List of Corrections

University of Mannheim

Bachelor Thesis

Reinforcement Learning

by

Felix Benning

born on the 27.11.1996 in Nürtingen
matriculation number 1501817

in the
Fakulty for 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

Chapter 1

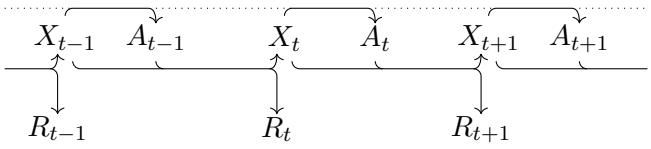
Markov Decision Processes

1.1 Introduction

In this introduction I try to convey an intuition for Markov Decision Processes. Readers familiar with the subject can skip directly to the more 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). That 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 case of 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 different

behaviours. Therefore MDPs are not defined as a stochastic process, but rather as a rulebook on how to create a stochastic process from a given 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. Which means 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.

To illustrate the uses of such a framework, I have selected a few examples from White (1985):

FiXme: more diverse examples

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.

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

$\mathcal{X} := \mathbb{N}$ 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

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

And the stock level on the next day will be

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

We express this with the *state transition function* f defined as

$$\begin{aligned} &f(x, a, d) \coloneqq (x - d \wedge x + a,\ p(d \wedge x) - a - q(x - d \wedge x)) \\ &\implies f(x_t, a_t, d_t) = (x_{t+1}, r_{t+1}) \end{aligned}$$

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 reduces 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

$$\pi(\cdot \mid x) \text{ Probability distribution over } \mathcal{A}$$

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.

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

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 (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. Although 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.

FiXme: Make this a definition of some sort?

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 $\mathbb{P}(X_0 = x) > 0$ holds for all $x \in \mathcal{X}$ ensuring that conditioning on every state is possible.

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) *behavior (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.

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

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

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

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

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

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

3. *Determinisitic 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. 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 $((X_t, A_t, R_{t+1}), t \in \mathbb{N}_0)$

FiXme: remark? lemma? prop?

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

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

(analogous A_t well defined)

Proof. As we want to use cumulative distribution functions (cdf) for this proof, we first embed the $cX \times \mathbb{R}$ 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 injective sigmoid function $\sigma\colon \mathbb{R} \to (0, 1)$, 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 and the corresponding cdf

$$\begin{aligned}\tilde{\mathcal{P}}(\cdot \mid x, a) &:= \mathbb{P}_{f \circ Y} \text{ where } Y \sim \mathcal{P}(\cdot \mid x, a) \\ F_{x,a}(y) &:= \tilde{\mathcal{P}}((-\infty, y] \mid x, a)\end{aligned}$$

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

$$F_{x,a}^{\leftarrow}(U) \sim \tilde{\mathcal{P}}(\cdot \mid x, a) \overset{\text{A.1,2}}{\Longrightarrow} f^{-1} \circ F_{x,a}^{\leftarrow}(U) \sim \mathcal{P}(\cdot \mid x, a)$$

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

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

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

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

$$\begin{aligned}\mathbb{P}(f^{-1} \circ F^{\leftarrow}_{X_t, A_t}(U) \in A \mid \mathcal{H}_0^t) &= \frac{\mathbb{P}(f^{-1} \circ F^{\leftarrow}_{x_t, a_t}(U) \in A, \mathcal{H}_0^t)}{\mathbb{P}(\mathcal{H}_0^t)} \\ &\overset{\text{U indep.}}{=} \mathbb{P}(f^{-1} \circ F^{\leftarrow}_{x_t, a_t}(U) \in A) \\ &= \ldots = \mathbb{P}(f^{-1} \circ F^{\leftarrow}_{X_t, A_t}(U) \in A \mid \mathcal{H}_t^t) \qquad \square\end{aligned}$$

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

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

$(*)$ *Some* of the History is irrelevant if all of the History is irrelevant.

FiXme: some history irrelevant in appendix?

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. $\square$

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 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. 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*

FiXme: does this series exist? argument neccessary? c.f. assumption 1 - enough?

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

Definition 1.2.9. Let $(Y_{(x,a)}, R_{(x,a)}) \sim \mathcal{P}(\cdot \mid x, a)$ be a random variable.

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

With the return we can define value functions.

Definition 1.2.10. Let $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathcal{P})$ be a MDP. Assume an exploring start, i.e. select X_0 such that $\mathbb{P}(X_0 = x) > 0$ for all states x. 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.

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

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

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

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

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

[1] Well defined because $\mathbb{P}(X_0 = x) > 0$. Note that the definition is independent of the distribution of X_0 and V^π can thus be defined as an inherent property of the MDP even 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

FiXme: he quotes Blackwell 1965 - quote blackwell directly?

- 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 a MDP

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

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 we assume an underlying MDP 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 fix-point equation. And that this fix-point equation satisfies the requirements of the Banach Fix-point Theorem (BFT). Which means we will get a free approximation method of the (optimal) value functions along the way.

1.3 Fix-Point Equations for Value Functions

We will start by finding fixpoint 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 will primarily use them as tools to show the fix-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 fixpoint 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 fix point equation.

To find these fixpoint 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. We will use A.1.1

$$\mathbb{E}[X \mid A] = \sum_{n\in\mathbb{N}} \mathbb{E}[X \mid A\cap B_n]\mathbb{P}(B_n \mid A) \text{ for } \mathbb{P}\left(\biguplus_{n\in\mathbb{N}} B_n\right) = 1$$

quite a bit in this proof.

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

This is almost what we want since $r(x,a) = \mathbb{E}[R_{(x,a)}]$, so we just need to look at the conditional expectation

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

$(*)$ We rename

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

then $(\tilde{X}_t, \tilde{A}_t, \tilde{R}_{t+1}, t \in \mathbb{N}_0)$ is an evaluation of the MDP with the (stationary!) policy π. □

FiXme: or realization or "Markov Action Process"? or something else?

Corollary 1.3.3. *For $\pi \in \Pi_S$, this fix 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 $\pi \in \Pi_S^D$ this fix-point equation holds

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

and this fix-point equation

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

Proof. Be $\pi \in \Pi_S$, 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 1.3.2.

Alternatively: the V^π fix 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 fix-point theorem (BFT) for the first time. But to do that we first need to make an assumption.

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

This implies that the immediate reward is bounded

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

But more importantly

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

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 fix-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 $V \in B(\mathcal{X})$, $x \in \mathcal{X}$ 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.

Remark 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. 1.3.3).

T^π meets the requirements of the Banach fixed-point theorem for $\gamma < 1$, this implies that the fixpoints above are *unique* fixpoints and can be approximated with the canonical iteration.

FiXme: proof BFT? - appendix?

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 $V, W \in B(\mathcal{X})$: FiXme: show: is com metric space - apper

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

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. *Observe that*

1. *T^π is an affine operator.*
2. *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, FiXme: comma, fullstop, nothing?

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

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 a 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 uniquness of the fix-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 apporximate 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 show 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 $V^*(x) = \infty$ for $x \in \mathbb{N}$, is achived by the seqences of behaviors

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

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

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

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 A 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 A 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 be

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

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 $T^{\hat{\pi}}$ (1.3.6) inductively with $(T^{\hat{\pi}})^1 V \geq (T^{\hat{\pi}})^0 V$, we get

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

thus

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

1.4.2 Fix-Point Equations for Optimal Value functions

With this statement we can now prove the building blocks for the fix-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:

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

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:

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

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 (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 $\delta > 0$, and

$$\begin{aligned}1 = p(\mathcal{X} \mid x,a) &= p\left(\bigcup_{\delta \to 0} M_\delta \,\middle|\, x,a \right) = \lim_{\delta \to 0} p(M_\delta \mid x,a) \\ &= \lim_{\delta \to 0} \sum_{y \in M_\delta} p(y \mid x,a)\end{aligned}$$

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

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

And also because M_δ is finite and 1.4.3 we know that

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

FiXme: slightly ugly vs. focus on important bits

This finally implies

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

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 1.3.3 we know $V^\pi(x) = Q^\pi(x, \pi(x))$ 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

FiXme: exist in set?
ters up the equation
ng it harder to read,
math. more correct

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

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

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

In either case we get

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

By induction using the monotonicity of T^{π} (1.3.6) we get

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

thus

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

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

$$\begin{aligned} \sup_{a \in \mathcal{A}_x} Q^*(x,a) &= \sup_{a \in \mathcal{A}_x} \sup_{\pi \in \Pi} \mathbb{E}^{\pi}[\mathcal{R} \mid X_0 = x, A_0 = a] \\ &= \sup_{(\pi_t, t \in \mathbb{N}_0)} \sup_{a \in \mathcal{A}_x} \mathbb{E}^{\pi}[\mathcal{R} \mid X_0 = x, A_0 = a] \\ &\overset{A_0 = a}{=} \sup_{(\pi_t, t \in \mathbb{N})} \sup_{a \in \mathcal{A}_x} \mathbb{E}^{\pi}[\mathcal{R} \mid X_0 = x, A_0 = a] \\ &\overset{(*)}{=} \sup_{(\pi_t, t \in \mathbb{N})} \sup_{\pi_0} \mathbb{E}^{\pi}[\mathcal{R} \mid X_0 = x] \\ &= V^*(x) \end{aligned} \tag{1.11}$$

($*$) "$\leq$" is trivial, since a deterministic π_0 can achieve the same as fixing an 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 like this

$$\begin{aligned}Q^\pi(x,a) &= \mathbb{E}^\pi[R_1 \mid X_0 = x, A_0 = a] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t R_{t+1} \,\middle|\, X_0 = x, A_0 = a\right] \\ &= r(x,a) + \gamma \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+2} \,\middle|\, X_0 = x, A_0 = a\right] \qquad (1.12)\end{aligned}$$

This implies

$$\begin{aligned}Q^*(x,a) &= \sup_{\pi \in \Pi} Q^\pi(x,a) \\ &\overset{(1.12)}{=} r(x,a) + \gamma \sup_{(\pi_t, t \in \mathbb{N})} \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t R_{t+2} \,\middle|\, X_0 = x, A_0 = a\right] \\ &\overset{A.1.1}{=} r(x,a) + \gamma \sup_{(\pi_t, t \in \mathbb{N})} \sum_{y \in \mathcal{X}} p(y \mid x, a) \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t R_{t+2} \,\middle|\, X_0 = x, A_0 = a, X_1 = y\right] \\ &= r(x,a) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x, a) \underbrace{\sup_{(\pi_t, t \in \mathbb{N})} \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t R_{t+2} \,\middle|\, X_0 = x, A_0 = a, X_1 = y\right]}_{(*)}\end{aligned}$$

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 (1.11) twice

$$
\begin{aligned}
(*) &\overset{(1.11)}{=} \sup_{(\pi_t, t\geq 2)} \sup_{a_1 \in \mathcal{A}_y} \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+2} \;\middle|\; X_0 = x, A_0 = a, X_1 = y, A_1 = a_1\right] \\
&\overset{\text{Markov}}{=} \sup_{(\pi_t, t\geq 2)} \sup_{a_1 \in \mathcal{A}_y} \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+2} \;\middle|\; X_1 = y, A_1 = a_1\right] \\
&\overset{(1.11)}{=} \sup_{(\pi_t, t\geq 2)} \sup_{\pi_1} \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+2} \;\middle|\; X_1 = y\right] \\
&= \sup_{(\tilde{\pi}_t, t\in\mathbb{N}_0)} \mathbb{E}^{\tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t \tilde{R}_{t+1} \;\middle|\; \tilde{X}_0 = y\right] = V^*(y)
\end{aligned}
$$

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

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

With this Lemma it will be easy to show that the optimal value functions are fix-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 fix-points of T^**

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

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 □

Now we just need to show that the Bellman Optimality Operator satisfies the Banach Fix-point Theorem.

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

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

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

Proof. This proof was taken from 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

$$|\sup_{a \in \mathcal{A}_x} f(a) - \sup_{b \in \mathcal{A}_x} g(b)| \leq \sup_{a \in \mathcal{A}_x} |f(a) - g(a)| \tag{1.13}$$

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 $V, W \in B(\mathcal{X})$

$$\begin{aligned}
\|T^*V - T^*W\|_\infty &= \sup_{x \in \mathcal{X}} |T^*V(x) - T^*W(x)| \\
&\overset{(1.13)}{\leq} \sup_{x \in \mathcal{X}} \sup_{a \in \mathcal{A}_x} \left| \gamma \sum_{y \in \mathcal{X}} p(y \mid x, a)(V(y) - W(y)) \right| \\
&\leq \sup_{x \in \mathcal{X}} \sup_{a \in \mathcal{A}_x} \gamma \underbrace{\sum_{y \in \mathcal{X}} p(y \mid x, a)}_{=1} \|V - W\|_\infty \\
&= \gamma \|V - W\|_\infty
\end{aligned}$$

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

$$\begin{aligned}
\|T^*Q - T^*P\|_\infty &= \sup_{x \in \mathcal{X}} \left| \gamma \sum_{y \in \mathcal{X}} p(y \mid x, a) \left(\sup_{a' \in \mathcal{A}_y} Q(y, a') - \sup_{b' \in \mathcal{A}_y} P(y, b') \right) \right| \\
&\overset{(1.13)}{\leq} \sup_{x \in \mathcal{X}} \gamma \sum_{y \in \mathcal{X}} p(y \mid x, a) \sup_{a' \in \mathcal{A}_y} |Q(y, a') - P(y, a')| \\
&\leq \gamma \|Q - P\|_\infty
\end{aligned}$$

□

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.

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) $\Rightarrow$ (ii)" Let $x \in \mathcal{X}$ be arbitrary, then

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

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

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

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) $\Rightarrow$ (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

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

"(iv) $\Rightarrow$ (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 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 fix-point of T^{π^*} is unique (1.3.5). □

Remark 1.5.2. From (iv) follows: if an optimal stochastic stationary policy exists, then there exists an optimal deterministic stationary policy as well. Since 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 to find an optimal value function. This heuristic only works of course, if the maximum exists. 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 $\pi^x \coloneqq (\pi^x_t, t \in \mathbb{N}_0)$ with

$$\pi^x_t(\cdot \mid y) \coloneqq \mathbb{P}^\pi(A_t \mid X_t = y, X_0 = x) \tag{1.14}$$

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

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

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

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

□

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.

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

□

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. For the optimal policy π exist $\pi^x, x \in \mathcal{X}$ Markovian policies with

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

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

$$\hat{\pi}(\cdot \mid x) \coloneqq \pi_0^x(\cdot \mid x) \qquad x \in \mathcal{X} \tag{1.15}$$

For which we now need to show that it is optimal.

$$
\begin{aligned}
V^*(x) = V^{\pi^x}(x) &= \mathbb{E}^{\pi^x}[R_1 \mid X_0 = x] + \gamma \mathbb{E}^{\pi^x}\left[\sum_{t=0} \gamma^t R_{t+2} \,\middle|\, X_0 = x\right] \\
&= \sum_{a \in \mathcal{A}_x} \mathbb{E}[R_{(x,a)}] \pi_0^x(a \mid x) \\
&\quad + \gamma \sum_{y \in \mathcal{X}} \mathbb{P}^{\pi^x}(X_1 = y \mid X_0 = x) \mathbb{E}^{\pi^x}\left[\sum_{t=0} \gamma^t R_{t+2} \,\middle|\, X_1 = y, X_0 = x\right]
\end{aligned} \tag{1.16}
$$

Where the conditional probability is

$$
\begin{aligned}
&\mathbb{P}^{\pi^x}(X_1 = y \mid X_0 = x) \\
&= \sum_{a \in \mathcal{A}_x} \mathbb{P}^{\pi^x}(X_1 = y, A_0 = a \mid X_0 = x) \\
&= \sum_{a \in \mathcal{A}_x} \mathbb{P}^{\pi^x}(X_1 = y \mid X_0 = x, A_0 = a) \mathbb{P}^{\pi^x}(A_0 = a \mid X_0 = x) \\
&= \sum_{a \in \mathcal{A}_x} p(y \mid x, a) \pi_0^x(a \mid x) = \sum_{a \in \mathcal{A}_x} p(y \mid x, a) \hat{\pi}(a \mid x) \\
&= \mathbb{P}^{\hat{\pi}}(X_1 = y \mid X_0 = x)
\end{aligned} \tag{1.17}
$$

And the conditional expectation is

$$
\begin{aligned}
&\mathbb{E}^{\pi^x}\left[\sum_{t=0} \gamma^t R_{t+2} \,\middle|\, X_1 = y, X_0 = x\right] \\
&= \sum_{a \in \mathcal{A}_y} \mathbb{E}^{\pi^x}\left[\sum_{t=0} \gamma^t R_{t+2} \,\middle|\, X_1 = y, A_1 = a, X_0 = x\right] \pi_1^x(a \mid y) \\
&\overset{\text{markov}}{=} \sum_{a \in \mathcal{A}_y} \mathbb{E}^{\pi^x}\left[\sum_{t=0} \gamma^t R_{t+2} \,\middle|\, X_1 = y, A_1 = a\right] \pi_1^x(a \mid y) \\
&= \mathbb{E}^{\pi^x}\left[\sum_{t=0} \gamma^t R_{t+2} \,\middle|\, X_1 = y\right] = \mathbb{E}^{\tilde{\pi}^x}\left[\sum_{t=0} \gamma^t \tilde{R}_{t+1} \,\middle|\, \tilde{X}_0 = y\right] \\
&= V^{\tilde{\pi}^x}(y)
\end{aligned} \tag{1.18}
$$

with

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

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 $\tilde{\pi}^x$. We can conclude that

FiXme: evaluated MDP?

$$
V^{\tilde{\pi}^x}(z) = V^*(z) \qquad \forall z \in \mathcal{X} : \mathbb{P}^{\pi^x}(X_1 = z \mid X_0 = x) > 0 \tag{1.19}
$$

otherwise we can use the fact that $V^{\pi^z}(z) = V^*(z)$ 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 again).

FiXme: ugly - better solution?

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

If you apply the same steps to $\hat{\pi}^x$ as to π^x in (1.16), everything but the summand z in the large sum will stay the same. And 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 (1.19) 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 fix-point of $T^{\hat{\pi}}$ and the fix-point is unique we have completed our proof, since then $V^{\hat{\pi}} = V^*$ which means $\hat{\pi}$ is optimal. □

In Summary: With an arbitrary optimal policy we can construct an optimal stationary policy with 1.5.7, and if there is an optimal stationary policy, there is also an optimal deterministic stationary policy (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 1.4.3).

FiXme: is this Title fix?

Proposition 1.5.8 (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

$$r(x, \pi^\varepsilon(x)) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x, \pi^\varepsilon(x)) V^*(y) + \delta \geq V^*(x) \tag{1.20}$$

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

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

By induction we get

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

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

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

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 $(Q_k, k \in \mathbb{N}_0)$ is called a *value iteration* if

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

This converges geometrically with regard to the $\|\cdot\|_\infty$-norm due to T^* fulfilling the requirements of the BFT (1.4.7). In general this iteration can of course only work on a finite MDP, since taking the supremum over an infinite amount of actions T^* for an infinite amount of states is impossible without some assumptions.

To find a close to optimal action this iteration can be stopped after a finite amount of steps, since from $\|V_k - V^*\|_\infty \leq \delta/2$ follows

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

So picking a π^ε as in 1.5.8 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 (1.20).

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 a *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

$$\pi_{k+1}(x) = \arg\max_{a \in \mathcal{A}_x} Q^{\pi_k}(x, a) \tag{1.21}$$

which means

$$\begin{aligned} V^{\pi_k} &\leq T^* V^{\pi_k}(x) \qquad (1.22) \\ &= \sup_{a \in \mathcal{A}_x} \left\{ r(x,a) + \gamma \sum_{y \in \mathcal{X}} p(y \mid x, a) V^{\pi_k}(y) \right\} \\ &\overset{1.3.2}{=} \sup_{a \in \mathcal{A}_x} Q^{\pi_k}(x, a) \\ &\overset{(1.21)}{=} Q^{\pi_k}(x, \pi_{k+1}(x)) \\ &= T^{\pi_{k+1}} V^{\pi_k}(x) \end{aligned}$$

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

$$V^{\pi_{k+1}} = \lim_{n\to\infty} (T^{\pi_{k+1}})^n V^{\pi_k} \geq T^{\pi_{k+1}} V^{\pi_k} = T^* V^{\pi_k} \tag{1.23}$$

This implies by induction $V^{\pi_k} \geq (T^*)^k V^{\pi_0}$

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

The statement follows from the fact, that Q^{π_k} can be expressed as a formula of V^{π_k} (c.f. 1.3.3) □

Converges faster than value iteration with every step (c.f. (1.23)) but requires calculation of Q^{π_k} in every step.

But in the finite case policy iteration has some interesting properties

Corollary 1.6.4.

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

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

Proof. (i) (Szepesvári 2010) Consider equation (1.22).

If it is an equality, then the current step is optimal since the value function is a fix 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) Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ 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

$$r^\pi := (r(x_1, \pi(x_1)), \ldots, r(x_n, \pi(x_n)))^T \in \mathbb{R}^n$$

$$P^\pi := \begin{pmatrix} p(x_1 \mid x_1, \pi(x_1)) & \cdots & p(x_n \mid x_1, \pi(x_1)) \\ \vdots & & \vdots \\ p(x_1 \mid x_n, \pi(x_n)) & \cdots & p(x_n \mid x_n, \pi(x_n)) \end{pmatrix}$$

This allows us to write

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

And since the Banach fixpoint 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

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

FiXme: Write down Algorithm?

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) we will examine advantages and disadvantages of various algorithms and discuss possible variations and extensions. In the next chapter we will show almost sure convergence of the basic algorithms introduced in this chapter and illustrate their connection to stochastic approximation.

FiXme: check outline of the plan

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.

But let us ignore the second problem 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

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

and with $Y_t = X_{t+1}$ 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 $\hat{r}$, $\hat{p}$

FiXme: add code/ ref code Dynamic Programming

FiXme: numerical stability analysis?

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^2	γ	G	
	γ^3			
	γ^4			
	γ^5			
	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.

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 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

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

is a bias free estimator for $V^\pi(X_k)$. *First-visit Monte Carlo* uses the return after the first visit of a state $x \in \{X_1, \ldots, X_t\}$ 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

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

$V^\pi(x)$ for every state $x \in \mathcal{X}$ which is visited with positive probability given behaviour π and starting distribution X_0.

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. But every visit Monte Carlo will turn out to be a special case of $TD(\lambda)$.

FiXme: will TD proof work?

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

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

for a moment. Then Monte Carlo converges whatever policy we select. Similar to policy iteration (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 (1.22). 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. We will discuss the most straightforward example here, under the assumption that we can calculate Q^{π_n} somehow.

FiXme: check claim

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 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.

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

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 a ε-soft optimal policy*

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

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

[1] w.l.o.g. only one greedy action

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 that

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

and together with $q(B \mid x, a) \coloneqq \mathcal{P}(\mathcal{X} \times B \mid x, a)$ the complementary marginal distribution 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 together imply

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

Since the fixpoint is unique it follows that

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

Since 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. $Q^{\tilde{\pi}_n}$

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

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 cleared 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.

FiXme: illustrations?

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 $(X_t, A_t, R_{t+1}, t \in \mathbb{N}_0)$ realized with a stationary behaviour π, we know that

FiXme: realized?

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

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)**.**

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

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. We therefore define

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

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

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

FiXme: Do I need to generalize the def?

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 $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ where 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 of the existing batch as efficiently as possible, there were some attempts to

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 $\Psi(Q)(\cdot \mid y)$
        $Q(x,a) \leftarrow Q(x,a) + \alpha_n(x,a)[R + \gamma Q(y,b) - Q(x,a)]$
        $x \leftarrow y$; $a \leftarrow b$;
    until x is terminal
end while

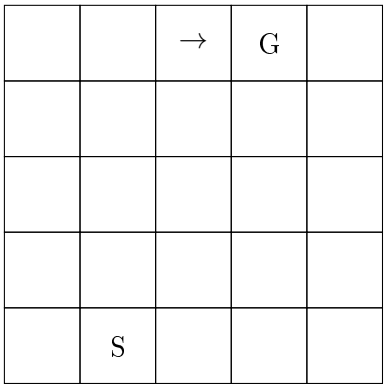
Where α_n is the learning rate.

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 2.4 after explaining the 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 the 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 to find a compromise between Monte Carlo and TD which $TD(\lambda)$ tries to do. The other approach is growing batch learning.

2.4 Growing Batch Learning

FiXme: section instead of subsection? chance place? - after TD(lambda), after Q-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, can be expensive (causing damage to hardware) and/or 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 maybe 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 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 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 that $\hat{p}$ 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" on-line 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 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 $T^\pi V(X_t)$, $R_{t+1} + \gamma V(X_{t+1})$ 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.

FiXme: write down algorithm?

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. By using the learning algorithm of the mixture $TD(\lambda)$ of TD and Monte Carlo 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.

Sutton and Barto (1998) call the 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

To try and mix TD and Monte Carlo, recall that Monte Carlo uses

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

as the estimator 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 (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 in turn can be interpreted as an estimator for $V^{\pi}(X_t)$. 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

FiXme: "corrected n-step truncated return" + explanation?

This leads to 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* error reduction property

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

Proof. We inductively prove that

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

at which point the claim follows from the contraction property of T^π. The induction basis is just the TD case, c.f. (2.1). The induction step follows

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

□

2.5.2 TD(λ) Forward View

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

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

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. So why not continue at every step with equal probability λ? Let N be the associated 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. The expected Return is guaranteed to be well defined by Assumption 1, but by now it is probabily sensible to tighten this assumption to certain boundedness.

FiXme: check claim, should I change assumption 1 in the first place?

The resulting algorithm (2.6) is called TD(λ). The notation (2.7) will be useful later.

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

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

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 α_t. To be more precise Sutton and Barto (1998) call this the *forward view* of TD(λ). This stems from the fact, that we update the value of V in the place x_t with *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 $\Delta V_t^\lambda(X_t)$ using (2.5)

$$\begin{aligned}
\Delta V_t^\lambda(X_t) &= \sum_{n=0}^{\infty}(\gamma\lambda)^n R_{t+1+n} + (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\gamma^n V_t(X_{t+n}) - V_t(X_t) \\
&= \sum_{n=0}^{\infty}(\gamma\lambda)^n R_{t+1+n} + \sum_{n=1}^{\infty}(\lambda\gamma)^{n-1}\gamma V_t(X_{t+n}) - \sum_{n=0}^{\infty}(\lambda\gamma)^n V_t(X_{t+n}) \\
&= \sum_{n=0}^{\infty}(\gamma\lambda)^n \left[R_{t+1+n} + \gamma V_t(X_{t+1+n}) - V_t(X_{t+n})\right] \\
&= \sum_{n=t}^{\infty}(\gamma\lambda)^{n-t} \left[R_{n+1} + \gamma V_t(X_{n+1}) - V_t(X_n)\right] \\
&\approx \sum_{n=t}^{\infty}(\gamma\lambda)^{n-t} \underbrace{\left[R_{n+1} + \gamma V_n(X_{n+1}) - V_n(X_n)\right]}_{=\Delta V_n^0(X_n)}
\end{aligned}$$

Where $\Delta V_n^0(X_n)$ is simply the update direction of TD(0). Now why does this approximation make sense?

First of all, V_{n+1} stays the same to V_n in every place but X_n. Assuming that the sequence of states wanders about a little bit, the first few V_n will all be equal to V_t. And once the sequence of states comes back around, the hope is that $n-t$ would be large enough, for $(\gamma\lambda)^{n-t} \approx 0$. Additionally, the difference between V_n and V_t is small, even for repeated states, if the learning rate α_n is sufficiently small.

Now to the question, why is this approximation useful for us? Well TD(0) is already on-line, as the updates happen with every step. So we could just "broadcast" the update deltas $\Delta_n^0(X_n)$ back to previous states, which can then get updated in the same direction. How much they get updated depends on the number of visits and the time between visits. Imagine the state X_t is never visited again afterwards. Then over time it receives the $\Delta_n^0(X_n)$ following X_t which get added to $V_n(X_t)$ with the factor $(\gamma\lambda)^{n-t}$. Then they eventually sum to something akin to $\Delta V_t^\lambda(X_t)$. The backwards view essentially tries to assign discounted credit/blame to previous states. Reducing λ translates to a faster fading of the *eligibility* for credit/blame. In the same manner the factor

$$e_n(x) = \sum_{k=0}^{n}\alpha_k(X_k)(\gamma\lambda)^{n-k}\mathbb{1}_{X_k=x}$$

is called *eligibility trace* of state x for $\Delta V_n^0(X_n)$.

Using (2.7) we will now show that, assuming the same V_0, the estimate V_T of the the forward view will be the same as the estimate generated by summing

over the TD(0) deltas $\Delta V_n^0(X_n)$ multiplied with the eligibility trace.

$$\begin{aligned}
V_T(x) - V_0(x) &= \sum_{k=0}^{T-1} \alpha_k(X_k) \mathbb{1}_{X_k=x} \Delta V_k^\lambda(X_k) \\
&\approx \sum_{k=0}^{T-1} \alpha_k(X_k) \mathbb{1}_{X_k=x} \sum_{n=k}^{\infty} (\gamma\lambda)^{n-k} \Delta V_n^0(X_n) \\
&= \sum_{k=0}^{T-1} \alpha_k(X_k) \mathbb{1}_{X_k=x} \sum_{n=k}^{T-1} (\gamma\lambda)^{n-k} \Delta V_n^0(X_n) \\
&\overset{\text{Fub.}}{=} \sum_{n=0}^{T-1} \Delta V_n^0(X_n) \sum_{k=0}^{n} \alpha_k(X_k) (\gamma\lambda)^{n-k} \mathbb{1}_{X_k=x} \\
&= \sum_{n=0}^{T-1} \Delta V_n^0(X_n) e_n(x)
\end{aligned}$$

Note that this could 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)

```
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(x)
        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 moved the α_n into the eligibility trace to allow for a changing learning rate over n. 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.

2.5.4 True Online TD(λ)

2.6 Q-learning

Q-learning 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

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

here the update rule is

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

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 (2.3.1), which we can fix with model learning algorithms like Dyna-Q, as hinted at in 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. But since we also need to do some exploration this can not be guaranteed. Watkin's Q(λ) therefore limits itself to n-steps with n smaller than the number of steps until the next exploratory action. Peng's Q(λ) 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.

FiXme: Quote papers, write down pseudo algos?

But at first glance Q-learnings bigger flaw seems 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 actions.

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. Decreasing T, decreases exploration. For

$x^* = \max_{i=1,\dots,n} x_i$ we can see that in limit

$$\begin{aligned}
\exp(x_i/T) &= \exp\left(\frac{x^*}{T} + \frac{x_i - x^*}{T}\right) = \exp(x^*/T)\exp((x_i - x^*)/T) \\
\implies \lim_{T\to 0} \frac{\exp(x_i/T)}{\sum_{i=1}^n \exp(x_i/T)} &= \lim_{T\to 0} \frac{\exp(x_i/T)}{\exp(x^*/T)} \frac{1}{\sum_{i=1}^n \exp((x_i - x^*)/T)} \\
&= \lim_{T\to 0} \frac{\exp((x_i - x^*)/T)}{\sum_{i=1}^n \exp((x_i - x^*)/T)} \\
&= \begin{cases} 0 & x_i \leq x^* \\ \frac{1}{\#\{j|x_j = x^*\}} & x_i = x^* \end{cases}
\end{aligned}$$

While increasing T, increases exploration

$$\lim_{T\to\infty} \frac{\exp(x_i/T)}{\sum_{i=1}^n \exp(x_i/T)} = \frac{1}{n}$$

2.7.3 Directed Exploration

Boltzmann exploration fixes most of the suicidal tendencies that the ε-greedy policy expresses in our cliff edge example (in 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 assuming boundedness of rewards we can use Hoeffding's inequality. To avoid the boundedness assumption we

could use for example, that independently identical distributed (iid) X_i 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. We denote this confidence interval with

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

In the same spirit Strehl and Littman (2008) define

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

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 on how recent the last try was.

FiXme: Dyna-Q+ ?

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 benefitted 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.

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.

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

This is the approach Pathak et al. (2017) recently came up with. Taking inspiration from humans, they argue that curiosity provides intrinsic rewards for actions which do not immediately result in extrinsic reinforcement. Their basic idea is, to train a function approximation algorithm (e.g. an artificial neuronal net) to predict the next state y given state x and action a. This is the world model of the agent. 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 to avoid the agent getting addicted to white noise, they add another module.

2.8 Function Approximation

Chapter 3

Stochastic Approximation – Convergence Proofs

Appendix A

Appendix

A.1 Basic 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)

$$\begin{aligned}\mathbb{E}[X \mid C] &= \frac{1}{\mathbb{P}(C)} \int_C X d\mathbb{P} = \sum_{n\in\mathbb{N}} \frac{1}{\mathbb{P}(C)} \int_{C\cap B_n} X d\mathbb{P} \\ &= \sum_{n\in\mathbb{N}} \frac{\mathbb{P}(C \cap B_n)}{\mathbb{P}(C)} \frac{1}{\mathbb{P}(C \cap B_n)} \int_{C\cap B_n} X d\mathbb{P} \\ &= \sum_{n\in\mathbb{N}} \mathbb{E}[X \mid C \cap B_n]\mathbb{P}(B_n \mid C)\end{aligned}$$

□

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)

$$\mathbb{P}(F^{\leftarrow}(U) \leq x) \overset{\text{(i)}}{=} \mathbb{P}(U \leq F(x)) = F(x)$$

□

A.2 Analysis

Bibliography

Dearden, Richard, Nir Friedman, and Stuart Russell (1998). "Bayesian Q-Learning". In: p. 8.

Kaelbling, Leslie Pack (1993). *Learning in Embedded Systems*. MIT press.

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.

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.

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.

Sutton, Richard S. and Andrew G. Barto (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.

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.