) \sum_{k=0}^{H-1} \gamma^k \sum_{t'=t+1}^{t+H} \mathbb{E}\left[R_{t+k+1} \frac{\partial}{\partial \psi_j} \log p(A_{t'-1} \mid H_{t'-1}, \boldsymbol{\psi}) \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi} \right] \, d\boldsymbol{\theta} \, d\boldsymbol{\mu}.$$

Let E denote one of the terms inside the innermost summation above for which $t' \geq t + k + 2$. In that case, a history $h_{t'-1}$ that extends history h_t will contain the reward r_{t+k+1}, which implies

$$E = \sum_{h_{t'-1}} \sum_{a_{t'-1}} p(a_{t'-1}, h_{t'-1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) r_{t+k+1} \frac{\partial}{\partial \psi_j} \log p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}).$$

The equation above can be rewritten as

$$E = \sum_{h_{t'-1}} \sum_{a_{t'-1}} p(h_{t'-1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) p(a_{t'-1} \mid h_{t'-1}, h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) r_{t+k+1} \frac{\partial}{\partial \psi_j} \log p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}).$$

Because $p(a_{t'-1} \mid h_{t'-1}, h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) = p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) = p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi})$,

$$E = \sum_{h_{t'-1}} \sum_{a_{t'-1}} p(h_{t'-1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}) r_{t+k+1} \frac{\partial}{\partial \psi_j} \log p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}).$$

By reversing the likelihood ratio trick,

$$E = \sum_{h_{t'-1}} \sum_{a_{t'-1}} p(h_{t'-1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) r_{t+k+1} \frac{\partial}{\partial \psi_j} p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}).$$

By pushing constants outside the innermost summation and using the fact that $\frac{\partial}{\partial \psi_j} 1 = 0$,

$$E = \sum_{h_{t'-1}} p(h_{t'-1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) r_{t+k+1} \frac{\partial}{\partial \psi_j} \sum_{a_{t'-1}} p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}) = 0.$$

Therefore, each term E may be discarded from the previous expression for $\frac{\partial}{\partial \psi_j} V_H^{\boldsymbol{\psi}}(h_t)$, which gives

$$\frac{\partial}{\partial \psi_j} V_H^{\boldsymbol{\psi}}(h_t) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) \mathbb{E}\left[\sum_{k=0}^{H-1} \gamma^k R_{t+k+1} \sum_{t'=t+1}^{t+k+1} \frac{\partial}{\partial \psi_j} \log p(A_{t'-1} \mid H_{t'-1}, \boldsymbol{\psi}) \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi} \right] d\boldsymbol{\theta}\ d\boldsymbol{\mu}.$$

By reordering the summations, the expression above can be conveniently rewritten as

$$\frac{\partial}{\partial \psi_j} V_H^{\boldsymbol{\psi}}(h_t) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) \mathbb{E}\left[\sum_{t'=t+1}^{t+H} \frac{\partial}{\partial \psi_j} \log p(A_{t'-1} \mid H_{t'-1}, \boldsymbol{\psi}) \sum_{k=t'-t-1}^{H-1} \gamma^k R_{t+k+1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi} \right] d\boldsymbol{\theta}\ d\boldsymbol{\mu}.$$

Finally, the gradient $\nabla_{\boldsymbol{\psi}} V_H^{\boldsymbol{\psi}}(h_t)$ with respect to $\boldsymbol{\psi}$ of $V_H^{\boldsymbol{\psi}}(h_t)$ at $\boldsymbol{\psi}$ is given by

$$\nabla_{\boldsymbol{\psi}} V_H^{\boldsymbol{\psi}}(h_t) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) \mathbb{E}\left[\sum_{t'=t+1}^{t+H} \nabla_{\boldsymbol{\psi}} \log p(A_{t'-1} \mid H_{t'-1}, \boldsymbol{\psi}) \sum_{k=t'-t-1}^{H-1} \gamma^k R_{t+k+1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi} \right] d\boldsymbol{\theta}\ d\boldsymbol{\mu}.$$

This result has a subtle intuitive interpretation. In gradient ascent, positive expected outcomes across plausible models contribute towards making the probability of a decision higher, while negative expected outcomes across plausible models contribute towards making the probability of a decision lower.

For a given history h_t, consider a sequence $\left(\left(\boldsymbol{\theta}^{(i)}, \boldsymbol{\mu}^{(i)}\right)\right)_{i=1}^{N}$ obtained by sampling model parameters such that $\left(\boldsymbol{\theta}^{(i)}, \boldsymbol{\mu}^{(i)}\right) \sim p(\cdot \mid h_t)$ for every i. For each i, consider also a sequence $\left(h_{t+H}^{(i,j)}\right)_{j=1}^{M}$ obtained by sampling histories such that $h_{t+H}^{(i,j)} \sim p(\cdot \mid h_t, \boldsymbol{\theta}^{(i)}, \boldsymbol{\mu}^{(i)}, \boldsymbol{\psi})$ for every j, and let $h_{t+H}^{(i,j)} = h_t, a_t^{(i,j)}, r_{t+1}^{(i,j)}, s_{t+1}^{(i,j)}, \ldots, a_{t+H-1}^{(i,j)}, r_{t+H}^{(i,j)}, s_{t+H}^{(i,j)}$. A corresponding Monte Carlo estimate of the gradient $\nabla_{\boldsymbol{\psi}} V_H^{\boldsymbol{\psi}}(h_t)$ with respect to $\boldsymbol{\psi}$ of $V_H^{\boldsymbol{\psi}}(h_t)$ at $\boldsymbol{\psi}$ is given by

$$\frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{t'=t+1}^{t+H} \nabla_{\boldsymbol{\psi}} \log p(a_{t'-1}^{(i,j)} \mid h_{t'-1}^{(i,j)}, \boldsymbol{\psi}) \sum_{k=t'-t-1}^{H-1} \gamma^k r_{t+k+1}^{(i,j)}.$$

Bayes-adaptive recurrent policy gradients combines such an estimate with stochastic gradient ascent to search for a better adaptive policy after each interaction with an environment (Alg. 15).

Although the Monte Carlo estimate of the gradient $\nabla_{\boldsymbol{\psi}} V_H^{\boldsymbol{\psi}}(h_t)$ presented above is unbiased, the corresponding estimator may have high variance. A so-called baseline function may be employed in an attempt to reduce this variance by using the fact that the gradient $\nabla_{\boldsymbol{\psi}} V_H^{\boldsymbol{\psi}}(h_t)$ with respect to $\boldsymbol{\psi}$ of $V_H^{\boldsymbol{\psi}}(h_t)$ at $\boldsymbol{\psi}$ is given by

$$\nabla_{\boldsymbol{\psi}} V_H^{\boldsymbol{\psi}}(h_t) = \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) \mathbb{E}\left[\sum_{t'=t+1}^{t+H} \nabla_{\boldsymbol{\psi}} \log p(A_{t'-1} \mid H_{t'-1}, \boldsymbol{\psi}) \left[\left[\sum_{k=t'-t-1}^{H-1} \gamma^k R_{t+k+1} \right] - B^{(h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})}(H_{t'-1}) \right] \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi} \right] d\boldsymbol{\theta}\ d\boldsymbol{\mu},$$

for any choice of baseline function $B^{(h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})} : \mathcal{H} \to \mathbb{R}$ that maps histories to real numbers and optionally depends on $h_t, \boldsymbol{\theta}, \boldsymbol{\mu}$ and $\boldsymbol{\psi}$. In order to show this, note that the expression on the right side may be written as

$$\nabla_{\boldsymbol{\psi}} V_H^{\boldsymbol{\psi}}(h_t) - \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \boldsymbol{\mu} \mid h_t) \sum_{t'=t+1}^{t+H} \mathbb{E}\left[\nabla_{\boldsymbol{\psi}} \log p(A_{t'-1} \mid H_{t'-1}, \boldsymbol{\psi}) B^{(h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})}(H_{t'-1}) \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi} \right] d\boldsymbol{\theta}\ d\boldsymbol{\mu}$$

based on a previously available expression for the gradient $\nabla_{\boldsymbol{\psi}} V_H^{\boldsymbol{\psi}}(h_t)$.

Let E denote the j-element of each term inside the summation in the expression above such that

$$E = \mathbb{E}\left[\frac{\partial}{\partial \psi_j} \log p(A_{t'-1} \mid H_{t'-1}, \boldsymbol{\psi}) B^{(h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})}(H_{t'-1}) \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi} \right].$$

Algorithm 15 Bayes-adaptive recurrent policy gradients

```
Input: prior over (one-step dynamics) models, number of interactions $T$, search horizon $H$, number of models to be drawn $N$, number of histories to be drawn $M$, learning rate $\eta$, discount factor $\gamma$
$\boldsymbol{\psi} \leftarrow$ arbitrary adaptive policy parameters
$s_0 \leftarrow$ initial state given by the environment
$h_0 \leftarrow s_0$
for each $t \in \{0, \ldots, T-1\}$ do
    for each $i \in \{1, \ldots, N\}$ do
        $\boldsymbol{\theta}^{(i)}, \boldsymbol{\mu}^{(i)} \leftarrow$ model drawn from the posterior $p(\cdot \mid h_t)$
        for each $j \in \{1, \ldots, M\}$ do
            $h^{(i,j)}_{t+H} \leftarrow$ history that extends $h_t$ drawn from $p(\cdot \mid h_t, \boldsymbol{\theta}^{(i)}, \boldsymbol{\mu}^{(i)}, \boldsymbol{\psi})$
            $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} + \eta \sum_{t'=t+1}^{t+H} \nabla_{\boldsymbol{\psi}} \log p(a^{(i,j)}_{t'-1} \mid h^{(i,j)}_{t'-1}, \boldsymbol{\psi}) \sum_{k=t'-t-1}^{H-1} \gamma^k r^{(i,j)}_{t+k+1}$
        end for
    end for
    $a_t \leftarrow$ action drawn from the adaptive policy $\tilde{\pi}^{(\boldsymbol{\psi})}$ for the history $h_t$
    $r_{t+1}, s_{t+1} \leftarrow$ reward and state given by the environment for action $a_t$
    $h_{t+1} \leftarrow (h_t, (a_t, r_{t+1}, s_{t+1}))$
end for
```

By making the expectation explicit,

$$E = \sum_{h_{t'-1}} \sum_{a_{t'-1}} p(a_{t'-1}, h_{t'-1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) \frac{\partial}{\partial \psi_j} \log p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}) B^{(h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})}(h_{t'-1}).$$

By factoring the joint probability mass function,

$$E = \sum_{h_{t'-1}} \sum_{a_{t'-1}} p(h_{t'-1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) p(a_{t'-1} \mid h_{t'-1}, h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) \frac{\partial}{\partial \psi_j} \log p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}) B^{(h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})}(h_{t'-1}).$$

Because $p(a_{t'-1} \mid h_{t'-1}, h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) = p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) = p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi})$,

$$E = \sum_{h_{t'-1}} \sum_{a_{t'-1}} p(h_{t'-1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}) \frac{\partial}{\partial \psi_j} \log p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}) B^{(h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})}(h_{t'-1}).$$

By reversing the likelihood ratio trick,

$$E = \sum_{h_{t'-1}} \sum_{a_{t'-1}} p(h_{t'-1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) \frac{\partial}{\partial \psi_j} p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}) B^{(h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})}(h_{t'-1}).$$

By pushing constants outside the innermost summation and using the fact that $\frac{\partial}{\partial \psi_j} 1 = 0$,

$$E = \sum_{h_{t'-1}} p(h_{t'-1} \mid h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi}) B^{(h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})}(h_{t'-1}) \frac{\partial}{\partial \psi_j} \sum_{a_{t'-1}} p(a_{t'-1} \mid h_{t'-1}, \boldsymbol{\psi}) = 0,$$

which shows that the expression subtracted from the gradient $\nabla_{\boldsymbol{\psi}} V_H^{\boldsymbol{\psi}}(h_t)$ is equal to zero.

A typical choice of baseline function approximates the value of a history for the remaining horizon such that

$$B^{(h_t, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi})}(h_{t'-1}) \approx \mathbb{E}\left[\sum_{k=t'-t-1}^{H-1} \gamma^k R_{t+k+1} \mid h_{t'-1}, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\psi} \right].$$

If such a baseline is adopted for gradient ascent, outperforming the expected outcome across plausible models contributes towards making the probability of a decision higher, while underperforming the expected outcome across plausible models contributes towards making the probability of a decision lower. Unfortunately, it is generally difficult to approximate the desired value, and even a perfect approximation would not guarantee that the corresponding estimator would have a reduced variance.

The posterior sampling algorithm is a heuristic alternative to Bayes-adaptive control algorithms (Alg. 16). This algorithm starts by drawing a model from the posterior. A dynamic programming algorithm (such as policy iteration or value iteration) is then used to find an optimal policy for this model. Finally, this policy is used to interact with the environment for a fixed number of time steps H, and the entire process repeats. Intuitively, increased certainty leads to decreased exploration. This heuristic has good asymptotic performance in episodic settings [7].

Algorithm 16 Posterior sampling control algorithm

```
Input: prior over (one-step dynamics) models, number of interactions $T$, commitment time $H$, discount factor $\gamma$
$s_0 \leftarrow$ initial state given by the environment
$h_0 \leftarrow s_0$
for each $t \in \{0, \dots, T-1\}$ do
    if $H$ divides $t$ then
        $\boldsymbol{\theta}, \boldsymbol{\mu} \leftarrow$ model drawn from the posterior $p(\cdot \mid h_t)$
        $\mathcal{P}, \mathcal{R} \leftarrow$ one-step dynamics functions corresponding to model $\boldsymbol{\theta}, \boldsymbol{\mu}$
        $\pi \leftarrow$ optimal policy for the one-step dynamics functions $\mathcal{P}$ and $\mathcal{R}$ and discount factor $\gamma$
    end if
    $a_t \leftarrow \pi(s_t)$
    $r_{t+1}, s_{t+1} \leftarrow$ reward and state given by the environment for action $a_t$
    $h_{t+1} \leftarrow (h_t, (a_t, r_{t+1}, s_{t+1}))$
end for
```

The posterior predictive algorithm is another heuristic alternative to Bayes-adaptive control algorithms (Alg. 17). This algorithm starts by using a dynamic programming algorithm (such as policy iteration or value iteration) to find an optimal policy for the so-called expected Markov decision process. This policy is then used to interact with the environment for a fixed number of time steps H, and the entire process repeats.

Algorithm 17 Posterior predictive control algorithm

```
Input: prior over (one-step dynamics) models, number of interactions $T$, commitment time $H$, discount factor $\gamma$
$s_0 \leftarrow$ initial state given by the environment
$h_0 \leftarrow s_0$
for each $t \in \{0, \dots, T-1\}$ do
    if $H$ divides $t$ then
        $\mathcal{P}, \mathcal{R} \leftarrow$ one-step dynamics functions of the expected Markov decision process given the history $h_t$
        $\pi \leftarrow$ optimal policy for the one-step dynamics functions $\mathcal{P}$ and $\mathcal{R}$ and discount factor $\gamma$
    end if
    $a_t \leftarrow \pi(s_t)$
    $r_{t+1}, s_{t+1} \leftarrow$ reward and state given by the environment for action $a_t$
    $h_{t+1} \leftarrow (h_t, (a_t, r_{t+1}, s_{t+1}))$
end for
```

The one-step dynamics functions $\mathcal{P}$ and $\mathcal{R}$ of the expected Markov decision process given the history h_t are defined by

$$\mathcal{P}^a_{ss'} = \frac{\alpha_{s'} + M^{(s,a)}_{s'}}{\sum_{s''} \alpha_{s''} + M^{(s,a)}_{s''}},$$

where $M^{(s,a)}_{s'}$ is the number of times in the history h_t that action a in state s resulted in state s', and

$$\mathcal{R}^a_{ss'} = \sum_r r \left[\frac{\alpha'_{i(r)} + N^{(s,a)}_{i(r)}}{\sum_{r'} \alpha'_{i(r')} + N^{(s,a)}_{i(r')}} \right],$$

where $N^{(s,a)}_{i(r)}$ is the number of times in the history h_t that action a in state s resulted in reward r.

The survey by Ghavamzadeh et al. [8] covers additional tabular Bayesian reinforcement learning algorithms.

License

References

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

[2] Peters, J. and Schaal, S. *Policy gradient methods for robotics.* International Conference on Intelligent Robots and Systems, 2006.

[3] Schulman, J. and Abbeel, P. *Deep Reinforcement Learning*, Available at `http://rll.berkeley.edu/deeprlcourse`, 2016.

[4] Silver, D. *Reinforcement learning*, Available at `http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html`, 2016.

[5] Wierstra, D., Förster, A., Peters, J., Schmidhuber, J. *Recurrent policy gradients.* Logic Journal of IGPL, 2010.

[6] Guez, A. *Sample-based Search Methods for Bayes-Adaptive Planning*, PhD thesis, University College London, 2015

[7] Osband, I. *Deep exploration via randomized value functions*, PhD thesis, Stanford University, 2016.

[8] Ghavamzadeh, M., Mannor, S., Pineau, J. and Tamar, A. *Bayesian reinforcement learning: A survey.* Foundations and Trends in Machine Learning, 8(5-6), pp.359-483, 2015.