Reinforcement Learning

Prof. Christian Bauckhage

lecture 07
foundations of reinforcement learning

given all our previous lectures, we can finally study reinforcement learning

in this lecture, we therefore have our first look at the modeling tools which are most commonly used to formalize the reinforcement learning problem

in particular, we introduce *Markov decision processes*, *(expected) returns*, *stochastic policies*, *value functions*, and *quality functions*

in our upcoming lectures, we will study all these concepts in great detail

advise:

work through these slides! they lay the foundation for everything to come!

outline

recap

discrete state space

$\Leftrightarrow$ a finite or countably infinite set $\mathcal{S} = \{s_1, s_2, \ldots\}$

stochastic process

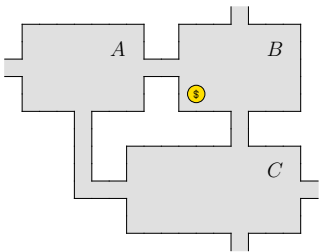
$\Leftrightarrow$ a set of rvs $\{S_t \mid t \in \mathbb{N}\}$ where each $S_t \in \mathcal{S}$

discrete time Markov chain (DTMC)

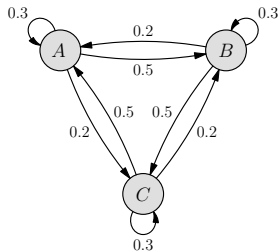
$\Leftrightarrow$ a stochastic process such that for all $t \in \mathbb{N}$

$$p(S_t \mid S_{t-1}, \ldots, S_0) = p(S_t \mid S_{t-1}) = p(S_t = s' \mid S_{t-1} = s) \equiv p(s' \mid s)$$

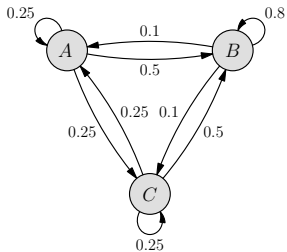
Markov chains provide simple (“first-order”) models of behavior



an environment



λ_1: patrolling the map



λ_2: guarding the treasure

reinforcement learning

reinforcement learning

- $\Leftrightarrow$ the problem faced by an *agent* that uses *informed trial-and-error explorations* to learn how to *behave* in a dynamic environment
- $\Leftrightarrow$ the problem faced by an agent that uses trial-and-error to learn to decide *what* to do *when* in order to achieve a *long-term goal*

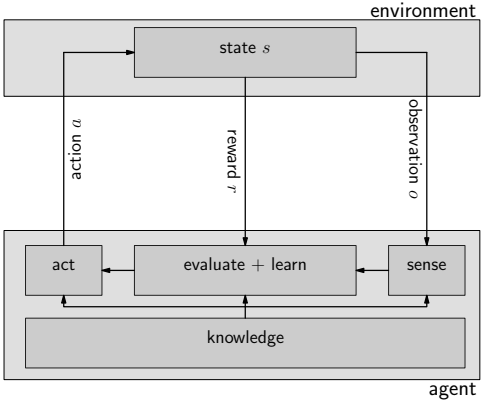
there are two major strategies for computational behavior learning

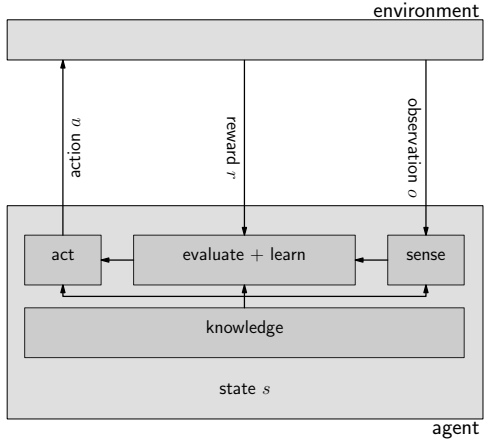
1) *search* the space of all behaviors to find those that work well (e.g. using MCTS, genetic algorithms, . . .)

2) **apply *statistical techniques* and *dynamic programming* to estimate the utilities of certain actions in certain states**

both approaches can be- and often are combined,
but we henceforth solely focus on the second one

further fancy models of intelligent agents





remarks

yes! there is a philosophical ambiguity . . .

- some people think of states as describing an agent's environment
- other people think of states as describing an agent's internal condition

from a practical point of view, this does not matter . . .

- as long as we are aware of what we consider as a state and then stick to our modeling choice (when coding), we can use the same upcoming math to reason about and implement decision making agents

$\Leftrightarrow$ **we can and henceforth will simply talk about "the state an agent is in"**

note

both above diagrams describe agents who interact with an environment by executing

actions a

which may change

states s

and also trigger

rewards r

but what about observations o ?

let's simplify our lives by making several simplifying assumptions . . .

simplifying assumptions (1)

we henceforth assume that

observations o provide complete information about states s

$\Leftrightarrow$ observations o *determine* states $\Leftrightarrow$ states s are *observable*

$\Rightarrow$ **we can ignore observations and simply reason about states**

observability is a term from control theory

states not fully described by observations are called *partially observable* states but, for now, we won't worry about those ...

simplifying assumptions (2)

we henceforth assume that

the sets of states and actions

$$\mathcal{S} = \{s_1, s_2, \ldots\}$$

$$\mathcal{A} = \{a_1, a_2, \ldots\}$$

are both discrete

rewards are real numbers $r \in \mathcal{R} \subset \mathbb{R}$

the sets $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$ are of finite sizes

(much) later, we will relax these assumptions

note

behavior is best seen as a *stochastic* temporal phenomenon

$\Rightarrow$ we should think of states, actions, and rewards as *random variables*

$\Leftrightarrow$ at each time step t, the decision making agent finds itself in a state

$$S_t \in \mathcal{S}$$

selects and performs an action

$$A_t \in \mathcal{A}$$

upon which it receives a reward

$$R_{t+1} \in \mathcal{R}$$

and transits to

$$S_{t+1} \in \mathcal{S}$$

yet, when we run a decision making agent, the rvs S_t, A_t, and R_t will "collapse" to certain particular values

to refer to such *instantiations* of the rvs S_t, A_t, and R_t, we write

$$
\begin{aligned}
s[t] \in \mathcal{S} \quad &\Leftrightarrow \quad \text{particular state an agent is in at time } t \\
a[t] \in \mathcal{A} \quad &\Leftrightarrow \quad \text{particular action an agent takes at time } t \\
r[t+1] \in \mathcal{R} \quad &\Leftrightarrow \quad \text{particular reward an agent receives at time } t+1
\end{aligned}
$$

note

our simplifying assumptions allow us to study the reinforcement learning problem from the perspective of (finite) *Markov decision processes (MDPs)*

MDPs provide a mathematical formalization of sequential decision making

they provide an idealized mathematical model of the reinforcement learning problem which allows for precise theoretical statements

as usual in AI, there is a gap between the model's mathematical tractability and the breadth of its (real world) applicability

the last statement basically says that phenomena of intelligence are too complex for mathematical modeling; w.r.t. reinforcement learning it means that an MDP is but a **useful *approximative* model** of what we really would want to describe . . .

Markov decision process (MDP)

$\Leftrightarrow$ a tuple $(\mathcal{S}, \mathcal{A}, T, R)$ where

$$\mathcal{S} \equiv \text{set of states}$$

$$\mathcal{A} \equiv \text{set of actions}$$

$$\begin{aligned} T(s', s, a) &\equiv \text{state transition probabilities} \\ &= p(S_{t+1} = s' \mid S_t = s, A_t = a) \end{aligned}$$

$$\begin{aligned} R(s, a) &\equiv \text{expected immediate rewards / reinforcements} \\ &= \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] \end{aligned}$$

note

here is how we can make sense of the notion of *expected* immediate rewards

$$R(s,a) = \mathbb{E}\big[R_{t+1} \;\big|\; S_t = s, A_t = a\big] = \sum_{r \in \mathcal{R}} r \cdot p\big(r \;\big|\; s, a\big)$$

but what is r ? and what is $p(r \,|\, s, a)$? neither occurs in the MDP definition !!!

so, let's consider this ...

note

if an agent could compute a function $Util(s)$ to asses a state's *utility*, then we could let

$$r = Util(s')$$

and think of the expected immediate reward w.r.t. s and a as follows

$$R(s,a) = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right] = \sum_{s' \in \mathcal{S}} Util(s') \cdot p(s' \mid s, a)$$

such a function $Util(s)$ is not included in most common MDP definitions

it may seem crazy to say this, but "theoretically" $Util(s)$ is not necessary

we brought it up, as it ties back to our earlier discussions and (hopefully) removes a point of great confusion for RL novices ...

note

let's dwell on the point that $Util(s)$ is "theoretically" unnecessary

one can develop the (MDP-based) theory of RL and most RL algorithms that we know of without having to worry about how to implement $R(s,a)$

indeed, this is what most textbooks, lectures, blogs, vlogs we know of do

however, as soon as it comes to practice and coding, the question of how to implement / represent / realize $R(s,a)$ becomes critical

in fact, so called *reward shaping* is a crucial success factor in applied RL

next, we proceed as in most textbooks but at least we now already know about the need for *reward shaping*

note

if $|\mathcal{S}|$ and $|\mathcal{A}|$ are finite, the transition function T and reward function R can be implemented as (possibly humongous) arrays or *lookup tables*

$\Leftrightarrow$ if $|\mathcal{S}|$ and $|\mathcal{A}|$ are finite, then both functions T and R can be *tabulated*

in our following discussion, we will pretend these tables are available, i.e. that somebody has provided us with all the numbers they contain

for our initial theoretical developments and considerations, this is OK in most practical settings, however, it is not

(much) later on, we will see how to handle such real world settings ...

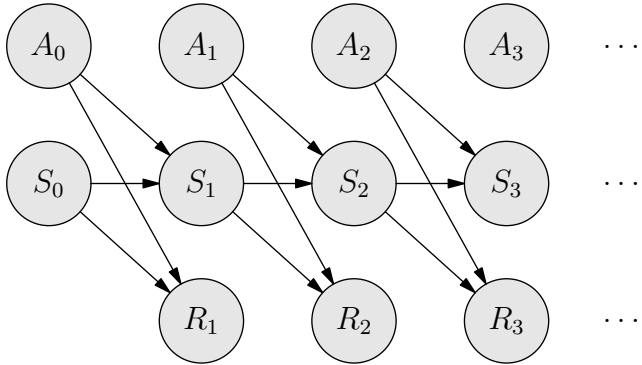
note

alternatively, we could have followed the definition
in *the* RL textbook by **Sutton** and **Barto** and said

an MDP is a stochastic process over rvs S_t, A_t, and R_t where

$$p\big(S_{t+1} = s', R_{t+1} = r \;\big|\; S_t = s, A_t = a\big) = p\big(s', r \;\big|\; s, a\big)$$

Bayesian network representation



this Bayes net definition subsumes the one we presented before, since we have

$$T(s', s, a) = p(S_{t+1} = s' \mid S_t = s, A_t = a) = \sum_{r \in \mathcal{R}} p(s', r \mid s, a)$$

$$\begin{aligned} R(s, a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] &= \sum_{r \in \mathcal{R}} r \cdot \sum_{s' \in \mathcal{S}} p(s', r \mid s, a) \\ &= \sum_{r \in \mathcal{R}} \sum_{s' \in \mathcal{S}} r \cdot p(s', r \mid s, a) \end{aligned}$$

this explicitly resolves confusions as to expected rewards ...

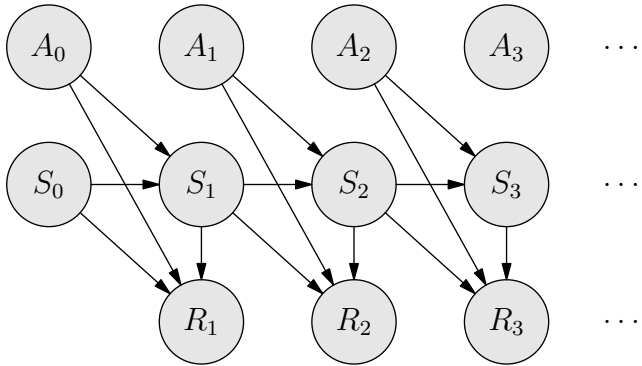
some authors also think of the reward function as a three-argument function

$$R'''\big(s', s, a\big) = \mathbb{E}\big[\, R_{t+1} \;\big|\; S_{t+1} = s', S_t = s, A_t = a \big] = \sum_{r \in \mathcal{R}} r \cdot \frac{p\big(s', r \;\big|\; s, a\big)}{p\big(s' \;\big|\; s, a\big)}$$

this is the most general view on MDPs allowing for the most flexible modeling

however, —to our knowledge— it is rarely necessary in practical applications

Bayesian network representation



some authors also think of the reward function as a univariate function $R'(s)$

this is not unreasonable because we can relate the idea to our own definition

$$
\begin{aligned}
R(s,a) &= \mathbb{E}\big[\, R_{t+1} \;\big|\; S_t = s, A_t = a \big] \\
&= \sum_{r \in \mathcal{R}} r \cdot p\big(r \;\big|\; s,a\big) && \text{(r and $p(r|s,a)$ are again but auxiliary)} \\
&= \sum_{r \in \mathcal{R}} \sum_{s' \in \mathcal{S}} r \cdot p\big(r \;\big|\; s,a,s'\big) \cdot p\big(s' \;\big|\; s,a\big) \\
&= \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} r \cdot p\big(r \;\big|\; s,a,s'\big) \cdot p\big(s' \;\big|\; s,a\big) \\
&\equiv \sum_{s' \in \mathcal{S}} R'\big(s'\big) \cdot p\big(s' \;\big|\; s,a\big)
\end{aligned}
$$

if you have not yet seen marginalization as on the previous slide, then ...

observe that

$$p(X \mid Z) = \frac{p(X,Z)}{p(Z)} = \sum_y \frac{p(X, Y=y, Z)}{p(Z)} = \sum_y \frac{p(X \mid Y=y, Z)\, p(Y=y, Z)}{p(Z)} = \sum_y \frac{p(X \mid Y=y, Z)\, p(Y=y \mid Z)\, p(Z)}{p(Z)}$$

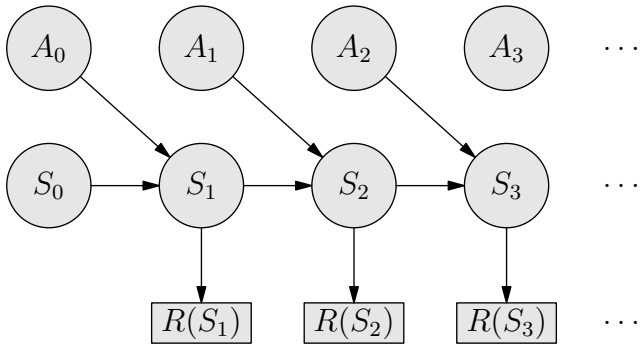
$$= \sum_y p(X \mid Z, Y=y)\, p(Y=y \mid Z)$$

equivalently

$$p(X \mid Z) = \sum_y p(X, Y=y \mid Z) = \sum_y \frac{p(X, Y=y, Z)}{p(Z)} = \sum_y \frac{p(X \mid Y=y, Z)\, p(Y=y, Z)}{p(Z)} = \sum_y \frac{p(X \mid Y=y, Z)\, p(Y=y \mid Z)\, p(Z)}{p(Z)}$$

$$= \sum_y p(X \mid Z, Y=y)\, p(Y=y \mid Z)$$

Bayesian network representation



circles denote rvs, boxes denote deterministic functions of rvs

remarks

"as usual", the Bayesian network view on MDPs is arguably more powerful

however, the majority of the literature (still) follows the state transition view

in fact, here is a quote from the 2nd edition of the **Sutton** and **Barto** book

> *Our characterization of the dynamics of an MDP in terms of $p(s', r \mid s, a)$ is slightly unusual. It is more common in the MDP literature to describe the dynamics in terms of the state transition probabilities $p(s' \mid s, a)$ and expected next rewards $r(s, a)$. In reinforcement learning, however, we more often have to refer to individual actual or sample rewards (rather than just their expected values). Our notation also makes it plainer that S_t and R_t are in general jointly determined, and thus must have the same time index. In teaching reinforcement learning, we have found our notation to be more straightforward conceptually and easier to understand.*

remarks

it is what it is . . .

reinforcement learning is a comparatively new discipline and its practitioners still struggle for a standardized notation and a common point of view

this poses one (of the many) challenges for people who are just beginning to learn about RL . . .

further terminology

if we *sample* an MDP, we obtain an (infinite) trajectory

$$s[0], a[0],\ r[1], s[1], a[1],\ r[2], s[2], a[2],\ r[3], s[3], a[3],\ \ldots$$

a trajectory of finite length is called an episode or a trial

$$s[0], a[0],\ r[1], s[1], a[1],\ \ldots,\ r[h], s[h], a[h]$$

a tuple $\big(s[t], a[t], r[t+1], s[t+1]\big)$ is said to be a transition

finally, a sequence of transitions is also called a roll-out

example episodes (all starting in the same state)

$[3, 1], \rightarrow$

$+10, [4, 1], ?$

$[3, 1], \leftarrow$

$+5, [2, 0], \leftarrow$

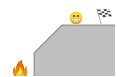
$-10, [1, 0], ?$

$[3, 1], \leftarrow$

$+5, [2, 0], \rightarrow$

$-1, [3, 0], \leftarrow$

$-1, [2, 0], \rightarrow$

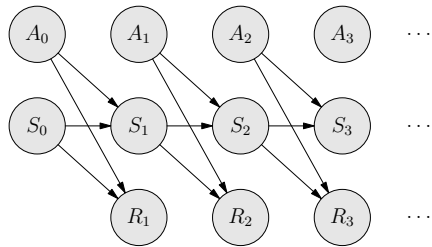


$-1, [3, 0], \rightarrow$

$+10, [4, 0], ?$

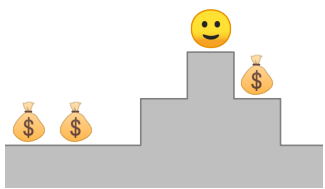
note

R_{t+1} is explicitly computed from S_t and A_t but it implicitly also depends on S_{t-1} and A_{t-1} because these are required to determine S_t ...



$\Rightarrow$ MDPs can be seen as a formalization of sequential decision making in which actions (state transitions) not only impact immediate rewards but also future states and thus future rewards

$\Leftrightarrow$ MDPs give rise to the notion of *delayed rewards* and the need to trade off immediate- and delayed rewards



given this, we can now state the ...

goal of a reinforcement learning agent

$\Leftrightarrow$ maximize *total amount of reward*

$\Leftrightarrow$ maximize not immediate reward but long term *cumulative reward*

note: the use of *reward signals* to formalize learning objectives distinguishes reinforcement learning from other machine learning paradigms

question

how to model cumulative rewards ?

answer

most commonly, it is done like this . . .

for learning episodic tasks which start in a state $S_t = s$ and end after / have a *horizon* of $h < \infty$ steps, we may simply consider the (expected) future return

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_{t+h}$$

learning to play a game poses an episodic task

e.g. no game of *tic tac toe* lasts longer than 9 moves
i.e. every game has final states where episodes end

for learning continuing tasks which last forever ($h \to \infty$), the above will likely not work (as returns might grow beyond all bounds and could thus not be maximized)

the common remedy is to consider the (expected) future discounted return

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{\tau=0}^{\infty} \gamma^\tau R_{t+1+\tau}$$

where the parameter $0 \leqslant \gamma < 1$ is called the discount rate or discount factor

control tasks are examples of continuing tasks

note

there is a hidden recursion because

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma \left[R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right] \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

so that (with $\gamma \neq 0$)

$$G_{t+1} = \frac{G_t - R_{t+1}}{\gamma}$$

in the literature, we often read

> the discount factor $0 \leqslant \gamma < 1$ is used to assign more importance to immediate rewards than to later ones

indeed, a reward received τ steps in the future is only worth a fraction of $\gamma^{\tau-1}$ of what it would be worth in the present

moreover, if $\gamma = 0$ (so that $0^0 = 1$ by definition and $0^\tau = 0$ for all $\tau > 0$), an RL agent will only maximize immediate returns

an agent that learns with $\gamma = 0$ is called nearsighted / myopic
an agent that learns with $\gamma \approx 1$ is called farsighted / hyperopic

but let's be honest . . .

note

discounting is but a “trick” to bound the above infinite sum

because, if $\left|R_{t+1}\right| \leqslant r_{\max} \in \mathbb{R}$ at all times t and if $0 \leqslant \gamma < 1$, then

$$\begin{aligned}\sum_{\tau=0}^{\infty} \gamma^{\tau} R_{t+1+\tau} &\leqslant \sum_{\tau=0}^{\infty} \gamma^{\tau} r_{\max} \\ &= r_{\max} \sum_{\tau=0}^{\infty} \gamma^{\tau} \\ &= r_{\max} \frac{1}{1-\gamma}\end{aligned}$$

recall (our very important) exercise 1.1

the series of powers of the discount factor γ

$$\sum_{\tau=0}^{\infty} \gamma^{\tau}$$

is but the specific geometric series with $a = 1$

$\Rightarrow$ we do indeed have the rather pivotal equality

$$\sum_{\tau=0}^{\infty} \gamma^{\tau} = \frac{1}{1-\gamma}$$

note

observe that (expected) future discounted returns are all about the future and do not care about the past

to simplify notation, we henceforth assume that $t = 0$ denotes the present in which the agent is in a known state $S_0 = s$ so that we may simply write

$$G_0 = \sum_{t=0}^{\infty} \gamma^t R_{t+1} = \sum_{t=0}^{\infty} \gamma^t R\big(S_t, A_t\big)$$

this expression is also called infinite-horizon discounted model with $0 \leqslant \gamma < 1$

(it subsumes finite-horizon models ($h < \infty$) w/o discount if we allow $\gamma = 1$ and $\forall\, t > h : R_{t+1} = 0$)

question

but what about maximizing returns ?

question

but what about maximizing returns ?

answer

we haven't talked about *policies* yet . . .

we therefore recall from lecture 01 . . .

a stochastic policy is a *probabilistic map*

$$\pi : \mathcal{S} \times \mathcal{A} \to \big[0, 1\big]$$

with $0 \leqslant \pi\big(s, a\big) \equiv \pi\big(a \mid s\big) \equiv p\big(a \mid s\big) \leqslant 1$ and $\sum_{a} \pi\big(a \mid s\big) = 1$

note

$\Rightarrow$ a stochastic policy

$$\pi(a \mid s) = p(A_t = a \mid S_t = s)$$

is a distribution over actions given states

MDP policies only regard / involve the current state, not a history of states

throughout, we will assume that policies are *stationary* (time-independent)

$$\forall\, t : A_t \sim \pi$$

question

why policies ?

answer

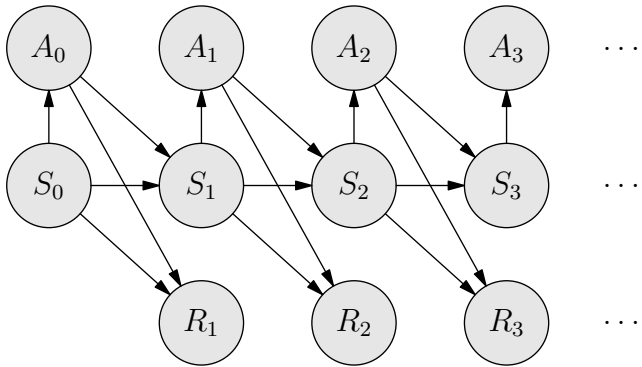
if an agent follows a policy, it has a criterion for deciding "*what* to do *when*"

actions are not selected arbitrarily but in "some" informed manner based on "some" background knowledge / experience

informed decision making provides "some" control over long-term returns

introducing *stochastic policies*, our MDP model becomes . . .

Bayesian network representation



note

the problem at the heart of the MDP framework of reinforcement learning is to *learn* a policy which specifies which action a an agent should choose in state s

but before we can study *policy learning*, we must consider how (stochastic) policies impact our previous considerations . . .

above, we assumed we know $S_t = s$ and $A_t = a$

knowing s and a allows us to retrieve the value

$$R(s, a) = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

from a table

but what if we only know $S_t = s$ and $A_t \sim \pi(a \mid s)$?

in other words, what is "$\mathbb{E}\left[R_{t+1} \mid S_t = s, A_t \sim \pi\right]$" ?

if all we know is $A_t \sim \pi(a \mid s)$, we can average it out to get

$$
\begin{aligned}
\mathbb{E}_{A_t \sim \pi}\Big[\mathbb{E}\big[R_{t+1} \mid S_t = s, A_t\big]\Big] &\equiv \mathbb{E}_{\pi}\Big[\mathbb{E}\big[R_{t+1} \mid S_t = s, A_t\big]\Big] \\
&= \mathbb{E}_{\pi}\Big[\sum_r r \cdot p\big(r \mid s, A_t\big)\Big] \\
&= \sum_a \Big[\sum_r r \cdot p\big(r \mid s, a\big)\Big] \cdot \pi\big(a \mid s\big) \\
&= \sum_a \sum_r r \cdot \frac{p\big(r, s, a\big)}{p\big(s, a\big)} \cdot \frac{p\big(s, a\big)}{p\big(s\big)} \\
&= \sum_a \sum_r r \cdot p\big(r, a \mid s\big) \\
&= \sum_r r \cdot p\big(r \mid s\big) \\
&= \mathbb{E}\big[R_{t+1} \mid S_t = s\big]
\end{aligned}
$$

to retrace what we just did, recall the
law of total expectation (or *tower law*)

$$
\begin{aligned}
\mathbb{E}[X] &= \sum_x x \cdot p(X = x) \\
&= \sum_x x \cdot \sum_y p(X = x, Y = y) \\
&= \sum_x x \cdot \sum_y p(X = x \mid Y = y) \cdot p(Y = y) \\
&= \sum_y p(Y = y) \cdot \sum_x x \cdot p(X = x \mid Y = y) \\
&= \sum_y p(Y = y) \cdot \mathbb{E}[X \mid Y = y] \\
&= \mathbb{E}\Big[\mathbb{E}[X \mid Y]\Big]
\end{aligned}
$$

while we are at it . . .

if X is conditionally independent (*ci*) of Y_t given Y_{t+1}

$$X \perp\!\!\!\perp Y_t \mid Y_{t+1} \Leftrightarrow p\big(X \mid Y_{t+1}, Y_t\big) = p\big(X \mid Y_{t+1}\big)$$

then we also have the (soon to be important) expectation

$$\begin{aligned}
\mathbb{E}\big[X \mid Y_t = y\big] &= \sum_x x \cdot p\big(X = x \mid Y_t = y\big) \\
&= \sum_x x \cdot \sum_{y'} p\big(X = x \mid Y_{t+1} = y', Y_t = y\big) \cdot p\big(Y_{t+1} = y' \mid Y_t = y\big) \\
&\overset{ci}{=} \sum_x x \cdot \sum_{y'} p\big(X = x \mid Y_{t+1} = y'\big) \cdot p\big(Y_{t+1} = y' \mid Y_t = y\big) \\
&= \sum_{y'} \sum_x x \cdot p\big(X = x \mid Y_{t+1} = y'\big) \cdot p\big(Y_{t+1} = y' \mid Y_t = y\big) \\
&= \sum_{y'} \mathbb{E}\big[X \mid Y_{t+1} = y'\big] \cdot p\big(Y_{t+1} = y' \mid Y_t = y\big) \qquad (\star)
\end{aligned}$$

note

to denote an expectation $\mathbb{E}_{A_t \sim \pi}\big[\cdot\big]$ w.r.t. a policy $\pi\big(a \mid s\big)$,
we already wrote and henceforth always will simply write

$$\mathbb{E}_{A_t \sim \pi}\big[\cdot\big] = \mathbb{E}_{\pi}\big[\cdot\big]$$

almost everybody does this

question

if stochastic policies impact immediate rewards, then what about long-term returns ?

answer

we need to extend the notion of infinite horizon discounted model to *value functions*

value function / state-value function

- $\Leftrightarrow$ a function $V_\pi : \mathcal{S} \to \mathbb{R}$ that computes the *value of a state* under a given policy π
- $\Leftrightarrow$ a function $V_\pi : \mathcal{S} \to \mathbb{R}$ that computes the expected return when starting in $s \in \mathcal{S}$ and following policy π thereafter

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \middle| S_0 = s \right] = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \middle| S_0 = s \right]$$

quality function / state-action-value function

$\Leftrightarrow$ a function $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that computes the expected return starting from s, taking action a, and then following policy π

$$Q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \middle| S_0 = s, A_0 = a \right]$$

note

V_π and Q_π are interrelated

$$V_\pi(s) = \sum_a Q_\pi(s,a)\,\pi(a \mid s)$$

moreover ...

previously, we wrote

$$G_0 = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$$

and have seen that

$$G_0 = R_1 + \gamma G_1$$

$\Rightarrow$ we have

$$
\begin{aligned}
V_\pi(s) &= \mathbb{E}_\pi\big[G_0 \;\big|\; S_0 = s\big] \\
&= \mathbb{E}_\pi\big[R_1 + \gamma\, G_1 \;\big|\; S_0 = s\big] \\
&= \mathbb{E}_\pi\big[R_1 \;\big|\; S_0 = s\big] + \mathbb{E}_\pi\big[\gamma\, G_1 \;\big|\; S_0 = s\big] \\
&= \mathbb{E}_\pi\big[R_1 \;\big|\; S_0 = s\big] + \gamma\, \mathbb{E}_\pi\big[G_1 \;\big|\; S_0 = s\big]
\end{aligned}
$$

the second term resembles a value function but is conditioned on S_0 instead on S_1

however ...

using ($\star$) on slide 55, we find

$$\begin{aligned}
\mathbb{E}_\pi\big[G_1 \;\big|\; S_0 = s\big] &= \sum_{s'} \mathbb{E}_\pi\big[G_1 \;\big|\; S_1 = s'\big] \cdot p\big(S_1 = s' \;\big|\; S_0 = s\big) \\
&= \sum_a \sum_{s'} \mathbb{E}_\pi\big[G_1 \;\big|\; S_1 = s'\big] \cdot p\big(s' \;\big|\; s, a\big) \cdot \pi\big(a \;\big|\; s\big) \\
&= \sum_a \sum_{s'} V_\pi\big(s'\big) \cdot p\big(s' \;\big|\; s, a\big) \cdot \pi\big(a \;\big|\; s\big)
\end{aligned}$$

most bloggers / vloggers don't understand this step and get it wrong

$\Rightarrow$ since

$$\mathbb{E}_\pi\big[R_1 \;\big|\; S_0 = s\big] = \sum_a R(s,a)\,\pi(a \mid s)$$

and

$$\mathbb{E}_\pi\big[G_1 \;\big|\; S_0 = s\big] = \sum_a \sum_{s'} V_\pi(s')\,p(s' \mid s,a)\,\pi(a \mid s)$$

we find

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi\big[G_0 \;\big|\; S_0 = s\big] \\ &= \mathbb{E}_\pi\big[R_1 \;\big|\; S_0 = s\big] + \gamma\,\mathbb{E}_\pi\big[G_1 \;\big|\; S_0 = s\big] \\ &= \sum_a \Big[R(s,a) + \gamma \sum_{s'} V_\pi(s')\,p(s' \mid s,a)\Big] \cdot \pi(a \mid s) \end{aligned}$$

Bellman equation for $V_\pi(s)$

$$V_\pi(s) = \sum_a \Big[R(s,a) + \gamma \sum_{s'} V_\pi(s')\, p(s' \mid s,a) \Big] \cdot \pi(a \mid s)$$

similarly, we have

$$Q_\pi(s,a) = \mathbb{E}_\pi\left[G_0 \mid S_0 = s, A_0 = a\right] = \mathbb{E}_\pi\left[R_1 + \gamma G_1 \mid S_0 = s, A_0 = a\right] = \underbrace{\mathbb{E}_\pi\left[R_1 \mid S_0 = s, A_0 = a\right]}_{=R(s,a)} + \gamma\, \mathbb{E}_\pi\left[G_1 \mid S_0 = s, A_0 = a\right]$$

and we note that

$$\begin{aligned}
\mathbb{E}_\pi\left[G_1 \mid s, a\right] = \sum_{s'} \mathbb{E}_\pi\left[G_1 \mid s', s, a\right] p(s' \mid s, a) &= \sum_{s'} \sum_{a'} \mathbb{E}_\pi\left[G_1 \mid s', s, a, a'\right] p(s' \mid s, a)\, p(a' \mid s', s, a) \\
&\overset{ci}{=} \sum_{s'} \sum_{a'} \mathbb{E}_\pi\left[G_1 \mid s', a'\right] p(s' \mid s, a)\, p(a' \mid s') \\
&= \sum_{s'} \sum_{a'} Q_\pi(s', a')\, p(s' \mid s, a)\, p(a' \mid s')
\end{aligned}$$

we therefore have

$$Q_\pi(s,a) = R(s,a) + \gamma \sum_{s'} \sum_{a'} Q_\pi(s', a')\, p(s' \mid s, a)\, p(a' \mid s')$$

Bellman equation for $Q_\pi(s,a)$

$$Q_\pi(s,a) = R(s,a) + \gamma \sum_{s'} \sum_{a'} Q_\pi(s',a')\, \pi(a' \mid s')\, p(s' \mid s,a)$$

note: this differs from what you will find on most websites
however, it agrees with *our* modeling assumptions !

note

the *real* problem at the heart of MDPs is to determine an optimal policy

$$\pi_* : \mathcal{S} \times \mathcal{A} \to \left[0, 1\right]$$

such that action selection according to $\pi_*\left(a \mid s\right)$ entails optimal behavior

this statement is not yet precise,
so that we will get back to it later
for now, we point out ...

a policy π is better than a policy π', written as $\pi \geqslant \pi'$, if its expected return is always *greater than or equal to* that of π'

$$\pi \geqslant \pi' \quad \Leftrightarrow \quad \forall\, s : V_{\pi}(s) \geqslant V_{\pi'}(s)$$

note

this induces a partial ordering over the set of all policies (if one policy is better than another at one state but worse elsewhere, the two policies are incomparable)

note / recall that every partially ordered set has *at least* one maximal element

$\Rightarrow$ there must exist at least one policy better than all others, $\exists\, \pi_* : \forall\, s : V_{\pi_*}(s) \geqslant V_\pi(s)$

$\Rightarrow$ **for any Markov decision process, there exists at least one optimal policy** π_*

in case there are several optimal policies, it is traditional to simply call them all π_*

optimal value function

$\Leftrightarrow$ the maximum value function over all policies

$$V_*(s) = \max_\pi V_\pi(s)$$

optimal quality function

$\Leftrightarrow$ the maximum quality function over all policies

$$Q_*(s,a) = \max_\pi Q_\pi(s,a)$$

optimal policy

given an optimal quality function, we can compute

$$\pi_*(a \mid s) = \begin{cases} 1 & \text{if } a = \operatorname*{argmax}_{a'} Q_*(s, a') \\ 0 & \text{otherwise} \end{cases}$$

Theorem (w/o proof)

for any MDP

there exists an optimal policy better than all other policies

$$\exists\, \pi_* : \ \forall\, \pi : \ \pi_* \geqslant \pi$$

all optimal policies achieve the optimal value function

$$V_{\pi_*} = V_*$$

all optimal policies achieve the optimal quality function

$$Q_{\pi_*} = Q_*$$

Theorem

an MDP can have several optimal policies π_*, π'_*, ... but only one optimal value function V_*

Proof (by contradiction).

suppose there are two optimal policies π_* and π'_* which have two *distinct* optimal value functions V_* and V'_*

if V_* and V'_* are distinct, there must exist a state s with $V_*(s) \neq V'_*(s)$ and we may assume w.l.o.g. that $V_*(s) < V'_*(s)$

however, if $V_*(s)$ is less than $V'_*(s)$, then π_* can't be optimal as it would be better to follow π'_*

$\Rightarrow$ if π_* and π'_* are both optimal, V_* and V'_* must be the same

□

note: we will see examples in the exercises !!!

note

there are two fundamental questions when working with MDPs

1) **how to *find* an *optimal policy* given a model $(\mathcal{S}, \mathcal{A}, T, R)$?**
2) **how to *learn* an *optimal policy* without having a model ?**

in the next couple of lectures, we address both questions

summary

we now know about

the reinforcement learning problem

Markov decision processes (MDPs)

value functions and policies

Bellman equations