MDPs: modeling

Dice game

Example: dice game

For each round $r = 1, 2, \ldots$

- You choose stay or quit.
- If quit, you get $\$10$ and we end the game.
- If stay, you get $\$4$ and then I roll a 6-sided dice.
  - If the dice results in 1 or 2, we end the game.
  - Otherwise, continue to the next round.

Start Stay Quit

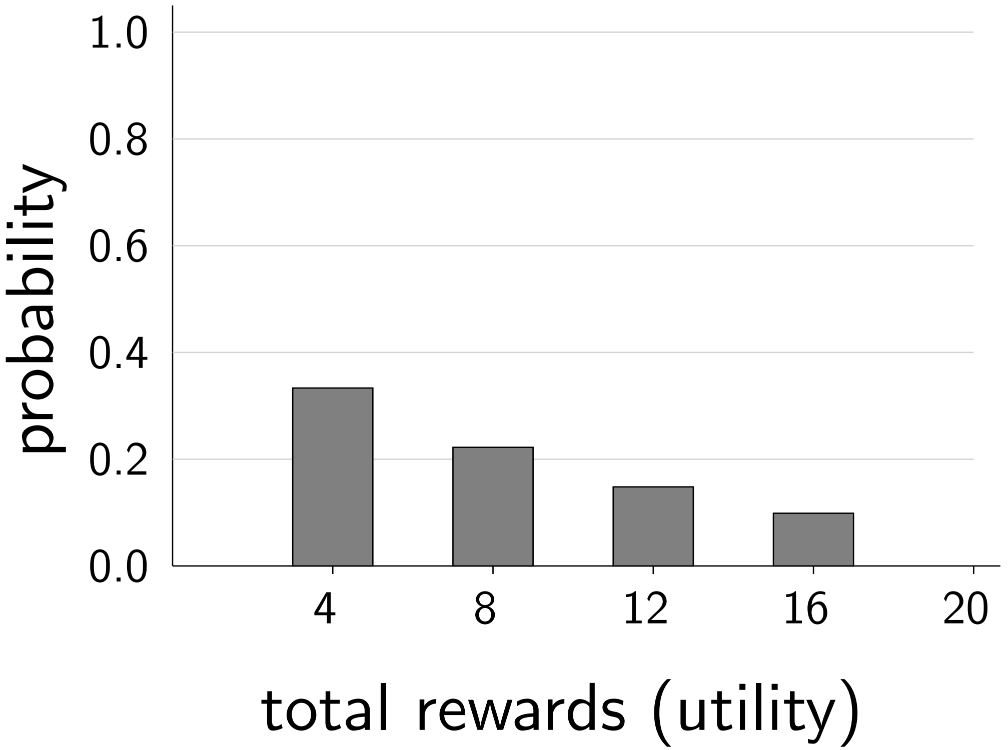
Dice:

Rewards: 0

- We'll see more volcanoes later, but let's start with a much simpler example: a dice game. What is the best strategy for this game?

Rewards

If follow policy "stay":



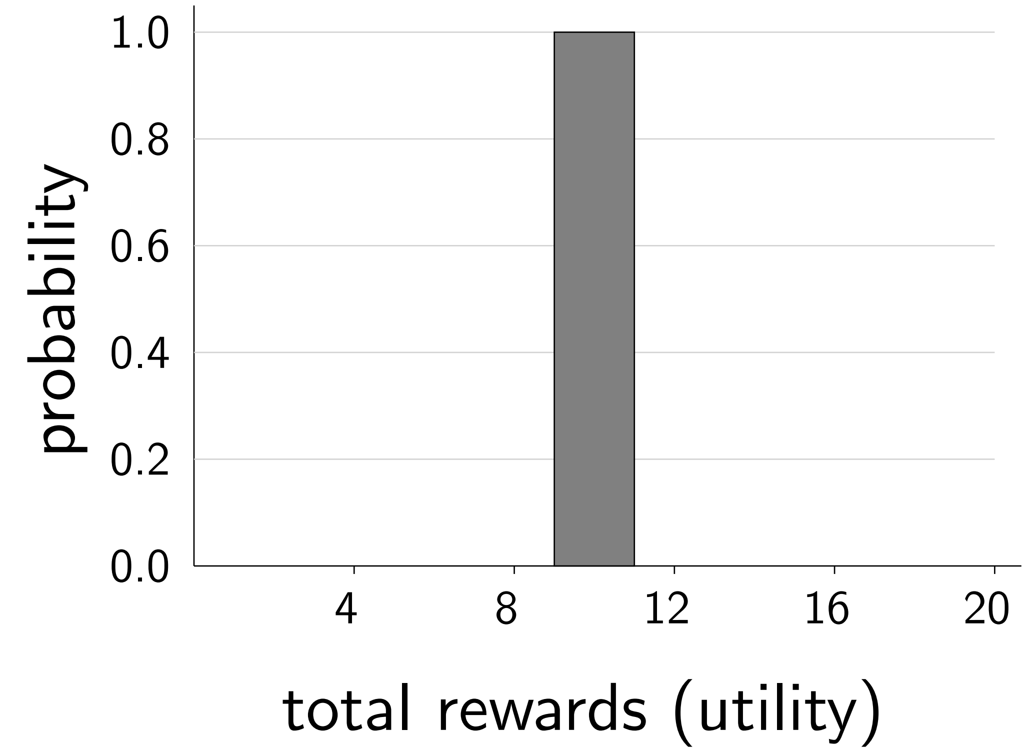
Expected utility:

$$\frac{1}{3}(4) + \frac{2}{3} \cdot \frac{1}{3}(8) + \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{3}(12) + \cdots = 12$$

- Let's suppose you always stay. Note that each outcome of the game will result in a different sequence of rewards, resulting in a **utility**, which is in this case just the sum of the rewards.
- We are interested in the **expected** utility, which you can compute to be 12.

Rewards

If follow policy "quit":



Expected utility:

$$1(10) = 10$$

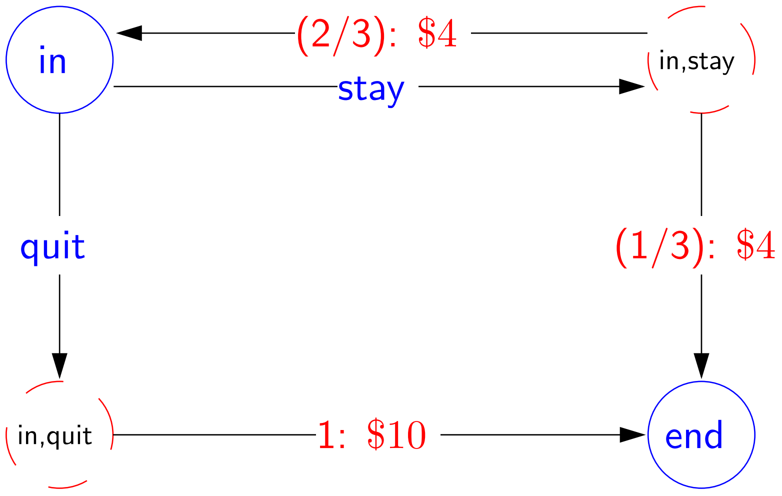
- If you quit, then you'll get a reward of 10 deterministically. Therefore, in expectation, the "stay" strategy is preferred, even though sometimes you'll get less than 10.

MDP for dice game

Example: dice game

For each round $r = 1, 2, \ldots$

- You choose stay or quit.
- If quit, you get $\$10$ and we end the game.
- If stay, you get $\$4$ and then I roll a 6-sided dice.
  - If the dice results in 1 or 2, we end the game.
  - Otherwise, continue to the next round.



- While we already solved this game directly, we'd like to develop a more general framework for thinking about not just this game, but also other problems such as the volcano crossing example. To that end, let us formalize the dice game as a **Markov decision process** (MDP).
- An MDP can be represented as a graph. The nodes in this graph include both **states** and **chance nodes**. Edges coming out of states are the possible actions from that state, which lead to chance nodes. Edges coming out of a chance nodes are the possible random outcomes of that action, which end up back in states. Our convention is to label these chance-to-state edges with the probability of a particular **transition** and the associated reward for traversing that edge.

Markov decision process

Definition: Markov decision process

States: the set of states

$s_{\text{start}} \in$ States: starting state

Actions(s): possible actions from state s

$T(s, a, s')$: probability of s' if take action a in state s

Reward(s, a, s'): reward for the transition (s, a, s')

IsEnd(s): whether at end of game

$0 \leq \gamma \leq 1$: discount factor (default: 1)

- A **Markov decision process** has a set of states States, a starting state s_{start}, and the set of actions Actions(s) from each state s.
- It also has a **transition distribution** T, which specifies for each state s and action a, a distribution over possible successor states s'. Specifically, we have that $\sum_{s'} T(s, a, s') = 1$ because T is a probability distribution (more on this later).
- Associated with each transition (s, a, s') is a reward, which could be either positive or negative.
- If we arrive in a state s for which IsEnd(s) is true, then the game is over.
- Finally, the discount factor γ is a quantity which specifies how much we value the future and will be discussed later.

Search problems

Definition: search problem

States: the set of states

$s_{\text{start}} \in$ States: starting state

Actions(s): possible actions from state s

Succ(s, a): where we end up if take action a in state s

Cost(s, a): cost for taking action a in state s

IsEnd(s): whether at end

- Succ$(s, a) \Rightarrow T(s, a, s')$
- Cost$(s, a) \Rightarrow$ Reward(s, a, s')

- MDPs share many similarities with search problems, but there are differences (one main difference and one minor one).
- The main difference is the move from a deterministic successor function $\text{Succ}(s, a)$ to transition probabilities over s'. We can think of the successor function $\text{Succ}(s, a)$ as a special case of transition probabilities: $T(s, a, s') = \begin{cases} 1 & \text{if } s' = \text{Succ}(s, a) \\ 0 & \text{otherwise} \end{cases}$.
- A minor difference is that we've gone from minimizing costs to maximizing rewards. The two are really equivalent: you can negate one to get the other.

Transitions

Definition: transition probabilities

The **transition probabilities** $T(s, a, s')$ specify the probability of ending up in state s' if taken action a in state s.

Example: transition probabilities

s	a	s'	$T(s, a, s')$
in	quit	end	1
in	stay	in	$2/3$
in	stay	end	$1/3$

- Just to dwell on the major difference, transition probabilities, a bit more: for each state s and action a, the transition probabilities specifies a distribution over successor states s'.

Probabilities sum to one

Example: transition probabilities

s	a	s'	$T(s,a,s')$
in	quit	end	1
in	stay	in	$2/3$
in	stay	end	$1/3$

For each state s and action a:

$$\sum_{s' \in \text{States}} T(s,a,s') = 1$$

Successors: s' such that $T(s,a,s') > 0$

- This means that for each given s and a, if we sum the transition probability $T(s, a, s')$ over all possible successor states s', we get 1.
- If a transition to a particular s' is not possible, then $T(s, a, s') = 0$. We refer to the s' for which $T(s, a, s') > 0$ as the successors.
- Generally, the number of successors of a given (s, a) is much smaller than the total number of states. For instance, in a search problem, each (s, a) has exactly one successor.

Transportation example

Example: transportation

Street with blocks numbered 1 to n.

Walking from s to $s+1$ takes 1 minute.

Taking a magic tram from s to $2s$ takes 2 minutes.

How to travel from 1 to n in the least time?

Tram fails with probability 0.5.

[semi-live solution]

- Let us revisit the transportation example. As we all know, magic trams aren't the most reliable forms of transportation, so let us asume that with probability $\frac{1}{2}$, it actually does as advertised, and with probability $\frac{1}{2}$ it just leaves you in the same state.

What is a solution?

Search problem: path (sequence of actions)

MDP:

Definition: policy

A **policy** π is a mapping from each state $s \in \text{States}$ to an action $a \in \text{Actions}(s)$.

Example: volcano crossing

s	$\pi(s)$
(1,1)	S
(2,1)	E
(3,1)	N
...	...

- So we now know what an MDP is. What do we do with one? For search problems, we were trying to find the minimum cost **path**.
- However, fixed paths won't suffice for MDPs, because we don't know which states the random dice rolls are going to take us.
- Therefore, we define a **policy**, which specifies an action for every single state, not just the states along a path. This way, we have all our bases covered, and know what action to take no matter where we are.
- One might wonder if we ever need to take different actions from a given state. The answer is no, since like as in a search problem, the state contains all the information that we need to act optimally for the future. In more formal speak, the transitions and rewards satisfy the **Markov property**. Every time we end up in a state, we are faced with the exact same problem and therefore should take the same optimal action.