Theory of Reinforcement Learning

DI Michael Pfeiffer
Institute for Theoretical Computer Science, TU Graz
pfeiffer@TUGraz.at

October 16, 2006

1 Theoretical Model of Reinforcement Learning

Reinforcement Learning (RL) deals with agents that sense and act in their environment and learn to choose optimal actions to achieve its goals [2]. In this document we summarize the mathematical models that have been developed for RL and show some proofs that give you the necessary background knowledge to understand the concepts behind learning algorithms.

The formal reinforcement learning model [1, 2, 3] for a deterministic environment consists of :

- A set of possible **states** S (the *state space*)
- A set of possible **actions** A from which the agent can choose (the *action space*)
- A numerical **reward function** $r : S \times A \to \mathbb{R}$
- A **transition function** $\delta : S \times A \to S$

In this model the agent interacts with the environment in the following way, which is also depicted in Figure 1: At time t the agent perceives a description s_t of the current state and then chooses an action $a_t \in A_{s_t}$, where A_s denotes the set of available actions in state s. The action changes the state of the environment to $s_{t+1} = \delta(s_t, a_t)$ and produces a *reward* or *reinforcement signal* $r_{t+1} = r(s_t, a_t)$ that is passed to the agent. This reinforcement signal indicates the quality of the selected action. The task of the agent is therefore to find a **policy**

$$\pi : S \to A$$

that chooses an action $a_t = \pi(s_t)$ in state s_t at time-step $t = 0, 1, 2, \ldots$ so as to maximize the possible cumulative reward over time.

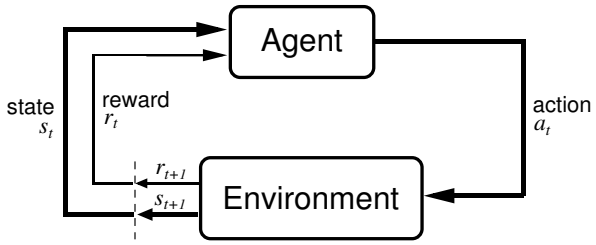

Figure 1: Agent-Environment interaction (from [1])

The above model fits only deterministic transition and reward models, but frequently we also have to deal with stochastic environments in which the next state and reward are only given by a probability distribution over possible values. We define the following quantities:

- $\mathcal{P}^a_{ss'} = P\{s_{t+1} = s' | s_t = s, a_t = a\}$ is called **transition probability**.
- $\mathcal{R}^a_{ss'} = E[r_{t+1} | s_t = s, a_t = a, s_{t+1} = s']$ is the **expected reward**.

The transition probability $\mathcal{P}^a_{ss'}$ gives for a given state s and action a the probability of the next state s'. In a deterministic environment, $\mathcal{P}^a_{ss'} = 1$ for exactly one state, namely $s' = \delta(s, a)$ and zero for all other states. $\mathcal{R}^a_{ss'}$ gives us the mean of a probability distribution for the reward for taking action a in state s and landing in state s'. Note that the reward in stochastic environments depends on three arguments, s, a, and s', so there is an explicit dependency on the next state s'. This was not the case for deterministic environments, because there the subsequent state is completely defined by s and a. To make notation easier we define the transition function $\delta(s, a)$ and the reward function $r(s, a, s')$ for stochastic environments as follows:

- $\delta(s, a)$ in stochastic environments returns the next state by sampling from the probability distribution $\mathcal{P}^a_{ss'}$.
- $r(s, a, s')$ in stochastic environments returns the next reward by sampling from a probability distribution with mean $\mathcal{R}^a_{ss'}$.

Similarly to stochastic transitions and rewards we can also define **stochastic policies** $\pi : S \times A \to [0, 1]$ as probability distributions over actions. $\pi(s, a)$ is then the probability of selecting action a in state s.

1.1 The Markov Property

In this framework the next state and reward depend only on the current state information and the action taken (even in stochastic environments). This has important consequences: the agent does not need to remember its history of perceptions and actions, but only needs to be aware of the current state to select its next action. But what is *the current state*? We have to distinguish between the **state of the environment** and the agent's **perceptions**. In general the agent does not have the complete information about everything in the environment, but only has some sensors that provide a representation of the environment's state to the agent. A mobile robot e.g. usually has distance sensors or cameras, but it cannot know what lies behind obstacles. On the other hand, the state representation does not have to consist only of immediate sensations, but the agent can also build up its own representation by processing the sequence of incoming signals. If the agent e.g. can only sense its current position, it can keep an internal memory of past positions to calculate its velocity. What we would like to have is that every relevant information for the decision making process is included in the state signal s. This is usually more than the immediate sensations, but never more than the complete history of all past sensations [1].

A state signal that fulfills this requirements is said to be **Markovian** or to have the **Markov property**, which is formally defined as

Definition 1.1 (Markov Property).

$$P\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, ..., r_1, s_0, a_0\} = P\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\} \quad (1)$$

In other words this means that the effects of an action taken in a state depend only on that state and not on the complete history of states, actions and rewards [1]. The left and right hand side in equation (1) need to be the same for all histories of previous states. It is a property of the state signal, but we will also say in this case that the environment and the task as a whole possess the Markov property.

An example for a state signal that has the Markov property is the configuration of a board game like chess or checkers, because only the current situation is important for the next decision, not the sequence of actions that led to the current board. On the other hand, if the task is to reach a

target position with a mobile robot and the state signal consists only of the immediate measurements of distance sensors, the task is clearly *non-Markovian*. Most of the theory that has been developed for reinforcement learning relies on the Markov property of the state signal. However, even if the state signal does not strictly fulfill this property, we can often successfully apply learning algorithms by seeing the state signal only as an approximation to a Markov state signal. Most theoretical guarantees however do not apply.

We are now ready to define a mathematical framework for reinforcement learning:

Definition 1.2 (Markov Decision Process (MDP))**.** A **Markov Decision Process**, short **MDP**, is a tuple $(S, A, \mathcal{P}, \mathcal{R})$ where

- S is the state space
- A is the action space
- $\mathcal{P} : S \times A \times S \to [0, 1]$ is the transition probability
- $\mathcal{R} : S \times A \times S \to \mathbb{R}$ is the expected reward

and the state signal $s \in S$ fulfills the Markov property.

MDPs are the most common framework for reinforcement learning, although the Markov property is in general a rather strong assumption. For most theoretical results we will only deal with **finite MDPs**, where both S and A are assumed to be finite.

1.2 Rewards and Return

We have already defined that the goal of the agent is to maximize the cumulative reward over time. The reward signal is the only information for the agent to evaluate its policy, there is no other information that tells the agent whether an actions was *correct* or *wrong*. This means that the reward function has to capture all important aspects of the desired behavior, without producing locally optimal policies. Here are some examples:

Example 1.1. To make an agent learn to reach a goal state in a maze we set the reward to $+1$ if it reaches that state, and zero otherwise. If want to make it reach the goal as fast as possible, we give it a reward of -1 for every step that does not lead to the goal state.

Example 1.2. To make an agent learn to play chess we give it a reward of $+1$ if it wins a match, -1 if it loses, or 0 for all intermediate steps and draws.

Example 1.3. An example of a flawed reward function: Again we want to make the agent reach a goal state as fast as possible. We therefore give it a reward of $+1$ for every action that moves the agent closer to the goal, and 0 otherwise. The optimal policy in this case is not to walk towards the goal as fast as possible, but to walk around forever, occasionally moving closer to the goal, but never reaching it. The agent can then accumulate infinite reward.

In mathematical terms we want to optimize at any time t the expected sum of future rewards

$$R_t = \sum_{k=0}^{\infty} r_{t+k+1}$$

which we call the **return**. In the simplest case we can assume that we have a finite time horizon, i.e. the interaction ends after at most $T < \infty$ time steps. We call such a subsequence of interactions an **episode** or **trial**, and states at which such a sequence stops are called **terminal states**. We assume that after reaching a terminal state the environment is reset to a starting state, which may also come from a distribution over the state space. Tasks with finite time horizons are called **episodic** tasks in the literature [1].

Some tasks however may not break into episodes but rather go on continually. We call this tasks **continuing tasks**. In this case we have an infinite time horizon, and the above definition of the return becomes problematic. The infinite sum of rewards may easily become infinite itself. Therefore we introduce a mathematical trick to avoid infinite returns.

Definition 1.3 (Discount Factor, Discounted Return)**.** Let $\gamma \in [0, 1]$ be a constant which we call **discount factor**. Then the **discounted return** at time t is defined as

$$R_t = r_{t+1} + \gamma r_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2}$$

We see that future rewards are *discounted* by the factor γ, i.e. their contribution to the discounted return diminishes exponentially with the number of time steps. If $\gamma < 1$ and the rewards are bounded then R_t is guaranteed to be finite. If $\gamma = 0$ we only learn to optimize immediate rewards, and the more we increase γ the more *farsighted* the agent becomes. Introducing a discounting factor may also have a strong effect on the optimal policy that we want to learn. Therefore the discount factor γ is often viewed as the fifth component in the definition of a MDP (see Def. 1.2).

Example 1.4. Let's look again at the first reward function defined in example 1.1. If we set the goal reward to $+1$ and all other rewards to zero, the optimal policy depends only on the discount factor. If $\gamma = 1$ the agent could first move around forever before reaching the goal, and still its return from any state would be $+1$. If however $\gamma < 1$, the return at a state k steps away from the goal can maximally be $\gamma^k < 1$, which is monotonically decreasing in k. Therefore the optimal policy would be to move towards the goal as fast as possible.

Using the trick of discounting we can use a unified notation for episodic and continuing tasks. For episodic tasks we introduce an additional *absorbing* state, from which all transitions lead back to itself, and all rewards are zero (see Figure 2). Combining the framework of MDPs with discounted returns leads to one of the main concepts in RL, namely value functions.

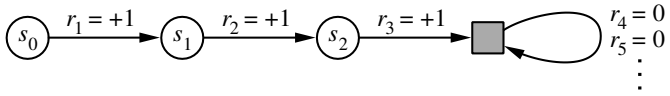

Figure 2: Absorbing state for episodic tasks (from [1])

1.3 Value Functions

In order to estimate the quality of a policy π is, the most common way is to estimate how good it is to be in a given state, or how good it is to choose a certain action in that state. We define the **value** of a state s under a policy π as the expected discounted return if we start from state s and follow policy π [1]:

Definition 1.4 (Value Function)**.**

$$V^\pi(s) = E_\pi \left[R_t | s_t = s \right] = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \;\middle|\; s_t = s \right] \tag{3}$$

Here E_π defines the expected value, if the agent follows the policy π. In order to evaluate different actions in a state we define the **action value** $Q^\pi(s, a)$ of a state-action pair under policy π as the expected discounted return if we take action a in state s and then follow policy π:

Definition 1.5 (Action-Value Function, Q-Function)**.**

$$Q^\pi(s, a) = E_\pi \left[R_t | s_t = s, a_t = a \right] = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \;\middle|\; s_t = s, a_t = a \right] \tag{4}$$

How we can actually calculate or estimate value- and action-value functions is the main subject of this course and will be explained in much more detail throughout the year. For now it is enough to understand intuitively what these functions tell us. The value function V^π tells us at which states we can expect the most return, following the current policy. So if the agent has an optimal value function and it knows which states it can reach from its current state, it should always go to the one with the highest value. The action-value function Q^π on the other hand tells the agent for the current state, how much future reward the available actions will probably yield, if thereafter actions are chosen according to policy π. With the knowledge of Q^π the agent does not need to know in which state it will land after action a, but it can simply choose the action with the highest Q-value.

An important observation is that there is a recursive relationship between the value of a state and the values of its successor states. Let us first look at the simpler case where transitions, rewards and policies are all deterministic. Then we can write

$$V^\pi(s) = r(s, \pi(s)) + \gamma V^\pi(\delta(s, \pi(s))) \tag{5}$$

This means that the value of one state is simply the immediate reward of executing the action returned by the policy plus the value of the next state. Similarly we note that there is also a relationship between V^π and Q^π:

$$\begin{aligned} V^\pi(s) &= Q^\pi(s, \pi(s)) && (6)\\ Q^\pi(s, a) &= r(s, a) + \gamma V^\pi(\delta(s, a)) = && (7)\\ &= r(s, a) + \gamma Q^\pi(\delta(s, a), \pi(\delta(s, a))) && (8) \end{aligned}$$

In equation (6) we use that the action-value of $(s, \pi(s))$ is simply the value of s, because we follow policy π at every step. In equation (7) we make use of the fact that the Q-function assumes that after executing action a for one step we follow policy π for all other steps. So the action-value of the pair (s, a) is the immediate reward of the action a in state s plus the discounted value of the successor state under policy π. We can also establish the recursive relationship between Q-values in (8) by simply inserting (6) in (7). The same relationships hold of course in the stochastic case, only the notation is a bit more complex. Let π be a stochastic policy, and write $\sum_{s \in S}$ and $\sum_{a \in A}$ with the corresponding probabilities to denote expectations over the whole state or action space. Then we can write the relationships, known as **Bellman equations** for V^π and Q^π [1], as

Theorem 1.1 (Bellman Equations for V^π and Q^π)**.**

$$\begin{aligned} V^\pi(s) &= E_\pi \left[Q^\pi(s, \pi(s)) \right] = && (9)\\ &= \sum_{a \in A_s} \pi(s, a) \sum_{s' \in S} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^\pi(s') \right) && (10)\\ Q^\pi(s, a) &= E_\pi \left[r(s, a, s') + \gamma V^\pi(s') \right] = && (11)\\ &= \sum_{s' \in S} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^\pi(s') \right) = && (12)\\ &= \sum_{s' \in S} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma \left(\sum_{a' \in A_{s'}} \pi(s', a') Q^\pi(s', a') \right) \right) && (13) \end{aligned}$$

Since these equations have to hold for all $s \in S$ and $a \in A$, in a finite MDP with known environment dynamics we could set up a system of equations with one equation per state or state-action pair and solve it to obtain the unique solution which is the value or action-value function. We will later show that the Bellman equations are the key to learn value function and later arrive at an optimal policy.

1.4 Optimal Value Functions

We still have not defined what makes a policy *optimal*, but using value functions, we can introduce a partial ordering on policies.

Definition 1.6 (Partial Ordering of Policies)**.** We write $\pi \geq \pi'$, meaning that policy π is better or equal than another policy π', if

$$V^{\pi}(s) \geq V^{\pi'} \text{ for every state } s \in S \tag{14}$$

This definition implies that by following policy π we receive at least as much reward as if we followed policy π' (at every possible state!). Note that this is only a partial ordering, meaning that it may be the case that neither $\pi \geq \pi'$ nor $\pi' \geq \pi$ holds. In this case the policies are not comparable. Using the ordering of policies, we can finally define what we mean with an **optimal** policy.

Definition 1.7 (Optimal Policy)**.** A policy π^* is optimal if

$$\pi^* \geq \pi \ \forall \pi \text{ or equivalently } V^{\pi^*}(s) = \max_{\pi} V^{\pi}(s) \ \forall s \in S \tag{15}$$

Note that the optimality of a policy is directly coupled with the optimality of its value function. An optimal policy has the highest possible value for every state in the state space. We define these values as $V^*(s) = V^{\pi^*}(s)$. Optimal policies also have optimal action-values, and so $Q^{\pi^*}(s,a) = \max_{\pi} Q^{\pi}(s,a)$ for all states $s \in S$ and all actions $a \in A_s$. Similarly to V^* we define $Q^*(s,a) := Q^{\pi^*}(s,a)$. From the definition it is clear that

$$\max_{a \in A_s} Q^*(s,a) = V^*(s) \tag{16}$$

because $Q^*(s,a)$ is the expected return if we take action a in state s and then follow an optimal policy. Using (16) and the Bellman equations (12) that holds for all action-value functions, we arrive at the so-called **Bellman optimality equation** for V^*

Theorem 1.2 (Bellman Optimality Equation for V^*)**.**

$$\begin{aligned} V^*(s) &= \max_{a \in A_s} Q^*(s,a) = & (17) \\ &= \max_{a \in A_s} \sum_{s' \in S} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^*(s') \right) & (18) \end{aligned}$$

Similarly we have the **Bellman optimality equation** for Q^*:

Theorem 1.3 (Bellman Optimality Equation for Q^*)**.**

$$\begin{aligned} Q^*(s,a) &= \sum_{s' \in S} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^*(s') \right) = & (19) \\ &= \sum_{s' \in S} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma \max_{a' \in A_{s'}} Q^*(s', a') \right) & (20) \end{aligned}$$

1.4.1 Policy Improvement

There are still many open questions. We know that optimal value functions satisfy the Bellman optimality equations, but can there also be sub-optimal value functions that fulfill them? Is there always an optimal policy or can it be that the system of Bellman optimality equations is unsolvable? And finally: How can we find an optimal value function and the corresponding policy? The idea is to start with any random policy, calculate its value function, and use this information to find an improved policy. To save space for notation we first prove that we can limit our search to the space of deterministic policies, starting with a lemma:

Lemma 1.1. *Assume π and π' are (possibly stochastic) policies with*

$$E_{\pi'}\left[Q^{\pi}(s,\pi'(s))\right] \geq V^{\pi}(s)\ \forall s \in S$$

Then

$$V^{\pi'}(s) \geq E_{\pi'}\left[Q^{\pi}(s,\pi'(s))\right] \geq V^{\pi}(s)\ \forall s \in S$$

Proof. Let $s_0 \in S$ be an arbitrary state. Using the definition of the Q-function we can write

$$E_{\pi'}\left[Q^{\pi}(s_0,\pi'(s_0))\right] = E_{\pi'}\left[r_1 + \gamma \cdot V^{\pi}(s_1)\right]$$

where we use the simplified notation s_i and r_i to denote the (possibly stochastic) result of applying policy π' in s_{i-1}. Using the assumption of the lemma we write

$$\begin{aligned} E_{\pi'}\left[r_1 + \gamma \cdot V^{\pi}(s_1)\right] &\leq E_{\pi'}\left[r_1 + \gamma \cdot E_{\pi'}\left[Q^{\pi}(s_1,\pi'(s_1))\right]\right] = \\ &= E_{\pi'}\left[r_1 + \gamma \cdot r_2 + \gamma^2 V^{\pi}(s_2)\right] \end{aligned}$$

where again the simplified notation has been used. By iterating the same argument we have

$$E_{\pi'}\left[Q^{\pi}(s_0,\pi'(s_0))\right] \leq \ldots \leq E_{\pi'}\left[\sum_{i=1}^{N} \gamma^{i-1} r_i + \gamma^N V^{\pi}(s_N)\right]$$

As $N \to \infty$ the right side converges to a sum of rewards obtained by following policy π' from state s_0, which is nothing else than $V^{\pi'}(s_0)$. Since s_0 was arbitrary, the lemma is proven. □

How can we interpret this lemma? We have a value and Q-function for policy π and a second policy π'. We know that $V^{\pi}(s) = E_{\pi}\left[Q^{\pi}(s,\pi(s))\right]$, because in both cases we calculate expected values under policy π. Now we look what happens at every state if we first take an action suggested by π' and thereafter follow π again. The expected return is then given by $E_{\pi'}\left[Q^{\pi}(s,\pi'(s))\right]$, because we still use the Q-function of π but replace only the first step. If this is larger or equal than $V^{\pi}(s)$ for all states $s \in S$, the lemma guarantees that policy π' is actually a better or equal policy. A simple conclusion would be that a policy that always takes the action with the highest Q-value is better, or in the worst case as good as the original policy. We call the step of replacing a policy π by $\pi^{+}(s) = \arg\max_{a \in A_s} Q^{\pi}(s,a)$ **policy improvement**.

Corollary 1.1. *For every policy π there exists a deterministic policy π' such that $\pi' \geq \pi$. As a special case: If there exists a stochastic optimal policy π, then there exists also a deterministic optimal policy π' such that $\pi' \geq \pi$.*

Proof. As exercise. □

Corollary 1.2. *If the same assumptions as in lemma 1.1 apply, and additionally $E_{\pi'}\left[Q^{\pi}(s_0,\pi'(s_0))\right] > V^{\pi}$ for some $s_0 \in S$, then $\pi' > \pi$, i.e. $\pi' \geq \pi$ and not $\pi \geq \pi'$.*

Proof. Same proof as for lemma 1.1, but starting in the special state s_0. □

1.5 Bellman Optimality Equations

Theorem 1.4 (Bellman Optimality Equations)**.** *Assume that A and S are finite. For every policy π_0 there exists some* **deterministic** *policy π with $\pi \geq \pi_0$ and π satisfies the* **Bellman optimality equations**

$$V^{\pi}(s) = \max_{a \in A_s} Q^{\pi}(s,a)\ \forall s \in S$$

Proof. We define a sequence of deterministic policies $\pi_1, \pi_2, \ldots$ such that $\pi_0 \leq \pi_1 < \pi_2 < \ldots$. We first make use of corollary 1.1 to choose a deterministic policy π_1 such that $\pi_1 \geq \pi_0$. If π_1 already satisfies the Bellman optimality equations, then we are done.

Otherwise there is a state $s_0 \in S$ such that $V^{\pi_1}(s_0) < \max_{a \in A_{s_0}} Q^{\pi_1}(s_0, a)$. Let a_0 be an action such that $V^{\pi_1}(s_0) < Q^{\pi_1}(s_0, a_0)$, then we define a new policy π_2 as

$$\pi_2(s) = \begin{cases} a_0 & \text{if } s = s_0 \\ \pi_1(s) & \text{otherwise} \end{cases}$$

According to corollary 1.2 we know that $\pi_1 < \pi_2$, and by iterating this argument we have $\pi_0 \leq \pi_1 < \pi_2 < \pi_3 < \ldots$.

Since we assumed that the state and action space are finite, we know that there can only be $|A| \cdot |S|$ different deterministic policies. Hence no infinite chain of policies can occur, and after finitely many improvement steps we arrive at a deterministic policy that fulfills the Bellman optimality equations. □

This theorem tells us, that there is always a deterministic policy that fulfills the Bellman optimality equations. We will now prove that the Bellman condition is already a sufficient criterion for optimality. (We have already shown in theorem 1.2 that it is a necessary condition.)

Theorem 1.5. *Assume that A and S are finite and $0 \leq \gamma < 1$. Then there exists* **at most one** *function $V^* : S \to \mathbb{R}$ which satisfies the Bellman optimality equations $V^*(s) = \max_{a \in A_s} Q^*(s, a)$ for all $s \in S$.*

Proof. We define the following norm between value functions

$$\|V - \tilde{V}\| = \max_{s \in S} |V(s) - \tilde{V}(s)| \quad \forall\, V, \tilde{V} : S \to \mathbb{R} \tag{21}$$

We also define an operator $\mathcal{A} : \mathbb{R}^S \to \mathbb{R}^S$, where $\mathbb{R}^S = \{f | f : S \to \mathbb{R}\}$ denotes the set of all functions that map S into $\mathbb{R}$. We use the symbol Q^V to denote the action-value function corresponding to V :

$$\mathcal{A}[V](s) = \max_{a \in A_s} Q^V(s, a) = \max_{a \in A_s} \sum_{s \in S'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V(s') \right) \tag{22}$$

Fact: $\mathcal{A}[V] = V$ is equivalent to V satisfying the Bellman optimality equations.

We will now show that $\mathcal{A}$ is a *contracting* operator in the norm defined in (21), i.e. $\|\mathcal{A}[V] - \mathcal{A}[\tilde{V}]\| \leq \gamma \|V - \tilde{V}\| < \|V - \tilde{V}\|$. We therefore fix an arbitrary state $s \in S$ and assume that $\mathcal{A}[V](s) \geq \mathcal{A}[\tilde{V}](s)$ (for the opposite case we just exchange V and $\tilde{V}$). Set $a^* = \arg\max_a Q^V(s, a)$. Then

$$\mathcal{A}[V](s) = \sum_{s'} \mathcal{P}^{a^*}_{ss'} \left(\mathcal{R}^{a^*}_{ss'} + \gamma V(s') \right) \tag{23}$$

and because a^* need not be optimal for $\tilde{V}$, we have

$$\mathcal{A}[\tilde{V}](s) \geq \sum_{s'} \mathcal{P}^{a^*}_{ss'} \left(\mathcal{R}^{a^*}_{ss'} + \gamma \tilde{V}(s') \right) \tag{24}$$

This implies

$$\begin{aligned} 0 \quad &\leq \quad \mathcal{A}[V](s) - \mathcal{A}[\tilde{V}](s) \leq && \text{(using (23) and (24))} \\ &\leq \quad \gamma \cdot \textstyle\sum_{s'} \mathcal{P}^{a^*}_{ss'} \left(V(s') - \tilde{V}(s') \right) \leq && \text{(max is larger than weighted sum)} \\ &\leq \quad \gamma \cdot \|V - \tilde{V}\| \end{aligned}$$

Since the above relationship holds for any arbitrary $s \in S$, we have shown that $\|\mathcal{A}[V] - \mathcal{A}[\tilde{V}]\| \leq \gamma \cdot \|V - \tilde{V}\| < \|V - \tilde{V}\|$. We still have to prove that there can be at most one function that fulfills the

Bellman optimality equations, which is equivalent to $\mathcal{A}[V] = V$. Assume for a contradiction that V and $\tilde{V}$ both satisfy the Bellman optimality equations and $V \neq \tilde{V}$, i.e. $\exists s \in S : V(s) \neq \tilde{V}(s)$. Using the contraction property of $\mathcal{A}$ and $\gamma < 1$ we find that

$$0 \quad < \quad \|V - \tilde{V}\| = \|\mathcal{A}[V] - \mathcal{A}[\tilde{V}]\| \leq \gamma \cdot \|V - \tilde{V}\| < \|V - \tilde{V}\|$$

This is a clear contradiction, and therefore V and $\tilde{V}$ have to be identical, if they satisfy the Bellman optimality equations, which proves the theorem. □

Using Theorem 1.5 we can easily prove that the Bellman optimality equations are a **sufficient** criterion for optimality:

Corollary 1.3. *Every policy π for which V^π satisfies the Bellman optimality equations*

$$V^\pi(s) = \max_{a \in A_s} Q^\pi(s, a) \ \forall s \in S$$

is optimal.

Proof. As exercise. □

We can now also guarantee (at least for the finite case) the existence of an optimal policy:

Corollary 1.4. *If S and A are finite, then there exists always an optimal policy.*

Proof. Theorem 1.4 guarantees that in the finite case there exists always a deterministic policy that satisfies the Bellman optimality equations. Through corollary 1.3 we know that this policy is optimal. □

These important theoretical results answer all our initial questions. We have a necessary and sufficient condition for an optimal policy, and we have proven the existence of at least one optimal policy. One important remark is that in Theorem 1.5 we have shown only the uniqueness of the optimal value function, but there may be more than one policy that leads to the same value function. So the optimal policy is not necessarily unique, only its value function.

Before discussing algorithms to learn the optimal value function, we prove one more result that illustrates the mechanism of policy improvement.

Corollary 1.5. *If A and S are finite and $0 \leq \gamma < 1$, then for any $V : S \to \mathbb{R}$ the sequence $\mathcal{A}^k[V]$ converges to a fix-point of $\mathcal{A}$ for $k \to \infty$ (where $\mathcal{A}^k[V] = \underbrace{\mathcal{A}[\mathcal{A}[\ldots\mathcal{A}[}_{k \text{ times}} V]]]$ denotes the k-times iterated application of the operator $\mathcal{A}[.]$).*

Proof. Since we know that a solution to the Bellman optimality equations always exists, there exists always a fix-point $V^* = \mathcal{A}[V^*]$ of $\mathcal{A}$. By the contraction property of $\mathcal{A}$ (see proof of Theorem 1.5) we know that $\|\mathcal{A}[V] - \underbrace{\mathcal{A}[V^*]}_{=V^*}\| \leq \gamma \cdot \|V - V^*\|$. By iterating this argument we have

$$\|\mathcal{A}^k[V] - V^*\| = \mathcal{A}^k[V] - \mathcal{A}^k[V^*]\| \leq \gamma^k \cdot \|V - V^*\| \to_{k \to \infty} 0$$

So as $k \in \mathbb{N}$ goes to infinity, the improved value function $\mathcal{A}^k[k]$ converges towards the unique fix-point V^*, which is the optimal value function. □

1.5.1 Finding Policies from Value Functions

Assuming that we know the optimal value function, or the optimal action-value function, we can easily derive an optimal policy from it:

Definition 1.8. Given V^* or Q^* the optimal policy is the so-called **greedy** policy

$$\pi^*(s) = \arg\max_{a \in A_s} \sum_{s' \in S} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^*(s') \right) = \arg\max_{a \in A_s} Q^*(s,a) \tag{25}$$

We see that if we have only the optimal value function we need to average over all possible successor states, and therefore need to know the dynamics of the environment to select the optimal action. If we know Q^* we just need to evaluate it for all available actions, the environment dynamics are already captured by the action-value function. Please note that the greedy policy is only optimal if the optimal value function is known. During learning, when we have only approximations to V^* or Q^*, we therefore have to use non-greedy policies to ensure *exploration*.

A Definition of Symbols

$P\{.\}$	Probability
$P\{.\|.\}$	Conditional probability
$E[.]$	Expected value
$E_\pi[.]$	Expected value with respect to a given policy π

References

[1] R. Sutton, A. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.

[2] T. Mitchell. *Machine Learning.* McGraw-Hill, Singapore, 1997.

[3] L. Kaelbling, M. Littman, A. Moore. *Reinforcement Learning: A Survey.* in: Journal of Artificial Intelligence Research 4, 1996.