University of Mannheim

Bachelor Thesis

Reinforcement Learning

by

Felix Benning

born on the 27.11.1996 in Nürtingen
matriculation number 1501817

in the
Faculty of Mathematics in Business and Economics
Supervisor: Prof. Dr. Leif Döring

Due Date: 11.05.2019

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.

This thesis was not previously presented to another examination board and has not been published.

City, Date Signature

Contents

Introduction **vii**

1 Markov Decision Processes **1**
1.1 Introduction 1
1.2 Model Formulation 4
1.3 Fixed Point Equations for Value Functions 11
1.4 Optimal Value Functions 15
1.5 Optimal policies 23
1.6 Dynamic Programming 30

2 Reinforcement Learning Algorithms **35**
2.1 Introduction 35
2.2 Monte Carlo 37
2.3 Temporal Difference Learning TD 43
2.4 Growing Batch Learning 46
2.5 TD(λ) – Mixing Monte Carlo and TD 48
2.6 Q-learning 55
2.7 Exploration 56
2.8 Function Approximation 62

3 Stochastic Approximation – Convergence **69**
3.1 Introduction 69
3.2 Learning Rates 70
3.3 Application to Reinforcement Learning 73
3.4 Proof of Theorem 3.3.1 79

A Appendix **93**
A.1 Probability Theory 93
A.2 Analysis 97
A.3 Autoencoder 98

Introduction

Let us consider a system to be a intelligent, if it collects information from its sensors and transforms this information into an "appropriate" response. Then the question where this input comes from (eyes, ears, cameras, etc.), or what kind of input it is (visual, audio, etc.), is as irrelevant for this definition, as the question what a response is (motor activity, sound, etc.). An intelligent system is simply a function mapping inputs to "appropriate" outputs. The question what "appropriate" means, is in contrast a lot more important and difficult to answer. But if we assume that there exists a "correct" response, then this correct response is itself a function from the input space to the output space. Artificial intelligence is therefore simply an artificial implementation of this correct response function.

If we have complete knowledge of this function, we can encode this function as an executable program manually. But more often than not, we do not really understand the rules according to which the inputs are supposed to be transformed into outputs.

For this reason we might want a machine to "learn" these rules by itself, i.e. we want it to approximate the correct response function. There are two larger subcategories in this category of *machine learning*.

In *supervised learning* we do not know the correct response function, but we know the correct response for certain inputs intuitively. An example for this case is image classification of animals. There, we do not know how the input (pixels of the picture) map to the output (name of the animal), but we can tell for a given picture what animal is depicted intuitively, i.e. we can provide examples of the correct response function to the learning algorithm. Supervised Learning is therefore concerned with generalization from examples. And while approximating a function with a finite sample of (error prone) function evaluations is a relatively old problem in numerical analysis and statistics, supervised learning is often associated with more recent approaches like artificial neuronal networks.

While we are still able to provide examples in supervised learning, *unsupervised learning* has to work with even less. This category includes clustering and principle component analysis, as well as *reinforcement learning*. Reinforcement learning is applied to problems, where we "know the correct response if we see it", but can not provide examples ourselves. This is for example the case with

walking. We know how walking looks like, but we have a hard time giving an example for the correct output signals sent to the motors in the leg. It is also used in cases where we do not know the correct response and have never seen it, but are able to compare different response functions. Examples range from games like chess, to profit maximization of a company.
In general reinforcement learning is used in cases, where humans or the environment in general can rate the solution, and provide rewards for good outcomes. Similar to the conditioning of animals, desirable actions are *reinforced* with rewards.
But in order to write algorithms, which maximize their rewards, we first need to formulate the relationship of possibly delayed rewards with actions (outputs) of the learning algorithm in certain states (given certain inputs). The model for this relationship – which virtually all modern reinforcement learning algorithms are based on – is the Markov Decision Process.
We will therefore introduce this model and its properties in the first chapter and continue with a review of common reinforcement algorithms in the second chapter. In the third chapter, we will explore the relation between the theory of stochastic approximation and reinforcement learning, since this relation can be exploited to obtain almost sure convergence for a large portion of reinforcement algorithms.

Chapter 1

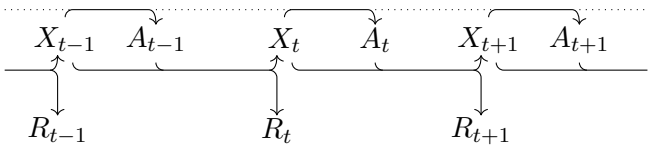
Markov Decision Processes

1.1 Introduction

I will try to convey intuition for Markov Decision Processes in this Introduction. Readers familiar with the subject can skip directly to the formal model formulation.

A Markov Decision Process appears to be quite similar to a Markov Process at first glance. Where a Markov Process is a stochastic process, i.e. a sequence of random variables $(X_t, t \in \mathbb{N}_0)$, which is memoryless (Markov). This means knowledge of the current state, makes knowledge about past states useless for predicting future states.

Markov Decision Processes (MDPs) introduce actions $(A_t, t \in \mathbb{N}_0)$ and rewards $(R_t, t \in \mathbb{N})$ to this model. Where the transition to the next state X_{t+1} given a state X_t and action A_t is Markov (memoryless), just like in Markov Processes.



But this apparent similarity can be misleading. While Markov Processes are defined as a stochastic process (i.e. a sequence of states) with properties as described above, MDPs cannot be defined like that. The reason for this is, that the next state X_{t+1} depends on the current state and action A_t, which means that the sequence of states cannot be defined without the sequence of actions. But defining the sequence of actions requires an action selection rule, a behaviour, a decision. So defining MDPs as a stochastic process, would result in every behaviour resulting in a different MDP. But since MDPs want to model decisions, this does not make much sense. Talking about optimal behaviours in a framework which can only be defined for a set behaviour, is nonsensical. What is needed is a model framework which is invariant to differ-

ent behaviours. Therefore MDPs are not defined as a stochastic process, but rather as a rulebook on how to create a stochastic process given a behaviour. The questions we ask the model are also different. MDPs are more interested into evaluating different behaviours and finding optimal ones, while Markov Processes try to describe an existing phenomenon. Though we will find, that – given a behaviour which is memoryless as well (without the dotted lines in the diagram) – the resulting stochastic process $(X_t, A_t, R_{t+1}, t \in \mathbb{N}_0)$ is a Markov Process, so the theory of Markov Processes could in principle be applied. But this will not be our focus.
Just like with Markov chains, the memorylessness could be circumvented by including the history in the current state, massively increasing the state space in the process. However it is questionable whether this would yield any interesting results, as then no state is visited twice. So it is of no use to an actor to learn the value of an action in a certain state without further assumptions. While the theory of MDPs found an extremely broad application as the backbone for reinforcement algorithms, and by proxy robotics, more classical applications include (White 1985):

1. Resource Management: The state is the resource level
    - Inventory Management: The resource is the inventory, the possible action is to order resupply, influencing the inventory (state) together with the stochastic demand, and the reward is the profit. The essential trade-off is the cost of storage versus lost sales from a stock-out.
    - Fishing: The resource is the amount of fish, the action is the amount fished, the reward is directly proportional to the amount fished, and the repopulation is the random element.
    - Pumped storage Hydro-power: The state is the amount of water in the higher reservoir and the electricity price, the action is to use water to generate electricity or wait for higher prices.
    - Beds in a hospital: How many empty beds are needed for emergencies?
2. Stock trading: The state is the price level and stock and liquidity owned.
3. Maintenance: When does a car/road become too expensive to repair?
4. Evacuation in response to flood forecasts

Lastly, to ease ourselves into the abstract definition of MDPs, let us do one example in more depth. Since examples in reinforcement learning, which make use of stochasticity, quickly become overly complex, we will stay with a classical example. Examples for reinforcement learning will naturally come up in the second chapter.

Example 1.1.1 (Inventory Management)**.** We will look at a retail store with just one good for simplicity. Let

$\mathcal{X} \coloneqq \mathbb{N}$ be the set of possible quantities of goods in stock,
$\mathcal{A} \coloneqq \mathbb{N}$ be the set of possible orders for resupply.

We will now introduce the mechanics of actions and rewards without stochastic behaviour and then change that later.
Let us assume that the ordered goods $a_t \in \mathcal{A}$ arrive in the morning the next day. The goods sold are the demand d_t, if the current stock $x_t \in \mathcal{X}$ can meet the demand. So the amount sold is actually $d_t \wedge x_t = \min\{d, x_t\}$. Assume orders are paid for in advance and bought at price 1, while the goods are sold at price p. The cost of storage is q per item and day. Then the profit at the end of day t is

$$r_{t+1} = p(d_t \wedge x_t) - a_t - q(x_t - d_t \wedge x_t).$$

And the stock level on the next day will be

$$x_{t+1} = x_t - d_t \wedge x_t + a_t.$$

Defining the *state transition function*

$$f(x, a, d) \coloneqq (x - d \wedge x + a,\ p(d \wedge x) - a - q(x - d \wedge x))\,,$$

allows us to express the day to day transition as

$$f(x_t, a_t, d_t) = (x_{t+1}, r_{t+1}).$$

Now assume that the demand is a random variable D_t which are independent and identically distributed (iid) for all t. This makes all the other objects (except for the actions) necessarily random variables too. Then distribution of (X_{t+1}, R_{t+1}) conditional on $X_t = x, A_t = a$ is defined to be the *transition probability kernel* $\mathcal{P}$ with

$$\mathcal{P}(\cdot \mid x, a) \coloneqq \mathbb{P}_{f(x,a,D_t)} \overset{\text{iid}}{=} \mathbb{P}_{f(x,a,D_0)}.$$

Note that knowledge of the transition kernel $\mathcal{P}$ is equivalent to knowledge of the transition function f and the distribution of the demand. Transition kernels reduce the clutter caused by exogenous random variables, which are different from application to application and are unknown until the next state is realized, at which point their knowledge is useless. We will therefore define this MDP to be $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathcal{P})$.
A stationary behaviour in this example would be a probability distribution over the possible orders, given the current state, i.e.

$$\pi(\cdot \mid x) \text{ is a probability distribution over } \mathcal{A}.$$

Actions are then selected by picking a random variable from this distribution,

$$A_t \sim \pi(\cdot \mid X_t).$$

Non-stationary behaviours are probability distributions over the action space which can depend on the entire history of states, actions and rewards.

We will just make one more generalization in the actual definition of MDPs. We will allow the action space $\mathcal{A}$ to depend on the state $x \in \mathcal{X}$.

1.2 Model Formulation

Most of the definitions in this chapter are adaptations from Szepesvári (2010). But to properly define the transition probabilities given an action in a certain state, let us define a probability kernel first.

Definition 1.2.1 (Kernel)**.** Let $(Y, \sigma_Y), (X, \sigma_X)$ be measure spaces.

$$\begin{aligned} \lambda \colon X \times \sigma_Y \to \mathbb{R} \text{ is called a } &\textit{(probability) kernel} \\ :\iff \lambda(\cdot, A) \colon x \mapsto \lambda(x, A) &\text{ is measurable, and} \\ \lambda(x, \cdot) \colon A \mapsto \lambda(x, A) &\text{ is a (probability) measure.} \end{aligned}$$

Since we will interpret probability kernels as distributions over Y given a certain condition $x \in X$, the notation $\lambda(\cdot \mid x) \coloneqq \lambda(x, \cdot)$ helps this intuition.

Definition 1.2.2. $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathcal{P})$ is called a *(finite) Markov Decision Process* (MDP), where

- $\mathcal{X}$ is a countable (finite) set of states,
- $\mathcal{A} = (\mathcal{A}_x)_{x \in \mathcal{X}}$ with A_x the countable (finite) set of possible actions in state x,
- $\mathcal{P} \colon (\mathcal{X} \times \mathcal{A}) \times \sigma_{\mathcal{X} \times \mathbb{R}} \to \mathbb{R}$ is a probability kernel, with the notation

$$\mathcal{X} \times \mathcal{A} \coloneqq \{(x, a) : x \in \mathcal{X}, a \in \mathcal{A}_x\}. \tag{1.1}$$

- $\mathcal{X} \times \mathbb{R}$ represents the next state and the reward. So $\mathcal{P}(\cdot \mid x, a)$ represents the probability distribution over the next states and rewards given an action a in the state x.
- $\mathcal{P}$ is called the *transition probability kernel*, or in short transition kernel.

Remark 1.2.3. Most authors split the transition kernel into a state transition kernel and a reward kernel (e.g. Puterman 2005). But since it is easier to define a marginal distribution from a joint distribution than vice versa, and since this notation is more compact I will stick to the definition from Szepesvári (2010). We will spare ourselves the complications of changing transition kernels over time. Just like the memorylessness this could be circumvented by including

time in the state. But if an algorithm is to learn which actions are optimal in which state, then both changing transition kernels over time and state spaces where you never visit any state twice (since the time is included) make it impossible to learn "the rules of the game" without assumptions on how the rules change over time. If such assumptions can be made it is usually easier to bake them into the state space. For example if you would like to introduce changing demand distributions over time in our inventory management example (Example 1.1.1), you could add the current expected demand to the state and the current market penetration. Then changes in the demand distribution can be modelled with a stationary transition kernel.

Some authors prefer to use "cost" over "rewards". This simply flips the notation and maximization problems become minimization problems. I prefer rewards, as negative rewards have an easier interpretation. According to Puterman (2005) some authors call this tuple a Markov Decision Problem instead of Markov Decision Process, presumably to reserve the term Markov Decision Process for the resulting sequence of states, actions and rewards $(X_t, A_t, R_{t+1}, t \in \mathbb{N}_0)$, aligning the Definition with the definition of a Markov process. However this does not appear to be common practice.
Nevertheless we still need to construct that stochastic process from the MDP when we have an action selection rule.
First, we need to select the random variable X_0 of the initial state. The initial state is not included in the definition of an MDP because later objects will be defined conditional on the current state. They are thus invariant to different starting distributions, as long as we have *exploring starts*, i.e. $\mathbb{P}(X_0 = x) > 0$ holds for all $x \in \mathcal{X}$ ensuring that conditioning on every state is possible. But since the definitions are independent of the starting state X_0, we can extend these definitions to degenerate distributions of X_0.
Second, we need an action selection rule

Definition 1.2.4. An A_t selection-rule $\pi = (\pi_t, t \in \mathbb{N}_0)$ is called (history dependent) *behaviour (policy)*, where

$$\pi_t \colon \begin{cases} (\mathcal{X} \times \mathcal{A} \times \mathbb{R})^t \times \mathcal{X} \times \sigma_{\mathcal{A}} \to \mathbb{R} \\ (h, x, A) \mapsto \pi_t(A \mid h, x) \end{cases} \quad \text{is a probability kernel,}$$

and $A_t \sim \pi_t(\cdot \mid (X_0, A_0, R_1), \ldots, (X_{t-1}, A_{t-1}, R_t), X_t))$.[1]
The following special cases can be viewed as policy subsets with some abuse of notation:

1. *Markov policies* $\pi = (\pi_t, t \in \mathbb{N}_0)$ are memoryless policies, i.e.

$$\pi_t \colon \begin{cases} \mathcal{X} \times \sigma_{\mathcal{A}} \to \mathbb{R} \\ (x, A) \mapsto \pi_t(A \mid x) \end{cases} \quad \text{with } A_t \sim \pi_t(\cdot \mid X_t).$$

[1] note that $\mathcal{X} \times \sigma_{\mathcal{A}} \coloneqq \{(x, A) : x \in \mathcal{X}, A \in \sigma_{\mathcal{A}_x}\}$ is defined just like $\mathcal{X} \times \mathcal{A}$ (c.f. (1.1))

A_t is selected such that it has the Markov property (well defined c.f. Remark 1.2.5), i.e.

$$\mathbb{P}[A_t = a \mid X_t] = \mathbb{P}[A_t = a \mid X_t, (X_{t-1}, A_{t-1}, R_t), \dots (X_0, A_0, R_1)]$$

2. *Stationary policies* are policies which do not change over time i.e. $\pi_{t+1} = \pi_t$. Since π_0 can only depend on the first state, they are naturally Markov and can be defined as

$$\pi\colon \begin{cases} \mathcal{X} \times \sigma_{\mathcal{A}} \to \mathbb{R} \\ (x, A) \mapsto \pi(A \mid x) \end{cases} \qquad \text{with } A_t \sim \pi(\cdot \mid X_t).$$

3. *Deterministic stationary policies* are specified by[2]

$$\pi\colon \mathcal{X} \to \mathcal{A}_x \quad \text{with } A_t = \pi(X_t).$$

Similarly, deterministic Markov policies and deterministic history dependent policies can be defined. The sets of these policies will be denoted like this

	Stationary		Markov		History dependent
Deterministic	Π_S^D	$\subseteq$	Π_M^D	$\subseteq$	Π^D
	$\cap\!\mid$		$\cap\!\mid$		$\cap\!\mid$
Stochastic	Π_S	$\subseteq$	Π_M	$\subseteq$	Π

Now we define inductively: $(X_{t+1}, R_{t+1}) \sim \mathcal{P}(\cdot \mid X_t, A_t)$ with the Markov property (well defined c.f. Remark 1.2.5), i.e.

$$\begin{aligned} &\mathbb{P}[(X_{t+1}, R_{t+1}) = (x, r) \mid (X_t, A_t)] \\ &= \mathbb{P}[(X_{t+1}, R_{t+1}) = (x, r) \mid (X_t, A_t), (X_{t-1}, A_{t-1}, R_t), \dots (X_0, A_0, R_1)] \end{aligned} \tag{1.2}$$

resulting in the stochastic process $((X_t, A_t, R_{t+1}), t \in \mathbb{N}_0)$.

Remark 1.2.5. $(X_{t+1}, R_{t+1}) \sim \mathcal{P}(\cdot \mid X_t, A_t)$ with the Markov property, is well defined i.e.: There exists a $\mathcal{X} \times \mathbb{R}$-valued random variable (X_{t+1}, R_{t+1}) such that

$$(X_{t+1}, R_{t+1}) \sim \mathcal{P}(\cdot \mid X_t, A_t) \text{ and it satisfies the Markov property.}$$

The analogous statement for the selection of A_t according to a Markov policy is true.

[2] $\pi\colon \mathcal{X} \to \mathcal{A}_x$ is notation for $\pi\colon \mathcal{X} \to \bigcup_{x \in \mathcal{X}} \mathcal{A}_x : \pi(x) \in \mathcal{A}_x$.

Proof. As we want to use cumulative distribution functions (cdf) for this proof, we first embed the $\mathcal{X} \times \mathbb{R}$ in $\mathbb{R}$. Since $\mathcal{X}$ is countable there exists an injective measurable mapping $g\colon \mathcal{X} \to \mathbb{N}$, and because there exist an homeomorphism[3] $\sigma\colon \mathbb{R} \to (0,1)$, there exists an injective measurable function

$$f\colon \begin{cases} \mathcal{X} \times \mathbb{R} \to \mathbb{R} \\ (x,r) \to g(x) + \sigma(r). \end{cases}$$

This allows us to define a probability kernel on the real numbers

$$\tilde{\mathcal{P}}(\cdot \mid x,a) \coloneqq \mathbb{P}_{f \circ Y} \qquad \text{where} \quad Y \sim \mathcal{P}(\cdot \mid x,a),$$

and the corresponding cdf

$$F_{x,a}(y) \coloneqq \tilde{\mathcal{P}}((-\infty, y] \mid x,a).$$

Now we select $U \sim \mathcal{U}(0,1)$ independent from the entire history, then due to A.1.4

$$F^{\leftarrow}_{x,a}(U) \sim \tilde{\mathcal{P}}(\cdot \mid x,a) \overset{\text{A.1.2}}{\Longrightarrow} f^{-1} \circ F^{\leftarrow}_{x,a}(U) \sim \mathcal{P}(\cdot \mid x,a),$$

where $F^{\leftarrow}$ is the pseudo-inverse (A.1.3). Thus

$$(X_{t+1}, R_{t+1}) \coloneqq f^{-1} \circ F^{\leftarrow}_{X_t,A_t}(U) \sim \mathcal{P}(\cdot \mid X_t, A_t)$$

fulfils the first requirement and is also Markov. To show the Markov property we first define a shorthand for the history

$$\mathcal{H}^t_s \coloneqq \{(X_t, A_t, R_{t+1}) \in H_t, \dots, (X_s, A_s, R_{s+1}) \in H_s\},$$

where H_i is defined as

$$H_i \coloneqq \{(x_i, a_i)\} \times U_i \in \sigma_{\mathcal{X} \times \mathcal{A} \times \mathbb{R}}.$$

With this notation we can now finish the proof:

$$\begin{aligned} \mathbb{P}\left(f^{-1} \circ F^{\leftarrow}_{X_t,A_t}(U) \in A \mid \mathcal{H}^t_0\right) &= \frac{\mathbb{P}(f^{-1} \circ F^{\leftarrow}_{x_t,a_t}(U) \in A, \mathcal{H}^t_0)}{\mathbb{P}(\mathcal{H}^t_0)} \\ &\overset{\text{U indep.}}{=} \mathbb{P}(f^{-1} \circ F^{\leftarrow}_{x_t,a_t}(U) \in A) \\ &= \dots = \mathbb{P}(f^{-1} \circ F^{\leftarrow}_{X_t,A_t}(U) \in A \mid \mathcal{H}^t_t) \end{aligned}$$

This proof relies heavily on our countable state set, more general cases would warrant the use of Kolmogorov's extension theorem. But Markov policies are still covered by the same type of proof. □

[3] continuous, bijective, inverse continuous – but we only really need measurable.

Proposition 1.2.6. *A (stationary) Markov policy π induces a (time homogeneous) Markov chain $(X_t, A_t, R_{t+1}, t \in \mathbb{N}_0)$.*

Proof. Using the shorthand for the history defined in the last proof together with

$$\mathbb{P}(A \cap B \mid C) = \frac{\mathbb{P}(A \cap B \cap C)}{\mathbb{P}(B \cap C)} \frac{\mathbb{P}(B \cap C)}{\mathbb{P}(C)} = \mathbb{P}(A \mid B \cap C)\mathbb{P}(B \mid C),$$

we can show the Markov property of the resulting chain:

$$\begin{aligned}
&\mathbb{P}\left[(X_t, A_t, R_{t+1}) \in \{(x,a)\} \times U \mid \mathcal{H}_0^{t-1}\right] \\
&= \mathbb{P}\left[R_{t+1} \in U \mid (X_t, A_t) = (x,a), \mathcal{H}_0^{t-1}\right] \mathbb{P}\left[(X_t, A_t) = (x,a) \mid \mathcal{H}_0^{t-1}\right] \\
&\overset{(1.2)}{=} \mathbb{P}\left[R_{t+1} \in U \mid (X_t, A_t) = (x,a)\right] \underbrace{\mathbb{P}\left[A_t = a \mid X_t = x\right]}_{=\pi_t(a|x)} \mathbb{P}\left[X_t = x \mid \mathcal{H}_{t-1}^{t-1}\right] \qquad (1.3) \\
&\overset{(*)}{=} \mathbb{P}\left[R_{t+1} \in U \mid (X_t, A_t) = (x,a), \mathcal{H}_{t-1}^{t-1}\right] \mathbb{P}\left[(X_t, A_t) = (x,a) \mid \mathcal{H}_{t-1}^{t-1}\right] \\
&= \mathbb{P}\left[(X_t, A_t, R_{t+1}) \in \{(x,a)\} \times U \mid \mathcal{H}_{t-1}^{t-1}\right]
\end{aligned}$$

$(*)$ *Some* of the history is irrelevant if all of the history is irrelevant (c.f. A.1.7). Left to show is, that for stationary policies the Markov chain is time homogeneous. When the policy is stationary, equation (1.3) simplifies to

$$\mathcal{P}(\mathcal{X} \times U \mid x, a)\pi(a \mid x)\mathcal{P}(\{x\} \times \mathbb{R} \mid x_{t-1}, a_{t-1}).$$

So the distribution of (X_t, A_t, R_{t+1}) given $\mathcal{H}_{t-1}^{t-1}$ is independent of t. The Markov chain is thus time homogeneous. □

Remark 1.2.7. If there is just one possible action in every state, the MDP is equivalent to a normal Markov Process. Since then there is a mapping $f\colon \mathcal{X} \to \mathcal{A}_x$ mapping the state to the only admissible action. Which implies that $A_t = f(X_t)$ which forces every behaviour to be equal to f. Therefore f is a deterministic stationary behaviour and Prop. 1.2.6 applies.

We can mostly ignore Markov policies because our transition kernel is stationary, which makes stationary policies the appropriate set of behaviours. Allowing changing transition probabilities over time, breaks virtually all of the following proofs. In such an environment of changing transition probabilities, Markov policies are the appropriate set to consider (c.f. Puterman 2005).

Definition 1.2.8. Define random variables $(Y_{(x,a)}, R_{(x,a)}) \sim \mathcal{P}(\cdot \mid x, a)$, then

$$r(x,a) \coloneqq \mathbb{E}[R_{(x,a)}] \quad \text{is called } \textit{immediate reward function.}$$

Definition 1.2.9. An MDP together with a discount factor $\gamma \in [0,1]$ is a

- *discounted* reward MDP for $\gamma < 1$,
- *undiscounted* reward MDP for $\gamma = 1$.

This allows us to define the *return*:

$$\mathcal{R} \coloneqq \sum_{t=0}^{\infty} \gamma^t R_{t+1}$$

In order for the expected return to be well defined in the discounted case, we need at least:

Assumption 1. $\forall (x,a) \in \mathcal{X} \times \mathcal{A}: \quad \mathbb{E}[|R_{(x,a)}|] \leq R \in \mathbb{R}$

Which due to Fubini implies, that the expected return is well defined. It also implies that the immediate reward is bounded:

$$\|r\|_\infty = \sup_{(x,a)\in\mathcal{X}\times\mathcal{A}} |\mathbb{E}[R_{(x,a)}]| \leq R$$

In order for the return to be always well defined in the discounted case, we would need:

Assumption 2. $\forall (x,a) \in \mathcal{X} \times \mathcal{A}: \quad |R_{(x,a)}| \leq R \in \mathbb{R}$

The undiscounted case will be of less interest (c.f. Section 1.6.1), but one possible way to handle that case, is to assume epsisodic MDPs (c.f. Def. 1.2.13). In order to define value functions, we will only need the expected return to be well defined for now.

Definition 1.2.10. Let $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathcal{P})$ be an MDP. Then every behaviour π results in a stochastic process $((X_t, A_t, R_{t+1}), t \in \mathbb{N}_0)$. We indicate with the superscript (e.g. $\mathbb{E}^\pi$) which behaviour generated the process in the following definitions.

$$V^\pi: \begin{cases} \mathcal{X} \to \mathbb{R} \\ x \mapsto \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x] \end{cases}$$ is called *value function* for π, [1]

$$Q^\pi: \begin{cases} \mathcal{X} \times \mathcal{A} \to \mathbb{R} \\ (x,a) \mapsto \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x, A_0 = a] \end{cases}$$ is called *action value function* for π, [2]

$$V^*: \begin{cases} \mathcal{X} \to \mathbb{R} \\ x \mapsto \sup\limits_{\pi\in\Pi} V^\pi(x) \end{cases}$$ is called *optimal value function*,

$$Q^*: \begin{cases} \mathcal{X} \times \mathcal{A} \to \mathbb{R} \\ (x,a) \mapsto \sup\limits_{\pi\in\Pi} Q^\pi(x,a) \end{cases}$$ is called *optimal action value function*,

and π is called *optimal* $: \iff V^* = V^\pi$.

[1] Well defined for exploring starts ($\mathbb{P}(X_0 = x) > 0$). And since the definition is independent of the distribution of X_0, V^π can be defined as an inherent property of the MDP and extended to cases when the actual start of the MDP is not exploring.

[2] Well defined because $A_1 \sim \pi_1(\cdot \mid (x, a, r_0), x_1)$ is defined for all a regardless of π_0

Remark 1.2.11. With the distribution of X_0 set (or X_0 being realized with a fixed value x), the distribution of X_t, A_t, R_{t+1} is inductively determined for all $t \in \mathbb{N}_0$. The conditional expectation is thus unique for a given $X_0 = x$, for all possible realizations of the MDP with a given behaviour. This means V^π, Q^π are well defined.

Remark 1.2.12. While the following proofs do not require discrete state and action spaces per se (as all the sums could just be replaced by integrals), the optimal value functions need not be measurable (Puterman 2005, p. 157). To remedy this Puterman suggests to either

- impose regularity conditions which would ensure measurability,
- use outer integrals to avoid measurability issues altogether,
- or extend notions of measurability.

But since the real world is finite in basically every case anyway we will spare ourselves the trouble.

Some authors include a Time set T in the tuple $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathcal{P})$ (e.g. Puterman 2005) this allows for finite horizons but not for continuous time, since the transition kernel is defined for discrete steps.
As finite processes are not our focus, we will not do that. But it is possible to achieve finite processes in infinite times with the use of terminal states. The catch is that – unless you include the time in the state – you can not guarantee the end of a process at a fixed time. Although including a finite time set in the state space does not blow it up as much as an infinite time set does. So this solution could possibly be acceptable.

Definition 1.2.13. Let $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathcal{P})$ be an MDP

$$x \in \mathcal{X} \text{ is called a } \textit{terminal (absorbing)} \text{ state} \quad :\iff \quad \forall s \in \mathbb{N} : \quad \mathbb{P}(X_{t+s} = x \mid X_t = x) = 1$$

An MDP with terminal states is called *episodic*. An *episode* is the random time period $(1, \dots, T)$ until a terminal state is reached.

Remark 1.2.14. The reward in a terminal state is zero by convention, i.e. x terminal state implies for all actions $a \in \mathcal{A}_x$, that $R_{(x,a)} = 0$.

1.2.1 Outlook

To avoid clutter we will from now on assume an underlying MDP with discounted rewards and with the accompanying definitions and notation.
In the following sections we will try to show that deterministic stationary policies are generally a large enough set of policies to choose from when searching for optimal or close to optimal policies. To be exact: We will find, that the supremum over all policies of value functions is the same as the supremum over

just the deterministic stationary policies. The other types of policies will get sandwiched in between.
We will show this fact by proving that both solve the same fixed point equation. And that this fixed point equation satisfies the requirements of the Banach Fixed Point Theorem (BFT). Which means, we will get a free approximation method of the (optimal) value functions along the way.

1.3 Fixed Point Equations for Value Functions

We will start by finding fixed point equations for V^π and Q^π which fulfil the requirements of the BFT. These can be used to simply calculate the value of one policy. But we can also use them as tools to show the fixed point equations for the optimal value functions. One intuitive example for such a use, is that you can try to show that V^* fulfils the fixed point equation for V^π and use uniqueness to argue that π has to be optimal. Other cases are a bit more technical and will use a type of monotonicity of the fixed point equation.
To find these fixed point equations we will try to set V^π and Q^π in relation to each other. For the most part we can focus on deterministic stationary policies, as the other policies will get sandwiched between these policies and the set of all policies.
To make some notation shorter, we will now define the state transition kernel as the marginal probability distribution of the entire transition kernel.

Definition 1.3.1.

$$p\colon \begin{cases} (\mathcal{X}\times\mathcal{A})\times\sigma_{\mathcal{X}} \to \mathbb{R} \\ (x,a,Y) \mapsto \mathcal{P}(Y\times\mathbb{R} \mid x,a) \end{cases} \quad \text{is the } \underline{\textit{state}} \textit{ transition kernel.}$$

Notation: $p(y \mid x,a) \coloneqq p(\{y\} \mid x,a)$ with $(x,a,y) \in \mathcal{X}\times\mathcal{A}\times\mathcal{X}$

Now we can start to explore the relation of V^π and Q^π.

Proposition 1.3.2. *Let $\pi \in \Pi_S$ be a stationary behaviour, then*

$$Q^\pi(x,a) = r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y \mid x,a) V^\pi(y)$$

Proof.

$$\begin{aligned}
&Q^\pi(x,a) \\
&= \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x, A_0 = a] \\
&= \mathbb{E}^\pi[R_1 \mid X_0 = x, A_0 = a] + \gamma \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t R_{t+2} \,\middle|\, X_0 = x, A_0 = a\right] \\
&\overset{\text{A.1.1}}{=} \underbrace{\mathbb{E}[R_{(x,a)}]}_{r(x,a)} + \gamma \sum_{y\in\mathcal{X}} \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t R_{t+2} \,\middle|\, X_0 = x, A_0 = a, X_1 = y\right] p(y \mid x,a)
\end{aligned}$$

This has almost the desired form. Just looking at the conditional expectation we get

$$\begin{aligned}
&\mathbb{E}^\pi\left[\sum_{t=0}^{\infty}\gamma^t R_{t+2}\;\middle|\; X_0=x, A_0=a, X_1=y\right]\\
&=\sum_{b\in\mathcal{A}}\mathbb{E}^\pi\left[\sum_{t=0}^{\infty}\gamma^t R_{t+2}\;\middle|\; X_0=x, A_0=a, X_1=y, A_1=b\right]\\
&\qquad\qquad\cdot\mathbb{P}^\pi(A_1=b\mid X_0=x, A_0=a, X_1=y)\\
&\overset{\text{Markov}}{=}\sum_{b\in\mathcal{A}}\mathbb{E}^\pi\left[\sum_{t=0}^{\infty}\gamma^t R_{t+2}\;\middle|\; X_1=y, A_1=b\right]\pi(b\mid y)\\
&=\mathbb{E}^\pi\left[\sum_{t=0}^{\infty}\gamma^t R_{t+2}\;\middle|\; X_1=y\right]\\
&\overset{(*)}{=}\mathbb{E}^\pi\left[\sum_{t=0}^{\infty}\gamma^t \tilde{R}_{t+1}\;\middle|\; \tilde{X}_0=y\right]\overset{(**)}{=}V^\pi(y),
\end{aligned}$$

$(*)$ by renaming

$$\tilde{X}_t \coloneqq X_{t+1}, \qquad \tilde{A}_t \coloneqq A_{t+1}, \qquad \tilde{R}_t \coloneqq R_{t+1}.$$

$(**)$ holds, since $(\tilde{X}_t, \tilde{A}_t, \tilde{R}_{t+1}, t\in\mathbb{N}_0)$ is another sequence generated by the MDP with the (stationary!) policy π. □

Corollary 1.3.3. *For $\pi\in\Pi_S$, this fixed point equation holds:*

$$V^\pi(x)=\mathbb{E}^\pi[r(x,A_0)\mid X_0=x]+\gamma\sum_{y\in\mathcal{X}}\mathbb{P}^\pi(X_1=y\mid X_0=x)V^\pi(y).$$

For $\pi\in\Pi_S^D$ the fixed point equation

$$\begin{aligned}
V^\pi(x)&=Q^\pi(x,\pi(x))\\
&=r(x,\pi(x))+\gamma\sum_{y\in\mathcal{X}}p(y\mid x,\pi(x))V^\pi(y),
\end{aligned}$$

and this fixed point equation hold:

$$Q^\pi(x,a)=r(x,a)+\gamma\sum_{y\in\mathcal{X}}p(y\mid x,a)Q^\pi(x,\pi(x)).$$

Proof. Let $\pi \in \Pi_S$, then:

$$\begin{aligned}
V^\pi(x) &= \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x] \\
&= \sum_{a \in \mathcal{A}_x} \underbrace{\mathbb{E}^\pi[\mathcal{R} \mid X_0 = x, A_0 = a]}_{=Q^\pi(x,a)} \pi(a \mid x) \qquad (1.4) \\
&= \sum_{a \in \mathcal{A}_x} \left(r(x,a) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x, a) V^\pi(y) \right) \pi(a \mid x) \\
&= \sum_{a \in \mathcal{A}_x} r(x,a)\pi(a \mid x) + \gamma \sum_{(y,a) \in \mathcal{X} \times \mathcal{A}_x} V^\pi(y) p(y \mid x, a) \pi(a \mid x) \qquad (1.5) \\
&= \mathbb{E}^\pi[r(x, A_0) \mid X_0 = x] + \gamma \sum_{(y,a) \in \mathcal{X} \times \mathcal{A}_x} V^\pi(y) \mathbb{P}^\pi(X_1 = y, A_0 = a \mid X_0 = x) \\
&= \mathbb{E}^\pi[r(x, A_0) \mid X_0 = x] + \gamma \sum_{y \in \mathcal{X}} V^\pi(y) \mathbb{P}^\pi(X_1 = y \mid X_0 = x)
\end{aligned}$$

And for the case, where π is a deterministic stationary policy:

$$V^\pi(x) = \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x] = \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x, A_0 = \pi(x)] = Q^\pi(x, \pi(x)).$$

The rest follows from proposition Prop. 1.3.2.
Alternatively: the V^π fixed point equation is a special case of the equation above. One just needs to realize that $r(x, \pi(x)) = \mathbb{E}^\pi[r(x, A_0) \mid X_0 = x]$ and:

$$\begin{aligned}
&\mathbb{P}^\pi(X_1 = y \mid X_0 = x) \\
&= \sum_{a \in \mathcal{A}_x} \mathbb{P}^\pi(X_1 = y \mid X_0 = x, A_0 = a) \underbrace{\mathbb{P}^\pi(A_0 = a \mid X_0 = x)}_{=\delta_{a\pi(x)}} \\
&= \mathbb{P}^\pi(X_1 = y \mid X_0 = x, A_0 = \pi(x)) \\
&= p(y \mid x, \pi(x))
\end{aligned}$$

□

With this relation we can use the Banach fixed point theorem (BFT) for the first time. But first note, that Assumption 1 implies

$$\mathbb{E}^\pi[|\mathcal{R}| \mid X_0 = x] \leq \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t |R_{t+1}| \,\middle|\, X_0 = x \right] \leq \frac{R}{1-\gamma},$$

which implies

$$\|V^\pi\|_\infty, \|V^*\|_\infty, \|Q^\pi\|_\infty, \|Q^*\|_\infty \leq R/(1-\gamma).$$

In particular they are bounded functions. We will denote the set of bounded function on M with

$$B(M) \coloneqq \{f \colon M \to \mathbb{R} : \|f\|_\infty < \infty\},$$

which happens to be a complete metric space which we will need for the Banach fixed point Theorem to work. Bounded Value functions will also be necessary

for a lot of other arguments later on, since the discount factor allows us to discard rewards after a finite time period as negligible. In finite MDPs this assumption is of course always fulfilled.

Definition 1.3.4. For a policy $\pi \in \Pi_S$, the mapping $T^\pi \colon B(\mathcal{X}) \to B(\mathcal{X})$ with

$$T^\pi V(x) \coloneqq \mathbb{E}^\pi[r(x, A_0) \mid X_0 = x] + \gamma \sum_{y \in \mathcal{X}} \mathbb{P}^\pi(X_1 = y \mid X_0 = x) V(y)$$

for $V \in B(\mathcal{X})$ and $x \in \mathcal{X}$ is called the *Bellman Operator*.
For the special case of $\pi \in \Pi_S^D$ this mapping can be written as

$$T^\pi V(x) \coloneqq r(x, \pi(x)) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x, \pi(x)) V(y).$$

With some abuse of notation we can define the Bellman operator on action value functions $T^\pi \colon B(\mathcal{X} \times \mathcal{A}) \to B(\mathcal{X} \times \mathcal{A})$ for $\pi \in \Pi_S^D$ with

$$T^\pi Q(x, a) \coloneqq r(x, a) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x, a) Q(y, \pi(y)) \qquad Q \in B(\mathcal{X} \times \mathcal{A}).$$

As mentioned previously, we can mostly ignore stochastic stationary policies as their optimal value functions will get sandwiched between deterministic and general policies. But we will need the slightly more general version of the operator for the value functions V^π for a proof about optimal policies later on. In contrast we only need the the special case for the operator on action value functions Q^π.

Theorem 1.3.5. *For all stationary policies $T^\pi V^\pi = V^\pi$ holds and for deterministic stationary policies $T^\pi Q^\pi = Q^\pi$ holds (c.f. Cor. 1.3.3).*
T^π meets the requirements of the Banach fixed point theorem for $\gamma < 1$. This implies that the fixed points above are unique *fixed points and can be approximated with the canonical iteration.*

Proof. $(B(\mathcal{X}), \|\cdot\|_\infty)$ is a non-empty, complete metric space and the mapping maps onto itself. It is left to show, that T^π is a contraction. Be $V, W \in B(\mathcal{X})$:

$$\begin{aligned}
\|T^\pi V - T^\pi W\|_\infty &= \|\gamma \sum_{y \in \mathcal{X}} \mathbb{P}^\pi(X_1 = y \mid X_0 = \cdot\,)(V(y) - W(y))\|_\infty \\
&\leq \gamma \sup_{x \in \mathcal{X}} \left\{ \sum_{y \in \mathcal{X}} \mathbb{P}^\pi(X_1 = y \mid X_0 = x) \|V - W\|_\infty \right\} \\
&= \gamma \|V - W\|_\infty \sup_{x \in \mathcal{X}} \underbrace{\left\{ \sum_{y \in \mathcal{X}} \mathbb{P}^\pi(X_1 = y \mid X_0 = x) \right\}}_{=1} \\
&= \gamma \|V - W\|_\infty
\end{aligned}$$

The proof for $T^\pi \colon B(\mathcal{X} \times \mathcal{A}) \to B(\mathcal{X} \times \mathcal{A})$ is analogous. □

Lemma 1.3.6. *For $W_1, W_2 \in B(\mathcal{X})$ write:*

$$W_1 \leq W_2 : \iff W_1(x) \leq W_2(x) \quad \forall x \in \mathcal{X}.$$

Then the operator T^π is monotonous:

$$W_1 \leq W_2 \implies T^\pi W_1 \leq T^\pi W_2$$

.

Proof. Be $W_1, W_2 \in B(\mathcal{X})$, $W_1 \leq W_2$ and $x \in \mathcal{X}$, then

$$T^\pi W_2(x) - T^\pi W_1(x) = \gamma \sum_{y \in \mathcal{X}} \mathbb{P}^\pi(X_1 = y \mid X_0 = x) \underbrace{(W_2(y) - W_1(y))}_{\geq 0} \geq 0 \quad \square$$

1.4 Optimal Value Functions

Until we have shown that the supremum over the small subset of deterministic stationary behaviors is the same as over all behaviors, we need to make the notational disctinction between different optimal value functions.

Definition 1.4.1.

$$\tilde{V}(x) \coloneqq \sup_{\pi \in \Pi_S^D} V^\pi(x)$$
$$\tilde{Q}(x,a) \coloneqq \sup_{\pi \in \Pi_S^D} Q^\pi(x,a)$$

As previously stated: The goal is to show that these two pairs of optimal value functions are actually the same using the uniqueness of the fixed point of the BFT.
The intuition for why this should be the case, comes from the fact that the MDP is memory-less. So winding up in the same state, results in the agent having the exact same decision problem. And if the agent decided for one optimal action in the past, then this action will again be optimal. For this reason the supremum over all behaviors should be equal to the supremum over stationary behaviors.
And if an optimal policy randomizes over different actions, then the values of theses actions must all be equally high which means that the set of deterministic policies is large enough. Since just picking one and sticking with it should be just as good.

1.4.1 Finite Outmatching

Before we get to tackle this problem, we first need to adress another one. Notice how we take the supremum for every state x individually?

$$\tilde{V}(x) = \sup_{\pi \in \Pi_S^D} V^\pi(x)$$

Since the stationary policy still allows us to condition on the state, it is intuitive to assume that we do not need different sequences of policies to approximate V^* for each state $x \in \mathcal{X}$. This will eventually turn out to be true using statements about the optimal value functions. We would not be successful to show this directly from the definition. As sequences approximating the different suprema are indexed by the (possibly countable) state space, what we would need to find is a single sequence of policies, which matches all the policies in this countable set of sequences in every step of the sequence. This turns out to be an impossible task. Here is an example for that.

Example 1.4.2 (The Genie Cubicles)**.** Imagine an infinite (countable) amount of cubicles $\mathbb{N} \subset \mathcal{X}$, with a genie in every cubicle. You can wish for an arbitrary amount of money $\mathcal{A} = \mathbb{N}$, after that you have to leave (end up in the terminal state 0, i.e.: $\mathcal{X} = \mathbb{N}_0$). Then of course $\tilde{V}(x) = \infty$ for $x \in \mathbb{N}$, is achived by the sequences of deterministic stationary behaviors

$$(\pi_x^{(n)}, n \in \mathbb{N}) \quad \text{with } \pi_x^{(n)}(y) = x + n \quad \forall y \in \mathcal{X}$$

Then there is no policy $\pi^{(n)}$ which can match every $\pi_x^{(n)}$ for all $x \in \mathbb{N}$.
Even if $\tilde{V}$ was finite, we could modify the example by cutting gifts off above a certain threshold with behaviors approaching that threshold.

So we can not match an infinite set of behaviors. Which means we will have to settle for a finite version right now. This version will help us to handle the optimal value functions. But with the later facts about optimal value functions we can then define a sequence of policies which approach the optimal value functions uniformly confirming our earlier suspicions, that the suprema can be attained with a single sequence of policies.

Proposition 1.4.3. *Be $n \in \mathbb{N}$ and $\{\pi_1, \dots, \pi_n\} \subseteq \Pi_S^D$, then:*

$$\exists \hat{\pi} \in \Pi_S^D : \quad \max_{i=1,\dots,n} V^{\pi}(x) \leq V^{\hat{\pi}}(x) \qquad \forall x \in \mathcal{X}$$

Proof. The idea is to to pick the same action in state x, as the policy which generates the most value out of this state, i.e. $\max_{i=1,\dots,n} V^{\pi_i}(x)$. So define

$$\hat{\pi} : \begin{cases} \mathcal{X} \to \mathcal{A} \\ x \mapsto \pi_{\underset{i=1,\dots,n}{\operatorname{argmax}} V^{\pi_i}(x)}(x). \end{cases}$$

One might be surprised that such an appearingly short sighted policy should surpass every policy in the finite set. At first it might seem that different policies achive their values of their state through different, maybe incompatible strategies. Would a strategy which takes a policy because it changes transition probabilities in a way that leads to a high-payoff state not be sabotaged by switching policies erratically? It is important to realize that if a policy τ

attaches a high value to a state because it provides easy access to states which can yield a high payoff, then the other policies will either exploit the high payoff later on as well, or the policy τ will once again be the maximum. Either way it would result in $\hat{\pi}$ exploiting the high payoff later on.
For the maximum value we write

$$V(x) \coloneqq \max_{i=1,\dots,n} V^{\pi_i}(x).$$

And let

$$m(x) \coloneqq \arg\max_{i=1,\dots,n} V^{\pi_i}(x),$$

then the modified policy $\hat{\pi}$ improves V in all states:

$$\begin{aligned} T^{\hat{\pi}}V(x) &= r(x, \pi_{m(x)}(x)) + \gamma \sum_{y\in\mathcal{X}} p(y \mid x, \pi_{m(x)}(x))V(y) \\ &\overset{V\geq V^{\pi_{m(x)}}}{\geq} r(x, \pi_{m(x)}(x)) + \gamma \sum_{y\in\mathcal{X}} p(y \mid x, \pi_{m(x)}(x))V^{\pi_{m(x)}}(y) \\ &\overset{1.3.3}{=} V^{\pi_{m(x)}}(x) = V(x) \end{aligned}$$

By using the monotonicity of $T^{\hat{\pi}}$ (Lemma 1.3.6) inductively with $(T^{\hat{\pi}})^1 V \geq (T^{\hat{\pi}})^0 V$, we get

$$(T^{\hat{\pi}})^n V \geq (T^{\hat{\pi}})^{n-1} V,$$

and thus finish the proof with:

$$V^{\hat{\pi}} = \lim_{n\to\infty} (T^{\hat{\pi}})^n V \geq V \geq V^{\pi_i} \qquad \forall i = 1, \dots, n \qquad \square$$

1.4.2 Fix-Point Equations for Optimal Value functions

With this statement we can now prove the building blocks for the fixed point equations.

Lemma 1.4.4.

(i) $\tilde{Q}(x,a) = r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y \mid x, a)\tilde{V}(y)$

(ii) $\tilde{V}(x) = \sup_{a\in\mathcal{A}_x} \tilde{Q}(x,a)$

(iii) $V^*(x) = \sup_{a\in\mathcal{A}_x} Q^*(x,a)$

(iv) $Q^*(x,a) = r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y \mid x, a)V^*(y)$

Proof. (i) The smaller or equal part is easy:

$$\begin{aligned}\tilde{Q}(x,a) &= \sup_{\pi\in\Pi_S^D}\left\{r(x,a)+\gamma\sum_{y\in\mathcal{X}}p(y\mid x,a)V^\pi(y)\right\}\\ &\le r(x,a)+\gamma\sum_{y\in\mathcal{X}}p(y\mid x,a)\underbrace{\sup_{\pi\in\Pi_S^D}V^\pi(y)}_{=\tilde{V}(y)}\end{aligned}$$

For the other direction we need to do a bit more work. Since the $r(x,a)$ and γ, are unaffected by the supremum what is left to show is:

$$\sup_{\pi\in\Pi_S^D}\sum_{y\in\mathcal{X}}p(y\mid x,a)V^\pi(y)\ge\sum_{y\in\mathcal{X}}p(y\mid x,a)\tilde{V}(y)$$

This problem should look familiar. It is the question whether there is a single sequence of policies that can match multiple sequences of policies indexed by the state space $y\in\mathcal{X}$. As we can find a policy which can outmatch a finite set of policies (Prop. 1.4.3), we will consider only the $y\in\mathcal{X}$ with the largest probability to occur and match these sequences with a single sequence. Since we then have just a single policy in the sum, we can estimate up by taking the outer supremum over deterministic policies. That is the idea, which can be executed as follows:
The set $M_\delta \coloneqq \{y\in\mathcal{X} : p(y\mid x,a)>\delta\}$ is finite for all $\delta>0$. Together with the continuity of measures this yields:

$$1=p\left(\bigcup_{\delta\to0}M_\delta\;\middle|\;x,a\right)=\lim_{\delta\to0}p(M_\delta\mid x,a)=\lim_{\delta\to0}\sum_{y\in M_\delta}p(y\mid x,a)$$

Therefore for all $\varepsilon>0$ exists a $\delta>0$ such that

$$\sum_{y\in M_\delta^\complement}p(y\mid x,a)<\frac{\varepsilon/4}{R/(1-\gamma)}. \tag{1.6}$$

Be $(\pi_y^{(n)}, n\in\mathbb{N})$ with $V^{\pi_y^{(n)}}(y)\nearrow\tilde{V}(y)\quad(n\to\infty)$. Since M_δ is finite, there exists $N\in\mathbb{N}$ such that

$$|\tilde{V}(y)-V^{\pi_y^{(n)}}(y)|<\varepsilon/2\qquad\forall n\ge N,\forall y\in M_\delta^\complement. \tag{1.7}$$

And also because M_δ is finite, and Prop. 1.4.3 we know that

$$\exists\hat{\pi}^{(n)}\in\Pi_S^D:\quad V^{\hat{\pi}^{(n)}}\ge V^{\pi_y^{(n)}}\qquad\forall y\in M_\delta. \tag{1.8}$$

This finally implies

$$\begin{aligned}
&\sum_{y\in\mathcal{X}} p(y\mid x,a)\tilde{V}(y) - \sum_{y\in\mathcal{X}} p(y\mid x,a)V^{\hat{\pi}^{(n)}}(y)\\
&\leq \sum_{y\in M_\delta} p(y\mid x,a)(\tilde{V}(y) - V^{\hat{\pi}^{(n)}}(y)) + \sum_{y\in M_\delta^{\complement}} p(y\mid x,a)\quad \underbrace{|\tilde{V}(y) - V^{\hat{\pi}^{(n)}}(y)|}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \leq \|\tilde{V}\|_\infty + \|V^{\hat{\pi}^{(n)}}\|_\infty\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \leq 2R/(1-\gamma)\\
&\overset{(1.8)}{\leq} \sum_{y\in M_\delta} p(y\mid x,a)\underbrace{(\tilde{V}(y) - V^{\pi_y^{(n)}}(y))}_{<\varepsilon/2 \quad (1.7)} + 2R/(1-\gamma)\underbrace{\sum_{y\in M_\delta^{\complement}} p(y\mid x,a)}_{<\frac{\varepsilon/4}{R/(1-\gamma)} \quad (1.6)}\\
&\leq \varepsilon \qquad \forall n\geq N.
\end{aligned}$$

This results in

$$\begin{aligned}
\sum_{y\in\mathcal{X}} p(y\mid x,a)\tilde{V}(y) &\leq \varepsilon + \sum_{y\in\mathcal{X}} p(y\mid x,a)V^{\hat{\pi}^{(n)}}(y)\\
&\leq \varepsilon + \sup_{\pi\in\Pi_S^D}\sum_{y\in\mathcal{X}} p(y\mid x,a)V^{\pi}(y).
\end{aligned}$$

Since this equation holds for all $\varepsilon > 0$ we are finished.
(ii) By Cor. 1.3.3 we know $V^\pi(x) = Q^\pi(x,\pi(x))$ and thus:

$$\begin{aligned}
\tilde{V}(x) &= \sup_{\pi\in\Pi_S^D} V^\pi(x) = \sup_{\pi\in\Pi_S^D} Q^\pi(x,\pi(x))\\
&\leq \sup_{a\in\mathcal{A}_x}\sup_{\pi\in\Pi_S^D} Q^\pi(x,a) = \sup_{a\in\mathcal{A}_x}\tilde{Q}(x,a) \qquad (1.9)\\
&\overset{\text{(i)}}{=} \sup_{a\in\mathcal{A}_x}\left\{ r(x,a) + \gamma\sum_{y\in\mathcal{X}} p(y\mid x,a)\sup_{\pi\in\Pi_S^D} V^\pi(y)\right\} \qquad (1.10)
\end{aligned}$$

Assume (1.9) is a true inequality for some $x\in\mathcal{X}$. Since the suprema in (1.10) can be arbitrarily closely approximated we get

$$\exists\pi,\exists a: \quad \tilde{V}(x) < r(x,a) + \gamma\sum_{y\in\mathcal{X}} p(y\mid x,a)V^\pi(y).$$

Define a slightly changed deterministic policy with this π and a:

$$\hat{\pi}\colon \begin{cases} \mathcal{X}\to\mathcal{A}_y & \\ y\mapsto \begin{cases}\pi(y) & y\neq x\\ a & y=x\end{cases}\end{cases}$$

Define $W_n \coloneqq (T^{\hat{\pi}})^n V^\pi$, then a case by case analysis of $y \neq x$ and $y = x$ results in:

$$\begin{aligned} W_1(y) &= T^{\hat{\pi}} V^\pi(y) \\ &\overset{y \neq x}{=} T^\pi V^\pi(y) = V^\pi(y) \\ &\overset{y=x}{=} r(x, \hat{\pi}(x)) + \gamma \sum_{z \in \mathcal{X}} p(z \mid x, \hat{\pi}(x)) V^\pi(z) \\ &= r(x,a) + \gamma \sum_{z \in \mathcal{X}} p(z \mid x, a) V^\pi(z) \\ &> \tilde{V}(x) \geq V^\pi(x) \end{aligned}$$

So in either case we get

$$W_1(y) \geq V^\pi(y) = W_0(y). \tag{1.11}$$

By induction with induction basis (1.11) using the monotonicity of $T^{\hat{\pi}}$ (Lemma 1.3.6), we get

$$W_{n+1} = T^{\hat{\pi}} W_n \geq T^{\hat{\pi}} W_{n-1} = W_n,$$

which leads to a contraction:

$$\begin{aligned} V^{\hat{\pi}}(x) &= \lim_{n \to \infty} (T^{\hat{\pi}})^n V^\pi(x) = \lim_{n \to \infty} W_n(x) \geq W_1(x) \\ &= r(x,a) + \gamma \sum_{z \in \mathcal{X}} p(z \mid x, a) V^\pi(z) \\ &> \tilde{V}(x) \quad \lightning \quad \hat{\pi} \in \Pi_S^D \end{aligned}$$

(iii) When taking the supremum over two variables the order does not matter, therefore:

$$\begin{aligned} \sup_{a \in \mathcal{A}_x} Q^*(x,a) &= \sup_{a \in \mathcal{A}_x} \sup_{\pi \in \Pi} \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x, A_0 = a] \\ &= \sup_{(\pi_t, t \in \mathbb{N}_0)} \sup_{a \in \mathcal{A}_x} \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x, A_0 = a] \\ &\overset{A_0 = a}{=} \sup_{(\pi_t, t \in \mathbb{N})} \sup_{a \in \mathcal{A}_x} \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x, A_0 = a] \\ &\overset{(*)}{=} \sup_{(\pi_t, t \in \mathbb{N})} \sup_{\pi_0} \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x] \\ &= V^*(x) \end{aligned} \tag{1.12}$$

$(*)$ "$\leq$" is trivial, since a deterministic π_0 can achieve the same as a fixed action $A_0 = a$.

"$\geq$" follows from this inequality which holds for all π (especially for all π_0):

$$\begin{aligned}\mathbb{E}^\pi[\mathcal{R} \mid X_0 = x] &\overset{A.1.1}{=} \sum_{a\in\mathcal{A}_x} \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x, A_0 = a]\pi_0(a \mid x) \\ &\leq \sup_{a\in\mathcal{A}_x} \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x, A_0 = a] \sum_{a\in\mathcal{A}_x} \pi_0(a \mid x) \\ &= \sup_{a\in\mathcal{A}_x} \mathbb{E}^\pi[\mathcal{R} \mid X_0 = x, A_0 = a]\end{aligned}$$

(iv) For an arbitrary policy π we can expand the action value function

$$\begin{aligned}Q^\pi(x,a) &= \mathbb{E}^\pi[R_1 \mid X_0 = x, A_0 = a] + \mathbb{E}\left[\sum_{t=1}^\infty \gamma^t R_{t+1} \,\middle|\, X_0 = x, A_0 = a\right] \\ &= r(x,a) + \gamma\mathbb{E}\left[\sum_{t=0}^\infty \gamma^t R_{t+2} \,\middle|\, X_0 = x, A_0 = a\right]. \qquad (1.13)\end{aligned}$$

This implies

$$\begin{aligned}Q^*(x,a) &= \sup_{\pi\in\Pi} Q^\pi(x,a) \\ &\overset{(1.13)}{=} r(x,a) + \gamma \sup_{(\pi_t, t\in\mathbb{N})} \mathbb{E}^\pi\left[\sum_{t=0}^\infty \gamma^t R_{t+2} \,\middle|\, X_0 = x, A_0 = a\right] \\ &\overset{A.1.1}{=} r(x,a) + \gamma \sup_{(\pi_t, t\in\mathbb{N})} \sum_{y\in\mathcal{X}} p(y \mid x, a)\mathbb{E}^\pi\left[\sum_{t=0}^\infty \gamma^t R_{t+2} \,\middle|\, X_0 = x, A_0 = a, X_1 = y\right] \\ &= r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y \mid x, a) \underbrace{\sup_{(\pi_t, t\in\mathbb{N})} \mathbb{E}^\pi\left[\sum_{t=0}^\infty \gamma^t R_{t+2} \,\middle|\, X_0 = x, A_0 = a, X_1 = y\right]}_{(*)}.\end{aligned}$$

The last equality follows from the fact, that the behaviours can condition on the entire history. In particular it can condition on $X_1 = y$ which means that moving the supremum into the sum does not actually open any room for improvement.

What is left to show, is that $(*)$ equals $V^*(y)$. To show this we use the same trick as in (1.12) twice

$$\begin{aligned}(*) &\overset{(1.12)}{=} \sup_{(\pi_t, t\geq 2)} \sup_{a_1\in\mathcal{A}_y} \mathbb{E}^\pi\left[\sum_{t=0}^\infty \gamma^t R_{t+2} \,\middle|\, X_0 = x, A_0 = a, X_1 = y, A_1 = a_1\right] \\ &\overset{\text{Markov}}{=} \sup_{(\pi_t, t\geq 2)} \sup_{a_1\in\mathcal{A}_y} \mathbb{E}^\pi\left[\sum_{t=0}^\infty \gamma^t R_{t+2} \,\middle|\, X_1 = y, A_1 = a_1\right] \\ &\overset{(1.12)}{=} \sup_{(\pi_t, t\geq 2)} \sup_{\pi_1} \mathbb{E}^\pi\left[\sum_{t=0}^\infty \gamma^t R_{t+2} \,\middle|\, X_1 = y\right] \\ &= \sup_{(\tilde{\pi}_t, t\in\mathbb{N}_0)} \mathbb{E}^{\tilde{\pi}}\left[\sum_{t=0}^\infty \gamma^t \tilde{R}_{t+1} \,\middle|\, \tilde{X}_0 = y\right] = V^*(y)\end{aligned}$$

Where $(\tilde{X}_t, \tilde{A}_t, \tilde{R}_{t+1}, t \in \mathbb{N}_0)$ is also a sequence generated by $\mathcal{M}$ with

$$\tilde{X}_t \coloneqq X_{t+1}, \quad \tilde{A}_t \coloneqq A_{t+1}, \quad \tilde{R}_t \coloneqq R_{t+1}, \quad \tilde{\pi}_t \coloneqq \pi_{t+1}. \qquad \square$$

With this Lemma it will be easy to show that the optimal value functions are fixed points of the following mappings.

Definition 1.4.5. The mapping $T^* \colon B(\mathcal{X}) \to B(\mathcal{X})$ with

$$T^*V(x) \coloneqq \sup_{a \in \mathcal{A}_x} \left\{ r(x,a) + \sum_{y \in \mathcal{X}} p(y \mid x, a) V(y) \right\} \qquad V \in B(\mathcal{X}),\ x \in \mathcal{X},$$

is called the *Bellman Optimality Operator*. With some more abuse of notation, define the mapping $T^* \colon B(\mathcal{X} \times \mathcal{A}) \to B(\mathcal{X} \times \mathcal{A})$ with

$$T^*Q(x,a) \coloneqq r(x,a) + \sum_{y \in \mathcal{X}} p(y \mid x, a) \sup_{a' \in \mathcal{A}_y} Q(y, a') \quad V \in B(\mathcal{X}),\ (x,a) \in \mathcal{X} \times \mathcal{A}.$$

Corollary 1.4.6. *All the optimal value functions are fixed points of T^*, i.e.*

$$T^*\tilde{V} = \tilde{V} \qquad T^*\tilde{Q} = \tilde{Q}$$
$$T^*V^* = V^* \qquad T^*Q^* = Q^*$$

Proof. We simply assemble the building blocks from the previous Lemma 1.4.4:

$$V^*(x) \overset{\text{(iii)}}{=} \sup_{a \in \mathcal{A}_x} Q^*(x,a) \overset{\text{(iv)}}{=} \sup_{a \in \mathcal{A}_x} \left\{ r(x,a) + \sum_{y \in \mathcal{X}} p(y \mid x, a) V^*(y) \right\} = T^*V^*(x)$$

The others are analogous. $\square$

Now we just need to show that the Bellman Optimality Operator satisfies the Banach fixed point theorem.

Theorem 1.4.7. *T^* satisfies the requirements of the Banach fixed point theorem for $\gamma < 1$, in particular*

$$V^*(x) = \tilde{V}(x)$$

is the unique fixed point of T^*. The suprema over all the other policy sets get sandwiched in between.

Proof. (Szepesvári 2010, p. 79) Since $B(\mathcal{X})$ and $B(\mathcal{X} \times \mathcal{A})$ are complete metric spaces and the mapping T^* maps onto itself we only need to show that it is a contraction. For that we first need to show for any two functions $f, g,$:

$$|\sup_{a \in \mathcal{A}_x} f(a) - \sup_{b \in \mathcal{A}_x} g(b)| \leq \sup_{a \in \mathcal{A}_x} |f(a) - g(a)| \tag{1.14}$$

To do that, we assume without loss of generality that the absolute value on the left side is itself (otherwise switch f and g):

$$\sup_{a\in\mathcal{A}_x} f(a) - \sup_{b\in\mathcal{A}_x} g(b) \leq \sup_{a\in\mathcal{A}_x} f(a) - g(a) \leq \sup_{a\in\mathcal{A}_x} |f(a) - g(a)|$$

Now let $V, W \in B(\mathcal{X})$, then we get a contraction:

$$\begin{aligned}
\|T^*V - T^*W\|_\infty &= \sup_{x\in\mathcal{X}} |T^*V(x) - T^*W(x)| \\
&\overset{(1.14)}{\leq} \sup_{x\in\mathcal{X}} \sup_{a\in\mathcal{A}_x} \left| \gamma \sum_{y\in\mathcal{X}} p(y \mid x, a)(V(y) - W(y)) \right| \\
&\leq \sup_{x\in\mathcal{X}} \sup_{a\in\mathcal{A}_x} \gamma \underbrace{\sum_{y\in\mathcal{X}} p(y \mid x, a)}_{=1} \|V - W\|_\infty \\
&= \gamma \|V - W\|_\infty
\end{aligned}$$

Similarly for $Q, P \in B(\mathcal{X} \times \mathcal{A})$:

$$\begin{aligned}
\|T^*Q - T^*P\|_\infty &= \sup_{x\in\mathcal{X}} \left| \gamma \sum_{y\in\mathcal{X}} p(y \mid x, a) \left(\sup_{a'\in\mathcal{A}_y} Q(y, a') - \sup_{b'\in\mathcal{A}_y} P(y, b') \right) \right| \\
&\overset{(1.14)}{\leq} \sup_{x\in\mathcal{X}} \gamma \sum_{y\in\mathcal{X}} p(y \mid x, a) \sup_{a'\in\mathcal{A}_y} |Q(y, a') - P(y, a')| \\
&\leq \gamma \|Q - P\|_\infty
\end{aligned}$$

□

Alternatively one can show that T^* fulfills Blackwell's condition for a contraction.

1.5 Optimal policies

Now that we proved the uniqueness of the optimal value functions, we need to ask the question whether this supremum can be attained with a single policy. And if not, if it can be approximated over all states by the same sequence of policies (c.f. Prop. 1.4.3).
Let us first consider the case where the supremum can be attained, since this includes the important special case of finite MDPs.

Proposition 1.5.1. *Be $\pi^* \in \Pi_S$, then the following statements are equivalent:*

(i) $\pi^* \in \Pi_S$ *is optimal ($V^* = V^{\pi^*}$),*

(ii) $\forall x \in \mathcal{X} : V^*(x) = \sum_{a\in\mathcal{A}_x} \pi^*(a \mid x) Q^*(x, a)$,

(iii) $\forall x \in \mathcal{X} : \pi^* = \arg\max_{\pi^*\in\Pi_S} \sum_{a\in\mathcal{A}_x} \pi^*(a \mid x) Q^*(x, a)$,

(iv) $\pi^*(a \mid x) > 0 \implies Q^*(x,a) = V^*(x) = \sup_{b \in \mathcal{A}_x} Q^*(x,b)$.

"Actions are concentrated on the set of actions that maximize $Q^*(x,\cdot)$."
(This also implies: $Q^*(x,a) < V^*(x) \implies \pi^*(a \mid x) = 0$.)

Proof. "(i) ⇒ (ii)" Let $x \in \mathcal{X}$ be arbitrary, then we can sandwich (1.15):

$$\begin{aligned} V^*(x) = V^{\pi^*}(x) &\overset{(1.4)}{=} \sum_{a \in \mathcal{A}_x} \pi^*(a \mid x) Q^{\pi^*}(x,a) \\ &\leq \sum_{a \in \mathcal{A}_x} \pi^*(a \mid x) Q^*(x,a) \\ &\leq \underbrace{\sum_{a \in \mathcal{A}_x} \pi^*(a \mid x)}_{=1} \sup_{b \in \mathcal{A}_x} Q^*(x,b) \\ &\overset{1.4.4}{=} V^*(x) \end{aligned} \tag{1.15}$$

"(ii) ⇒ (iii)" Let $\pi \in \Pi_S$, $(x,a) \in \mathcal{X} \times \mathcal{A}$ be arbitrary. Then with (1.4) we have

$$\sum_{a \in \mathcal{A}_x} \pi(a \mid x) Q^*(x,a) \leq \underbrace{\sum_{a \in \mathcal{A}_x} \pi(a \mid x)}_{=1} \sup_{b \in \mathcal{A}_x} Q^*(x,b) = V^*(x).$$

Therefore $V^*(x)$ is an upper bound for every $\pi \in \Pi_S$, and since π^* attains this upper bound it is a maximum.
"(iii) ⇒ (iv)" Be $\pi^*(a \mid x) > 0$ for some $a \in \mathcal{A}_x, x \in \mathcal{X}$. Then there exists no $b \in \mathcal{A}_x$ with $Q^*(x,b) > Q^*(x,a)$. Otherwise we can define the behaviour

$$\hat{\pi}(\cdot \mid x) \colon \begin{cases} \mathcal{A}_x \to [0,1] \\ c \mapsto \begin{cases} 0 & c = a \\ \pi^*(b \mid x) + \pi^*(a \mid x) & c = b \\ \pi^*(c \mid x) & \text{else.} \end{cases} \end{cases}$$

This results in a contradiction:

$$\begin{aligned} &\sum_{c \in \mathcal{A}_x} \hat{\pi}(c \mid x) Q^*(x,c) \\ &= [\pi^*(b \mid x) + \underbrace{\pi^*(a \mid x)}_{>0}] \underbrace{Q^*(x,b)}_{>Q^*(x,a)} + \sum_{c \in \mathcal{A}_x \setminus \{a,b\}} \pi^*(c \mid x) Q^*(x,c) \\ &> \sum_{c \in \mathcal{A}_x} \pi^*(c \mid x) Q^*(x,c) \quad \lightning \ \pi^* \text{ attains maximum} \end{aligned}$$

"(iv) ⇒ (ii)" Be $x \in \mathcal{X}$, since $\pi^*(\cdot \mid x)$ is a probability distribution on $\mathcal{A}_x$ there exists $a \in \mathcal{A}_x$ such that $\pi^*(a \mid x) > 0$. Define

$$M_x \coloneqq \{a \in \mathcal{A}_x : \pi^*(a \mid x) > 0\}.$$

Then $Q^*(x,a) = V^*(x)$ for all $a \in M_x$ by prerequisite, therefore

$$\begin{aligned} V^*(x) &= \sum_{a \in M_x} \pi^*(a \mid x) V^*(x) = \sum_{a \in M_x} \pi^*(a \mid x) Q^*(x,a) \\ &= \sum_{a \in \mathcal{A}_x} \pi^*(a \mid x) Q^*(x,a). \end{aligned}$$

"(ii) $\Rightarrow$ (i)" Using (iv) from Lemma 1.4.4 for an $x \in \mathcal{X}$:

$$\begin{aligned} V^*(x) &= \sum_{a \in \mathcal{A}_x} \pi^*(a \mid x) Q^*(x,a) \\ &= \sum_{a \in \mathcal{A}_x} \pi^*(a \mid x) \left(r(x,a) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x,a) V^*(y) \right) \\ &= \sum_{a \in \mathcal{A}_x} \pi^*(a \mid x) r(x,a) + \gamma \sum_{y \in \mathcal{X}} \sum_{a \in \mathcal{A}_x} \pi^*(a \mid x) p(y \mid x,a) V^*(y) \\ &\overset{(1.5)}{=} \mathbb{E}[r(x, A_0) \mid X_0 = x] + \gamma \sum_{y \in \mathcal{X}} \mathbb{P}(X_1 = y \mid X_0 = x) V^*(y) \\ &= T^{\pi^*} V^*(x) \end{aligned}$$

Therefore $V^{\pi^*} = V^*$ since the fixed point of T^{π^*} is unique (Thm. 1.3.5). □

Corollary 1.5.2. *If optimal stationary policy exists, an optimal deterministic stationary policy exists as well.*

Proof. Due to (iv) in Prop. 1.5.1 such a policy can be constructed by choosing one of the actions a for every $x \in \mathcal{X}$ which result in $Q^*(x,a) = V^*(x)$. These actions have to exist if there is a stochastic stationary policy which is optimal. □

Before we show, that if an arbitrary (history dependent) optimal policy exists, there also exists a stationary policy which is optimal, I want to note, that from (iii) follows a first heuristic for finding an optimal policy. This heuristic only works of course, if the maximum exists. But this is never a problem in finite MDPs.

Definition 1.5.3. $Q \colon \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ an action value function, $\tilde{\pi} \colon \mathcal{X} \to \mathcal{A}_x$ with

$$\tilde{\pi}(x) \coloneqq \arg\max_{a \in \mathcal{A}_x} Q(x,a), \qquad x \in \mathcal{X}.$$

$\tilde{\pi}(x)$ is called *greedy* with respect to Q in $x \in \mathcal{X}$. $\tilde{\pi}$ is called *greedy* w.r.t. Q.

Remark 1.5.4.

- 1.5.1(iv) implies that greedy w.r.t. Q^* is optimal. This means that knowledge of Q^* is sufficient to select the best action.

- 1.4.4 implies that knowledge of V^*, r, p is sufficient as well.

Now we show how we can construct an optimal stationary policy from an arbitrary optimal policy. We will break this problem into two parts. First we construct an optimal Markovian policy, and then we construct an optimal stationary policy from the optimal Markovian policy. The first step is taken from Puterman (2005, pp. 134–137).

Proposition 1.5.5 (Puterman)**.** *Let $\pi \in \Pi$. Then, for each $x \in \mathcal{X}$, there exists a Markovian policy π^x satisfying*

$$\mathbb{P}^{\pi^x}(X_t = y, A_t = a \mid X_0 = x) = \mathbb{P}^{\pi}(X_t = y, A_t = a \mid X_0 = x) \quad \forall t \in \mathbb{N}_0,$$

where $\mathbb{P}^\pi$ indicates that the sequence $((X_t, A_t, R_{t+1}), t \in \mathbb{N}_0)$ is constructed with policy π.

Proof. Fix $x \in \mathcal{X}$. For each $(y, a) \in \mathcal{X} \times \mathcal{A}$ define $\pi^x := (\pi^x_t, t \in \mathbb{N}_0)$ with

$$\pi^x_t(\cdot \mid y) := \mathbb{P}^\pi(A_t \mid X_t = y, X_0 = x). \tag{1.16}$$

We will now show the statement for this π^x by induction. The base case is:

$$\begin{aligned}\mathbb{P}^\pi(X_0 = y, A_0 = a \mid X_0 = x) &= \mathbb{P}^\pi(A_0 = a \mid X_0 = x)\delta_{xy} \\ &= \pi^x_0(a \mid x)\delta_{xy} = \mathbb{P}^{\pi^x}(A_0 = a \mid X_0 = x)\delta_{xy} \\ &= \mathbb{P}^{\pi^x}(X_0 = y, A_0 = a \mid X_0 = x)\end{aligned}$$

Assume the claim holds for $\{0, \dots, t-1\}$. Then using Markovian transition kernel

$$\mathbb{P}^\pi[X_t = y \mid (X_{t-1}, A_{t-1}) = (z, a), X_0 = x] = p(y \mid z, a),$$

together with A.1.1, we get:

$$\begin{aligned}\mathbb{P}^\pi(X_t = y \mid X_0 = x) &= \sum_{(z,a)\in\mathcal{X}\times\mathcal{A}} p(y \mid z, a)\mathbb{P}^\pi[(X_{t-1}, A_{t-1}) = (z, a) \mid X_0 = x] \\ &\overset{\text{ind.}}{=} \sum_{(z,a)\in\mathcal{X}\times\mathcal{A}} p(y \mid z, a)\mathbb{P}^{\pi^x}[(X_{t-1}, A_{t-1}) = (z, a) \mid X_0 = x] \\ &= \mathbb{P}^{\pi^x}(X_t = y \mid X_0 = x)\end{aligned}$$

Using A.1.1 again we get:

$$\begin{aligned}&\mathbb{P}^\pi[X_t = y, A_t = a \mid X_0 = x] \\ &= \underbrace{\mathbb{P}^\pi[A_t = a \mid X_t = y, X_0 = x]}_{=\pi^x_t(a|y)}\mathbb{P}^\pi[X_t = y \mid X_0 = x] \\ &= \mathbb{P}^{\pi^x}[A_t = a \mid X_t = y, X_0 = x]\mathbb{P}^{\pi^x}[X_t = y \mid X_0 = x] \\ &= \mathbb{P}^{\pi^x}[X_t = y, A_t = a \mid X_0 = x]\end{aligned}$$

□

Corollary 1.5.6 (Puterman)**.** *For (an optimal) policy $\pi \in \Pi$ exist Markovian policies π^x for all $x \in \mathcal{X}$ such that*

$$V^{\pi}(x) = V^{\pi^x}(x).$$

Proof.

$$\begin{aligned}
V^{\pi}(x) &= \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty}\gamma^t R_{t+1} \,\middle|\, X_0 = x\right] \\
&\overset{\text{Fubini}}{=} \sum_{t=0}^{\infty}\gamma^t \mathbb{E}^{\pi}[R_{t+1} \mid X_0 = x] \\
&= \sum_{t=0}^{\infty}\gamma^t \sum_{(y,a)\in\mathcal{X}\times\mathcal{A}} \mathbb{E}^{\pi}[R_{t+1} \mid X_t = y, A_t = a, X_0 = x] \\
&\qquad\qquad \cdot \mathbb{P}^{\pi}(X_t = y, A_t = a \mid X_0 = x) \\
&= \sum_{t=0}^{\infty}\gamma^t \sum_{(y,a)\in\mathcal{X}\times\mathcal{A}} \mathbb{E}[R_{(x,a)}]\mathbb{P}^{\pi}(X_t = y, A_t = a \mid X_0 = x) \\
&= \sum_{t=0}^{\infty}\gamma^t \sum_{(y,a)\in\mathcal{X}\times\mathcal{A}} \mathbb{E}[R_{(x,a)}]\mathbb{P}^{\pi^x}(X_t = y, A_t = a \mid X_0 = x)
\end{aligned}$$

The same transformations with π^x instead of π backwards yield $V^{\pi}(x) = V^{\pi^x}(x)$. □

Using the results from Puterman we can now show:

Theorem 1.5.7. *Let $\pi \in \Pi$ be an optimal policy (i.e. $V^* = V^{\pi}$), then there exists an optimal stationary policy $\hat{\pi}$.*

Proof. Because of Prop. 1.5.5, for the optimal policy π exist Markovian policies π^x for $x \in \mathcal{X}$ with

$$V^*(x) = V^{\pi}(x) = V^{\pi^x}(x).$$

From these Markovian policies we can define the stationary policy $\hat{\pi}$ with

$$\hat{\pi}(\cdot \mid x) \coloneqq \pi_0^x(\cdot \mid x) \qquad x \in \mathcal{X}.$$

For which we need to show that it is optimal. First, we have:

$$\begin{aligned}
V^*(x) = V^{\pi^x}(x) &= \mathbb{E}^{\pi^x}[R_1 \mid X_0 = x] + \gamma\mathbb{E}^{\pi^x}\left[\sum_{t=0}^{\infty}\gamma^t R_{t+2} \,\middle|\, X_0 = x\right] \\
&= \sum_{a\in\mathcal{A}_x} \mathbb{E}[R_{(x,a)}]\pi_0^x(a \mid x) \\
&\qquad + \gamma\sum_{y\in\mathcal{X}} \mathbb{P}^{\pi^x}(X_1 = y \mid X_0 = x)\mathbb{E}^{\pi^x}\left[\sum_{t=0}^{\infty}\gamma^t R_{t+2} \,\middle|\, X_1 = y, X_0 = x\right]
\end{aligned} \tag{1.17}$$

Where the conditional probability equals:

$$\begin{aligned}
&\mathbb{P}^{\pi^x}(X_1 = y \mid X_0 = x) \\
&= \sum_{a \in \mathcal{A}_x} \mathbb{P}^{\pi^x}(X_1 = y, A_0 = a \mid X_0 = x) \\
&= \sum_{a \in \mathcal{A}_x} \mathbb{P}^{\pi^x}(X_1 = y \mid X_0 = x, A_0 = a)\mathbb{P}^{\pi^x}(A_0 = a \mid X_0 = x) \\
&= \sum_{a \in \mathcal{A}_x} p(y \mid x, a)\pi_0^x(a \mid x) = \sum_{a \in \mathcal{A}_x} p(y \mid x, a)\hat{\pi}(a \mid x) \\
&= \mathbb{P}^{\hat{\pi}}(X_1 = y \mid X_0 = x)
\end{aligned} \tag{1.18}$$

And the conditional expectation is:

$$\begin{aligned}
&\mathbb{E}^{\pi^x}\left[\sum_{t=0} \gamma^t R_{t+2} \,\middle|\, X_1 = y, X_0 = x\right] \\
&= \sum_{a \in \mathcal{A}_y} \mathbb{E}^{\pi^x}\left[\sum_{t=0} \gamma^t R_{t+2} \,\middle|\, X_1 = y, A_1 = a, X_0 = x\right] \pi_1^x(a \mid y) \\
&\overset{\text{Markov}}{=} \sum_{a \in \mathcal{A}_y} \mathbb{E}^{\pi^x}\left[\sum_{t=0} \gamma^t R_{t+2} \,\middle|\, X_1 = y, A_1 = a\right] \pi_1^x(a \mid y) \\
&= \mathbb{E}^{\pi^x}\left[\sum_{t=0} \gamma^t R_{t+2} \,\middle|\, X_1 = y\right] = \mathbb{E}^{\tilde{\pi}^x}\left[\sum_{t=0} \gamma^t \tilde{R}_{t+1} \,\middle|\, \tilde{X}_0 = y\right] \\
&= V^{\tilde{\pi}^x}(y)
\end{aligned} \tag{1.19}$$

Where we renamed the variables

$$\tilde{X}_t \coloneqq X_{t+1}, \qquad \tilde{A}_t \coloneqq A_{t+1}, \qquad \tilde{R}_t \coloneqq R_{t+1}, \qquad \tilde{\pi}_t^x = \pi_{t+1}^x,$$

creating the sequence $((\tilde{X}_t, \tilde{A}_t, \tilde{R}_{t+1}), t \in \mathbb{N}_0)$ consistent with the MDP together with policy $\tilde{\pi}^x$. And if it where not the case, that

$$V^{\tilde{\pi}^x}(z) = V^*(z) \qquad \forall z \in \mathcal{X} : \mathbb{P}^{\pi^x}(X_1 = z \mid X_0 = x) > 0, \tag{1.20}$$

we can use the property of the policy π^z

$$V^{\pi^z}(z) = V^*(z),$$

and improve the optimal policy π^x by swapping out the policy $(\pi_t^x, t \in \mathbb{N})$ with $(\pi_{t-1}^z, t \in \mathbb{N})$ conditional on $X_0 = x$ and $X_1 = z$ (losing the Markov property in the process):

$$\begin{aligned}
&\hat{\pi}_0^x(a \mid x) && \coloneqq \pi_0^x(a \mid x) && \\
&\hat{\pi}_t^x(a \mid X_0 = x, X_1 = z, X_t = x_t) && \coloneqq \pi_{t-1}^z(a \mid x_t) && \text{for } t \geq 1 \\
&\hat{\pi}_t^x(a \mid X_0 = x, X_1 \neq z, X_t = x_t) && \coloneqq \pi_t^x(a \mid x_t) && \text{for } t \geq 1 \\
&\hat{\pi}_t^x(a \mid y) && \coloneqq \pi_t^x(a \mid y) && \text{for } y \neq x
\end{aligned}$$

And if the same steps are applied to $\hat{\pi}^x$ as to π^x in (1.17), everything but the summand z in the large sum will stay the same. And since we assumed (1.20) was false, this summand increases because

$$V^*(z)\mathbb{P}^{\pi^x}(X_1 = z \mid X_0 = x) > V^{\tilde{\pi}^x}(z)\mathbb{P}^{\pi^x}(X_1 = z \mid X_0 = x).$$

But that implies that $V^*(x) < V^{\hat{\pi}^x}(x)$, which is a contradiction. Therefore (1.20) holds. Using this fact, we can conclude:

$$\begin{aligned} V^*(x) &= V^{\pi^x}(x) \\ &= \sum_{a \in \mathcal{A}_x} \mathbb{E}[R_{(x,a)}]\pi_0^x(a \mid x) + \gamma \sum_{y \in \mathcal{X}} \mathbb{P}^{\pi^x}(X_1 = y \mid X_0 = x) V^{\tilde{\pi}^x}(y) \\ &= \sum_{a \in \mathcal{A}_x} \mathbb{E}[R_{(x,a)}]\hat{\pi}(a \mid x) + \gamma \sum_{y \in \mathcal{X}} \mathbb{P}^{\hat{\pi}}(X_1 = y \mid X_0 = x) V^*(y) \\ &= T^{\hat{\pi}} V^*(x) \end{aligned}$$

Since V^* is a fixed point of $T^{\hat{\pi}}$ and the fixed point is unique we have completed our proof, since then $V^{\hat{\pi}} = V^*$ which means $\hat{\pi}$ is optimal. □

Corollary 1.5.8. *If an arbitrary optimal policy exists, an optimal deterministic stationary policy exists.*

Proof. Thm. 1.5.7 + Cor. 1.5.2 □

Therefore the set of deterministic stationary policies is large enough to choose from if you are looking for an optimal policy. But what happens if there are no optimal policies? If there are no optimal policies, can one sequence of deterministic policies get arbitrarily close to the optimal value function for *every* starting state? We are now able to answer the question we were not quite ready to answer in Finite Outmatching (c.f Prop. 1.4.3).

Proposition 1.5.9 (ε-Optimal Policies)**.** *For every $\varepsilon > 0$ exists a deterministic stationary policy π^ε such that $V^* \leq V^{\pi_\varepsilon} + \varepsilon$.*

Proof. Be $\varepsilon > 0$ arbitrary, define $\delta := \varepsilon(1 - \gamma)$. Because of this

$$V^*(x) = T^*V^*(x) = \sup_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x, a) V^*(y) \right\},$$

there exists an $\pi^\varepsilon(x)$ for all $x \in \mathcal{X}$ such that

$$r(x, \pi^\varepsilon(x)) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x, \pi^\varepsilon(x)) V^*(y) + \delta \geq V^*(x). \tag{1.21}$$

This defines a mapping $\pi^\varepsilon \colon \mathcal{X} \to \mathcal{A}_x$ with this property

$$\begin{aligned} T^{\pi^\varepsilon} V^*(x) &= r(x, \pi^\varepsilon(x)) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x, \pi_\varepsilon(x)) V^*(x) \\ &\geq V^*(x) - \delta = (V^* - \delta \mathbb{1})(x). \end{aligned} \tag{1.22}$$

By induction we get

$$(T^{\pi^\varepsilon})^n V^*(x) \geq \left(V^* - \left(\delta \sum_{k=0}^{n} \gamma \right) \mathbb{1} \right)(x).$$

Using the monotonicity of T^{π^ε} (Lemma 1.3.6) we can make the induction step:

$$\begin{aligned}
(T^{\pi^\varepsilon})^{n+1} V^*(x) &= T^{\pi^\varepsilon} (T^{\pi^\varepsilon})^{n-1} V^*(x) \\
&\overset{\text{ind.}}{\geq} T^{\pi^\varepsilon} \left(V^* - \left(\delta \sum_{k=0}^{n} \gamma^k \right) \mathbb{1} \right)(x) \\
&\overset{\text{affine}}{=} T^{\pi^\varepsilon} V^*(x) - \gamma \sum_{y \in \mathcal{X}} p(y \mid x, a) \left(\delta \sum_{k=0}^{n} \gamma^k \right) \mathbb{1}(y) \\
&\overset{(1.22)}{\geq} [V^*(x) - \delta] - \delta \sum_{k=0}^{n} \gamma^{k+1} \sum_{y \in \mathcal{X}} p(y \mid x, a) \\
&= V^*(x) - \delta \sum_{k=0}^{n+1} \gamma^k
\end{aligned}$$

This concludes our proof with:

$$\begin{aligned}
V^{\pi^\varepsilon}(x) &= \lim_{n \to \infty} (T^{\pi^\varepsilon})^n V^* \geq V^*(x) - \delta \lim_{n \to \infty} \sum_{k=0}^{n} \gamma \\
&= V^*(x) - \delta/(1-\gamma) = V^*(x) - \varepsilon
\end{aligned}$$

□

1.6 Dynamic Programming

If we have full knowledge of all transition probabilities and immediate rewards, we can use *value iteration* and *policy iteration* to approach the optimal value functions. These methods developed by Richard Bellman are known as "Dynamic Programming" as they recursively break down the problem using the Bellman equations. Since they use estimates of V^* to create better estimates, they are also a case of *bootstrapping*.

Definition 1.6.1. Let $V_0 \in B(\mathcal{X})$ or $Q_0 \in B(\mathcal{X} \times \mathcal{A})$ be arbitrary, then $(V_k, k \in \mathbb{N}_0)$ or $(Q_k, k \in \mathbb{N}_0)$ are called *value iterations* if

$$V_{k+1} \coloneqq T^* V_k \qquad \text{or} \qquad Q_{k+1} \coloneqq T^* Q_k$$

This converges geometrically with regard to the $\|\cdot\|_\infty$-norm, due to T^* fulfilling the requirements of the BFT (Thm. 1.4.7).
To find an ε-optimal policy, define $\delta \coloneqq \varepsilon(1-\gamma)$ as in Prop. 1.5.9. This iteration can then be stopped when $\|V_k - V^*\|_\infty \leq \delta/2$, which also implies

$$\|T^* V_k - V^*\|_\infty = \|T^* V_k - T^* V^*\|_\infty \leq \gamma\delta/2 \leq \delta/2.$$

So picking π^ε with regard to V_k instead of V^* like in Prop. 1.5.9, results in

$$\begin{aligned} r(x,\pi^\varepsilon(x)) + \gamma \sum_{y\in\mathcal{X}} p(y \mid x,\pi^\varepsilon(x)) V_k(y) + \delta/2 &\geq T^* V_k(x) + \delta/2 \\ &\geq V^*(x) + \delta \end{aligned}$$

From there the proof is the same as from (1.21).
The other iteration, policy iteration, works if a greedy policy can always be selected. Since this requires a supremum over actions to be attained, this can not be guaranteed in general. Though for finite MDPs this is always the case.

Definition 1.6.2. Let $\pi_0 \in \Pi_S^D$ be an arbitrary deterministic stationary policy. Then $(\pi_k, k \in \mathbb{N}_0)$ is called *policy iteration* if

$$\pi_{k+1} \in \Pi_S^D \text{ is greedy with regard to } Q^{\pi_k}.$$

Proposition 1.6.3. *Let $(\pi_k, k \in \mathbb{N}_0)$ be a policy iteration, then*

$$\|Q^{\pi_k} - Q^*\|_\infty \to 0 \quad (k \to \infty).$$

Proof. Let $(\pi_k, k \in \mathbb{N}_0)$ be a policy iteration. Then by definition

$$\pi_{k+1}(x) = \arg\max_{a\in\mathcal{A}_x} Q^{\pi_k}(x,a), \tag{1.23}$$

which means:

$$\begin{aligned} V^{\pi_k} &\leq T^* V^{\pi_k}(x) \\ &= \sup_{a\in\mathcal{A}_x} \left\{ r(x,a) + \gamma \sum_{y\in\mathcal{X}} p(y \mid x,a) V^{\pi_k}(y) \right\} \\ &\overset{1.3.2}{=} \sup_{a\in\mathcal{A}_x} Q^{\pi_k}(x,a) \\ &\overset{(1.23)}{=} Q^{\pi_k}(x,\pi_{k+1}(x)) \\ &= T^{\pi_{k+1}} V^{\pi_k}(x) \end{aligned} \tag{1.24}$$

Because of the monotonicity of $T^{\pi_{k+1}}$ (Lemma 1.3.6) this implies

$$V^{\pi_{k+1}} = \lim_{n\to\infty} (T^{\pi_{k+1}})^n V^{\pi_k} \geq T^{\pi_{k+1}} V^{\pi_k} = T^* V^{\pi_k}. \tag{1.25}$$

Which in turn implies by induction $V^{\pi_k} \geq (T^*)^k V^{\pi_0}$, and therefore:

$$\lim_{k\to\infty} \|V^* - V^{\pi_k}\|_\infty \leq \lim_{k\to\infty} \|V^* - (T^*)^k V^{\pi_0}\|_\infty \overset{1.4.7}{=} 0$$

This is easily translated into the action value version:

$$\begin{aligned} \|Q^* - Q^{\pi_k}\|_\infty &\overset{1.3.2}{\leq} \sup_{(x,a)\in\mathcal{X}\times\mathcal{A}} \left| \gamma \sum_{y\in\mathcal{X}} p(y \mid x,a)(V^*(y) - V^{\pi_k}(y)) \right| \\ &\leq \gamma \|V^* - V^{\pi_k}\|_\infty \end{aligned}$$

□

Policy iteration converges faster than value iteration with every step (c.f. (1.25)), and in the case of finite MDPs it actually converges in finitely many steps. And while it requires the calculation of Q^{π_k} in every step, this turns out to be less of a problem than one might think:

Corollary 1.6.4.

(i) *Policy iteration in finite MDPs converges in finitely many steps*

(ii) V^π *(and thus* Q^π*) can be calculated in one step*

Proof. (i) (Szepesvári 2010) Consider equation (1.24).
If it is an equality, then the current step is optimal since the value function is a fixed point of T^*.
If it is a true inequality, the value function truly increases in this iteration. But since there are only a finite set of deterministic stationary policies on finite MDPs

$$|\Pi_S^D| = |\{\pi : \mathcal{X} \to \mathcal{A}_x\}| = \prod_{x\in\mathcal{X}} |\mathcal{A}_x| < \infty$$

there is only a finite set of value functions V^π which means that it has to stop increasing after a finite number of steps.
(ii) Since the finite case is analogous to the (countably) infinite case, but easier to write down and understand, we will prove the finite case only. Let $\mathcal{X} = \{x_1, \dots, x_n\}$ be the finite state set. Then we can interpret $V\colon \mathcal{X} \to \mathbb{R}$ as a vector of $\mathbb{R}^n$ and define

$$r^\pi \coloneqq (r(x_1, \pi(x_1)), \dots, r(x_n, \pi(x_n)))^T \in \mathbb{R}^n$$
$$P^\pi \coloneqq \begin{pmatrix} p(x_1 \mid x_1, \pi(x_1)) & \cdots & p(x_n \mid x_1, \pi(x_1)) \\ \vdots & & \vdots \\ p(x_1 \mid x_n, \pi(x_n)) & \cdots & p(x_n \mid x_n, \pi(x_n)) \end{pmatrix}$$

This allows us to write

$$T^\pi V = r^\pi + \gamma P^\pi V.$$

And since the Banach fixed point iteration can be started with any bounded value function, it can in particular be started with the zero function or rather vector. Therefore by induction with induction basis ($k = 1$)

$$(T^\pi)^k 0 = r^\pi = \sum_{i=0}^{k-1} (\gamma P^\pi)^i r^\pi,$$

and induction step ($k \to k+1$)

$$(T^\pi)^{k+1} 0 = T^\pi \left(\sum_{i=0}^{k-1} (\gamma P^\pi)^i r^\pi \right) = r^\pi + \sum_{i=0}^{k-1} (\gamma P^\pi)^{i+1} r^\pi = \sum_{i=0}^{k} (\gamma P^\pi)^i r^\pi,$$

we get

$$V^\pi = \sum_{k=0}^{\infty} (\gamma P^\pi)^k r^\pi = (1 - \gamma P^\pi)^{-1} r^\pi. \qquad \square$$

1.6.1 Bootstrapping and Discounting

We used discounted MDPs for most of this chapter. $\gamma < 1$ was essential for the contraction property of the fixed point equation, which allowed us to apply the Banach fixed point theorem to it. But this requirement is not just sufficient but also necessary to ensure a unique fixed point in general. Imagine an MDP with just one state and no rewards. Then the recursion would be

$$V(x) = 0 + \sum_{y \in \{x\}} 1 \cdot V(y)$$

for V^π and V^*. Therefore any kind of real value for $V(x)$ is a fixed point. Methods which use some form of bootstrapping (i.e. recurrence) usually break down in the undiscounted case for this very reason. The expectation of value about a state can simply never be disproven, because the algorithm just assumes that the reward or punishment is just delayed and will come sooner or later. So even in episodic MDPs where finiteness of the entire return is guaranteed even in the undiscounted case, we can not expect to be able to use recursion to find the respective value functions. The first algorithm in the second chapter, Monte Carlo, will also be the only algorithm presented, which does not utilize bootstrapping, it is thus the only stable algorithm presented for $\gamma = 1$.

While discounting might seem unnatural at first, it is standard in economic theory. Asking anyone whether they want to receive some benefit now or later, all other things being equal, will be answered almost definitely with "now". It is also the reason why anyone would accept credit, receiving less money now than they pay back later. So if we want a machine to behave as we would expect (e.g. prefer checkmate in 2 turns over checkmate in 5 turns), discounting is in fact quite natural. Adjusting the discount parameter reflects an adjustment of the impatience of the algorithm. An individual will be indifferent between spending x now, and $(1 + r)x$ later, if their discount parameter is equal to

$$\gamma = \frac{1}{1 + r}.$$

And since one can only borrow at some rate r, if someone else is willing to lend at that rate, the interest rate will reflect the average discount parameter in equilibrium. Reality is surely a little bit more complex, but taking some interest rate in order to calculate γ can be a good rule of thumb. Interest rates between 1% and 10% result in discount parameters between 0.91 and 0.99.

Chapter 2

Reinforcement Learning Algorithms

2.1 Introduction

Dynamic Programming usually breaks down in the real world for two reasons:

1. The transition probabilities and immediate rewards are not known or hard to calculate.

2. The state and action space is too large to even compute one iteration of Dynamic Programming for every state-action tuple (e.g. possible positions and possible moves in every position in chess).

This is where *Reinforcement Learning* algorithms come in, which try to find solutions without having to sweep the entire state space. In this chapter based on Sutton and Barto (1998) (their second edition (2018) was used for statements which need to be up-to-date) we will examine advantages and disadvantages of various algorithms and discuss possible variations and extensions. In the next chapter we will attempt a proof of almost sure convergence of the basic algorithms introduced in this chapter, and illustrate their connection to stochastic approximation.
We separate their introduction and the convergence proofs, because – while the guarantee of almost sure convergence is reassuring – it is of little use for comparing various algorithms. Since one of the reasons for moving away from Dynamic Programming in the first place was, that we could not calculate the value function for the entire state space within a reasonable time frame, as the size of the state space is often too large for that. Therefore almost sure convergence should not be viewed as more than an entry requirement. For this reason most papers compare algorithms empirically on various example problems. And for some of the more complex algorithms convergence proofs simply do not exist yet.

So since the theoretical convergence properties are usually only ever an afterthought, it is more natural to introduce the various algorithms heuristically, explaining what specific problems they try to address with examples.

2.1.1 Naive Batch Learning

Let us ignore the second problem of large state spaces for a moment and consider the case where we only have the first problem (p and r unknown). Then we can learn from a sample $((X_t, A_t, Y_t, R_t), t \in \{0, \dots, T\})$ with

$$(Y_t, R_t) \sim \mathcal{P}(\cdot \mid X_t, A_t)$$

with the restriction $Y_t = X_{t+1}$ if the sample is generated sequentially by a behaviour. But there is no reason not to allow this more general sample which might be useful in cases where you can jump around in the state space and try different transitions at will.
We can then use this batch of transitions to calculate estimators $\hat{p}, \hat{r}$ for the state transitions and immediate rewards $\hat{r}$. And use Dynamic Programming on these estimators.

Algorithm 1 Naive Batch Learning Algorithm

1. Generate the history $(X_t, A_t, Y_t, R_t, t \in \{0, \dots, T\})$

```
for (y, x, a) ∈ 𝒳 × 𝒳 × 𝒜 do                          ▷ initialize variables
    rewards[x, a]←list()
    stateTransitions[y | x, a]← 0
    totalTransitions[x, a]← 0
end for
for t = 0, ..., T do
    rewards[X_t, A_t].append(R_{t+1})
    stateTransitions[Y_t | X_t, A_t]++
    totalTransitions[X_t, A_t]++
end for
for (x, a) ∈ 𝒳 × 𝒜 do
    r̂(x, a) ← average(rewards[x,a])
    for y ∈ 𝒳 do
        p̂(y | x, a) ← stateTransitions[y | x, a]/totalTransitions[x, a]
    end for
end for
```

2. Use Dynamic Programming on $\hat{r}$, $\hat{p}$

If the batch was generated by an exploration policy we separated the *exploration* from the later *exploitation*. The methods which do that are often grouped under the term *Batch Reinforcement Learning* or *off-line* learning.

This method works fine, if the state space is small and one can sample enough observations for every state and action in a reasonable timeframe. But if our state space is too large for that, it is impractical to wait for this.

2.2 Monte Carlo

One idea to tackle larger state spaces is, that it is often unnecessary to know the value function in every state.
To visualize this idea it is useful to imagine the state space to be a room the agent has to navigate.

	γ^1	γ^0	G	
	γ^2			
	γ^3			
	γ^4			
	S			

The state space $\mathcal{X}$ are the tiles on the floor, the actions $\mathcal{A}$ are the adjacent tiles, where the next state is with probability one equal to the action, and the reward is 0 for every transition but the transition to the goal (G) where the reward is 1 and the game ends. The start state is S. The value of choosing a tile is indicated in grey on the tile for a selection of tiles.

It is often enough for the agent to know the action value function for the states on his path, without knowing the value of states in the corners. Since he can then follow these "breadcrumbs" to find the goal reliably again. And it will quickly stop walking in circles if it goes into the direction of the highest value next time.
This idea is the basis for Monte Carlo algorithms. To make notation more brief we will introduce the algorithm to calculate the value function, the action value function will be analogous. Consider an episodic MDP and a behaviour π, then

$$\sum_{t=0}^{\infty} \gamma^t R_{t+1} = \sum_{t=0}^{T} \gamma^t R_{t+1}$$

is a bias free estimator for $V^\pi(X_0)$ for episode length T. As the definition of V^π was

$$V^\pi(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \,\middle|\, X_0 = x \right],$$

and because of the Markov property,

$$\sum_{t=k}^{T} \gamma^t R_{t+1}$$

Algorithm 2 First-visit Monte Carlo

for $x \in \mathcal{X}$ **do** ▷ initializing
    Returns$(x) \leftarrow$ list()
    $V^\pi(x) \leftarrow 0$
end for
while true **do** (forever) ▷ learning
    Generate an episode $((X_t, R_{t+1}), t \in \{0, \dots, T\})$ with behaviour π
    for $x \in \{X_0, \dots, X_T\}$ **do**
        $k \leftarrow \min\{t \mid X_t = x\}$
        Returns(x).append$(\sum_{t=k}^{T} \gamma^t R_{t+1})$
        $V^\pi(x) \leftarrow$ average(Returns(x))
    end for
end while

is a bias free estimator for $V^\pi(X_k)$. *First-visit Monte Carlo* uses the return after the first visit of a state $x \in \{X_1, \dots, X_t\}$ as an estimator for $V^\pi(x)$.
Because of the strong law of large numbers, first-visit Monte Carlo algorithm causes the value function estimation $\hat{V}^\pi(x)$ to converge with probability 1 to $V^\pi(x)$, for every state $x \in \mathcal{X}$ which is visited with positive probability given a stationary behaviour π and starting distribution X_0. This is due to the fact, that only one estimate for $V^\pi(x)$ for an $x \in \mathcal{X}$ is used per episode, so all estimates are independent. And since the return after the first visit to x has the same distribution as the return conditional on the starting state being x (A.1.8), these estimates are iid with expected value $V^\pi(x)$.
Every-visit Monte Carlo uses the Return after every visit of a state x to estimate $V^\pi(x)$. Since this means that the tail of the rewards can be included in multiple returns, the returns are not independent from each other anymore which make proving convergence a little bit more difficult than simply applying the strong law of large numbers. Though every visit Monte Carlo will turn out to be a special case of TD(λ) (c.f. (2.9), which lets us sketch some convergence proof ideas in Chapter 3.
The same method can be applied to learn the action value function Q^π. In this case we take the returns following a state *and* action as estimators for Q^π. But if the model is known and our problem is just a large state space calculating V^π is preferable, since $|\mathcal{X}| \leq |\mathcal{X} \times \mathcal{A}|$ means that the first algorithm needs fewer observations to converge and Q^π can be calculated with V^π given r and p.

2.2.1 From π to π^*

Let us assume exploring starts, i.e.

$$\mathbb{P}(X_0 = x) > 0 \qquad \forall x \in \mathcal{X},$$

for a moment. Then Monte Carlo converges whatever policy we select. Similar to policy iteration (Def. 1.6.2) we can then alternate between calculating Q^{π_n} and selecting π_{n+1} as a greedy policy with regard to Q^{π_n}. This is referred to as generalized policy iteration (GPI). If we would wait for our Monte Carlo approximation of Q^{π_n} to converge, π_n would converge to π^* for the same reason as policy iteration converges. But remember we are doing Monte Carlo in the first place, because the state space is too large to wait for an evaluation of every state. Which means in practice algorithms alternate between choosing a greedy policy with regard to V^{π_n} and generating an episode with policy π_n, averaging the estimates from this episode with the estimates of $V^{\pi_{n-1}}$. But since we have to assume

$$V^{\pi_{n-1}} \neq V^{\pi_n}$$

in general, using these old estimates is not bias free. It is easy to see that this procedure can only converge to π^* if it converges at all. Since there are only a finite number of deterministic stationary policies it would have to stay constant at some point, but for a constant policy Monte Carlo will converge, and then the policy can only stay constant if it is greedy with regards to its own value function, which forces it to be optimal (1.24). But it is still an open problem whether or not this alternating procedure converges at all (Sutton and Barto 2018, p. 99).
If we remove the assumption of exploring starts, we still need to ensure that every state is visited with positive probability for MC to converge. This requires the policy to do the exploring. There are multiple approaches to exploration which we will discuss later (Section 2.7). We will discuss the most straightforward example here, under the assumption that we can calculate Q^{π_n} somehow.

Definition 2.2.1. A stationary policy π is called

- *soft*, if it fulfils

$$\pi(a \mid x) > 0 \qquad \forall (x, a) \in \mathcal{X} \times \mathcal{A},$$

- ε-*soft* for some $1 \geq \varepsilon > 0$, if it fulfils

$$\pi(a \mid x) > \frac{\varepsilon}{|\mathcal{A}_x|} \quad \forall (x, a) \in \mathcal{X} \times \mathcal{A},$$

- ε-*greedy* with regard to Q, if it selects the greedy action w.r.t. Q with probability $(1-\varepsilon)$ and a (uniform) random action with probability ε, i.e.

$$\pi(a \mid x) = \begin{cases} (1-\varepsilon) + \frac{\varepsilon}{|\mathcal{A}_x|} & a \text{ is greedy}^1 \\ \frac{\varepsilon}{|\mathcal{A}_x|} & a \text{ is not greedy.} \end{cases}$$

[1] w.l.o.g. only one greedy action, otherwise pick one or mix

Note that an ε-greedy policy is ε-soft.
An ε-soft policy π^* is called *ε-soft optimal* if

$$V^{\pi^*}(x) = \sup_{\pi\ \varepsilon\text{-soft}} (x)V^{\pi} =: \tilde{V}^*(x).$$

Proposition 2.2.2. *Generalized policy iteration with ε-greedy policies converges to an ε-soft optimal policy*

Proof. To use the same argument as with standard policy iteration (1.24), we would need

$$T^*(V^{\pi_n}) = T^{\pi_{n+1}}(V^{\pi_n}).$$

But this is not true. To circumvent this problem we "move the requirement of ε-softness into the environment". I.e. we define an MDP $\tilde{M}$ where

$$\tilde{\mathcal{P}}(\cdot \mid x, a) \coloneqq (1-\varepsilon)\mathcal{P}(\cdot \mid x, a) + \frac{\varepsilon}{|\mathcal{A}_x|} \sum_{b \in \mathcal{A}_x} \mathcal{P}(\cdot \mid x, b).$$

This means, that with probability ε, the transition kernel will ignore the selected action and behave as if an uniformly random action was chosen.
We can transform ε-soft policies π from the old MDP to stationary policies $\tilde{\pi}$ of $\tilde{M}$ with a mapping f, where

$$f(\pi)(a \mid x) = \tilde{\pi}(a \mid x) \coloneqq \frac{\pi(a \mid x) - \frac{\varepsilon}{|\mathcal{A}_x|}}{1-\varepsilon} \geq 0.$$

And for every stationary policy $\tilde{\pi}$ of $\tilde{M}$ we can define the ε-soft policy π by

$$\pi(a \mid x) \coloneqq (1-\varepsilon)\tilde{\pi}(a \mid x) + \frac{\varepsilon}{|\mathcal{A}_x|},$$

which is the inverse mapping. Therefore f is a bijection between the ε-soft policies in the old MDP and all stationary policies in the new MDP. We now show, that the Value functions V^{π} stay invariant with regard to this mapping. First note:

$$\begin{aligned}\tilde{p}(y \mid x, a) &= \tilde{\mathcal{P}}(\{y\} \times \mathbb{R} \mid x, a) \\ &= (1-\varepsilon)p(y \mid x, a) + \frac{\varepsilon}{|\mathcal{A}_x|} \sum_{b \in \mathcal{A}_x} p(y \mid x, b)\end{aligned}$$

And together with $q(B \mid x, a) \coloneqq \mathcal{P}(\mathcal{X} \times B \mid x, a)$, the complementary marginal distribution which hast the same property, we can show:

$$\begin{aligned}\tilde{r}(x, a) &= \int t\ d\tilde{q}(t \mid x, a) = \int t\ d \left((1-\varepsilon)q(\cdot \mid x, a) + \frac{\varepsilon}{|\mathcal{A}_x|} \sum_{b \in \mathcal{A}_x} q(\cdot \mid x, b) \right)(t) \\ &= (1-\varepsilon) \int t\ dq(t \mid x, a) + \frac{\varepsilon}{|\mathcal{A}_x|} \sum_{b \in \mathcal{A}_x} \int t\ dq(t \mid x, a) \\ &= (1-\varepsilon)r(x, a) + \frac{\varepsilon}{|\mathcal{A}_x|} \sum_{b \in \mathcal{A}_x} r(x, b)\end{aligned}$$

Therefore we know:

$$\begin{aligned}
&\sum_{a\in\mathcal{A}_x} \tilde{\pi}(a \mid x)\tilde{r}(x,a) \\
&= \sum_{a\in\mathcal{A}_x} \left(\frac{\pi(a \mid x) - \frac{\varepsilon}{|\mathcal{A}_x|}}{1-\varepsilon}\right) \left((1-\varepsilon)r(x,a) + \frac{\varepsilon}{|\mathcal{A}_x|}\sum_{b\in\mathcal{A}_x} r(x,b)\right) \\
&= \sum_{a\in\mathcal{A}_x} \left(\pi(a \mid x) - \frac{\varepsilon}{|\mathcal{A}_x|}\right) r(x,a) + \frac{\varepsilon}{|\mathcal{A}_x|}\sum_{b\in\mathcal{A}_x} r(x,b) \underbrace{\sum_{a\in\mathcal{A}_x} \frac{\pi(a \mid x) - \frac{\varepsilon}{|\mathcal{A}_x|}}{1-\varepsilon}}_{=1} \\
&= \sum_{a\in\mathcal{A}_x} \pi(a \mid x) r(x,a)
\end{aligned}$$

And similarly:

$$\begin{aligned}
&\mathbb{P}^{\tilde{\pi}}(\tilde{X}_1 = y \mid \tilde{X}_0 = x) = \sum_{a\in\mathcal{A}_x} \tilde{\pi}(a \mid x)\tilde{p}(y \mid x,a) \\
&= \sum_{a\in\mathcal{A}_x} \left(\frac{\pi(a \mid x) - \frac{\varepsilon}{|\mathcal{A}_x|}}{1-\varepsilon}\right) \left((1-\varepsilon)p(y \mid x,a) + \frac{\varepsilon}{|\mathcal{A}_x|}\sum_{b\in\mathcal{A}_x} p(y \mid x,b)\right) \\
&= \ldots = \sum_{a\in\mathcal{A}_x} \pi(a \mid x) p(y \mid x,a) = \mathbb{P}^{\pi}(X_1 = y \mid X_0 = x)
\end{aligned}$$

The last two equations imply together:

$$\begin{aligned}
\tilde{T}^{\tilde{\pi}} V^{\pi}(x) &= \sum_{a\in\mathcal{A}_x} \tilde{\pi}(a \mid x)\tilde{r}(x,a) + \gamma \sum_{y\in\mathcal{X}} \mathbb{P}^{\tilde{\pi}}(\tilde{X}_1 = y \mid X_0 = x) V^{\pi}(y) \\
&= \sum_{a\in\mathcal{A}_x} \pi(a \mid x) r(x,a) + \gamma \sum_{y\in\mathcal{X}} \mathbb{P}^{\pi}(X_1 = y \mid X_0 = x) V^{\pi}(y) \\
&= T^{\pi} V^{\pi}(x) = V^{\pi}(x)
\end{aligned}$$

And since the fixpoint is unique it follows that

$$V^{\tilde{\pi}} = V^{\pi}.$$

And as f is bijective, this implies

$$\sup_{\tilde{\pi}\in\tilde{\Pi}_S} V^{\tilde{\pi}}(x) = \sup_{\pi \ \varepsilon\text{-soft}} V^{\pi}(x),$$

as well as:

$$\begin{aligned} Q^{\tilde{\pi}}(x,a) &= \tilde{r}(x,a) + \gamma \sum_{y \in \mathcal{X}} \tilde{p}(y \mid x,a) V^{\pi}(y) \\ &= (1-\varepsilon)\left(r(x,a) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x,a) V^{\pi}(y) \right) \\ &\qquad + \frac{\varepsilon}{|\mathcal{A}_x|} \sum_{b \in \mathcal{A}_x} \left(r(x,b) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x,b) \right) \\ &= (1-\varepsilon) Q^{\pi}(x,a) + \frac{\varepsilon}{|\mathcal{A}_x|} \sum_{b \in \mathcal{A}_x} \left(r(x,b) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x,b) \right) \end{aligned}$$

Which implies that

$$\arg\max_{a \in \mathcal{A}_x} Q^{\tilde{\pi}}(x,a) = \arg\max_{a \in \mathcal{A}_x} Q^{\pi}(x,a).$$

Therefore greedy with regard to Q^{π} and $Q^{\tilde{\pi}}$ is the same. Let π_n be an ε-soft policy, and let π_{n+1} be ε-greedy with regard to Q^{π_n}. Then $\tilde{\pi}_{n+1} \coloneqq f(\pi_{n+1})$ is greedy w.r.t. $Q^{\tilde{\pi}_n}$:

$$\begin{aligned} \tilde{\pi}_{n+1}(a \mid x) = f(\pi_{n+1})(a \mid x) &= \frac{\pi(a \mid x) - \frac{\varepsilon}{|\mathcal{A}_x|}}{1-\varepsilon} \\ &= \begin{cases} \frac{\left((1-\varepsilon) + \frac{\varepsilon}{|\mathcal{A}_x|}\right) - \frac{\varepsilon}{|\mathcal{A}_x|}}{1-\varepsilon} = 1 & a \text{ is greedy} \\ \frac{\frac{\varepsilon}{|\mathcal{A}_x|} - \frac{\varepsilon}{|\mathcal{A}_x|}}{1-\varepsilon} = 0 & a \text{ is not greedy} \end{cases} \end{aligned}$$

Since we proved policy iteration converges for greedy policies, this finishes our proof:

$$\sup_{\pi \ \varepsilon\text{-soft}} V^{\pi}(x) = \sup_{\tilde{\pi} \in \tilde{\Pi}_S} V^{\tilde{\pi}}(x) = \lim_{n \to \infty} V^{\tilde{\pi}_n}(x) = \lim_{n \to \infty} V^{\pi_n}(x) \qquad \square$$

2.2.2 Shortcomings of Monte Carlo

Sutton and Barto provide a very instructive example for the weakness of Monte Carlo. Recall that Monte Carlo tries to estimate V^{π}. So the example assumes a given behaviour and tries to evaluate the value of a situation.

Example 2.2.3 (Driving Home)**.** Let us assume that John Doe works in city A, and drives to his home in city B after work, every day. Since he wants to get home as quickly as possible, the value of the state is inversely proportional to the time it will take him from a certain position home. This means that estimating the value of a certain state is equivalent to estimating the time left to drive. The longer John drives this route, the better he will get, at

estimating the time it will take him to drive certain sections of the road. If there is a delay in an earlier section he has a good estimate for the remaining time, once he clears the obstruction. Now imagine he is driving home from a doctors appointment from city A. As soon as he enters the highway to city B, he is able to use his experience driving this route to estimate the remaining time quite precisely.
The Monte Carlo algorithm on the other hand *never* uses existing value estimations to estimate the value of a different state. If John Doe uses Monte Carlo estimates to guess the remaining time from the doctor, the accuracy of his estimates will only ever increase with the times he actually starts driving from the doctor's office.

Since Monte Carlo has more possible "starting points", or generally earlier points in the chain, in case of estimating Q^π, it is worse in this case. An extreme example would be two actions in the same state, which have the same effect. Even though Monte Carlo might have a good estimate of the value of the first action it will start completely anew for the second action.

2.3 Temporal Difference Learning TD

Temporal Difference learning (TD) first proposed by Sutton (1988) addresses this weakness of Monte Carlo algorithms. For a sequence $(X_t, A_t, R_{t+1}, t \in \mathbb{N}_0)$ generated by a stationary behaviour π, we know:

$$\begin{aligned}
&\mathbb{E}^\pi[R_{t+1} + \gamma V(X_{t+1}) \mid X_t = x] \\
&= \mathbb{E}^\pi[r(x, A_0) \mid X_0 = x] + \gamma \sum_{y \in \mathcal{X}} \mathbb{P}^\pi(X_1 = y \mid X_0 = x) V(y) \\
&= T^\pi V(x) \qquad (2.1)
\end{aligned}$$

The random variable $R_{t+1} + \gamma V(X_{t+1})$ is therefore a bias free estimator for $T^\pi V(X_t)$. But because of its variance we can not simply update $V(X_t)$. Instead TD moves the estimate into the direction of this observation. To get an intuition for why this makes sense consider this example.

Example 2.3.1 (Unwinding the Arithmetic Mean)**.** Let X_i be independent identically distributed (iid) random variables, then the arithmetic means is:

$$\begin{aligned}
\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i &= \frac{n-1}{n}\frac{1}{n-1}\sum_{i=1}^{n-1} X_i + \frac{1}{n}X_n \\
&= \left(1 - \frac{1}{n}\right)\overline{X}_{n-1} + \frac{1}{n}X_n \\
&= \overline{X}_{n-1} + \underbrace{\frac{1}{n}}_{\text{“learning rate”}}\underbrace{(X_n - \overline{X}_{n-1})}_{\text{“direction”}}
\end{aligned}$$

But Temporal Difference Learning tries to do Dynamic Programming (i.e. replacing V with $T^\pi V$, which would require a learning rate of 1) at the same time as trying to reduce the error of the estimate of $T^\pi V$, where the learning rate should be something like $1/n$. The smaller the learning rate, the more of the old estimate we retain. This has the disadvantage, that old estimates are (in a way) previous steps in dynamic programming, which are further away from V^π. Larger learning rates on the other hand retain more of the variance of the latest random variable in the estimator. So one might for example want to have larger steps at the beginning until the dynamic programming part converges, and then start to focus more on the variance reduction part. This implies that a wider range of learning rates might be sensible. More on that in the third chapter. We therefore define

$$V_{n+1}(x) \coloneqq \begin{cases} V_n(x) + \alpha_n[R_{n+1} + \gamma V_n(X_{n+1}) - V_n(x)] & x = X_n \\ V_n(x) & x \neq X_n, \end{cases} \tag{2.2}$$

where α_n is called the *learning rate*. As one might want to increase the learning rates for rarely visited areas of the MDP, and decrease it for well known parts, it makes sense to allow it to be a function of the history $((X_t, A_t, R_{t+1}), t \in \{0, \dots, n\})$ and thus allow it to be a random variable. We will omit this possible dependency for now and reconsider it in Chapter 3.
For a deterministic stationary behaviour π, we can show the analogue

$$\begin{aligned} &\mathbb{E}^\pi[R_{t+1} + \gamma Q(X_{t+1}, A_{t+1}) \mid X_t = x, A_t = a] \\ &= r(x,a) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x, a) Q(y, \pi(y)) \\ &= T^\pi Q(x,a). \end{aligned}$$

Note that we could easily extend this to all stationary policies (c.f. 1.3.2, 1.3.5), by defining

$$T^\pi Q(x,a) \coloneqq r(x,a) + \gamma \sum_{(y,b) \in \mathcal{X} \times \mathcal{A}} \mathbb{P}^\pi[X_{t+1} = y, A_{t+1} = b \mid X_t = x, A_t = a] Q(y,b).$$

To distinguish the TD algorithm, which calculates Q from the one which calculates V, Sutton and Barto call the first one *Sarsa*. The name stems from the tuple $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$, as S_t is their notation for the state.
Similar to the Monte Carlo algorithm, we use generalized policy iteration to approach π^*. But since we do not need to wait for an episode to finish to create new estimates, we can alternate even more quickly between calculating Q^{π_n} and selecting π_{n+1}. The most extreme form is now updating the policy before every transition in every time step. This is what is called *on-line* learning in contrast to the *off-line* mentioned in the introduction.
Batch Reinforcement Learning is often used synonymously with off-line learning. But as batch learning algorithms were designed to utilize the information

Algorithm 3 On-line Sarsa (Sutton and Barto 1998)

Assume there is a mapping Ψ, assigning an action value function Q a policy π (for example mapping Q on an ε-greedy policy w.r.t. Q).

initialize Q(x,a)
while True **do** (for every episode)
    initialize x as a realization of X_0
    choose a according to $\Psi(Q)(\cdot \mid x)$
    repeat(for each step n of the episode)
        do a, observe reward R and next state y
        choose b according to $\Psi(Q)(\cdot \mid y)$
        $Q(x,a) \leftarrow Q(x,a) + \alpha_n[R + \gamma Q(y,b) - Q(x,a)]$
        $x \leftarrow y$; $a \leftarrow b$
    until x is terminal
end while

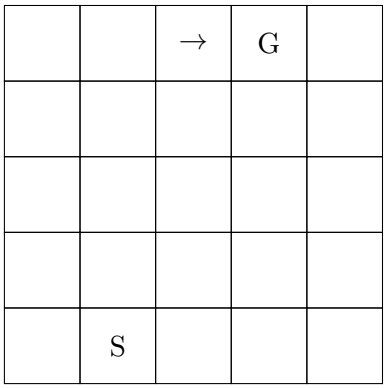
Where α_n is the learning rate.

of the existing batch as efficiently as possible, there were some attempts to modify them in order to provide intermediate results to advise action selection, transforming them into on-line algorithms. Lange et al. (2012) suggest to refer to these algorithms as *growing batch reinforcement learning*. We will discuss an example in Section 2.4, after explaining the need for it.

2.3.1 Shortcomings of TD

In a way TD is the opposite of Monte Carlo. While Monte Carlo did not use existing estimates of the value function, TD *only* uses existing estimates. This can cause problems as well.

Recall our room navigation example and assume that we initialize the TD algorithm with the value function $Q \equiv 0$. Then the algorithm picks an action, walks into that direction, gets reward zero, and updates Q in this place to zero. This continues until it reaches the goal where it updates the state and action that led it to the goal with a positive number. Which makes this action the greedy action (indicated by the arrow). But in all previous states TD still does not behave better than before. It will take a couple more episodes, to backpropagate the reward of the goal to the Q values at the start. This is quite bad in comparison to Monte Carlo, which only takes one episode to leave a trail of breadcrumbs



to the goal. One way to tackle this problem, is to try and find a compromise between Monte Carlo and TD like TD(λ). The other approach is growing batch learning.

2.4 Growing Batch Learning

One reason for TD's problems is, that it uses sampled transitions only once and then throws them away. This is not so much a problem when the MDP is cheap to sample. But in a lot of real world applications, experiences

- cost more time than most computations,
- are expensive (e.g. causing damage to hardware),
- and/or are outright dangerous (e.g. self-driving cars).

For this reason it is preferable, if the algorithm learns as much as possible from a given batch of experiences (transitions). To serve as a comparison, recall our initial naive batch learning algorithm (Algorithm 1). First it estimates the model parameters; after that it uses these estimates multiple times in every Dynamic Programming step. TD's estimates, on the other hand, are baked into the current estimate of V (or Q), which means that older samples are baked into older instances of V, which can not really be reused as they also represent previous instances of the "dynamic programming part" of TD (and also previous policies in the GPI case with Q).
So how could we modify this batch learning algorithm (using the entire batch of transitions) to be on-line, i.e. provide provisional estimates and incorporate additional information – growing batches?
First we can notice that we estimate $\hat{r}$ by calculating the algorithmic mean, which we can unwind as in Example 2.3.1. But note that this increases the number of operations for calculating the mean of n elements from

$$n-1 \text{ additions } +1 \text{ division} = n,$$

to

$$(n-1) \times (1 \text{ addition} + 1 \text{ subtraction } + 1 \text{ division}) = 3n-3,$$

basically tripling the number of operations.
The calculation of $\hat{p}$ is already partly on-line, as the increment in line 8 and 9 of the algorithm can be done in every transition. Recalculation of $\hat{p}$ in every step, would entail an iteration through all the $y \in \mathcal{X}$ whichever followed a particular (x, a), and might thus be better left for the time, when $\hat{p}$ is actually required.
To accelerate Dynamic Programming, we could use the previous estimation of V (Q) as a starting point, the next time we use DP. But DP still requires us to go over the entire state space (state-action space) for just one update, and most model parameter stay the same anyway, so more local updates are warranted.

The approach of algorithms like "Dyna" are basically hybrids of a "pure" online learning algorithm like TD (or – as we will later see – Q learning i.e. Dyna-Q) and a model based learning algorithm which runs in the background – ideally (for performance) as a separate process – and updates the value function with "localized" dynamic programming, i.e.

$$V^\pi(x) \leftarrow \sum_{a \in \mathcal{X}_x} \pi(a \mid x) \left(\hat{r}(x,a) + \sum_{y \in \mathcal{X}} \hat{p}(y \mid x,a) V^\pi(y) \right), \tag{2.3}$$

or

$$Q^\pi(x,a) \leftarrow \hat{r}(x,a) + \sum_{y \in \mathcal{X}} \sum_{b \in \mathcal{A}_y} \hat{p}(y \mid x,a) \pi(b \mid y) Q^\pi(y,b). \tag{2.4}$$

This leaves us with the question, which states to prioritize for updates. The simplest form of Dyna just picks these states randomly. From our example in Subsection 2.3.1 it should be quite obvious, that this is not the most efficient way to backpropagate rewards. But it should not be completely disregarded as it has one of, if not the smallest overhead of possible selection algorithms.
A second approach is *prioritized sweeping*, where the absolute difference between the sample of $T^\pi V(X_t)$ (i.e. $R_{t+1} + \gamma V(X_{t+1})$) and $V(X_t)$ (c.f. (2.2)) needs to be larger than a certain threshold for this state to be put into the update queue. The model learning part then updates all the states, which lead to this state, and inserts theses states into the queue themselves if their change is large too.
Another approach is *trajectory sampling* where a trajectory of state is generated by applying the current policy to the model ($\hat{p}$) of the real MDP. Then the algorithm could update the value functions along this chain of states, ideally backwards (to avoid the problem that TD has). This focuses the update on parts of the MDP, which are actually relevant, as they are possible to reach with the current policy. Just like it makes more sense to analyse different chess games as a joint string of moves, instead of analysing random positions which might never come up in a real game. Trajectory sampling can also be combined with the other approach to improve TD, using the learning algorithm of TD(λ) on these simulated trajectories.
The approaches we discussed here so far, are all *model based learning*, which could leave the impression that (growing) batch learning is synonymous with model based learning in contrast to *model free learning* methods like Monte Carlo and TD, which do not actually estimate the transition probabilities, and skip right to the value functions. But there are also model free growing batch learning methods.
One such example is *experience replay* coined by Lin (1992) which simply creates backups of past transitions and plays them back to the learning algorithms in a possibly different (e.g. backwards) order. The fact, that playing back a random transition is equivalent to sampling the model ($\hat{p}, \hat{r}$), motivates why

this has the desired effect if the selection of past transition is not unbalanced. Experience replay can be warranted if the sums in (2.3) or (2.4) are too large, i.e. when too many different states y can follow a state x. This is sometimes described as a high *branching factor*, where this factor could be defined as the average number of branches. Experience replay essentially trades memory (backing up all transitions) for computation speed, as it does not need to sum over all branches following a state. Variants might only back up certain transitions, which are selected according some measure of importance. Humans can for example remember very good or bad experiences better than "boring" situations. So maybe it is sufficient, to keep only transitions with very high or low rewards.
Sutton and Barto (1998) call model based learning methods "planning", and use the term "decision time planning" for algorithms which try to improve the knowledge of $Q(X_t, a)$ for all actions a, just before selecting the action A_t, by exploring the branches of the state based on its current model (essentially applying some localized Dynamic Programming to its model to aid the decision).

2.5 TD(λ) – Mixing Monte Carlo and TD

The trade-off between Monte Carlo and TD, is essentially the trade-off, between a small bias and a small variance. The estimator which Monte Carlo uses

$$\hat{V}^{\infty}(X_t) \coloneqq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T,$$

is an unbiased estimate for $V^{\pi}(X_t)$. Where T is the length of the episode. And since all these estimates get averaged, we can unwind this mean (Example 2.3.1)

$$V_n(X_t) = V_{n-1}(X_t) + \frac{1}{n}(\hat{V}_n^{\infty}(X_t) - V_{n-1}(X_t)).$$

On the other hand, recall that TD uses

$$\hat{V}^{(1)}(X_t) \coloneqq R_{t+1} + \gamma V(X_{t+1}),$$

as the estimator for $T^{\pi}V(X_t)$ which is not an unbiased estimator of $V^{\pi}(X_t)$, but has less variance due to its use of existing, possibly stable estimates (c.f. the driving home example). Recall that TD then adjusts its estimates as follows

$$V_n(X_t) = V_{n-1}(X_t) + \alpha_n(\hat{V}_n^{(1)}(X_t) - V_{n-1}(X_t)).$$

2.5.1 n-Step Return

This leads to the compromise, the *n-step return*, which we define as

$$\hat{V}^{(n)}(X_t) \coloneqq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(X_{t+n}).$$

Since rewards in the terminal state are by convention zero, we know for $n \geq T - t$,

$$\hat{V}^{(n)}(X_t) - \hat{V}^{\infty}(X_t) = \gamma^n V(X_{t+n}) = \gamma^n V(X_T).$$

If the value function correctly assigns value 0 to the terminal state X_T, then they are already equal for all $n \geq T - t$, otherwise it converges. This allows us to interpret Monte-Carlo as ∞-step return.

Lemma 2.5.1. *The n step return has the property*

$$\mathbb{E}^{\pi}[\hat{V}^{(n)}(X_t) \mid X_t = \cdot] = (T^{\pi})^n V,$$

Proof. (By induction) The induction basis is just the TD case, c.f. (2.1). The induction step follows:

$$\begin{aligned}
&\mathbb{E}^{\pi}[\hat{V}^{(n)}(X_t) \mid X_t = x] \\
&= \mathbb{E}^{\pi}[R_{t+1} + \gamma \hat{V}^{(n-1)}(X_{t+1}) \mid X_t = x] \\
&\overset{\text{A.1.1}}{=} \mathbb{E}^{\pi}[R_{t+1} \mid X_t = x] \\
&\qquad + \gamma \sum_{y \in \mathcal{X}} \mathbb{P}^{\pi}(X_{t+1} = y \mid X_t = x) \underbrace{\mathbb{E}^{\pi}[\hat{V}^{(n-1)}(X_{t+1}) \mid X_t = x, X_{t+1} = y]}_{\overset{\text{Markov+ind.}}{=} (T^{\pi})^{n-1} V(y)} \\
&= \mathbb{E}^{\pi}[r(x, A_0) \mid X_0 = x] + \gamma \sum_{y \in \mathcal{X}} \mathbb{P}^{\pi}(X_1 = y \mid X_0 = x)(T^{\pi})^{n-1} V(y) \\
&= (T^{\pi})^n V(x)
\end{aligned}$$

□

Corollary 2.5.2. *The n-step return has the* error reduction property*:*

$$\left\| \mathbb{E}^{\pi}[\hat{V}^{(n)}(X_t) \mid X_t = \cdot] - V^{\pi} \right\|_{\infty} \leq \gamma^n \|V - V^{\pi}\|_{\infty}$$

Proof. The claim follows from the contraction property of T^{π}. □

2.5.2 TD(λ) Forward View - λ-return

In that sense any kind of convex combination of n-step returns are estimates for V^{π} with a minimal error reduction of γ. This is true in particular for the geometric average, the λ-return (Watkins 1989):

$$\hat{V}^{\lambda}(X_t) \coloneqq (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} \hat{V}^{(n)}(X_t).$$

Why is the geometric average interesting? Well imagine incrementing over the time steps after t and at every step deciding, whether to call it a day (using the current value function as a replacement for the rest of the returns) or to continue. Since any time looks equally good or bad to stop, there is not really

a reason to pick any particular step.[1] So why not continue at every step with equal probability λ? Let N be the geometrically distributed random variable, which represents the stop time. Then assuming a realized MDP, denoting the realized chain of states by x_t, we observe

$$\mathbb{E}[\hat{V}^{(N)}(x_t)] = \sum_{n=1}^{\infty} \mathbb{P}(N=n)\hat{V}^{(n)}(x_t) = \sum_{n=1}^{\infty}(1-\lambda)\lambda^{n-1}\hat{V}^{(n)}(x_t) = \hat{V}^{\lambda}(x_t).$$

At this point it might be helpful to note, that for $\lambda = 0$ we get our 1-step basic TD estimate. For this reason TD is the special case TD(0). To continue this definition for $\lambda = 1$, we expand the estimation as follows:

$$\begin{aligned}
\hat{V}^{\lambda}(X_t) &= (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\hat{V}^{(n)}(X_t) \\
&= (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\left(\sum_{j=0}^{n-1}\gamma^j R_{t+1+j} + \gamma^n V(X_{t+n})\right) \\
&\overset{\text{Fub.}}{=} \sum_{j=0}^{\infty}\gamma^j R_{t+1+j}(1-\lambda)\underbrace{\sum_{n=j+1}^{\infty}\lambda^{n-1}}_{=\sum_{n=j}^{\infty}\lambda^n=\left(\frac{1}{1-\lambda}-\sum_{n=0}^{j-1}\lambda^n\right)} + (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n V(X_{t+n}) \\
&= \sum_{j=0}^{\infty}\gamma^j R_{t+1+j}\left(1-(1-\lambda)\frac{1-\lambda^j}{1-\lambda}\right) + (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n V(X_{t+n}) \\
&= \sum_{j=0}^{\infty}\gamma^j R_{t+1+j}\lambda^j + (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n V(X_{t+n}) \qquad (2.5) \\
&\xrightarrow{\lambda\to 1} \sum_{j=0}^{\infty}\gamma^j R_{t+1+j}
\end{aligned}$$

This is simply the return after t, which is well defined with Assumption 2. There are two different versions of TD(λ) constructed with the λ-return. The first one updates the value function with every step:

$$V_{t+1}(x) = V_t(x) + \alpha_t \underbrace{[\hat{V}_t^{\lambda}(X_t) - V_t(x)]}_{=:\Delta V_t^{\lambda}(X_t)} \mathbb{1}_{X_t=x} \qquad (2.6)$$

$$= V_0(x) + \sum_{k=0}^{t}\alpha_k \Delta V_k^{\lambda}(X_k)\mathbb{1}_{X_k=x} \qquad (2.7)$$

The index t indicates that $\hat{V}^{\lambda}$ uses V_t for its estimation of the remaining return. TD(0) is then just TD, and TD(1) is every-visit Monte Carlo for appropriate

[1] This is actually not quite true, it is most likely better to stop at states, which were visited often, since their estimates are probably better. But ex ante, before sampling the MDP, we can not make that distinction.

α_t. To be more precise, we want to average the returns after every time we visited a state. So assuming that we are in state X_t, the learning rate needs to be $1/(\#\text{previous visits} + 1)$ (c.f. Example 2.3.1), i.e.:

$$\alpha_t = \left(\sum_{k=0}^{t} \mathbb{1}_{X_k = X_t} \right)^{-1}$$

The second version of TD(λ) updates the value function after the end T_n of every episode n, and we will thus refer to it as batch TD(λ):

$$V_{n+1}(x) = V_n(x) + \sum_{k=0}^{T_n - 1} \alpha_k \underbrace{\left[\hat{V}_n^\lambda(X_k) - V_n(x) \right]}_{= \Delta V_n^\lambda(X_k)} \mathbb{1}_{X_k = x} \tag{2.8}$$

We would really have to index X_k with the episode $X_k^{(n)}$, since we get a state sequence in every episode. But since we will only need to talk about one episode, and in order to keep notation simple, I will omit this index. To understand this algorithm, it is best to consider only one episode at first. Batch TD(1) is also equal to Monte Carlo, for

$$\alpha_k = \left(n \sum_{i=0}^{T_n - 1} \mathbb{1}_{X_k = X_i} \right)^{-1}, \tag{2.9}$$

since then

$$V_{n+1}(x) = V_n(x) - \frac{1}{n} \left[\text{average returns after } x \text{ in episode } n - V_n(x) \right].$$

These two variants of TD(λ) are more specifically called the *forward view* of TD(λ) (Sutton and Barto 1998). This stems from the fact, that the λ-return in X_t requires knowledge of *future* returns, which forces us to wait for the end of the episode just like with Monte Carlo. For this reason this forward view of TD(λ) is not actually the version of TD(λ), which would actually be implemented. The forward view is simply easier to handle in convergence proofs, which is why it is also called the theoretical view of TD(λ).

2.5.3 TD(λ) Backward View

To motivate the *backward view* of TD(λ), we transform $\Delta V_m^\lambda(X_t)$ ($m = t$ or $m = n$ for batch TD(λ)) using (2.5):

$$\begin{aligned}
\Delta V_m^\lambda(X_t) &= \sum_{i=0}^{\infty} (\gamma\lambda)^i R_{t+1+i} + (1-\lambda) \sum_{i=1}^{\infty} \lambda^{i-1} \gamma^n V_m(X_{t+i}) - V_m(X_t) \\
&= \sum_{i=0}^{\infty} (\gamma\lambda)^i R_{t+1+i} + \sum_{i=1}^{\infty} (\lambda\gamma)^{i-1} \gamma V_m(X_{t+i}) - \sum_{i=0}^{\infty} (\lambda\gamma)^i V_m(X_{t+i})
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=0}^{\infty} (\gamma\lambda)^i \left[R_{t+1+i} + \gamma V_m(X_{t+1+i}) - V_m(X_{t+i})\right] \\
&= \sum_{i=t}^{\infty} (\gamma\lambda)^{i-t} \left[R_{i+1} + \gamma V_m(X_{i+1}) - V_m(X_i)\right] \\
&\overset{(*)}{=} \sum_{i=t}^{T-1} (\gamma\lambda)^{i-t} \underbrace{\left[R_{i+1} + \gamma V_m(X_{i+1}) - V_m(X_i)\right]}_{=\Delta V_m^0(X_i)}
\end{aligned} \tag{2.10}
$$

$(*)$ T is the end of the episode; note that the rewards in the terminal state are zero by convention and if we assume $V_m(y) = 0$ for y a terminal state, we get

$$R_{T+1} + \gamma V_m(X_{T+1}) - V(X_T) = 0$$

just like for all the following time steps. $V_m(y) = 0$ for every terminal state y is not a problematic assumption, since we can only end the episode if we can detect a terminal state, which easily allows us to set the value function to zero, if we did not initialize the value function iteration with a value function which assigns the terminal state zero anyway.

Note that $\Delta V_m^0(X_i)$ is simply the update direction of TD(0). And since TD(0) is already on-line – as the updates happen with every step – we could just "broadcast" the update deltas $\Delta_i^0(X_i)$ back to previous states. They can then get updated with these deltas.

In other words, we want to flip the summing over the deltas in (2.7) or (2.8), with the summation of all the TD(0) deltas. Here the version for $m = n$:

$$
\begin{aligned}
V_{n+1}(x) - V_n(x) &\overset{(2.8)}{=} \sum_{t=0}^{T-1} \alpha_t \mathbb{1}_{X_t=x} \Delta V_n^\lambda(X_t) \\
&= \sum_{t=0}^{T-1} \alpha_t \mathbb{1}_{X_t=x} \sum_{i=t}^{T-1} (\gamma\lambda)^{i-t} \Delta V_n^0(X_i) \\
&\overset{\text{Fub.}}{=} \sum_{i=0}^{T-1} \Delta V_n^0(X_i) \underbrace{\sum_{t=0}^{i} \alpha_t (\gamma\lambda)^{i-t} \mathbb{1}_{X_t=x}}_{=:e_i(x)}
\end{aligned}
$$

This backwards view essentially tries to assign discounted credit/blame to previous states. The $\Delta V_i^0(X_i)$, are the unexpected reward or cost, in state X_i at time i, and $e_i(x)$ measures the amount of credit/blame the state x is eligible for. e_i, is therefore called the *eligibility trace*. The partial sums are then interim estimates for V_{n+1}.

In order to adapt this idea for the first version of TD(λ) (2.7), we need to make an approximation, otherwise we can not move the $\Delta V_t^0(X_i)$ out of the

sum over t:

$$\begin{aligned}\Delta V_t^\lambda(X_t) &= \sum_{i=t}^{T-1}(\gamma\lambda)^{i-t}\left[R_{i+1}+\gamma V_t(X_{i+1})-V_t(X_i)\right]\\ &\approx \sum_{i=t}^{T-1}(\gamma\lambda)^{i-t}\underbrace{\left[R_{i+1}+\gamma V_i(X_{i+1})-V_i(X_i)\right]}_{=\Delta V_i^0(X_i)} \qquad (2.11)\end{aligned}$$

Now why does this approximation make sense? First of all, V_{i+1} stays the same to V_i in every place but X_i (c.f. (2.6)). Assuming that the sequence of states wanders about a little bit, the first few V_i will all be equal to V_t. And once the sequence of states comes back around, the hope is that $i-t$ would be large enough, for $(\gamma\lambda)^{i-t}\approx 0$. Additionally, the difference between V_i and V_t is small, even for repeated states, if the learning rate α_i is sufficiently small. This lets us write the version for (2.7):

$$\begin{aligned}V_T(x)-V_0(x) &\approx \sum_{t=0}^{T-1}\alpha_t\mathbb{1}_{X_t=x}\sum_{i=t}^{T-1}(\gamma\lambda)^{i-t}\Delta V_i^0(X_i)\\ &\overset{\text{Fub.}}{=} \sum_{i=0}^{T-1}\Delta V_i^0(X_i)\underbrace{\sum_{t=0}^{i}\alpha_t(\gamma\lambda)^{i-t}\mathbb{1}_{X_t=x}}_{=:e_i(x)}\end{aligned}$$

This version can easily be adapted for non episodic MDPs by replacing T with ∞, where V_∞ is the estimate of the value function in the limit.

Algorithm 4 On-line TD(λ) (Backward View - non batch version)

```
Initialize V
e(x) ← 0   ∀x ∈ 𝒳
while True do
    Initialize x as a realization of X_0
    repeat(for every step n of the episode)
        a ← action sampled from π(· | x)
        Take action a, observe reward R, next state x'
        Δ ← R + γV(x') − V(x)
        e(x) ← e(x) + α_n
        for y ∈ 𝒳 do
            V(y) ← V(y) + e(y)Δ
            e(y) ← γλe(y)
        end for
        x ← x'
    until x is terminal
end while
```

Just like with Monte Carlo and TD the action value (Q) version Sarsa(λ) is basically the same, and generalized policy iteration would again be used to find an optimal policy.
Note that I took the liberty, and moved the α_n into the eligibility trace to allow for changing learning rates over time. Sutton and Barto (1998) place the learning rate outside the eligibility trace which forces it to be constant in order to translate to the forward view.
While the theory for the batch version of TD(λ) is a lot cleaner, it should be noted that this version is rarely used in practice. True Online TD(λ) attempts to fix the hole that this approximation causes but is slightly more complex, and while the first version of TD(λ) generally performs considerably better than either TD(0) or Monte-Carlo, these two extremes are still often used due to simplicity, so TD(λ) will probably also stay relevant for some time, due to its relative simplicity compared to True Online TD(λ).

2.5.4 Variations - True Online TD(λ)

The eligibility traces above are also called *accumulating traces*, because every visit adds to the existing trace

$$e_{n+1}(x) = \gamma\lambda e_n(x) + \alpha_{n+1}\mathbb{1}_{X_{n+1}=x}.$$

This was the original trace proposed by Sutton (1988) when he presented TD(0) and TD(λ).
Replacing traces introduced by Singh and Sutton (1996) instead reset the trace, whenever a state is visited a second time. Notice how that is essentially equivalent to cancelling the sum in (2.11) at that time. The reason for this change was, that the accumulating traces made learning unstable because they could become too large. Recall that this problem occurs, when states are visited repeatedly. Since we argued that repeated visits would be far apart in order to justify the approximation, this gives us some insight as to why this might be the case.
True Online TD(λ), proposed by Van Seijen and Sutton (2014), circumvents the approximation altogether. Instead of utilizing the full λ-return

$$\hat{V}^{\lambda}(X_t) = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\hat{V}^{(n)}(X_t)$$

for the forward view, it utilizes the *truncated λ-return*

$$\hat{V}^{\lambda|t^*}(X_t) = (1-\lambda)\sum_{n=1}^{t^*-t-1}\lambda^{n-1}\hat{V}^{(n)}(X_t) + \lambda^{t^*-t-1}\hat{V}^{(t^*-t)}(X_t),$$

which is also a convex combination of n-step returns, but only requires knowledge of $X_t, \ldots, X_{t^*}$, instead of knowledge of the *entire* future. So in order to calculate

$$V_{t+1|t^*}(X_t) = V_{t|t^*}(X_t) + \alpha_t\left[\hat{V}^{\lambda|t^*}_{t|t^*}(X_t) - V_{t|t^*}\right]$$

(c.f. (2.6)), we only require knowledge of the history up to time t^*. So $V_{t^*|t^*}$ is our best guess for the value function V^π at time t^*. Though in order to calculate $V_{t^*+1|t^*+1}$ it seems like we need to start the entire recursion again for $V_{t|(t^*+1)}$ up to $t = t^* + 1$.

$$\begin{array}{llll} V_{0|0} & & & \\ V_{0|1} & V_{1|1} & & \\ V_{0|2} & V_{1|2} & V_{2|2} & \\ \vdots & & & \ddots \\ V_{0|t^*} & & \dots & V_{t^*|t^*} \end{array}$$

But Van Seijen and Sutton (2014) found a way to calculate $V_{t^*+1|t^*+1}$ directly from $V_{t^*|t^*}$. The resulting eligibility traces are referred to as *Dutch traces*. Lastly it is worth noting, that in order to keep the number of updates in line 10 of Algorithm 4 small, most algorithms set eligibility traces below a certain threshold to zero.

2.6 Q-learning

Q-learning (Watkins 1989) is virtually the same as Sarsa, just that Q-learning tries to skip policy iteration by trying to approximate Q^* directly, instead of Q^π. As it does not approximate the value function of the policy it uses currently, it is an *off-policy* algorithm, while Monte Carlo and TD are *on-policy* algorithms. Instead of

$$Q(X_t, A_t) \leftarrow Q(X_t, A_t) + \alpha_t[R_{t+1} + \gamma Q(X_{t+1}, A_{t+1}) - Q(X_t, A_t)],$$

here the update rule is

$$Q(X_t, A_t) \leftarrow Q(X_t, A_t) + \alpha_t[R_{t+1} + \gamma \max_{b \in \mathcal{A}_{X_t}} Q(X_{t+1}, b) - Q(X_t, A_t)],$$

because the expected value equals

$$\begin{aligned} &\mathbb{E}[R_{t+1} + \gamma \max_{b \in \mathcal{A}_{X_t}} Q(X_{t+1}, b) \mid X_t = x, A_t = a] \\ &= r(x, a) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x, a) \max_{b \in \mathcal{A}_x} Q(y, b) \\ &= T^* Q(x, a), \end{aligned}$$

instead of $T^\pi Q$ like in the case of Sarsa.

2.6.1 Shortcomings of Q-learning

Due to its heritage, it displays the same weakness as TD (Subsection 2.3.1), which we can fix with model learning algorithms like Dyna-Q, as hinted at in

Section 2.4. But the approach of TD(λ) generalizes only to some degree to Q-learning.
To create n-step estimates, the algorithm would need to actually pick the greedy action in every step (otherwise the expected Value is not $(T^*)^n Q$). But since we also need to do some exploration this can not be guaranteed. And if it would pick the greedy action every time, then the update would be equal to the TD(0) update anyway. Watkin's Q(λ) (Watkins 1989) therefore limits itself to n-steps, with n smaller than the number of steps until the next exploratory action. Peng's Q(λ) (Peng and Williams 1994) essentially pretends, this problem does not exist, and uses histories with non-greedy actions too. This means it "converges to something between Q^* and Q^π", the hope is that it would converge to Q^* if the policy becomes more and more greedy over time. While there is no proof that Peng's Q(λ) converges, "most studies have shown it to perform significantly better than Watkin's" empirically (Sutton and Barto 1998, p. 184). So we are now again at a point, where we simply do not know whether it converges or not.
At first glance, Q-learnings larger flaw might seem to be its tendency to overestimate itself. As it directly estimates Q^*, it's greedy actions assume that the optimal policy will be played afterwards. But given an ε-greedy policy that is not always the case. Sutton and Barto's example is a Cliff Edge, where the shortest Path is right beside the cliff, but a "safer path" is only a little bit longer. Q-learning tries to walk the short path but plunges down the cliff relatively often because of the ε part of its ε-greedy policy. While Sarsa with GPI converges to walking the safer path since it estimates Q^π account for the exploratory part of the policy. But this flaw could also be addressed with more sophisticated exploration policies.

2.7 Exploration

Until now, the only policy we discussed, which tries to tackle the exploration vs. exploitation trade-off, is the ε-greedy policy. This section is meant to give an overview over different approaches to exploration. For on-policy algorithms, convergence is tricky to discuss, as was already the case for the ε-greedy policy. For off-policy Algorithms, in particular Q-learning, the exploration policy only needs to guarantee that every state action pair is visited often enough (i.e. infinitely many times in the limit). But to allow for the use of a Q(λ) algorithm, the greedy actions need to be picked on most occasions.

2.7.1 Optimism

An extremely simple approach is, to initialize the estimate of the action value function Q higher than $R/(1-\gamma)$ (c.f. Assumption 1). Then unexplored state action pairs have this over-optimistic value, and exploring states lowers their value down towards their actual value. The algorithm will therefore

always select the least explored actions until expectations are lowered. While it is very easy to show convergence of a simple greedy policy iteration, it is virtually useless in our setting of large state and action spaces, since it will only exploit its knowledge once it thoroughly convinced itself that there are no better options.

2.7.2 Boltzmann Exploration

The ε-greedy policy is also known as *semi-uniform random exploration*, because it selects a random action uniformly, when exploration is rolled with probability ε. Boltzmann exploration does not just differentiate between exploratory actions and greedy actions, but weights actions depending on the estimation of their value

$$\pi(a \mid x) = \frac{\exp(Q(x,a)/T)}{\sum_{b \in \mathcal{A}_x} exp(Q(x,b)/T)},$$

where $T > 0$ is a "temperature" parameter, which can be used to change the algorithms tendency for exploration. For $q^* = \max_{i=1,\dots,n} q_i$ we can see that in the limit, decreasing T reduces exploration:

$$\begin{aligned}
\exp(q_i/T) = \exp\left(\frac{q^*}{T} + \frac{q_i - q^*}{T}\right) &= \exp(q^*/T)\exp((q_i - x^*)/T) \\
\implies \lim_{T\to 0} \frac{\exp(q_i/T)}{\sum_{j=1}^n \exp(q_j/T)} &= \lim_{T\to 0} \frac{\exp(q_i/T)}{\exp(q^*/T)} \frac{1}{\sum_{j=1}^n \exp((q_j - q^*)/T)} \\
&= \lim_{T\to 0} \frac{\exp((q_i - q^*)/T)}{\sum_{j=1}^n \exp((q_j - q^*)/T)} \\
&= \begin{cases} 0 & q_i \le q^* \\ \frac{1}{\#\{j \mid q_j = q^*\}} & q_i = q^* \end{cases}
\end{aligned}$$

While increasing T, increases exploration in the limit

$$\lim_{T\to\infty} \frac{\exp(q_i/T)}{\sum_{j=1}^n \exp(q_j/T)} = \frac{1}{n}$$

2.7.3 Directed Exploration

Boltzmann exploration fixes most of the suicidal tendencies that the ε-greedy policy expresses in our cliff edge example (Subsection 2.6.1). But since Boltzmann exploration still assigns every action a positive probability, it is still considered an *undirected* exploration method. The following methods are *directed* exploration methods.

Interval Estimation

Kaelbling (1993) suggested to use confidence intervals in more simple decision problems like the n-armed bandit. In the n-armed bandit the agent has to decide between n options which have stochastic returns. Sampling these options allows us to estimate their means and construct confidence intervals around them. Choosing the action with the highest upper bound of its confidence interval accounts for the trade-off between picking actions with high mean, and less explored actions with large confidence intervals. Consequently modifications where proposed which would allow for the construction of confidence intervals around the $Q(x,a)$ values in model based learning (e.g. Wiering and Schmidhuber 1998; Strehl and Littman 2008). It is probably futile trying to apply interval estimation to pure on-line learning, since at least much of the early variation in the Q values comes from "dynamic programming" and not stochastic uncertainty.
The basic idea behind model based interval estimation (MBIE) is, to create confidence intervals around $\hat{r}$ and $\hat{p}$. This is relatively easy for $\hat{r}(x,a)$ as it is simply a mean of reward samples. And by assuming boundedness of rewards, Strehl and Littman (2008) propose to use Hoeffding's inequality. To avoid the boundedness assumption we could for example use, that independently identical distributed (iid) X_i's have the property

$$\mathrm{Var}\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] = \frac{1}{n}\mathrm{Var}[X_1] \quad \text{and} \quad \hat{\sigma}_n \coloneqq \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2} \to \sigma_{X_1},$$

so we could use $\hat{\sigma}_n/\sqrt{n}$ as an estimator for the standard deviation of $\overline{X}$. And since $\overline{X}$ is approximately normal distributed because of the central limit theorem, the standard deviation is enough to construct a simple confidence interval. Regardless of how this confidence interval is constructed, we simply denote this interval by

$$CI(r_{x,a}) \coloneqq [\hat{r}(x,a) - \varepsilon^r, \hat{r}(x,a) + \varepsilon^r].$$

In the same spirit Strehl and Littman define

$$CI(p_{x,a}) \coloneqq \left\{ p \in [0,1]^{|\mathcal{X}|} : \sum_{y\in\mathcal{X}} p(y) = 1 \text{ and } \|p - \hat{p}(\cdot \mid x,a)\|_1 \leq \varepsilon^p \right\}$$

with an appropriately selected ε^p. This is of course less of a confidence *interval* and should maybe rather be called a confidence set.
Their idea is, that if we take the "upper limit" out of both "intervals" we get something akin to the upper limit $\tilde{Q}$ of the "confidence interval" for Q which would satisfy the recursion

$$\tilde{Q}(x,a) = \max_{R\in CI(r_{x,a})} R + \max_{p\in CI(p_{x,a})} \gamma \sum_{y\in\mathcal{X}} p(y) \max_{b\in\mathcal{A}_y} \tilde{Q}(y,b)$$

They show that this recursion is still a contraction, which means we could calculate $\tilde{Q}$ with the same (local) dynamic programming ideas which we use in model based learning. While the question whether or not this construct is in fact a confidence interval around Q, is not very pressing (since we are not interested in confidence intervals per se but rather exploration heuristics), the computational burden of a second recursion on top of the main recursion is more of a problem. Especially since this recursion includes two more maximizations, and while the first one is trivial once ε^r is calculated, the second one is not. Whether or not this is acceptable, depends on the computational resources available and on the expensiveness of samples from the MDP.

Alternatively one could try to simplify the estimation. Wiering and Schmidhuber (1998) simply forego the recursion and only use bounds around $\hat{r}$ and $\hat{p}$, ignoring the estimation error in the Q values of the following states. Other approaches simply award exploration bonuses based on the frequency of tries, or how recent the last try was.

Another problem is, that interval estimation allows the upper limit of the confidence interval around the optimal action to be lower than the average value of another action with positive probability. This unlikely case results in the algorithm never trying these optimal actions again. This is called sticking (Kaelbling 1993, p. 61). Last but not least, there is the question on how to initialize the size of the intervals when no samples (or too few) are available. Wiering and Schmidhuber (1998) suggest to begin with Boltzmann exploration, and switch to interval estimation later.

Bayesian Exploration

Dearden et al. (1998) suggest to select actions based on the probability that they are optimal, conditional on the samples collected. But just like with every Bayesian approach, calculating conditional probabilities requires a prior distribution of the Q values. This results in more parameters, with which the algorithm can be tuned. Therefore the authors warn that this could have benefited the performance of this algorithm in their empirical comparison. And while their algorithm was quite competitive, they also note that it is very computationally expensive.

2.7.4 Intrinsic Rewards

Most exploration approaches struggle, when the rewards for actions have a long delays (i.e. when rewards are sparse). Recall our room navigation example again, and imagine a second goal $\tilde{G}$ closer to the start with a smaller reward. Then the algorithm will most likely stumble into the secondary goal first and the action values around this secondary goal will relatively quickly be updated to lead towards this goal.

In case of the ε-greedy policy, the occasional exploratory actions will only cause a one step deviation from the shortest path to this secondary goal, and the algorithm will use its knowledge about the surroundings to quickly walk back towards this goal using subsequent greedy actions. While finding the larger goal, requires multiple subsequent exploratory actions, leading away from the secondary goal.

			G	
	↓			
↓	←	↓		
$\tilde{G}$	←	←	←	
↑	S	←		

Boltzmann exploration does not perform much better, as the actions leading to the secondary goal will be assigned higher probability, so finding the larger goal still requires a chain of unlikely actions.
And since estimating confidence intervals around Q values without samples, is not really possible, the performance of interval estimation depends on how it handles unknown states. If it uses Boltzmann exploration in the beginning, it will obviously struggle too. And setting high ranges for the initial interval estimations results in very similar behaviour to optimistic initialization of the Q values. As the algorithm would then explore until the entire state space is explored.
Another example for this sparse reward setting are small negative rewards littered around the state space. Most games have this problem, where there is only one big reward for beating the level, but there are many possibilities to lose during the level. This can result in the agent deciding, that the optimal way to play is not to play, i.e. achieve the small goal of avoiding to lose.
A naive fix to this problem would be to favour chains of exploratory actions (i.e. increase the likelihood of an exploratory action following another), or similar adjustments. An out-of-the-box solution is, to modify the existing rewards, artificially making them less sparse.

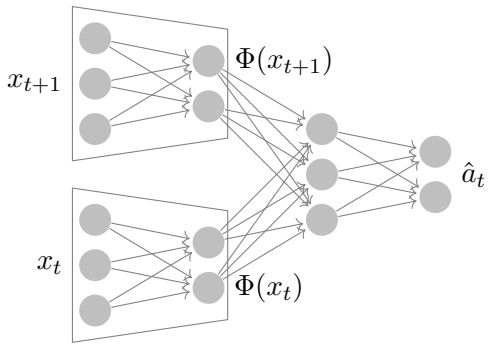
Curiosity-Driven Exploration

Inspired by human curiosity, intrinsic rewards for actions which do not immediately result in extrinsic reinforcement were suggested. There are mostly two types of intrinsic motivation algorithms (Pathak et al. 2017). One rewards reaching "novel" states, encouraging the agent to stay in unexplored territory before walking back to known extrinsic rewards. Where "novelty" could be a constant minus the number of visits, or anything else one might come up with.
In the second approach the agent builds a world model which tries to predict the next state x_{t+1} given state x_t and action a_t. This world model would be a function approximation algorithm (e.g. an artificial neuronal net) which would be trained on the samples generated. Surprising this world model results in an

additional reward for the reinforcement algorithm. This encourages the agent to take actions which it does not understand (i.e. which the world model does not predict). But in this basic form, the agent would get addicted to white noise, i.e. situations where the next state is completely unpredictable but also not necessarily meaningful.

Pathak et al. (2017) recently came up with a modification to fix this problem. They argue that the unpredictability is due to the fact, that these unpredictable state changes are not influenced by the agents actions. So they suggest a filter Φ for the state space, which maps the state to a smaller feature space, which only represents the influenceable state of the world. This feature map Φ is attained by training a neuronal net.

This neuronal net consists of two copies of a neuronal net with input neurons x_t and x_{t+1} respectively, which have fewer output neurons. These two outputs $\Phi(x_t)$ and $\Phi(x_{t+1})$ are then fed into a common net which has $\hat{a}_t$ as output. Similar to autoencoders (c.f. A.3) compressing information with a bottleneck, this neuronal net has to compress x_t and x_{t+1} to the middle layer $\Phi(x_t)$ and $\Phi(x_{t+1})$. But since it does not need to recover the original from this compression but rather a_t, it is meant to throw away the information about x_t which does not help to predict a_t, i.e. throw away parts of the state space which can not be influenced by an action.



The world model then has to predict $\Phi(x_{t+1})$ instead of x_{t+1}, which means that its error will not increase when it does not predict non-influenceable changes like white noise, as these are filtered out by Φ.

This set-up proved extremely effective in empirical tests. It even learned to play some games, it was tested on, without any extrinsic reward at all.

It might be argued, that this feature map Φ filters out too many things, like actions by other actors which are not influenced by the agent itself, but influence the agent. If they influence the agent though, and the agent could, or should react to these actions, then these events will be relevant for predicting the agents actions. Therefore they would not be filtered out. Only if the agent can not, or has no incentive to respond to these events is it irrelevant for predicting the agents actions and will be filtered. And since the agent could not or does not need to react to these events anyway, no harm is done.

2.8 Function Approximation

Up until now, we only discussed tabular methods, i.e. methods which store the value $V(x)$ of a state x for every state in a table. There are two reasons for discarding this strategy. First, the state space might be simply too large to store V in that way (e.g. chess). Second, algorithms do not generalize at all with this method. If an algorithm ends up in a state where it never was before, the state can be almost identical to another state, and the algorithm will still treat it as completely unknown. If the input is a picture, it would be enough to change one pixel, one speck of dust and suddenly the algorithm treats the situation as completely new.
The solution is to use function approximation methods (i.e. supervised learning) to approximate the value functions. Function approximation tries to find the function from a reduced set of functions, which approximates the real function best. This reduced set of functions, might be the set of linear functions (linear regressions), the set of polynomials (polynomial regression), the trigonometric functions (Fourier analysis), the set of linear combinations of some universal kernel (support vector machines), the set of functions achievable with some weights in a neuronal net, or anything else one might come up with. There is usually some motivation for a particular set. Often the set, or a related superset, is dense in the set of continuous functions – e.g. polynomials, trigonometric functions, the limits of linear combinations of universal kernels and linear combinations of sigmoid functions, which make up neuronal nets, are all dense in the set of continuous functions on some compact set.
Since the value function we are looking for will not be included in this reduced set of functions in general, we need to define some form of distance measure on the set of functions in order to pick the closest function in the reduced set. After that we can search for algorithms which converge to this closest function. This more general setting of a reduced set of possible functions has two additional benefits:

1. The reduced set of functions can exclude functions which can differentiate between hidden attributes of two states. Therefore this more general setting can model hidden information.

2. We can transform the limitation of memorylessness from a limitation of the MDP model to a limitation of the reduced (value) function set. I.e. we include the entire history in the state space, and force the reduced set of functions to only include value functions which depend on the last state only. Where the assumption, that the next state and reward depends only on the last state, was previously an assumption of the MDP, it now becomes an assumption of the function approximation. Function approximation with some form of memory, like recurrent neuronal nets with Long Short-Term Memory (LSTM), are then just a different reduced function set on the same MDP model.

With these two observations, the underlying MDP becomes similar in kind, to the underlying sample space Ω in probability theory. Whatever situation one might want to model, the MDP is large enough to perfectly model the situation by definition, and the restrictions are introduced later. In the case of probability theory by random variables, in the case of MDPs by the information available about the current state. This view tends to render all model based learning methods discussed in Section 2.4 useless, since it leaves the precise state, the agent is in, undefined, and the state space is too large to sample enough transitions in order to estimate transition probabilities anyway.
With this framework, video captured by a camera can be the observable part of the state, while everything that can not be observed by the camera is a hidden attribute of the state. But function approximation methods with memory can still achieve feats like object permanence, i.e. correctly assume objects still exists, after they have been covered up.
There are also approaches which use unsupervised dimension reduction techniques on the inputs (principal component analysis, autoencoders, etc.), and use the resulting feature map as the state space. This allows for generalization, since multiple similar states are collapsed into one state. But it is also a special case of function approximation of the value function, since this is too just a restriction on the set of possible (value) functions. There are multiple reasons, why it might make sense to put such a dimension reduction module before the value function approximation:

1. Consider the example of a robot using a camera to achieve a task. While letting the robot use its vision to try different behaviours takes real time of the robot moving around, training a dimension reduction algorithm on a bunch of pictures can be done without any real world interaction. Afterwards this pretrained function can be used in the value function approximation of the robot, speeding up the learning process. More generally:

   (a) data might already be available for some sub-task,

   (b) some sub task might be cheaper to train on its own than the entire system,

   (c) it might be easier to test reliability of subsystems, than getting enough data to verify the safety of the entire system (e.g. self-driving: a lot of data needed to verify the required safety improvement, since accidents are very rare)

2. The ability to use model based learning techniques might be reintroduced by creating a state space using a feature map from a dimension reduction module (e.g. a deep autoencoder (c.f. A.3) in Deep Fitted Q Iteration Lange et al. 2012). And even model free learning methods like experience replay can benefit from such a feature map, as it reduces the amount of

memory need to store these experiences, and uncompressed video takes up an incredible amount of storage.

3. Readers familiar with neuronal nets, might also be familiar with the vanishing/exploding gradient problem in deep neuronal nets with gradient decent. Pre-training lower layers promises to avoid this problem to some degree at least.

2.8.1 Mean Squared Value Error

As already noted, we require a distance measure on functions in order to determine the closest function in the reduced set. One such metric is

$$d(f,g) \coloneqq \sum_{x\in\mathcal{X}} \mu(x)[f(x)-g(x)]^2.$$

where μ is a probability measure on $\mathcal{X}$ weighting the states on how important they are. Let $\{x \mapsto v(x,\mathbf{w})\}$ be the reduced set of functions with parameters $\mathbf{w}$ (for example weights in a neuronal net). And v^* the objective function, then the *mean squared value error* is defined as

$$\overline{\mathrm{VE}}(\mathbf{w}) \coloneqq d(v(\cdot,\mathbf{w}),v^*) = \sum_{x\in\mathcal{X}} \mu(x)[v(x,\mathbf{w})-v^*(x)]^2.$$

Now let $X_0, X_1, \ldots$ be a sequence in the state space, then we can often argue

$$\mu_n(x) = \frac{1}{n}\sum_{i=0}^{n-1} \mathbb{1}_{x=X_i} \to \mu(x) \qquad (n\to\infty)$$

for some probability measure μ. If the sequence $X_0, X_1, \ldots$ is generated with some behaviour π in an MDP for example, we can usually determine a limiting distribution. In order to avoid a degenerate limiting distribution in episodic MDPs, this generally requires a restart of the MDP whenever we reach a terminal state. This is useful, since it allows us to write

$$\frac{1}{n}\sum_{i=0}^{n-1}[v(X_i,\mathbf{w})-v^*(X_i)]^2 = \sum_{x\in\mathcal{X}} \mu_n(x)[v(x,\mathbf{w})-v^*(x)]^2 \to \overline{\mathrm{VE}}(\mathbf{w}).$$

And while the usefulness of this equation in gradient decent is most likely the main reason for using this particular μ, there is also a theoretical justification for it, since we care more about the accuracy of the value function on the states we actually visit than on states we do not visit anyway.

2.8.2 Very Short Introduction to Gradient Decent

One heuristic to find the minimum of $\overline{\mathrm{VE}}$, is to move in the direction of the steepest decent. This is in the opposite direction of the gradient:

$$\begin{aligned}
-\nabla\overline{\mathrm{VE}}(\mathbf{w}) &= -\sum_{x\in\mathcal{X}} \mu(x)\nabla[v(x,\mathbf{w}) - v^*(x)]^2 \\
&= \lim_{n\to\infty} -\sum_{x\in\mathcal{X}} \mu_n(x)\nabla[v(x,\mathbf{w}) - v^*(x)]^2 \\
&= \lim_{n\to\infty} \nabla\frac{1}{n}\sum_{i=0}^{n-1} -[v(X_i,\mathbf{w}) - v^*(X_i)]^2 \\
&= \lim_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n-1} 2[v^*(X_i) - v(X_i,\mathbf{w})]\nabla v(X_i,\mathbf{w})
\end{aligned}$$

But instead of making one step of size $\tilde{\alpha}$ in this direction

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \underbrace{\tilde{\alpha}\frac{2}{n}}_{=:\alpha}\sum_{i=0}^{n-1}[v^*(X_i) - v(X_i,\mathbf{w}_t)]\nabla v(X_i,\mathbf{w}_t),$$

we could also make n individual steps

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha[v^*(X_t) - v(X_t,\mathbf{w}_t)]\nabla v(X_t,\mathbf{w}_t).$$

Since we update $\mathbf{w}_t$ with every step, these two variants are not equivalent! But due to the higher frequency of updates in the second variant the hope is, that it converges more quickly. If we now just replace $v^*(X_t)$ with U_t where $\mathbb{E}[U_t \mid X_t] = v^*(X_t)$, and allow the learning rate to vary, then we obtain a regression

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t[U_t - v(X_t,\mathbf{w}_t)]\nabla v(X_t,\mathbf{w}_t),$$

which is known as *stochastic gradient decent* SGD. But note that the specifier "stochastic" refers to the selection of $X_i \sim \mu$ and not U_t. According to Sutton and Barto (2018, p. 202) convergence proofs to a local minimum for SGD are available if the learning rate fulfils (3.2). In particular this covers Monte Carlo with

$$U_t = \sum_{n=t}^{\infty} \gamma^{n-t} R_{n+1}.$$

But since bootstrapping methods like TD and Q-learning, or more general n-step returns are not unbiased, these proofs are not applicable to them. Additionally the n-step return depends on the weights $\mathbf{w}$:

$$\hat{V}^{(n)}(X_t,\mathbf{w}) = R_{t+1} + \gamma R_{t+1} + \cdots + \gamma^{n-1}R_{t+n} + \gamma^n V(X_{t+n},\mathbf{w})$$

So this would have to be taken into account when differentiating with regard to $\mathbf{w}$. But the regression

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t[\hat{V}^{(n)}(X_t,w_t) - v(X_t,\mathbf{w}_t)]\nabla v(X_t,\mathbf{w}_t),$$

explicitly ignores this dependency and is thus grouped in *semi-gradient methods* with other bootstrapping algorithms, that ignore this dependency.

2.8.3 A counterexample

There are certain interaction effects between bootstrapping methods and function approximation methods, which make the combination unstable. Consider the state space $\mathcal{X} = \{1, 2\}$, let the action space $\mathcal{A} = \{1, 2\}$ simply determine the next state (i.e. $p(y \mid x, a) = \delta_{ya}$), and imagine there are no rewards ever. Then consider the set of linear functions

$$\{v : x \mapsto wx : w \in \mathbb{R}\}.$$

In particular the true value function $V^\pi = 0$ (for all π) is included in the set. Now consider the transition from state $X_0 = 1$ to state $X_1 = 2$, when starting with some positive w_0. Since there are no rewards, the update in TD(0) would be:

$$\begin{aligned} w_1 &= w_0 + \alpha[0 + \gamma v(X_1, w_0) - v(X_0, w_0)]\nabla v(X_0, w_0) \\ &= w_0 + \alpha[\gamma 2 w_0 - w_0] \cdot 1 \\ &= w_0 + \alpha \underbrace{(2\gamma - 1)}_{\geq 0 \iff \gamma \leq 1/2} w_0 \end{aligned}$$

So the value of w can actually increase! Even though it needs to move towards 0, since $v(x, 0) = V^\pi(x)$. It should be noted, that the valuation of the first state would increase in the tabular TD(0) version too, since the value of the state gets boosted by the fact, that it serves as an entryway to the higher valued state. But because of the interdependency of the evaluation of both states, the promise of this higher valued state causes the increase of the first state, which in turn increases the expectation for the value of the second state. So if we would just repeat the experience of a transition from state 1 to state 2, we would get into an ever increasing loop of expectations about the fantastic rewards that await us in state 2. But since TD(0) is actually on-policy, we will end up disappointed by the lack of rewards, when we actually reach state 2 and lower our expectations.
But off-policy algorithms are not constrained by this requirement. Q-learning for example, evaluates the next state based on the optimal action possible in that state

$$\max_{b \in \mathcal{A}_{X_{t+1}}} Q(X_{t+1}, b),$$

with no consideration whether or not action b is actually enacted. Since the only off-policy algorithm I introduced was Q-learning, we need to adapt our example for action value functions. By modifying the available set of functions to

$$\{q : (x, a) \mapsto wa : w \in \mathbb{R}\},$$

and assuming that the exploration policy always picks action 1, we always stay in the first state and thus end up with the iteration:

$$\begin{aligned} w_{t+1} &= w_t + \alpha[0 + \gamma \overbrace{\max_{b \in \mathcal{A}_{X_{t+1}}} q(X_{t+1}, b, w_t)}^{=q(1,2,w_t)} - \overbrace{q(X_t, A_t, w_t)}^{=q(1,1,w_t)}] \overbrace{\nabla q(X_t, A_t, w_t)}^{=A_t=1} \\ &= w_t + \alpha[(2\gamma - 1)w_t] > w_t \quad \iff \gamma > 1/2 \end{aligned}$$

The exploration policy in this counter example is of course specially designed to cause divergence. This leads Sutton and Barto (2018, p. 263) to hypothesize, that Q-learning might still be relatively stable for exploration policies, which are greedy enough, like ε-greedy policies with small enough epsilon. They claim "to the best of [their] knowledge, Q-learning has never been found to diverge in this case, but there has been no theoretical analysis".
Since it is the combination of function approximation, bootstrapping and off-policy learning, that causes such instability, Sutton and Barto (2018, p. 264) dub this combination the "deadly triad". They claim that if any two elements are present, instability can be avoided. This claim is a bit of an oversimplification. J. N. Tsitsiklis and Roy (1997) present a proof, that TD(λ) with linear function approximation converges to some $\mathbf{w}^\lambda$, which is not equal to the minimum[2] $\mathbf{w}^*$ in general, but can be bounded by:

$$d\left(v(\cdot, \mathbf{w}^\lambda), V^\pi\right) \leq \frac{1 - \lambda\gamma}{1 - \gamma} \underbrace{d(v(\cdot, \mathbf{w}^*), V^\pi)}_{=\min_{\mathbf{w}} \overline{\mathrm{VE}}(\mathbf{w})}$$

So TD(λ) is stable enough to avoid divergence if $\gamma < 1$ (c.f. Subsection 1.6.1) in the linear function approximation case, and its distance to the minimum in limit improves with λ. And while this is far from perfect it appears to be a workable solution in practical applications.
But for non linear function approximation J. N. Tsitsiklis and Roy (1997) also present an example, where TD(0) diverges. So it might simply be bootstrapping and function approximation which do not work well with each other.
The reason for this incompatibility might be, that both concepts try to accomplish some sort of generalization. Temporal difference learning is supposed to generalize the existing estimates from older paths through the state space, to evaluations of new paths which cross states from older paths (c.f. Example 2.2.3). So these two methods have some overlap when it comes to their goals, which could explain why they get in each others way. But TD has no access to spatial similarity, which function approximation based on continuous functions automatically has, while function approximation does not have the theory of dynamic programming baked in. So neither method is a special case of the other.

[2] slight oversimplification since the minimum is not necessarily unique if the regressors are not linearly independent

Chapter 3

Stochastic Approximation – Convergence

3.1 Introduction

We motivated the concept of learning rates, by unwinding the arithmetic mean for iid X_i (c.f. Example 2.3.1), which resulted in

$$\begin{aligned}\overline{X}_n &= \overline{X}_{n-1} + \frac{1}{n}(X_n - \overline{X}_{n-1}) \\ &= \overline{X}_{n-1} + \alpha_n(X_n - \overline{X}_{n-1}).\end{aligned}$$

The structure of this iteration has similarities to numeric approximation algorithms like newton's method

$$\theta_{n+1} = \theta_n - \frac{1}{f'(\theta_n)} f(\theta_n),$$

where the learning rate would be $-1/f'(\theta_n)$ in this case. Now if we set

$$f(\theta) = \mathbb{E}[X_1] - \theta,$$

then $X_n - \theta$ could be interpreted as a blurred function evaluation of $f(\theta)$, since

$$X_n - \theta = \underbrace{\mathbb{E}[X_1] - \theta}_{=f(\theta)} + \underbrace{X_n - \mathbb{E}[X_1]}_{=:\delta_n}.$$

This leads us straight to the algorithm proposed by Robbins and Monro (1951)

$$\theta_{n+1} = \theta_n + \alpha_n Y_n(\theta_n), \tag{3.1}$$

which converges to the zero θ^* of a non-increasing function $g(\theta) = \mathbb{E}[Y_n(\theta)]$ with probability one, if $Y_n(\theta)$ is bounded, $g'(\theta^*)$ exists and is negative, and the learning rate satisfies

$$\sum_{n=1}^{\infty} \alpha_n = \infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} \alpha_n^2 < \infty. \tag{3.2}$$

This algorithm was the first algorithm in a group of algorithms, which is now known as stochastic approximation. And while most of the technical requirements for convergence change with the algorithm, the requirements (3.2) seems to be essential to this problem formulation. Although general theorems like Dvoretzky's avoid this, by replacing the explicit functional form of (3.1) with a general transformation, which has to fulfil constraints formulated with multiple sequences. And these sequences individually fulfil only parts of (3.2), but both parts of this requirement still shine through.

3.2 Learning Rates

To interpret this requirement, examining the special case $\alpha_n = n^{-\beta}$ is instructive.

$$\sum_{n=1}^{\infty} n^{-\beta} = \infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} n^{-2\beta} < \infty$$

forces $1 \geq \beta > 1/2$. Therefore the learning rate needs to behave similar to a sequence between $1/n$ and $1/\sqrt{n}$.
The second part of this requirement is somewhat reminiscent of the central limit theorem, where the variance is kept positive by dividing the sum of estimates only by $1/\sqrt{n}$. Since we know for any $\beta > 1/2$ and iid X_k:

$$\operatorname{Var}\left(n^{-\beta}\sum_{k=1}^{n} X_k\right) = n^{1-2\beta}\operatorname{Var}(X_1) \to 0 \qquad (n \to \infty)$$

But if there is a deeper relationship between these two cases, it is not as obvious as it might seem from that initial observation. Because our "unwinding" strategy does not work for $\beta \neq 1$. In fact the weights generated by learning rates inherently sum to 1, assuming $\alpha_1 = 1$, which is a result of the assumption

$$\theta_1 = X_1 = \theta_0 + \alpha_1(X_1 - \theta_0).$$

These weights can be calculated inductively as follows:

$$\begin{aligned}
\theta_n &= \theta_{n-1} + \alpha_n(X_n - \theta_{n-1}) \\
&= (1-\alpha_n)\underbrace{\theta_{n-1}}_{\overset{\text{ind.}}{=}\sum_{k=1}^{n-1}\left(\alpha_k\prod_{i=k+1}^{n-1}(1-\alpha_i)\right)X_k \overset{n=2}{=} X_1} + \alpha_n X_n \\
&= \sum_{k=1}^{n} \underbrace{\left(\alpha_k \prod_{i=k+1}^{n}(1-\alpha_i)\right)}_{=:w_k^{(n)}} X_k
\end{aligned}$$

And by induction the sum of these weights is equal to 1:

$$\sum_{k=1}^{n} w_k^{(n)} = (1-\alpha_n) \underbrace{\sum_{k=1}^{n-1} w_k^{(n-1)}}_{=1} + \alpha_n = 1$$

Setting the learning rate, corresponds therefore with shifts in the weight distribution of the observations.
A common interpretation of the first part of (3.2) is, that no matter how far away initial estimates are from the true value, the sequence needs to converge to the true value. Therefore the sum of the learning rates needs to sum to infinity. Otherwise the sequence could only cover a finite distance, given a constant delta to the target, which would be decreasing with steps towards the target in reality. And by constructing cases where we get a nudge of size $1/n$ in the right direction with equal probability to a nudge in the wrong direction of size $1/(n+1)$, it is quite intuitive, why we need the ability to cover an infinite distance. Since we could slow progress arbitrarily by increasing n.
Additionally, for the application we want to use stochastic approximation for, the case can be made, that newer observations should be weighted more, but certainly not less. And by weighting every observation equally, the learning rate $1/n$ is already at the border. More formally, if we want to ensure increasing weights

$$w_1^{(n)} \leq \cdots \leq w_n^{(n)} \qquad \forall n \in \mathbb{N},$$

we only need to ensure

$$w_{n-1}^{(n)} \leq w_n^{(n)} \qquad \forall n \in \mathbb{N},$$

since

$$w_1^{(n-1)} \leq \cdots \leq w_{n-1}^{(n-1)} \quad \Longrightarrow \quad \underbrace{(1-\alpha_n) w_1^{(n-1)}}_{=w_1^{(n)}} \leq \cdots \leq \underbrace{(1-\alpha_n) w_{n-1}^{(n-1)}}_{=w_{n-1}^{(n)}}.$$

This translates to:

$$\begin{aligned} &\alpha_{n-1}(1-\alpha_n) = w_{n-1}^{(n)} \leq w_n^{(n)} = \alpha_n, \\ \iff &\alpha_{n-1} \leq \alpha_n (1+\alpha_{n-1}) \\ \iff &\frac{\alpha_{n-1}}{1+\alpha_{n-1}} \leq \alpha_n \end{aligned}$$

In our special case this requires:

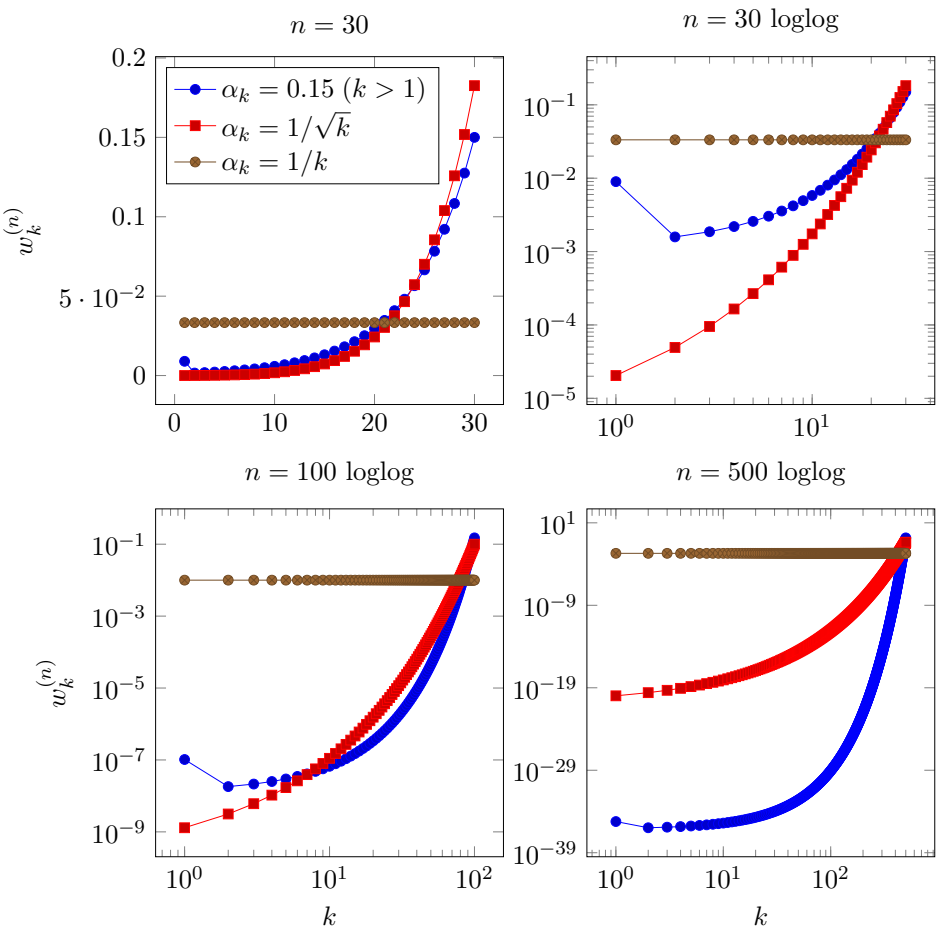
$$\begin{aligned} &\frac{1}{(n-1)^\beta + 1} = \frac{(n-1)^{-\beta}}{1+(n-1)^{-\beta}} = \frac{\alpha_{n-1}}{1+\alpha_{n-1}} \leq \alpha_n = n^{-\beta} \\ \Longrightarrow\ &n^\beta \leq (n-1)^\beta + 1 \\ \Longrightarrow\ &f(n) \coloneqq n^\beta - (n-1)^\beta - 1 \leq 0 \end{aligned}$$

For $n = 1$ this is true for any $\beta > 0$, but because of

$$f'(n) = \beta n^{\beta-1} - \beta(n-1)^{\beta-1} = \beta(n^{\beta-1} - (n-1)^{\beta-1}) \leq 0 \iff \beta \leq 1,$$

we require $\beta \leq 1$.

In applications the learning rates usually fulfil the first requirement. But constant learning rates are still quite common. This is justified by the fact, that in reality these algorithms do not run indefinitely. And for small n, constant learning rates behave relatively similar to learning rates close to $1/\sqrt{n}$. Since the weight $1/\sqrt{n}$ of the newest observation only becomes smaller than that of a constant learning rate α, once $n \geq 1/\alpha^2$. The larger problem with constant learning rates is probably the fact, that the weight of the first observation is comparatively high since $\theta_1 = X_1$ effectively sets $\alpha_1 = 1$. But this is only a problem in the very beginning for very small n, as you can see in the following plots:



3.3 Application to Reinforcement Learning

Most, if not all convergence proofs of reinforcement learning methods rely on stochastic approximation in one way or another. As an example, Watkins and Dayan's (1992) original proof for the convergence of Q-learning constructs a new MDP for every realization of the sequence $((X_t, A_t, R_{t+1}), t \in \mathbb{N}_0)$. This new MDP has the state space $\mathbb{N} \times \mathcal{X}$, where $\mathcal{X}$ is the state space of the old MDP. And this new MDP is designed in such a way, that $Q_n(x, a)$, generated by the realization of $((X_t, A_t, R_{t+1}), t \in \mathbb{N}_0)$ with Q-learning, is by definition equal to the optimal value function $Q^*((n, x), a)$ of the new MDP. Using stochastic approximation results, they then prove, that the transition probabilities and immediate rewards of this new MDP, converge to the old transition probabilities and immediate rewards in the sense:

$$r((n,x),a) \to r(x,a) \quad \text{and} \quad \underbrace{p(y \mid (n,x),a)}_{\coloneqq \sum_{m=1}^{n-1} p((m,y) \mid (n,x),a)} \to p(y \mid x,a) \quad (n \to \infty)$$
$$= \sum_{m=1}^{\infty} p((m,y) \mid (n,x),a)$$

And starting from a state (n, x), the chance of transitioning to a state (m, y) with $m < n_0$ can be bounded for any n_0. This means the states (n, x) in the new MDP behave similar to the states x in the old MDP, when n is high. This results in

$$Q_n(x,a) = Q^*((n,x),a) \approx Q^*(x,a) \qquad \text{for } n >> 0.$$

And carefully lowering the epsilon bounds, yields convergence with probability one.

But it seems much more desirable, to apply stochastic approximation directly to reinforcement learning algorithms. Because, while we do not look for the zero of a function, we do look for the fixed point of T^* in case of Q-learning, or T^π in case of TD(λ); and it seems like it should be unnecessary to have such an involved proof for convergence. The classic way to transform a fixed point problem into a zero problem would be to define

$$g(Q) \coloneqq T^*(Q) - Q,$$

which would result in the stochastic approximation algorithm

$$Q_{n+1}(x,a) = Q_n(x,a) + \alpha_n \underbrace{(\hat{Q}_n(x,a) - Q_n(x,a))}_{\mathbb{E}[\hat{Q}_n(x,a) - Q_n(x,a)]} ,$$
$$= T^* Q_n(x,a) - Q_n(x,a)$$
$$= g(Q)(x,a)$$

with

$$\hat{Q}_n(X_t, A_t) = R_{t+1} + \gamma \max_{b \in \mathcal{A}_{X_{t+1}}} Q_n(X_{t+1}, b)$$

in case of Q-learning. So it seems like we only need to find a general enough result in stochastic approximation, to cover all reinforcement learning algorithms.
One such attempt is made by Jaakkola et al. (1994), which transform the functional form above as follows:

$$\begin{aligned}
& Q_{n+1}(x,a) = Q_n(x,a) + \alpha_n(\hat{Q}_n(x,a) - Q_n(x,a)) \\
\iff \quad & \underbrace{Q_{n+1}(x,a) - Q^*(x,a)}_{=:\Delta_{n+1}(x,a)} = \underbrace{Q_n(x,a) - Q^*(x,a)}_{=:\Delta_n(x,a)} + \alpha_n(\hat{Q}_n(x,a) - Q_n(x,a)) \\
\iff \quad & \Delta_{n+1}(x,a) = \Delta_n(x,a) + \alpha_n(\underbrace{\hat{Q}_n(x,a) - Q^*(x,a)}_{=:F_n(x,a)} - \Delta_n(x,a))
\end{aligned}$$

With $\alpha_n = \beta_n$, this is essentially the iterative process of their theorem.

Theorem 3.3.1 (Jaakkola et al.)**.** *A random iterative process*

$$\Delta_{n+1}(x) = (1 - \alpha_n(x))\Delta_n(x) + \beta_n(x)F_n(x)$$

converges to zero w.p.1 under the following assumption:

1. *$x \in S$, where S is a finite set.*
2. *$\sum_n \alpha_n(x) = \infty$, $\sum_n \alpha_n^2(x) < \infty$, $\sum_n \beta_n(x) = \infty$, $\sum_n \beta_n^2(x) < \infty$ and*
$$\mathbb{E}[\beta_n(x) \mid H_n] \leq \mathbb{E}[\alpha_n(x) \mid H_n]$$
uniformly over x w.p.1.
3. *$\|\mathbb{E}[F_n(\cdot) \mid H_n, \beta_n]\|_\infty \leq \gamma \|\Delta_n\|_\infty$, where $\gamma \in (0,1)$.*
4. *$Var[F_n(x) \mid H_n, \beta_n] \leq C(1 + \|\Delta_n\|_\infty)^2$, where C is some constant.*

Here

$$H_n = \{X_n, X_{n-1}, \ldots, F_{n-1}, \ldots, \alpha_{n-1}, \ldots, \beta_{n-1}, \ldots\}$$

stands for the history up to until n. $F_n(x), \alpha_n(x)$ and $\beta_n(x)$ are allowed to depend on the past. $\alpha_n(x)$ and $\beta_n(x)$ are assumed to be non-negative and mutually independent given H_n.

"Essentially the iterative process of this theorem", because an important detail is different: Here α_n is not just a function of the history, but it is also indexed by the state (state-action). This might seem strange, since α_n is just used in one state action tuple:

$$Q_{n+1}(x,a) = \begin{cases} Q_n(x,a) + \alpha_n(\overbrace{R_{n+1} + \max_{b \in \mathcal{A}_{X_{n+1}}} Q_n(X_{n+1}, b)}^{\hat{Q}(x,a)} - Q_n(x,a)) & \substack{X_n = x, \\ A_n = a} \\ Q_n(x,a) & \text{else} \end{cases}$$

But looking closer at the process in the theorem, every state is updated in every iteration. In order to bring this definition in line with reinforcement learning algorithms, $\alpha(x)$ (and $\beta_n(x)$) is set to zero, for every state x, which is not supposed to be updated. The other way one might come up with, setting $\alpha_n(x) = \beta_n(x)$ and $F_n(x) = \Delta_n(x)$ for all states which are not supposed to be updated, would violate the third assumption.
In order to adhere to the third assumption, it is in fact assumed that an observation $\hat{Q}(x, a)$ is made in every state-action, every turn, and all these observations but one are simply discarded by $\alpha_n(x) = 0$.
Another source of confusion might be the fact, that $\alpha_n(x) = \beta_n(x)$ are supposed to be independent conditional on the history, but if they are simply functions of the history, then they are constants conditional on the history. So most use cases are in fact included.

Corollary 3.3.2 (Q-learning)**.** *Let $\mathcal{X} \times \mathcal{A}$ be finite, $\mathbb{E}[R^2_{(x,a)}] < c \in \mathbb{R}$ and $\alpha_n(x, a)$ a function of the history up to n, with*

$$\sum_{n=1}^{\infty} \alpha_n(x, a) = \infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} \alpha_n^2(x, a) < \infty \tag{3.3}$$

uniformly over all histories. Then Q-learning converges with probability one with regard to $\|\cdot\|_\infty$ to Q^*.

Note: Due to

$$\alpha_n(x, a) = 0 \qquad \text{for } (x, a) \neq (X_n, A_n)$$

and (3.3), we require that every state and action is visited an infinite amount of times.

Proof. 1. $S = \mathcal{X} \times \mathcal{A}$ is finite. 2. $\alpha_n(x, a) = \beta_n(x, a)$ fulfils the requirement. 3. Recall that $\alpha_n(x, a) = 0$ for most state-actions allows us to pretend there is an observation $\hat{Q}_n(x, a)$ in every state-action tuple. Therefore:

$$\begin{aligned}
\|\mathbb{E}[F_n(\cdot) \mid H_n, \beta_n]\|_\infty &= \max_{(x,a)\in\mathcal{X}\times\mathcal{A}} |\mathbb{E}[\hat{Q}_n(x, a) - Q^*(x, a) \mid H_n, \beta_n]| \\
&\overset{\text{Markov}}{=} \max_{(x,a)\in\mathcal{X}\times\mathcal{A}} \left| r(x, a) + \gamma \sum_{y\in\mathcal{X}} p(y \mid x, a) \max_{b\in\mathcal{A}_y} Q_n(y, b) - Q^*(x, a) \right| \\
&\overset{1.4.4}{=} \max_{(x,a)\in\mathcal{X}\times\mathcal{A}} \left| \gamma \sum_{y\in\mathcal{X}} p(y \mid x, a) \left(\max_{b\in\mathcal{A}_y} Q_n(y, b) - \max_{c\in\mathcal{A}_y} Q^*(y, c) \right) \right| \\
&\overset{(1.14)}{\leq} \max_{(x,a)\in\mathcal{X}\times\mathcal{A}} \gamma \underbrace{\sum_{y\in\mathcal{X}} p(y \mid x, a)}_{=1} \underbrace{\max_{b\in\mathcal{A}_y} |Q_n(y, b) - Q^*(y, b)|}_{\leq \|\Delta_n\|_\infty} \\
&\leq \gamma \|\Delta_n\|_\infty
\end{aligned}$$

4. To make our lives easier, we first prove:

$$\begin{aligned} & (a+b)^2 \leq 2a^2 + 2b^2 \qquad (3.4) \\ \iff \quad & a^2 + 2ab + b^2 \leq 2a^2 + 2b^2 \\ \iff \quad & 2ab \leq a^2 + b^2 \\ \iff \quad & 0 \leq a^2 - 2ab - b^2 = (a-b)^2 \end{aligned}$$

Using this, we get the last requirement:

$$\begin{aligned} & \mathrm{Var}[F_n(x,a) \mid H_n, \beta_n] \\ & \overset{\text{A.1.5}}{=} \min_f \mathbb{E}[(\hat{Q}_n(x,a) - Q^*(x,a) - f(H_n,\beta_n))^2 \mid H_n, \beta_n] \\ & \leq \mathbb{E}[(\underbrace{\hat{Q}_n(x,a)}_{=R_{(x,a)}+\gamma \max_{b\in\mathcal{A}_{Y_{(x,a)}}} Q_n(Y_{(x,a)},b)} - \gamma Q^*(y,c))^2 \mid H_n, \beta_n] \\ & \overset{(3.4)}{\leq} 2\mathbb{E}[R_{(x,a)}^2] + 2\gamma\mathbb{E}\left[\underbrace{\left(\max_{b\in\mathcal{A}_{Y_{(x,a)}}} Q_n(Y_{(x,a)},b) - Q^*(y,c)\right)^2}_{\leq \|\Delta_n\|_\infty^2} \,\middle|\, H_n, \beta_n\right] \\ & \leq C(1+\|\Delta_n\|_\infty)^2 \end{aligned}$$

□

Corollary 3.3.3 (TD(0))**.** *Let $\mathcal{X}$ ($\mathcal{X}\times\mathcal{A}$) be finite, $\mathbb{E}[R_{(x,a)}^2] < c \in \mathbb{R}$ and $\alpha_n(y)$ with $y \in \mathcal{X}$ ($y \in \mathcal{X}\times\mathcal{A}$) a function of the history up to n, with*

$$\sum_{n=1}^{\infty} \alpha_n(y) = \infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} \alpha_n^2(y) < \infty \qquad (3.5)$$

uniformly over all histories. Then TD learning (Sarsa) converges with probability one with regard to $\|\cdot\|_\infty$ to T^π (Q^π).

Proof. Analogous to Q-learning. □

When trying to apply this theorem to TD(λ), one stumbles into a problem: The λ-return is defined using returns at multiple timesteps. The λ-return essentially contains too much information about the future. So we can not put it into the history without problems for the conditional expectation of the the next λ-return. Jaakkola et al. (1994) try to circumvent this problem, by applying Theorem 3.3.1 to the batch TD(λ) and using an entire episode as one time step.
In order to rewrite the batch TD(λ) algorithm (2.8), such that Theorem 3.3.1 can be applied, we define $m_n(x)$ to be the number of visits to state x in episode n, i.e.:

$$m_n(x) \coloneqq \sum_{i=1}^{T_n-1} \mathbb{1}_{X_i=x}.$$

If we further denote the λ-return of the k-th visit to state x by

$$V_n^\lambda(x;k) \coloneqq \hat{V}_n^\lambda(X_{i_x(k)}) \quad \text{with} \quad i_x(k) \coloneqq \min\{j : j \geq i_x(k-1) \text{ and } X_j = x\},$$

we can rewrite (2.8), as:

$$\begin{aligned} V_{n+1}(x) &= V_n(x) + \sum_{j=0}^{T_n-1} \alpha_j [\hat{V}_n^\lambda(X_j) - V_n(x)] \mathbb{1}_{X_j=x} \\ &= V_n(x) + \sum_{k=1}^{m(x)} \alpha_{i_x(k)} [V_n^\lambda(x;k) - V_n(x)] \end{aligned} \tag{3.6}$$

And at least in the case of batch TD(1) (Monte Carlo), the learning rate can be written independent of the timestamp within the episode, using our state indexed learning rate:

$$\alpha_{i_x(k)} \overset{(2.9)}{=} \left(n \sum_{i=0}^{T_n-1} \mathbb{1}_{X_{i_x(k)}=X_i} \right)^{-1} = \left(n \sum_{i=0}^{T_n-1} \mathbb{1}_{x=X_i} \right)^{-1} = \frac{1}{n \cdot m_n(x)} =: \alpha_n(x)$$

Therefore we can transform (3.6) into:

$$\begin{aligned} V_{n+1}(x) &= V_n(x) + \sum_{k=1}^{m_n(x)} \alpha_{i_x(k)} [V_n^\lambda(x;k) - V_n(x)] \\ &= V_n(x) + \alpha_n(x) \left[\sum_{k=1}^{m(x)} V_n^\lambda(x;k) - m_n(x) V_n(x) \right] \\ &= [1 - \alpha_n(x) m_n(x)] V_n(x) + \alpha_n(x) \sum_{k=1}^{m(x)} V_n^\lambda(x;k) \end{aligned}$$

In order to get this recursion into the shape of the theorem, we subtract $V^\pi(x)$:

$$\begin{aligned} &\overbrace{V_{n+1}(x) - V^\pi(x)}^{=:\Delta_{n+1}(x)} \\ &= [1 - \underbrace{\alpha_n(x) m_n(x)}_{=:\tilde{\alpha}_n(x)}][V_n(x) - V^\pi(x)] + \alpha_n(x) \left[\sum_{k=1}^{m_n(x)} V_n^\lambda(x;k) - m_n(x) V^\pi(x) \right] \\ &= [1 - \tilde{\alpha}_n(x)]\Delta_n(x) + \underbrace{\alpha_n(x)\mathbb{E}[m_n(x)]}_{=:\beta_n(x)} \underbrace{\frac{1}{\mathbb{E}[m_n(x)]} \left[\sum_{k=1}^{m_n(x)} (V_n^\lambda(x;k) - V^\pi(x)) \right]}_{=:F_n(x)} \end{aligned}$$

In order to be able to apply the theorem, we need

$$\begin{aligned} \mathbb{E}[\alpha_n(x) m_n(x) \mid H_n] &= \mathbb{E}[\tilde{\alpha}_n(x) \mid H_n] \\ &\geq \mathbb{E}[\beta_n(x) \mid H_n] = \mathbb{E}[\alpha_n(x)\mathbb{E}[m_n(x)] \mid H_n], \end{aligned}$$

which looks like it would pose no problem, since the number of repeat visits to some state is independent of the previous episodes. But remember, that we wanted to use

$$\alpha_n(x) = \frac{1}{n \cdot m_n(x)}$$

and suddenly we would need

$$\mathbb{E}\left[\frac{\mathbb{E}[m_n(x)]}{m_n(x)} \mid H_n\right] \leq 1.$$

Since Jaakkola et al. (1994) do not address this, Monte Carlo seems to be excluded from their version of batch TD(λ). The closest we could get, would be to set

$$\alpha_n(x) \coloneqq \frac{1}{n\mathbb{E}[m_n(x)]}.$$

But then the question remains, how we could possibly know the average number of visits to state x in order to use this as our learning rate. Whether or not the other requirements for $\tilde{\alpha}_n$ and β_n are met, is another unaddressed problem. Now to $F_n(x)$, we have

$$\begin{aligned}
&\|\mathbb{E}[F_n(\cdot) \mid H_n]\|_\infty \\
&= \max_{x\in\mathcal{X}} \frac{1}{\mathbb{E}[m_n(x)]} \left| \mathbb{E}\left[\sum_{k=1}^{m_n(x)} (V_n^\lambda(x;k) - V^\pi(x)) \;\middle|\; H_n \right] \right| \\
&= \max_{x\in\mathcal{X}} \frac{1}{\mathbb{E}[m_n(x)]} \left| \mathbb{E}\left[\sum_{k=1}^{\infty} (V_n^\lambda(x;k) - V^\pi(x)) \mathbb{1}_{m_n(x)\geq k} \;\middle|\; H_n \right] \right| \\
&\overset{\text{Fub.}}{\leq} \max_{x\in\mathcal{X}} \frac{1}{\mathbb{E}[m_n(x)]} \sum_{k=1}^{\infty} \underbrace{\left| \mathbb{E}\left[(V_n^\lambda(x;k) - V^\pi(x)) \mathbb{1}_{m_n(x)\geq k} \;\middle|\; H_n \right] \right|}_{(*)}
\end{aligned}$$

Because of $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X \mid Y]]$ for any random variables X, Y, we can write:

$$\begin{aligned}
(*) &= \left| \mathbb{E}\left[\mathbb{E}\left[(V_n^\lambda(x;k) - V^\pi(x)) \mathbb{1}_{m_n(x)\geq k} \;\middle|\; \mathbb{1}_{m_n(x)\geq k}, H_n \right] \;\middle|\; H_n \right] \right| \\
&\leq \mathbb{E}\left[\left| \mathbb{E}\left[(V_n^\lambda(x;k) - V^\pi(x)) \;\middle|\; \mathbb{1}_{m_n(x)\geq k}, H_n \right] \right| \mathbb{1}_{m_n(x)\geq k} \;\middle|\; H_n \right] \\
&\overset{(*^2)}{=} \mathbb{E}\left[\underbrace{\left| \mathbb{E}\left[(V_n^\lambda(x;k) - V^\pi(x)) \;\middle|\; m_n(x) \geq k, H_n \right] \right|}_{\overset{(*^3)}{=} \left| \mathbb{E}\left[(\hat{V}_n^\lambda(x) - V^\pi(x)) \;\middle|\; H_n \right] \right|} \mathbb{1}_{m_n(x)\geq k} \;\middle|\; H_n \right] \\
&= \left| \mathbb{E}\left[\hat{V}_n^\lambda(x) \;\middle|\; H_n \right] - V^\pi(x) \right| \\
&\overset{2.5.2}{\leq} \gamma \|V_n - V^\pi\|_\infty
\end{aligned}$$

($*^2$) Due to

$$\mathbb{E}[X \mid \mathbb{1}_A] \overset{\text{A.1.6}}{=} \mathbb{E}[X \mid A]\mathbb{1}_A + \mathbb{E}[X \mid A^\complement]\mathbb{1}_{A^\complement},$$

we get for any random variable X and measurable set A:

$$\mathbb{E}[X \mid \mathbb{1}_A]\mathbb{1}_A = \mathbb{E}[X \mid A]\mathbb{1}_A^2 + \mathbb{E}[X \mid A^\complement] \underbrace{\mathbb{1}_{A^\complement}\mathbb{1}_A}_{=0} = \mathbb{E}[X \mid A]\mathbb{1}_A.$$

($*^3$) If we condition on visiting state x at least k times, then $V_n^\lambda(x;k)$ is distributed like any unconditioned λ-return after x due to the Markov property of the MDP.
We therefore get:

$$\begin{aligned}\|\mathbb{E}[F_n(\cdot) \mid H_n]\|_\infty &\leq \max_{x\in\mathcal{X}} \frac{1}{\mathbb{E}[m_n(x)]}\gamma\|V_n - V^\pi\|_\infty \underbrace{\sum_{k=1}^{\infty}\mathbb{E}[\mathbb{1}_{m_n(x)\geq k} \mid H_n]}_{=\mathbb{E}[m_n(x)]} \\ &= \gamma\|V_n - V^\pi\|_\infty\end{aligned}$$

Jaakkola et al. (1994) claim that the variance of $F_n(x)$ can similarly be bounded by

$$\mathbb{E}[m_n(x)^4]\|V_n\|_\infty^2,$$

and argue, that since the absorption time $m_n(x)$ is geometrically distributed in an absorbing Markov chain, and thus

$$\mathbb{E}[m_n(x)^4] \leq C$$

the variance of $F_n(x)$ fulfils the requirements of the theorem.
They also provide a proof for the convergence of on-line TD(λ). The basis of their method is the realization, that the difference between batch TD(λ) and on-line TD(λ) vanishes in the limit.

3.4 Proof of Theorem 3.3.1

I do not find the proof presented by Jaakkola et al. entirely convincing. I still want to present the theorem, because the gaps in their proof are generally not irredeemable, and I still think that this more general approach is the right approach for tackling the convergence of reinforcement learning algorithms. Additionally there are some interesting ideas in their somewhat incomplete proof.
Another well known proof, applying stochastic approximation more directly to Q-learning, is the proof by John N. Tsitsiklis (1994), which focuses on algorithms which utilize outdated Q-values due to parallelization, i.e. asynchronous stochastic approximation. Although cursory reading left the impression of proofs coming out of the blue, without much insight on why they work.
Lastly, there have been attempts to utilize the theory of ordinary differential equations, in order to achieve convergence proofs for stochastic approximation in more general settings (Kushner 1997; Borkar and Meyn 2000).

The proof by Jaakkola et al. of their theorem is built on Dvoretzky's (1956) very general theorem about one dimensional sequences. The proof for Dvoretzky's theorem itself can not be adapted for higher dimensional sequences, since it relies on arguments about the signum of the sequence.

Theorem 3.4.1 (Dvoretzky)**.** *Let*

$$\alpha_n, \beta_n, \gamma_n \colon \mathbb{R}^n \to \mathbb{R}_{\geq 0}, \qquad n \in \mathbb{N},$$

be measurable functions satisfying:

1. *The functions $\alpha_n(r_1, \ldots, r_n)$ are uniformly bounded, and*

$$\lim_{n\to\infty} \alpha_n(r_1, \ldots, r_n) = 0 \tag{3.7}$$

   uniformly for all sequences $r_1, r_2, \ldots$, i.e. $\lim_n \|\alpha_n\|_\infty \to 0$.

2. *The series*

$$\sum_{n=1}^{\infty} \beta_n(r_1, \ldots, r_n) \tag{3.8}$$

   is uniformly bounded (i.e. $\|\sum_n^\infty \beta_n\|_\infty < C$) and uniformly convergent (i.e. $\lim_k \|\sum_n^k \beta_n - \sum_n^\infty \beta_n\|_\infty = 0$)

3. *For all sequences $r_1, r_2, \ldots$ which satisfy*

$$\sup_n |r_n| < L \tag{3.9}$$

   for an arbitrary $L \in \mathbb{R}$,

$$\sum_{n+1}^{\infty} \gamma_n(r_1, \ldots, r_n) = \infty \tag{3.10}$$

   holds uniformly. (i.e. $\lim_k \inf_{\|r\|_\infty < L} \sum_n^k \gamma_n \to \infty$)

Let θ be a real number and T_n, $n \in \mathbb{N}$ be a measurable transformation satisfying

$$|T_n(r_1, \ldots, r_n) - \theta| \leq \max\{\alpha_n, (1+\beta_n)|r_n - \theta| - \gamma_n\} \tag{3.11}$$

for all real $r_1, \ldots, r_n$ with $\alpha_n = \alpha_n(r_1, \ldots, r_n)$, etc. Let X_1 and Y_n, $n \in \mathbb{N}$ be random variables. Define the recursion

$$X_{n+1} = T_n[X_1, \ldots, X_n] + Y_n \qquad \text{for } n \geq 1$$

Then the conditions

1. *$E[X_1^2] < \infty$,*

2. $\sum_{n=1}^{\infty} \mathbb{E}[Y_n^2] < \infty$,

3. $\mathbb{E}[Y_n \mid X_1, \ldots, X_n] = 0$ *with probability 1 for all* $n \in \mathbb{N}$,

imply convergence in L^2

$$\lim_{n\to\infty} \mathbb{E}[(X_n - \theta)^2] = 0,$$

and almost sure convergence

$$\mathbb{P}(\lim_{n\to\infty} X_n = \theta) = 1.$$

Proof. see Dvoretzky (1956) □

Remark 3.4.2. The condition (3.11) is weaker than (can be replaced by)

$$|T_n(r_1, \ldots, r_n) - \theta| \leq \max\{\alpha_n, (1 + \beta_n - \gamma_n)|r_n - \theta|\} \tag{3.12}$$

Proof. Essential to this proof is the construction of a sequence ρ_n such that $\rho_n \to 0$ but retains

$$\sum_{n}^{\infty} \gamma_n \rho_n = \infty \tag{3.13}$$

in the same sense as γ_n.
We will first discuss the simpler case where $\alpha_n, \beta_n, \gamma_n$ are simply real sequences, and later argue that the same proof can be applied to the defined real functions.
Let $K_1 = 0$, due to

$$\lim_{K\to\infty} \sum_{j=K_n+1}^{K} \gamma_j \to \infty \tag{3.14}$$

we can select K_{n+1} inductively, such that

$$\sum_{j=K_n+1}^{K_{n+1}} \gamma_j > n. \tag{3.15}$$

And we define

$$\rho_{K_n+1} \coloneqq \ldots \coloneqq \rho_{K_{n+1}} \coloneqq \frac{1}{n}. \tag{3.16}$$

Then we get

$$\sum_{j=1}^{K_{n+1}} \gamma_j \rho_j = \sum_{k=1}^{n} \sum_{j=K_k+1}^{K_{k+1}} \gamma_j \rho_j = \sum_{k=1}^{n} \frac{1}{k} \sum_{j=K_k+1}^{K_{k+1}} \gamma_j > \sum_{k=1}^{n} \frac{k}{k} = n \to \infty \quad (n \to \infty)$$

Due to γ_n diverging uniformly, we can select K_{n+1} such that

$$\inf_{\|r\|_\infty<L} \sum_{j=K_n+1}^{K_{n+1}} \gamma_j(r_1,\dots,r_j) > n.$$

Then we can prove the properties of the sequence ρ_n the same way, due to:

$$\inf_{\|r\|_\infty<L} \sum_{j=1}^{K_{n+1}} \gamma_j(r_1,\dots,r_j)\rho_j \geq \sum_{k=1}^{n} \inf_{\|r\|_\infty<L} \sum_{j=K_k+1}^{K_{k+1}} \gamma_j(r_1,\dots,r_j)\rho_j$$

We can now use this sequence ρ_n to make a case by case analysis.
If $|r_n - \theta| \geq \rho_n$, then

$$(1+\beta_n-\gamma_n)|r_n-\theta| \leq (1+\beta_n)|r_n-\theta| - \underbrace{\gamma_n\rho_n}_{=:\tilde{\gamma}_n} \tag{3.17}$$

If $|r_n - \theta| \leq \rho_n$, then

$$(1+\beta_n-\gamma_n)|r_n-\theta| \leq (1+\beta_n)\rho_n \tag{3.18}$$

Therefore we get the required inequality

$$|T_n(r_1,\dots,r_n)-\theta| \leq \max\left\{ \underbrace{\alpha_n, (1+\beta_n)\rho_n,}_{\tilde{\alpha}_n:=\max\{\alpha_n,(1+\beta_n)\rho_n\}} \quad (1+\beta_n)|r_n-\theta| - \tilde{\gamma}_n \right\}$$

Where $\tilde{\gamma}_n$ has the required property (3.10) due to (3.13) and $\tilde{\alpha}$ has the required property (3.7), since β_n converges uniformly due to (3.8). □

Lemma 3.4.3. *A random process*

$$w_{n+1}(x) = (1-\alpha_n(x))w_n(x) + \beta_n(x)\eta_n(x) \qquad x \in S$$

converges to zero with probability one, if

1. $\sum_n \alpha_n(x) = \infty$, $\sum_n \alpha_n^2(x) < \infty$, $\sum_n \beta_n(x) = \infty$ *and* $\sum_n \beta_n^2(x) < \infty$ *uniformly over every possible history with probability one.* α_n *and* β_n *are assumed to be non-negative and mutually independent given history* H_n.

2. $\mathbb{E}[\eta_n(x) \mid H_n, \beta_n]$ *and* $\mathbb{E}[\eta_n^2(x) \mid H_n, \beta_n] \leq C$ *w.p.1, where*

$$H_n = \{w_n, w_{n-1}, \dots, \eta_{n-1}, \dots, \alpha_{n-1}, \dots, \beta_{n-1}, \dots\}$$

Proof. According to Jaakkola et al., the lemma "follows directly from Dvoretzky's [...] theorem", which is presented above (Thm. 3.4.1). And if $\alpha_n(x)$ is a function of $(w_1,\dots,w_n)$ this is certainly true. The proof follows:

Since there are no interactions between the processes indexed by $x \in S$, we can essentially treat it like a one dimensional process for a given x and apply Dvoretzky's theorem. We set

$$T_n(r_1, \dots, r_n) \coloneqq (1 - \alpha_n(x)(r_1, \dots, r_n))w_n(x) \tag{3.19}$$
$$\leq \max\{\cdot, (1 + 0 - \alpha_n(x))|w_n(x)|\} \tag{3.20}$$

which fulfils (3.12) with $\alpha_n(x) = \gamma_n$. And

$$Y_n \coloneqq \beta_n(x)\eta_n(x)$$

which fulfils the required conditions

$$\sum_{n=1}^{\infty} \mathbb{E}[Y_n^2] = \sum_{n=1}^{\infty} \mathbb{E}[\beta_n(x)^2 \underbrace{\mathbb{E}[\eta_n^2(x) \mid H_n, \beta_n,]]}_{\leq C} \overset{\text{Fub.}}{=} C\mathbb{E}\left[\underbrace{\sum_{n=1}^{\infty} \beta_n(x)^2}_{<\infty \text{ (uniformly)}}\right] < \infty.$$

And since $w_n, \dots, w_1$ is part of H_n:

$$\mathbb{E}[Y_n \mid w_n, \dots, w_1] = \mathbb{E}\left[\beta_n(x) \underbrace{\mathbb{E}[\eta_n(x) \mid H_n, \beta_n]}_{=0} \middle| w_n, \dots, w_1\right] = 0.$$

But if $\alpha_n(x)$ is not a function of $(w_1, \dots, w_n)$ which is not required in their lemma formulation, and does not make sense, since the $w_1, \dots, w_n$ represent just an auxiliary process which is not known to the algorithm. The learning rate rather depends on the states visited.
But Dvoretzky's proof only really requires

$$\mathbb{E}[Y_n \mid T_n] = \mathbb{E}[Y_n \mid w_1, \dots, w_n] = 0$$

in order to argue, that

$$\int_A Y_n T_n dP = 0$$

for a set $A \in \sigma(w_1, \dots, w_n)$ from the sigma algebra generated by the history of the process. So the theorem can most likely be generalized for T_n a function of a history H_n, where

$$\mathbb{E}[Y_n \mid H_n] = 0$$

and $\sigma(w_1, \dots, w_n) \subseteq \sigma(H_n)$. But this is not quite a "direct" application. Additionally the lemma and consequently the theorems has oddly strict requirements. Since it is only really needed, that

$$\sum_n \alpha_n(x) = \infty \qquad \text{and} \qquad \sum_n \beta_n^2(x) < \infty,$$

and not the other two of the four requirements. And while α_n needs to fulfil its requirement uniformly, β_n is only really required to fulfil it in expected value. □

Besides the interaction effects between different x, the variance of $F_n(x)$ in (Thm. 3.3.1) is only bounded by the sequence itself and not by a constant, like in this lemma. The next two lemmas essentially try to bootstrap the boundedness of the variance, by artificially bounding the sequence and therefore the variance, proving convergence, and then remove the bounds on the sequence without losing convergence.

Lemma 3.4.4. *Consider a stochastic iteration*

$$X_{n+1}(x) = G_n(X_n, Y_n, x),$$

where G_n is a sequence of functions and Y_n is a random process. Assume that G_n is scale invariant w.r.t the first variable, i.e.

$$G_n(\lambda X_n, Y_n, x) = \lambda G_n(X_n, Y_n, x)$$

We say, we keep $\|X_n\|_\infty$ bounded by C_0 by scaling with $0 < \lambda < 1$ in the sense of $\tilde{X}_1 = X_1$ and

$$\tilde{X}_{n+1} = \lambda^{K_n} G_n(\tilde{X}_n, Y_n, x)$$

where

$$K_n = \min\{k \in \mathbb{N}_0 : \lambda^k \|G_n(\tilde{X}_n, Y_n, \cdot)\|_\infty \leq C_0\}$$

If $\tilde{X}_n$ converges almost surely to zero, then X_n converges to zero almost surely.

Remark 3.4.5. It should be noted, that the transformation from $(X_1, \ldots, X_n)$ to $(\tilde{X}_1, \ldots, \tilde{X}_n)$ is bijective, which means that the sigma algebras generated by the two sequences are the same. Functions depending on $(X_n, \ldots, X_1)$ can be rewritten as functions of $(\tilde{X}_n, \ldots, \tilde{X}_1)$, and properties of these functions depending on the infimum or supremum over the history (like the properties of $\alpha_n, \beta_n, \gamma_n$ in Dvoretzky's theorem) do not change.

Proof. Assume $\tilde{X}_n(\omega)$ converges to zero, then there exists a a natural number n_0, after which $\tilde{X}_n(\omega)$ stays below C_0 and thus $K_n(\omega) = 0$ for all $n \geq n_0$. Therefore the total amount of scaling is finite. And since G_n is scale invariant w.r.t to the first variable, it does not matter, when this scaling happens. Not scaling by $\lambda^{K_n(\omega)}$ at time step n translates to scaling all the elements after n_0 by

$$\lambda^{-\sum_{n=1}^{n_0} K_n(\omega)}.$$

And scaling a sequence converging to 0 by a finite amount does not change convergence. Therefore $X_n(\omega)$ converges too. Thus almost sure convergence of $\tilde{X}_n$ implies almost sure convergence of X_n. □

We can therefore "bootstrap convergence" for scale invariant processes, which we can apply to the following case:

Lemma 3.4.6. *A stochastic process*

$$X_{n+1}(x) = (1 - \alpha_n(x))X_n(x) + \gamma\beta_n(x)\|X_n\|_\infty$$

converges almost surely to zero, provided

1. *$x \in S$, where S is a finite set.*
2. *$\sum_n \alpha_n(x) = \infty$, $\sum_n \alpha_n^2(x) < \infty$, $\sum_n \beta_n(x) = \infty$, $\sum_n \beta_n^2(x) < \infty$ and*
$$\mathbb{E}[\beta_n(x) \mid H_n] \leq \mathbb{E}[\alpha_n(x) \mid H_n]$$
uniformly over x w.p.1.

Where

$$H_n = \{X_n, X_{n-1}, \dots, F_{n-1}, \dots, \alpha_{n-1}, \dots, \beta_{n-1}, \dots\}$$

and $\alpha_n(x)$ and $\beta_n(x)$ are assumed to be non-negative and mutually independent given H_n.

Proof (incomplete). Because of the previous lemma, we will instead look at the scaled process, but denote it with X_n anyway. More specifically

$$X_{n+1}(x) = \lambda^{K_n} G_n(X_n, \alpha_n, \beta_n, x)$$

with

$$G_n(X_n, \alpha_n, \beta_n, x) = (1 - \alpha_n(x))X_n(x) + \gamma\beta_n(x)\|X_n\|_\infty.$$

Let $C_1 > C_0$, where C_0 is the scaling parameter which bounds $\|X_n\|_\infty$. Our strategy is, to bound the sequence by smaller and smaller constants. We define

$$C_{n+1}(\varepsilon) \coloneqq C_1/(1+\varepsilon)^n \to 0 \qquad (n \to \infty).$$

And for a $\varepsilon_0 > 0$ with $1/(1+\varepsilon_0) > \gamma$, we write $C_n \coloneqq C_n(\varepsilon_0)$, and observe

$$C_n(\varepsilon) - \gamma C_{n-1}(\varepsilon) = \left(\frac{1}{1+\varepsilon} - \gamma\right) C_{n-1}(\varepsilon) > 0 \qquad \text{for } 0 < \varepsilon \leq \varepsilon_0. \tag{3.21}$$

Step 1: In this first step, we want to reduce the bound of the sequence from C_1 to C_2. Since we care about the behaviour of the sequence in the limit and $\alpha_n(x) \to 0$ uniformly we can assume without loss of generality $\alpha_n(x) < 1$ and thus

$$|X_{n+1}(x)| \leq |G_n(X_n, \alpha_n, \beta_n, x)| \leq (1 - \alpha_n(x))|X_n(x)| + \gamma\beta_n(x)C_1. \tag{3.22}$$

At this point we got rid of the interdependency of the different $x \in S$, and bounded the sequence. The goal is now, to iteratively apply Dvoretzky's theorem. The inequality (3.22) implies:

$$\underbrace{|X_{n+1}(x)| - \gamma C_1}_{=:\Delta_{n+1}^{C_1}(x)} \leq (1 - \alpha_n(x)) \underbrace{(|X_n(x)| - \gamma C_1)}_{=:\Delta_n^{C_1}(x)} + (\beta_n(x) - \alpha_n(x))\gamma C_1$$

$$\leq (1 - \alpha_n(x))\Delta_n^{C_1}(x) + \underbrace{\left(\beta_n(x) \underbrace{-\mathbb{E}[\beta_n(x) \mid H_n] + \mathbb{E}[\alpha_n(x) \mid H_n]}_{\geq 0} - \alpha_n(x)\right)\gamma C_1}_{=:Y_n(x)}$$

$$\overset{\text{ind.}}{\leq} (1 - \alpha_n(x))\tilde{\Delta}_n^{C_1}(x) + Y_n(x) =: \tilde{\Delta}_{n+1}^{C_1} \qquad \text{with } \tilde{\Delta}_1^{C_1} \coloneqq \Delta_1^{C_1}$$

And due to

$$\mathbb{E}[Y_n(x) \mid H_n] = 0,$$

and

$$\begin{aligned}
\sum_{n=1}^{\infty} \mathbb{E}[Y_n^2] &= \sum_{n=1}^{\infty} \mathbb{E}[\mathbb{E}[Y_n^2 \mid H_n]] \\
&\overset{(3.4)}{\leq} 2 \sum_{n=1}^{\infty} \mathbb{E}[\mathrm{Var}(\beta_n(x) \mid H_n)] + \mathbb{E}[\mathrm{Var}(\alpha_n(x) \mid H_n)] \\
&\overset{\text{A.1.5}}{\leq} 2 \sum_{n=1}^{\infty} \mathbb{E}[\mathbb{E}[\beta_n^2(x) \mid H_n]] + \mathbb{E}[\mathbb{E}[\alpha_n^2(x) \mid H_n]] \\
&\overset{\text{Fub.}}{=} 2\mathbb{E}\left[\sum_{n=1}^{\infty} \beta_n^2(x)\right] + 2\mathbb{E}\left[\sum_{n=1}^{\infty} \alpha_n^2(x)\right] < \infty,
\end{aligned}$$

we can apply Dvoretzky's theorem like in Lemma 3.4.3, and get

$$\tilde{\Delta}_n^{C_1}(x) \xrightarrow{\text{a.s.}} 0 \qquad (n \to \infty).$$

Since we can do this for all finite $x \in S$, this implies that $\|\tilde{\Delta}_n^{C_1}\|_\infty \xrightarrow{\text{a.s.}} 0$, which in turn implies:

$$\begin{aligned}
\lim_{n\to\infty} \mathbb{P}\left(\sup_{k\geq n} |X_k(x)| > C_2\right) &= \lim_{n\to\infty} \mathbb{P}\left(\sup_{k\geq n} \underbrace{|X_k(x)| - \gamma C_1}_{=\Delta_k^{C_1}(x) \leq \tilde{\Delta}_k^{C_1}(x)} > \underbrace{C_2 - \gamma C_1}_{=:\delta > 0 \ (3.21)}\right) \\
&\leq \lim_{n\to\infty} \mathbb{P}\left(\sup_{k\geq n} \tilde{\Delta}_k^{C_1}(x) > \delta\right) \\
&\leq \lim_{n\to\infty} \mathbb{P}\left(\sup_{k\geq n} |\tilde{\Delta}_k^{C_1}(x)| > \delta\right) = 0
\end{aligned}$$

Due to

$$\lim_{n\to\infty} \mathbb{P}\left(\sup_{k\geq n} \|X_k\|_\infty > C_2\right) \leq \lim_{n\to\infty} \sum_{x\in\mathcal{X}} \mathbb{P}\left(\sup_{k\geq n} |X_k(x)| > C_2\right) = 0,$$

we can select $M_1(p_1) \in \mathbb{N}$ such that the probability of

$$A_1(p_1) \coloneqq \left\{\sup_{k\geq M_1(p_1)} \|X_k\|_\infty \leq C_2\right\}$$

is larger than $p_1 \in [0, 1)$.

Step 2: In this step we try to show, that the probability of $\|X_n\|_\infty$ converging to zero is larger than zero. In order to do that we select $p_1, p_2, \cdots < 1$, such that

$$\prod_{n=1}^{\infty} p_n > 0 \left(\iff \sum_{n=1}^{\infty} \underbrace{-\log(p_n)}_{>0 \iff p_n < 1} < \infty \right).$$

We can now select $p_1^{(2)}, p_2^{(2)} \in [0,1)$ such that

$$p_2 \leq p_1^{(2)} p_2^{(2)}.$$

On the set $A_1\left(p_1^{(2)}\right)$, $\|X_k\|_\infty$ is bounded by C_2 for all $k \geq M_1\left(p_1^{(2)}\right)$. So we can repeat the argument from Step 1 to get

$$p_2^{(2)} \leq \mathbb{P}\left(\overbrace{\sup_{k \geq M_2\left(p_2^{(2)}\right)} \|X_k\|_\infty \leq C_3}^{=A_2\left(p_2^{(2)}\right)} \,\middle|\, A_1\left(p_1^{(2)}\right) \right),$$

and thus

$$\begin{aligned} p_2 \leq p_1^{(2)} p_2^{(2)} &\leq \mathbb{P}\left[A_2\left(p_2^{(2)}\right) \,\middle|\, A_1\left(p_1^{(2)}\right)\right] \mathbb{P}\left[A_1\left(p_1^{(2)}\right)\right] \\ &= \mathbb{P}\left[A_2\left(p_2^{(2)}\right), A_1\left(p_1^{(2)}\right)\right] \end{aligned}$$

More generally we can iterate Step 1 for

$$p_n \leq p_1^{(n)} \dots p_n^{(n)}$$

and

$$A_n(p) \coloneqq \left\{ \sup_{k \geq M_n(p)} \|X_k\|_\infty \leq C_{n+1} \right\}$$

by selecting $M_n(p)$ appropriately according to step 1, to fulfil

$$p_n \leq \mathbb{P}\left[A_n\left(p_n^{(n)}\right), \dots, A_1\left(p_1^{(n)}\right)\right] \tag{3.23}$$

This leads us to

$$\begin{aligned} 0 < \prod_{n=1}^{\infty} p_n &\leq \prod_{n=1}^{\infty} \mathbb{P}\left[A_n\left(p_n^{(n)}\right), \dots, A_1\left(p_1^{(n)}\right)\right] \\ &\leq \prod_{n=1}^{\infty} \mathbb{P}\left[A_n\left(p_n^{(n)}\right) \,\middle|\, A_{n-1}\left(p_{n-1}^{(n)}\right), \dots, A_1\left(p_1^{(n)}\right)\right] \\ &\overset{(*)}{\leq} \prod_{n=1}^{\infty} \mathbb{P}\left[A_n\left(p_n^{(n)}\right) \,\middle|\, A_{n-1}\left(p_{n-1}^{(n-1)}\right), \dots, A_1\left(p_1^{(1)}\right)\right] \\ &= \mathbb{P}\left(\bigcap_{n=1}^{\infty} A_n\left(p_n^{(n)}\right)\right) \\ &= \mathbb{P}\left(\bigcap_{n=1}^{\infty} \left\{\forall k \geq M_n\left(p_n^{(n)}\right): \quad \|X_k\|_\infty \leq C_{n+1}\right\}\right) \\ &\leq \mathbb{P}\left(\forall \delta > 0,\ \exists M \in \mathbb{N},\ \forall k \geq M: \quad \|X_k\|_\infty \leq \delta\right) \\ &= \mathbb{P}\left(\lim_{k \to \infty} \|X_k\|_\infty = 0\right) \end{aligned} \tag{3.24}$$

$(*)$ is the missing step in this proof. The intuition for this inequality follows: We can assume w.l.o.g., that $p_i^{(n)} \geq p_i^{(i)}$ for $i \leq n$, since we want the product of the $p_i^{(n)}$ to be larger than p_n. Thus we can also assume w.l.o.g. that

$$M_i^{(n)} \coloneqq M_i\left(p_i^{(n)}\right) \geq M_i\left(p_i^{(i)}\right) =: M_i,$$

since this is the time step from which on we require $\|X_k\|_\infty \leq C_{i+1}$. And we only increase the probability by delaying the requirement. But if we condition on $\|X_k\|_\infty \leq C_{i+1}$ at an earlier time, then the probability, that

$$\|X_k\|_\infty \leq C_{n+1}$$

from M_n on, should increase. I.e.

$$\mathbb{P}\left[\sup_{k\geq M_n}\|X_k\|_\infty \leq C_{n+1} \,\middle|\, \sup_{k\geq M_{n-1}^{(n)}}\|X_k\|_\infty \leq C_n, \ldots, \sup_{k\geq M_1^{(n)}}\|X_k\|_\infty \leq C_1\right]$$
$$\leq \mathbb{P}\left[\sup_{k\geq M_n}\|X_k\|_\infty \leq C_{n+1} \,\middle|\, \sup_{k\geq M_{n-1}^{(n-1)}}\|X_k\|_\infty \leq C_n, \ldots, \sup_{k\geq M_1^{(1)}}\|X_k\|_\infty \leq C_1\right]$$

In order to satisfy this inequality, it might be necessary to require α_n, β_n to lack properties, which would slingshot sequences getting close to zero early back up. But since the learning rate should diminish, when the sequence gets closer to the value function, these assumptions would not have any practical implications. And since it is not possible to discern the distance to the value function in reality, the learning rate will be independent of this distance in virtually all practical applications. But since $\alpha_n(x)$ decreases $|X_n(x)|$ and $\beta_n(x)$ increases $\|X_n\|_\infty$ only by the factor γ, the existing requirement

$$\mathbb{E}[\beta_n(x) \mid H_n] \leq \mathbb{E}[\alpha_n(x) \mid H_n]$$

is probably enough.

Step 3: By using $C_{n+1}(\varepsilon)$ instead of $C_{n+1} = C_{n+1}(\varepsilon_0)$ in (3.24), we can shift the left side towards one. And since the product is convergent, it will be enough for the individual multiples to converge to one.

We define for $\varepsilon \leq \varepsilon_0$ and thus $C_n(\varepsilon) \geq C_n(\varepsilon_0)$

$$\begin{aligned} p_n(\varepsilon) &\coloneqq \mathbb{P}\left(\sup_{k\geq M_n^{(n)}}\|X_k\|_\infty \leq C_{n+1}(\varepsilon), \ldots, \sup_{k\geq M_1^{(n)}}\|X_k\|_\infty \leq C_2(\varepsilon)\right) \\ &\geq \mathbb{P}\left(\sup_{k\geq M_n^{(n)}}\|X_k\|_\infty \leq C_{n+1}(\varepsilon_0), \ldots, \sup_{k\geq M_1^{(n)}}\|X_k\|_\infty \leq C_2(\varepsilon_0)\right) \\ &\overset{(3.23)}{\geq} p_n \end{aligned}$$

Recall that $\|X_n\|_\infty$ is kept smaller than C_0 by scaling. And since $C_1 > C_0$ and

$$C_n(\varepsilon) \to C_1 \qquad (\varepsilon \to 0),$$

there exists $\varepsilon > 0$ such that

$$C_n(\varepsilon) > C_0 > \|X_n\|_\infty \qquad \forall n \in \mathbb{N}.$$

Therefore

$$p_n(\varepsilon) \to 1 \quad (\varepsilon \to 0)$$

And analogous to (3.24), we can show for every $0 < \varepsilon \leq \varepsilon_0$

$$\begin{aligned} 0 < \prod_{n=1}^{\infty} p_n \leq \prod_{n=1}^{\infty} p_n(\varepsilon) \leq \mathbb{P}\left(\lim_{k\to\infty} \|X_k\|_\infty = 0\right) \\ \implies 1 \overset{\text{A.2.2}}{=} \lim_{\varepsilon\to 0} \prod_{n=1}^{\infty} p_n(\varepsilon) \leq \mathbb{P}\left(\lim_{k\to\infty} \|X_k\|_\infty = 0\right) \end{aligned} \qquad \square$$

Proof theorem 3.3.1 (incomplete). We define $r_n \coloneqq F_n(x) - \mathbb{E}[F_n(x) \mid H_n, \beta_n]$ and two parallel processes

$$\begin{aligned} \delta_{n+1}(x) &\coloneqq (1 - \alpha_n(x))\delta_n(x) + \beta_n(x)\mathbb{E}[F_n(x) \mid H_n, \beta_n] \\ w_{n+1}(x) &\coloneqq (1 - \alpha_n(x))w_n(x) + \beta_n(x)r_n(x). \end{aligned}$$

Which decompose Δ_n for $\delta_1 = \Delta_1$ and $w_1 = 0$, since

$$\underbrace{\delta_{n+1}(x) + w_{n+1}(x)}_{\Delta_{n+1}(x)} = (1 - \alpha_n(x)) \underbrace{(\delta_n(x) + w_n(x))}_{\Delta_n(x)} + \beta_n(x)F_n(x)$$

Due to 3. $\|\mathbb{E}[F_n(\cdot) \mid H_n, \beta_n]\|_\infty \leq \gamma\|\Delta_n\|_\infty$, we can bound δ_n by

$$|\delta_{n+1}(x)| \leq (1 - \alpha_n(x))|\delta_n(x)| + \gamma\beta_n(x)\|\delta_n + w_n\|_\infty$$

which already looks similar to the form of Lemma 3.4.6. The idea is now, to argue that $\|w_n\|_\infty$ is negligible. Notice that

$$\mathrm{Var}[r_n \mid H_n, \beta_n] = \mathrm{Var}[F_n(x) \mid H_n, \beta_n] \leq C(1 + \|\Delta_n\|_\infty)^2$$

is bounded by a constant, if Δ_n is bounded. So we have another "boundedness bootstrapping" problem.

Step 1: If we bound Δ_n by scaling as in Lemma 3.4.4, then w_n converges almost surely to zero due to lemma 3.4.3, since scaling does neither affect the inequality (3.20) nor the expected value of Y_n and it only reduces the variance of Y_n. Therefore

$$\mathbb{P}\left(\sup_{k\geq n} \|w_k\|_\infty > \varepsilon\right) \to 0 \qquad (n \to \infty).$$

The following arguments will only rely on inequalities which are not affected by scaling, so we will ignore it for simplicity. Conditioning on the set

$$M_n = \left\{ \sup_{k \geq n} \|w_k\|_\infty \leq \varepsilon \right\},$$

we can choose $c \in \mathbb{R}$ such that $\tilde{\gamma} \coloneqq \gamma(c+1)/c < 1$ and thus for $\|\delta_k\|_\infty \geq \varepsilon c$

$$\gamma \|\delta_k + w_k\|_\infty \leq \gamma(\|\delta_k\|_\infty + \varepsilon) \leq \gamma \|\delta_k\|_\infty \left(\frac{c}{c} + \frac{\varepsilon c}{\|\delta_k\|_\infty} \frac{1}{c} \right) \leq \tilde{\gamma} \|\delta_k\|_\infty$$

So if $\|\delta_k\|_\infty$ stayed above εc we could bound it By

$$|\tilde{\delta}_{k+1}(x)| = (1 - \alpha_n(x))|\tilde{\delta}_k(x)| + \tilde{\gamma} \|\tilde{\delta}_k\|_\infty \tag{3.25}$$

which converges to zero due to Lemma 3.4.6 which is a contradiction. The missing argument is, why it can not fluctuate above and below this bound. But assuming we can show

$$\mathbb{P}\left(\limsup_{k \to \infty} \|\delta_k\|_\infty \leq \varepsilon c \mid M_n \right) = 1 \qquad \forall n \in \mathbb{N},$$

we immediately get

$$\begin{aligned} \mathbb{P}\left(\limsup_{k \to \infty} \|\delta_k\|_\infty \leq \varepsilon c \right) &\geq \mathbb{P}\left(\limsup_{k \to \infty} \|\delta_k\|_\infty \leq \varepsilon c, M_n \right) \\ &= \underbrace{\mathbb{P}\left(\limsup_{k \to \infty} \|\delta_k\|_\infty \leq \varepsilon c \mid M_n \right)}_{=1} \underbrace{\mathbb{P}(M_n)}_{\to 1 \ (n \to \infty)} \qquad \forall \varepsilon > 0 \end{aligned}$$

which yields

$$\begin{aligned} \mathbb{P}\left(\lim_{k \to \infty} \|\delta_k\|_\infty = 0 \right) &= \mathbb{P}\left(\limsup_{k \to \infty} \|\delta_k\|_\infty \leq 0 \right) \\ &= \lim_{\varepsilon \to 0} \mathbb{P}\left(\limsup_{k \to \infty} \|\delta_k\|_\infty \leq \varepsilon c \right) = 1 \end{aligned}$$

And due to $\Delta_n = \delta_n + w_n$, this implies $\Delta_n \to 0$ almost surely under the assumption that Δ_n stays bounded.

Step 2: What is left to show is, that the unscaled version of Δ_n converges, if the scaled version of Δ_n converges. We define

$$s_n(x) \coloneqq \frac{r_n}{\gamma(1 + \|\Delta_n\|_\infty)},$$

which fulfils

$$\mathbb{E}[s_n \mid H_n, \beta_n] = \frac{\mathbb{E}[r_n \mid H_n, \beta_n]}{\gamma(1 + \|\Delta_n\|_\infty)} = 0$$

and

$$\mathbb{E}[s_n^2(x) \mid H_n, \beta_n] = \frac{\mathbb{E}[r_n^2 \mid H_n, \beta_n]}{(1 + \|\Delta_n\|_\infty)^2} \leq \frac{C(1 + \|\Delta_n\|_\infty)^2}{\gamma^2(1 + \|\Delta_n\|_\infty)^2} = C/\gamma^2.$$

In contrast to r_n it therefore always fulfils the necessary properties for Lemma 3.4.3. So we split up w_n again into

$$\begin{aligned} u_{n+1}(x) &= (1 - \alpha_n(x))u_n(x) + \gamma\beta_n(x)\|\Delta_n\|_\infty s_n(x) \\ v_{n+1}(x) &= (1 - \alpha_n(x))v_n(x) + \gamma\beta_n(x)s_n(x), \end{aligned}$$

using $r_n(x) = \gamma(1 + \|\Delta_n\|)s_n(x)$. v_n converges to zero due to Lemma 3.4.3. What remains is

$$\begin{aligned} |\delta_{n+1}(x)| &\leq (1 - \alpha_n(x))|\delta_n(x)| + \gamma\beta_n(x)\|\delta_n + u_n + v_n\|_\infty \\ u_{n+1}(x) &= (1 - \alpha_n(x))u_n(x) + \gamma\beta_n(x)s_n(x)\|\delta_n + u_n + v_n\|_\infty \end{aligned}$$

Jaakkola et al. now argue, that due to the same argument as in (3.25) with v_k, we can make (δ_n, u_n) into a scale invariant process. Scaling Δ_n can easily be translated into scaling (δ_n, u_n), since

$$\lambda\Delta_n = \lambda(\delta_n + u_n + v_n).$$

And if we assume the convergence of Δ_n to zero is equivalent to the convergence of (δ_n, u_n) to zero, then (δ_n, u_n) converges due to Step 1 when scaled. And due to Lemma 3.4.4 the scaling of (δ_n, u_n) was not necessary in the first place and thus (δ_n, u_n) converges, which implies that Δ_n converges without scaling too.

The issue here is, that we can not make (δ, u_n) into a scale invariant process. At best we can create bounds which are scale invariant. Secondly the idea from (3.25) is also flawed (fluctuation). And third, while the convergence of (δ_n, u_n) to zero implies convergence of Δ_n to zero, the inverse is not necessarily true, but needed to argue with Step 1. This is therefore the largest hole in this proof.

The scale invariant bounds can be created as follows by induction with induction start $\tilde{\delta}_1 = |\delta_1|$ and $\tilde{u}_1 = |u_1|$ and induction step:

$$\begin{aligned} |\delta_{n+1}(x)| &\leq (1 - \alpha_n(x))|\delta_n(x)| + \tilde{\gamma}\beta_n(x)\|\delta_n + u_n\|_\infty \\ &\overset{\text{ind.}}{\leq} (1 - \alpha_n(x))|\tilde{\delta}_n(x)| + \tilde{\gamma}\beta_n(x)(\|\tilde{\delta}_n\|_\infty + \|\tilde{u}_n\|_\infty) =: \tilde{\delta}_{n+1}(x) \\ |u_{n+1}(x)| &\leq (1 - \alpha_n(x))|u_n(x)| + \tilde{\gamma}\beta_n(x)|s_n(x)|\|\delta_n + u_n\|_\infty \\ &\overset{\text{ind.}}{\leq} (1 - \alpha_n(x))|\tilde{u}_n(x)| + \tilde{\gamma}\beta_n(x)|s_n(x)|(\|\tilde{\delta}_n\|_\infty + \|\tilde{u}_n\|_\infty) =: \tilde{u}_{n+1}(x) \end{aligned}$$

Then $(\tilde{\delta}_n, \tilde{u}_n)$ is a scale invariant process. But its convergence is not necessarily equivalent with the convergence of Δ_n, and the scaling is not equivalent either, since scaling of Δ_n changes the denominator of the definition of $s_n(x)$ too. So $s_n(x)$ needs to explicitly use the unscaled version of Δ_n when $(\tilde{\delta}_n, \tilde{u}_n)$ is scaled, in order to make the process scale invariant. But this variant of scaling the process is not equivalent to scaling Δ_n anymore. □

Appendix A

Appendix

A.1 Probability Theory

Lemma A.1.1.

(i) $\mathbb{P}(A \cap B \mid C) = \mathbb{P}(A \mid B \cap C)\mathbb{P}(B \mid C)$

(ii) $\mathbb{P}(A \mid C) = \sum_{n\in\mathbb{N}} \mathbb{P}(A \mid B_n \cap C)\mathbb{P}(B_n \mid C)$ *for* $\mathbb{P}\left(\biguplus_{n\in\mathbb{N}} B_n\right) = 1$

(iii) $\mathbb{E}[X \mid C] = \sum_{n\in\mathbb{N}} \mathbb{E}[X \mid C \cap B_n]\mathbb{P}(B_n \mid C)$ *for* $\mathbb{P}\left(\biguplus_{n\in\mathbb{N}} B_n\right) = 1$

Proof. (i)

$$\mathbb{P}(A \cap B \mid C) = \frac{\mathbb{P}(A \cap B \cap C)}{\mathbb{P}(B \cap C)} \frac{\mathbb{P}(B \cap C)}{\mathbb{P}(C)} = \mathbb{P}(A \mid B \cap C)\mathbb{P}(B \mid C)$$

(ii)

$$\begin{aligned}\mathbb{P}(A \mid C) &= \mathbb{P}\left(A \cap \biguplus_{n\in\mathbb{N}} B_n \,\middle|\, C\right) = \sum_{n\in\mathbb{N}} \mathbb{P}(A \cap B_n \mid C) \\ &\overset{\text{(i)}}{=} \sum_{n\in\mathbb{N}} \mathbb{P}(A \mid B_n \cap C)\mathbb{P}(B_n \mid C)\end{aligned}$$

(iii)

$$\begin{aligned}\mathbb{E}[X \mid C] &= \frac{1}{\mathbb{P}(C)} \int_C X d\mathbb{P} \overset{(*)}{=} \sum_{n\in\mathbb{N}} \frac{1}{\mathbb{P}(C)} \int_{C\cap B_n} X d\mathbb{P} \\ &= \sum_{n\in\mathbb{N}} \frac{\mathbb{P}(C \cap B_n)}{\mathbb{P}(C)} \frac{1}{\mathbb{P}(C \cap B_n)} \int_{C\cap B_n} X d\mathbb{P} \\ &= \sum_{n\in\mathbb{N}} \mathbb{E}[X \mid C \cap B_n]\mathbb{P}(B_n \mid C)\end{aligned}$$

□

Lemma A.1.2. *Let $(\Omega, \mathcal{A}, \mu)$ be a measure space and a function f exists with*

$$f\colon \Omega \to \mathbb{R} \text{ injective and measureable,}$$
$$f^{-1}\colon f(\Omega) \to \Omega \text{ measureable.}$$

Then for X Ω-valued random variable and Y $f(\Omega)$-valued random variable

$$\mathbb{P}_{f\circ X} = \mathbb{P}_Y \iff \mathbb{P}_X = \mathbb{P}_{f^{-1}\circ Y}$$

Proof. "$\Leftarrow$" Let $A \in \mathcal{B}(\mathbb{R})$, then w.l.o.g. $A \subseteq f(\Omega)$ otherwise

$$\mathbb{P}_{f\circ X}(A) = \mathbb{P}(A \cap f(\Omega)) + \underbrace{\mathbb{P}_{f\circ X}(A \cap f(\Omega)^{\complement})}_{=0} = \ldots = \mathbb{P}_Y(A)$$

Thus $f \circ f^{-1}(A) = A$ holds, which finishes this direction with

$$\begin{aligned}\mathbb{P}_{f\circ X}(A) &= \mathbb{P}(X^{-1} \circ f^{-1}(A)) = \mathbb{P}_X(f^{-1}(A)) \\ &= Pr_{f^{-1}\circ Y}(f^{-1}(A)) = \mathbb{P}(Y^{-1} \circ f \circ f^{-1}(A)) \\ &= \mathbb{P}_Y(A)\end{aligned}$$

"$\Rightarrow$" Let $A \in \mathcal{A}$, then

$$\begin{aligned}\mathbb{P}_X(A) &= \mathbb{P}(X^{-1} \circ f^{-1} \circ f(A)) = \mathbb{P}_{f\circ X}(f(A)) \\ &= \mathbb{P}_Y(f(A)) = \mathbb{P}(Y^{-1} \circ f(A)) \\ &= \mathbb{P}_{f^{-1}\circ Y}(A)\end{aligned}$$

□

Definition A.1.3 (Pseudo-inverse)**.** Let F be a cumulative distribution function, then

$$F^{\leftarrow}(y) \coloneqq \inf\{x \in \mathbb{R} : F(x) \geq y\}$$

is called the *Pseudo-inverse* of F.

Lemma A.1.4. *Let F be a cdf, then*

(i) $F^{\leftarrow}(y) \leq x \iff y \leq F(x)$

(ii) $U \sim \mathcal{U}(0,1) \implies F^{\leftarrow}(U) \sim F$

Proof. (i) "$\Rightarrow$"

$$\begin{aligned}y &\leq \inf_{x\in\{z\in\mathbb{R}:F(z)\geq y\}} F(x) \overset{\text{F right-continuous}}{=} F(\inf\{z \in \mathbb{R} : F(z) \geq y\}) \overset{\text{def.}}{=} F(F^{\leftarrow}(y)) \\ &\leq F(x)\end{aligned}$$

Where the last inequality follows from the assumption $F^{\leftarrow}(y) \leq x$ and F being non decreasing.
"$\Leftarrow$" Follows simply from the fact that x is included in the set of the infimum.

$$y \leq F(x) \implies F^{\leftarrow}(y) = \inf\{z \in \mathbb{R} : F(z) \geq y\} \leq x$$

(ii) is a simple corollary from (i)

$$\mathbb{P}(F^{\leftarrow}(U) \leq x) \overset{\text{(i)}}{=} \mathbb{P}(U \leq F(x)) = F(x)$$

□

Lemma A.1.5. *For a real valued random variable X, an arbitrary random vector Y and f a measurable real valued function*

$$\mathrm{Var}[X \mid Y] = \min_f \mathbb{E}[(X - f(Y))^2 \mid Y]$$

Proof.

$$\begin{aligned}\mathbb{E}[(X - f(Y))^2 \mid Y] &= \mathbb{E}[X^2 \mid Y] - \mathbb{E}[2Xf(Y) \mid Y] + \mathbb{E}[f(Y)^2 \mid Y] \\ &= \mathbb{E}[X^2 \mid Y] - 2f(Y)\mathbb{E}[X \mid Y] + f(Y)^2\end{aligned}$$

Fr a fixed ω, $f(Y(\omega)) =: a$ is simply a constant. Therefore we can set the first derivative equal to zero:

$$\frac{d}{da}\mathbb{E}[(X - f(Y))^2 \mid Y](\omega) = -2\mathbb{E}[X \mid Y](\omega) + 2a \stackrel{!}{=} 0$$

Since the second derivative is $2 > 0$, $\mathbb{E}[X \mid Y](\omega) = f(Y(\omega))$ is the unique minimum. Which implies that $\mathrm{Var}[X \mid Y]$ is the minimum regardless of ω. □

Lemma A.1.6.

$$\mathbb{E}[X \mid \mathbb{1}_A] = \mathbb{E}[X \mid A]\mathbb{1}_A + \mathbb{E}[X \mid A^{\complement}]\mathbb{1}_{A^{\complement}}$$

Proof. Let $f(\mathbb{1}_A) = \mathbb{E}[X \mid \mathbb{1}_A]$, then for $B \in \sigma(\mathbb{1}_A) = \{\emptyset, A, A^{\complement}, \Omega\}$ we know

$$\int_B X d\mathbb{P} = \int_B f(\mathbb{1}_A) d\mathbb{P}.$$

In particular this requires

$$\int_A X d\mathbb{P} = \int_A f(\mathbb{1}_A) d\mathbb{P} = f(1)\mathbb{P}(A). \tag{A.1}$$

Thus we know

$$f(1) = \frac{1}{\mathbb{P}(A)} \int_A X d\mathbb{P} = \mathbb{E}[X \mid A].$$

With similar reasoning for $A^{\complement}$, we can prove the original claim:

$$\mathbb{E}[X \mid \mathbb{1}_A] = f(\mathbb{1}_A) = f(1)\mathbb{1}_A + f(0)\mathbb{1}_{A^{\complement}} = \mathbb{E}[X \mid A]\mathbb{1}_A + \mathbb{E}[X \mid A^{\complement}]\mathbb{1}_{A^{\complement}}.$$

□

Lemma A.1.7. *Some History is irrelevant, if the entire history is irrelevant or more generally*

$$\mathbb{E}[X \mid Z, Y_1, Y_2] = \mathbb{E}[X \mid Z] \implies \mathbb{E}[X \mid Z, Y_1] = \mathbb{E}[X \mid Z]$$

for arbitrary random variables X, Z, Y_1, Y_2 with Y_2 being $\mathcal{Y}$ valued, where $\mathcal{Y}$ is countable. This is true in particular for

$$\mathbb{P}(A \mid Z, Y_1, Y_2) = \mathbb{E}[\mathbb{1}_A \mid Z, Y_1, Y_2].$$

Proof. Due to the same argument as in (A.1)

$$\int_{\{Y=y\}} X d\mathbb{P} = \int_{\{Y=y\}} \mathbb{E}[X \mid Y] d\mathbb{P} = \mathbb{E}[X \mid Y = y]\mathbb{P}(Y = y),$$

we can write

$$\mathbb{E}[X \mid Y = y] = \frac{1}{\mathbb{P}(Y = y)} \int_{\{Y=y\}} X d\mathbb{P} = \mathbb{E}[X \mid \{Y = y\}]$$

which allows us to apply A.1.1 (iii):

$$\begin{aligned}
&\mathbb{E}[X \mid Z = z, Y_1 = y_1] \\
&\overset{\text{A.1.1}}{=} \sum_{y_2 \in \mathcal{Y}} \underbrace{\mathbb{E}[X \mid Z = z, Y_1 = y_1, Y_2 = y_2]}_{=\mathbb{E}[X|Z=z]} \mathbb{P}(Y_2 = y_2 \mid Z = z, Y_1 = y_1) \\
&= \mathbb{E}[X \mid Z = z] \underbrace{\sum_{y_2 \in \mathcal{Y}} \mathbb{P}(Y_2 = y_2 \mid Z = z, Y_1 = y_1)}_{=1}
\end{aligned}$$

□

Lemma A.1.8. *Let $((X_t, A_t, R_{t+1}), t \in \mathbb{N}_0)$ be a sequence generated with a stationary behavior π from an episodic MDP. Then for some $x \in \mathcal{X}$*

$$K \coloneqq \min\{k \in \mathbb{N}_0 : X_k = x\}$$

marks the first visit to x, and

$$\sum_{t=K}^{\infty} R_{t+1} \,\Big|\, K \sim \sum_{t=0}^{\infty} R_{t+1} \,\Big|\, X_0 = x.$$

I.e. the distribution of the reward after the first visit to x has the same distribution as the distribution of the reward conditional on the starting state x.

Proof. Because of

$$\{K = k\} = \{X_0 \neq x, \ldots, X_{k-1} \neq x, X_k = x\},$$

we can show for every measurable function f and for all $k \in \mathbb{N}_0$:

$$\begin{aligned}
&\mathbb{E}\left[f\left(\sum_{t=K}^{\infty} R_{t+1} \right) \,\middle|\, K = k \right] \\
&= \mathbb{E}\left[f\left(\sum_{t=k}^{\infty} R_{t+1} \right) \,\middle|\, X_0 \neq x, \ldots, X_{k-1} \neq x, X_k = x \right] \\
&\overset{\text{Markov}}{=} \mathbb{E}\left[f\left(\sum_{t=k}^{\infty} R_{t+1} \right) \,\middle|\, X_k = x \right] \\
&\overset{\text{stationary}}{=} \mathbb{E}\left[f\left(\sum_{t=0}^{\infty} R_{t+1} \right) \,\middle|\, X_0 = x \right]
\end{aligned}$$

Since this is independent of k, we get

$$\mathbb{E}\left[f\left(\sum_{t=K}^{\infty} R_{t+1}\right) \,\middle|\, K\right] = \mathbb{E}\left[f\left(\sum_{t=0}^{\infty} R_{t+1}\right) \,\middle|\, X_0 = x\right]$$

for every measurable f, which implies the distributions are the same. □

A.2 Analysis

Lemma A.2.1. *Let $0 < a_n(\varepsilon) \leq a_n(\varepsilon_0)$ for $n \in \mathbb{N}$ and $0 < \varepsilon \leq \varepsilon_0$, then*

$$\sum_{n=1}^{\infty} a_n(\varepsilon_0) < \infty \quad \textit{and} \quad \forall n \in \mathbb{N} : \lim_{\varepsilon \to 0} a_n(\varepsilon) = 0$$
$$\implies \lim_{\varepsilon \to 0} \sum_{n=1}^{\infty} a_n(\varepsilon) = 0$$

Proof. Select an arbitrary $\delta > 0$, then there is some $N \in \mathbb{N}$ such that

$$\sum_{n=N+1}^{\infty} a_n(\varepsilon_0) < \delta/2$$

and some $\varepsilon_0 > \varepsilon > 0$ such that

$$\sum_{n=1}^{N} a_n(\varepsilon) < \delta/2$$

and thus

$$\sum_{n=1}^{\infty} a_n(\varepsilon) \leq \sum_{n=N+1}^{\infty} a_n(\varepsilon_0) + \sum_{n=1}^{N} a_n(\varepsilon) < \delta$$ □

Lemma A.2.2. *Let $0 < p_n(\varepsilon) \leq p_n(\varepsilon_0)$ for $n \in \mathbb{N}$ and $0 < \varepsilon \leq \varepsilon_0$, then*

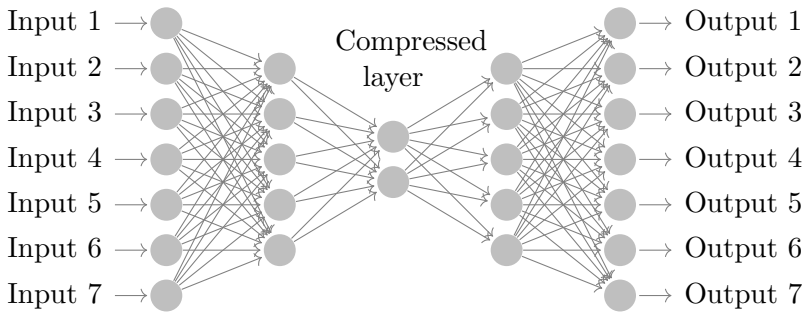
$$\prod_{n=1}^{\infty} p_n(\varepsilon_0) > 0 \quad \textit{and} \quad \forall n \in \mathbb{N} : \lim_{\varepsilon \to 0} p_n(\varepsilon) = 1$$
$$\implies \lim_{\varepsilon \to 0} \prod_{n=1}^{\infty} p_n(\varepsilon) = 1$$

Proof. Since the logarithm is continuous we get

$$\sum_{n=1}^{\infty} \log(p_n(\varepsilon_0)) > -\infty \quad \text{and} \quad \forall n \in \mathbb{N} : \lim_{\varepsilon \to 0} \log(p_n(\varepsilon)) = 0$$
$$\implies \sum_{n=1}^{\infty} -\log(p_n(\varepsilon_0)) < \infty \quad \text{and} \quad \forall n \in \mathbb{N} : \lim_{\varepsilon \to 0} \log(p_n(\varepsilon)) = 0$$
$$\overset{\text{A.2.1}}{\implies} \lim_{\varepsilon \to 0} \sum_{n=1}^{\infty} -\log(p_n(\varepsilon)) = 0$$
$$\implies \lim_{\varepsilon \to 0} \prod_{n=1}^{\infty} p_n(\varepsilon) = \exp\left(\lim_{\varepsilon \to 0} \log\left(\prod_{n=1}^{\infty} p_n(\varepsilon)\right)\right) = 1$$ □

A.3 Autoencoder

While an autoencoder is usually implemented as a neuronal net, the fact that it is a neuronal net is not essential. What is essential, is that it consists of two parameterized functions $g \circ f$, where f is function from $\mathbb{R}^n$ to $\mathbb{R}^k$ with $n > k$, and $g\colon \mathbb{R}^k \to \mathbb{R}^n$. The parameters are then adjusted, such that $g \circ f$ is as close as possible to the identity on R^n. The closeness metric depends on the context. Usually it depends on some average distance metric between input and output of a given dataset (for example a set of pictures). Then the function f can be used to compress high dimensional data to lower dimensional data, and g can be used to decompress data. Here is an example with $n = 7, k = 2$:



Bibliography

Borkar, V. and S. Meyn (Jan. 1, 2000). "The O.D.E. Method for Convergence of Stochastic Approximation and Reinforcement Learning". In: *SIAM Journal on Control and Optimization* 38.2, pp. 447–469. ISSN: 0363-0129. DOI: 10.1137/S0363012997331639.

Dearden, Richard, Nir Friedman, and Stuart Russell (1998). "Bayesian Q-Learning". In: *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, p. 8.

Dvoretzky, Aryeh (Jan. 1, 1956). *On Stochastic Approximation*. Columbia University, New York City, United States. URL: https://apps.dtic.mil/docs/citations/AD1028378 (visited on 03/16/2019).

Jaakkola, Tommi, Michael I. Jordan, and Satinder P. Singh (Nov. 1, 1994). "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms". In: *Neural Computation* 6.6, pp. 1185–1201. ISSN: 0899-7667. DOI: 10.1162/neco.1994.6.6.1185.

Kaelbling, Leslie Pack (1993). *Learning in Embedded Systems*. MIT press.

Kushner, Harold J. (1997). *Stochastic Approximation Algorithms and Applications*. In collab. with George Yin. Applications of Mathematics; 35. Berlin Heidelberg [u.a.]: u.a. Springer. xxi+417. ISBN: 978-0-387-94916-1.

Lange, Sascha, Thomas Gabel, and Martin Riedmiller (2012). "Batch Reinforcement Learning". In: *Reinforcement Learning: State-of-the-Art*. Ed. by Marco Wiering and Martijn van Otterlo. Adaptation, Learning, and Optimization. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 45–73. ISBN: 978-3-642-27645-3. DOI: 10.1007/978-3-642-27645-3_2. URL: https://doi.org/10.1007/978-3-642-27645-3_2 (visited on 02/22/2019).

Lin, Long-Ji (May 1, 1992). "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching". In: *Machine Learning* 8.3, pp. 293–321. ISSN: 1573-0565. DOI: 10.1007/BF00992699.

Pathak, Deepak et al. (July 2017). "Curiosity-Driven Exploration by Self-Supervised Prediction". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Honolulu, HI, USA: IEEE, pp. 488–489. ISBN: 978-1-5386-0733-6. DOI: 10.1109/CVPRW.2017.70.

Peng, Jing and Ronald J. Williams (Jan. 1, 1994). "Incremental Multi-Step Q-Learning". In: *Machine Learning Proceedings 1994*. Ed. by William W. Cohen and Haym Hirsh. San Francisco (CA): Morgan Kaufmann, pp. 226–232. ISBN: 978-1-55860-335-6. DOI: 10.1016/B978-1-55860-335-6.50035-0. URL: http://www.sciencedirect.com/science/article/pii/B9781558603356500350 (visited on 04/28/2019).

Puterman, Martin L. (2005). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. OCLC: 254152847. Hoboken, NJ: Wiley-Interscience. 649 pp. ISBN: 978-0-471-72782-8.

Robbins, Herbert and Sutton Monro (1951). "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407. ISSN: 0003-4851. URL: https://www.jstor.org/stable/2236626 (visited on 02/27/2019).

Singh, Satinder P. and Richard S. Sutton (Jan. 1, 1996). "Reinforcement Learning with Replacing Eligibility Traces". In: *Machine Learning* 22.1, pp. 123–158. ISSN: 1573-0565. DOI: 10.1023/A:1018012322525.

Strehl, Alexander L. and Michael L. Littman (Dec. 1, 2008). "An Analysis of Model-Based Interval Estimation for Markov Decision Processes". In: *Journal of Computer and System Sciences*. Learning Theory 2005 74.8, pp. 1309–1331. ISSN: 0022-0000. DOI: 10.1016/j.jcss.2007.08.009.

Sutton, Richard S. (Aug. 1, 1988). "Learning to Predict by the Methods of Temporal Differences". In: *Machine Learning* 3.1, pp. 9–44. ISSN: 1573-0565. DOI: 10.1007/BF00115009.

Sutton, Richard S. and Andrew G. Barto (1998). *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. Cambridge, Mass. [u.a.]: MIT Press. xviii+322. ISBN: 978-0-262-19398-6.

— (2018). *Reinforcement Learning: An Introduction*. Second edition. Adaptive Computation and Machine Learning Series. Cambridge, Massachusetts: The MIT Press. 526 pp. ISBN: 978-0-262-03924-6.

Szepesvári, Csaba (Jan. 1, 2010). "Algorithms for Reinforcement Learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 4.1, pp. 1–103. ISSN: 1939-4608. DOI: 10.2200/S00268ED1V01Y201005AIM009.

Tsitsiklis, J. N. and B. Van Roy (May 1997). "An Analysis of Temporal-Difference Learning with Function Approximation". In: *IEEE Transactions on Automatic Control* 42.5, pp. 674–690. ISSN: 0018-9286. DOI: 10.1109/9.580874.

Tsitsiklis, John N. (Sept. 1, 1994). "Asynchronous Stochastic Approximation and Q-Learning". In: *Machine Learning* 16.3, pp. 185–202. ISSN: 1573-0565. DOI: 10.1007/BF00993306.

Van Seijen, Harm and Richard S. Sutton (2014). "True Online TD(λ)". In: *Proceedings of the 31st International Conference on Machine Learning - Volume 32* (Beijing, China). ICML'14. JMLR.org, pp. I-692–I-700. URL:

http://dl.acm.org/citation.cfm?id=3044805.3044884 (visited on 03/08/2019).

Watkins, Christopher J. C. H. (1989). "Learning from Delayed Rewards". PhD Thesis. King's College, Cambridge.

Watkins, Christopher J. C. H. and Peter Dayan (May 1, 1992). "Q-Learning". In: *Machine Learning* 8.3-4, pp. 279–292. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/BF00992698.

White, Douglas J. (Dec. 1985). "Real Applications of Markov Decision Processes". In: *Interfaces* 15.6, pp. 73–83. ISSN: 0092-2102, 1526-551X. DOI: 10.1287/inte.15.6.73.

Wiering, Marco and Jürgen Schmidhuber (1998). "Efficient Model-Based Exploration". In: *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior: From Animals to Animats*. Vol. 6, pp. 223–228.