Chapter 17

Markov Decision Process

Question

- How would you get groceries in the least amount of time?

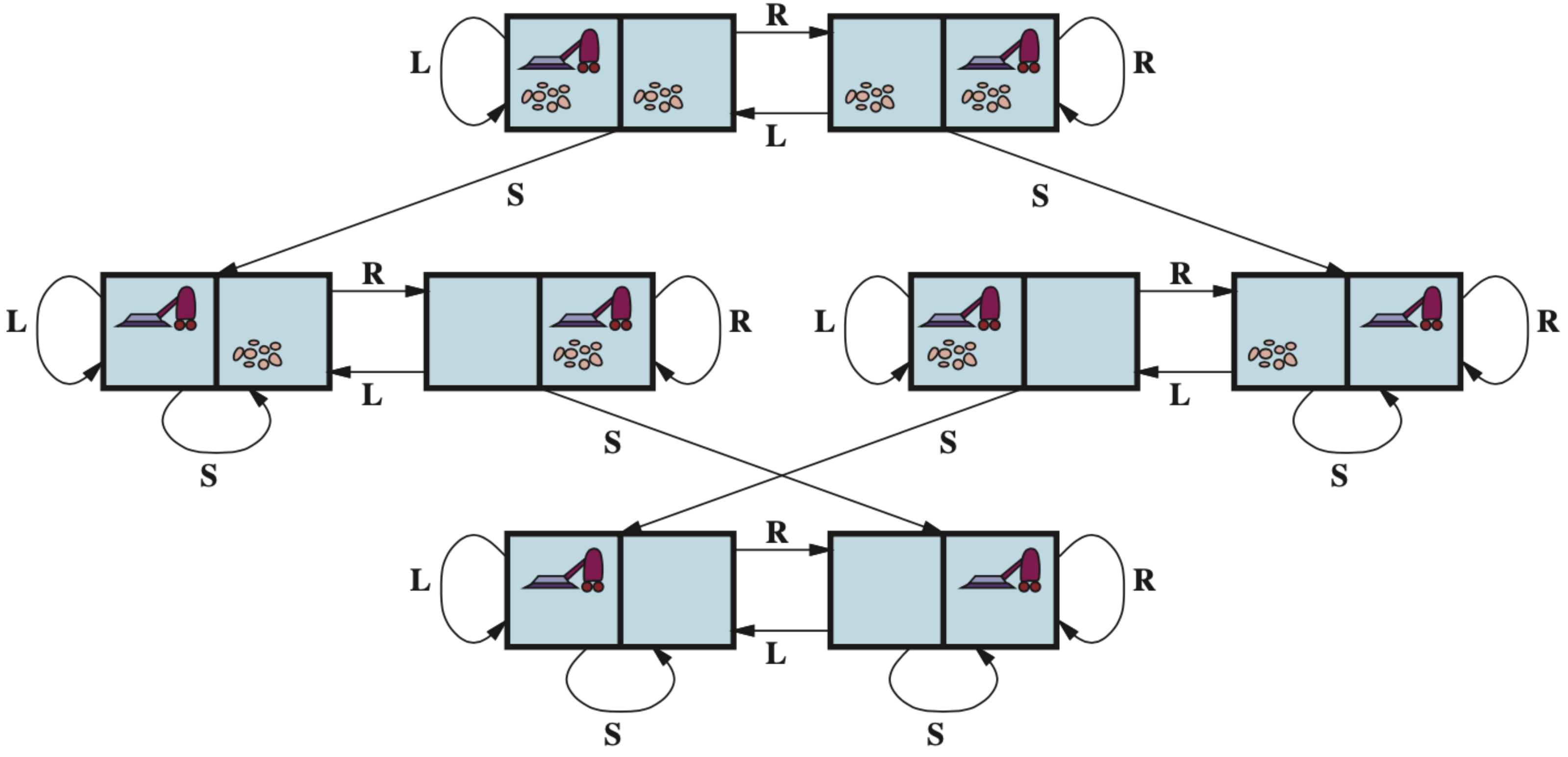
1. order grocery delivery
2. walk to the store
3. bike to the store
4. drive to the store
5. fly to the store

So far: search problems

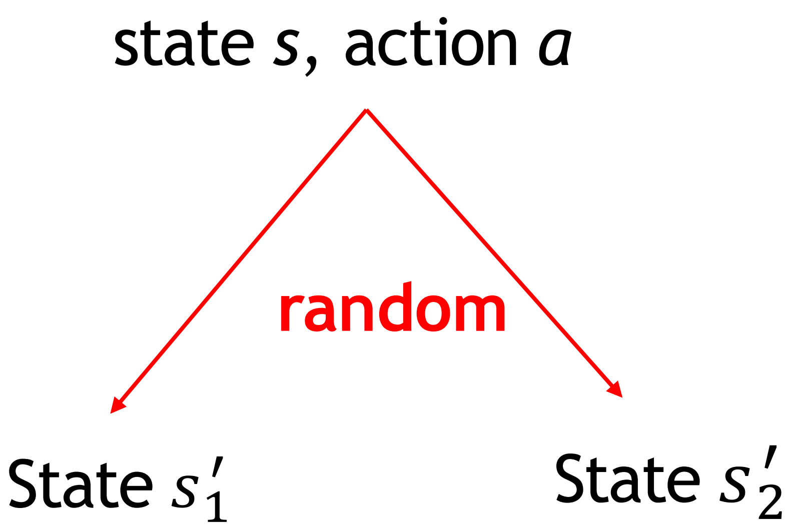
state *s*,
action *a*

deterministic

state Succ(*s*, *a*)



Uncertainty in the real world



Taking an action might lead to any one of many possible states!

History

- MDPs: Mathematical Model for decision making under uncertainty, first introduced in 1950s-60s.

- The term Markov refers to Andrey Markov as MDPs are extensions of Markov Chains, and they allow making decisions (taking actions or having choice).

Applications

- Robotics: decide where to move, but actuators can fail, hit unseen obstacles, etc.

- Resource allocation: decide what to produce, don't know the customer demand for various products

- Agriculture: decide what to plant, but don't know weather and thus crop yield

Markov Decision Process

An MDP can be represented as a graph:

- The nodes represent states $\mathcal{S}$
- A finite set of actions $\mathcal{A}$ to take when in a state: the edges represent possible actions to take when in that state
- The state transition matrix $\mathcal{P}(s, a, s')$: defines transition probabilities from all states s to all successor states s'
- The reward function $\mathcal{R}(s, a, s')$ gives the rewards for moving from one state to the next
- A discount factor γ in the range $0 \leq \gamma \leq 1$

Markov Property

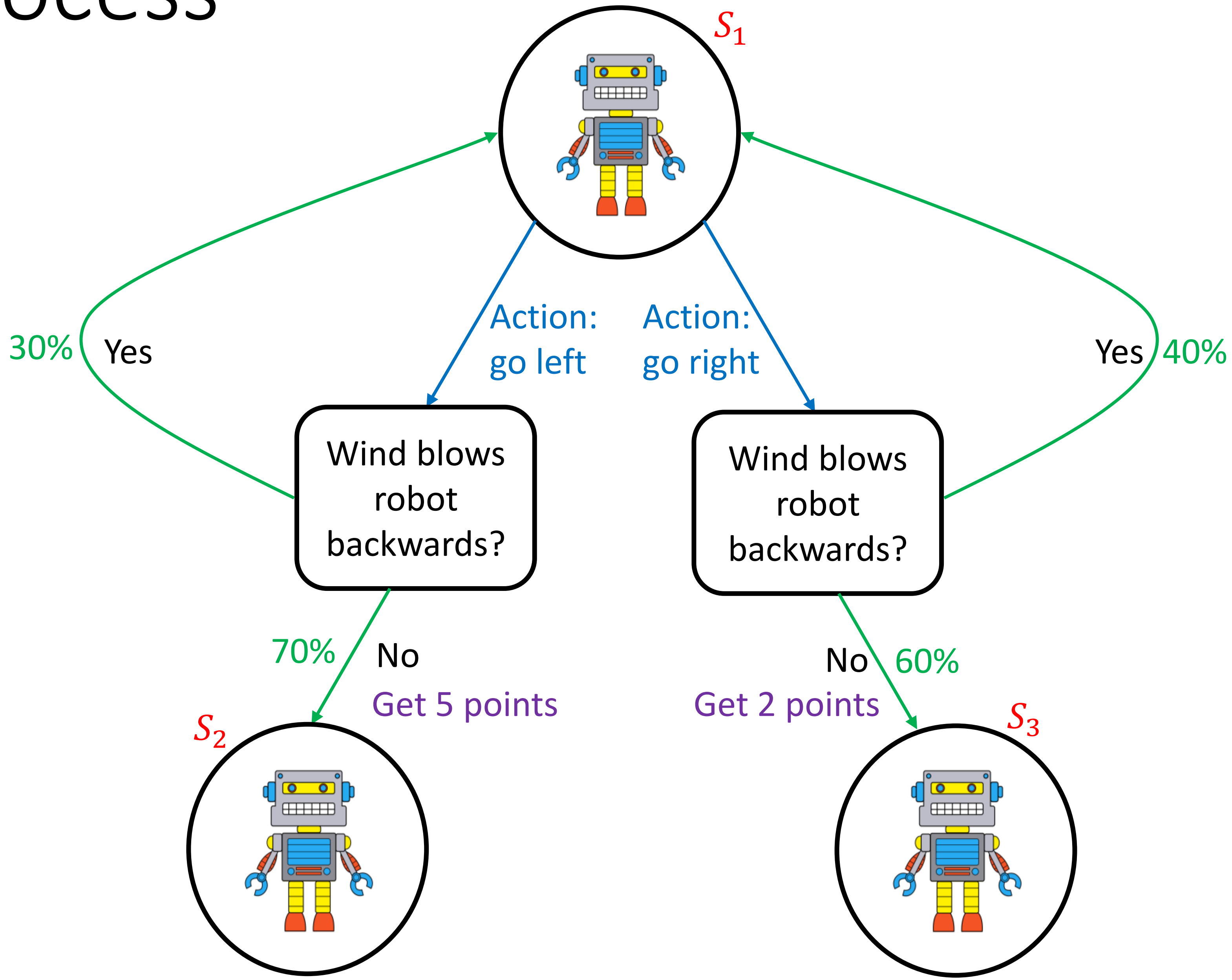
- A state s_t is Markov if and only if:

$$P[s_{t+1}|s_t] = P[s_{t+1}|s_1, \dots, s_t]$$

- The future is independent of the past, given the present
- The state captures all relevant information from the history
- Once the state is known, the history may be thrown away, i.e. state is a sufficient statistic of the future

Markov Decision Process

- States: s_1, s_2, s_3
- Actions: go left or go right
- Rewards:
    1. 5 points for going to s_2
    2. 2 points for going to s_3



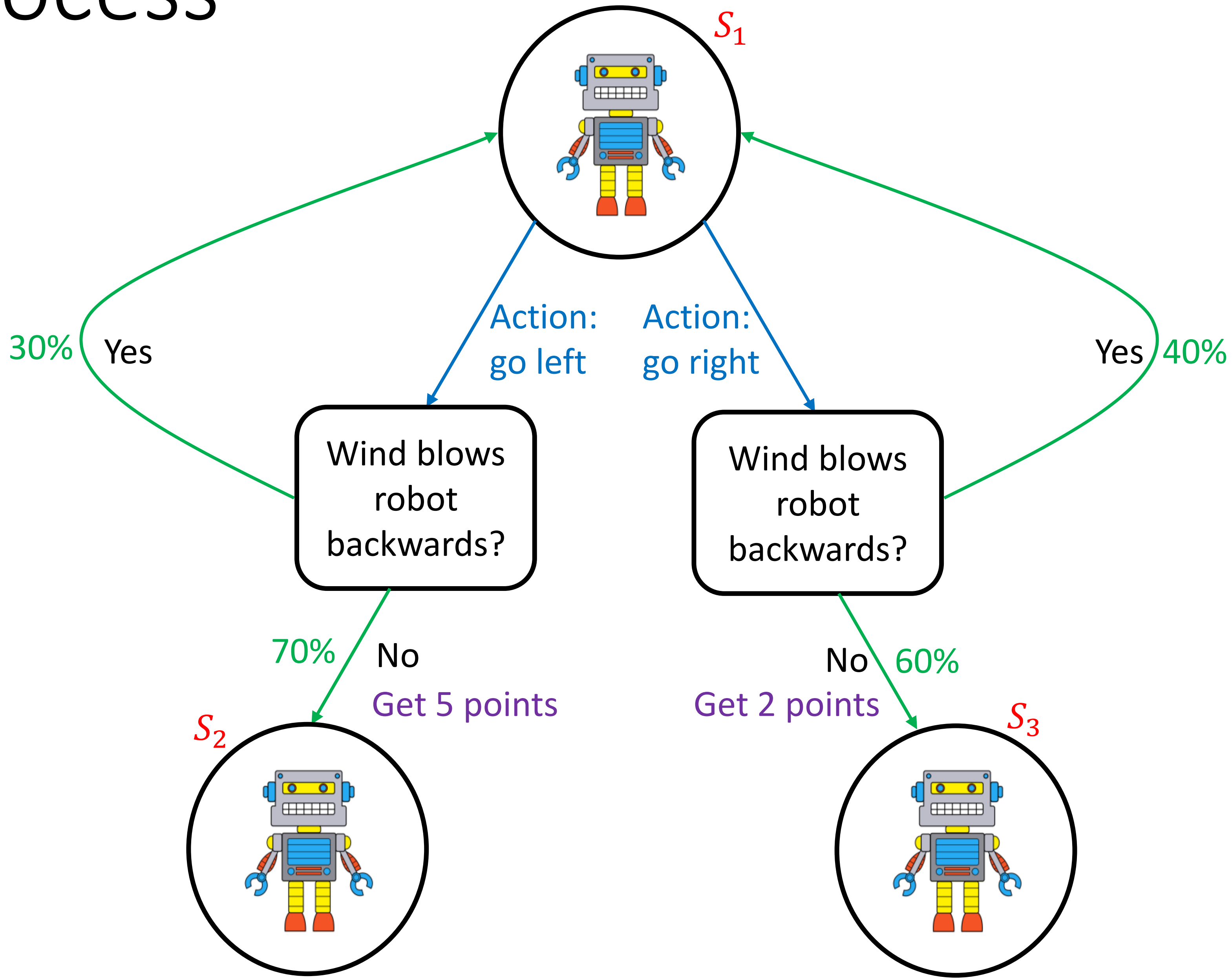
Markov Decision Process

- State Transition Matrix:

$$\mathcal{P}^{action} = \begin{bmatrix} P_{11}^{a} & P_{12}^{a} & P_{13}^{a} \\ P_{21}^{a} & P_{22}^{a} & P_{23}^{a} \\ P_{31}^{a} & P_{32}^{a} & P_{33}^{a} \end{bmatrix}$$

$$\mathcal{P}^{right} = \begin{bmatrix} 0.4 & 0 & 0.6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathcal{P}^{left} = \begin{bmatrix} 0.3 & 0 & 0.7 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Example: Dice Game

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

- You choose either to stay or quit

1. Quit: get 10 points and end the game.
2. Stay: get 4 points and then roll the dice:
    a) If the dice is 1 or 2, end the game.
    b) Otherwise, get 4 points, continue to the next round.

Question: how do you get the maximum points in this game?

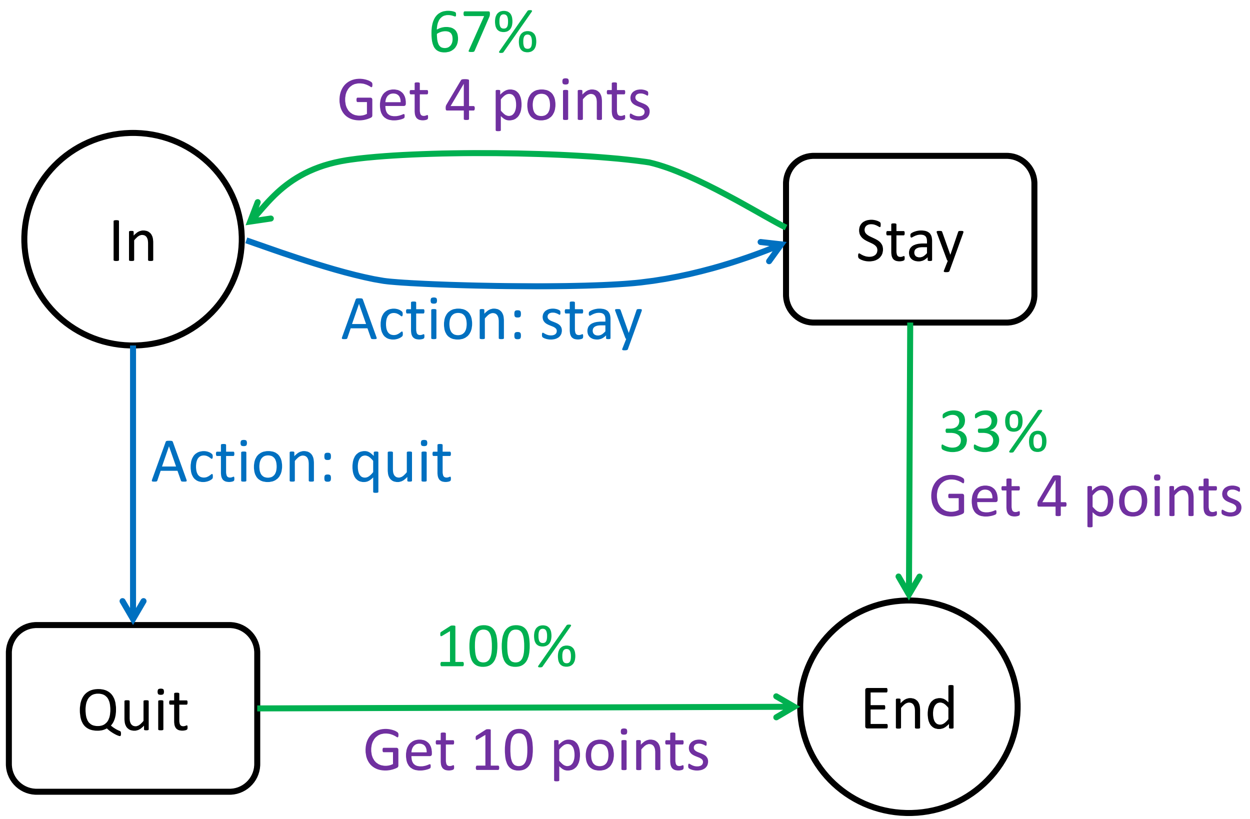
MDP

Question: how do you get the maximum points in this game?

- States: In, End
- Actions: stay, quit
- Rewards:
  1. 4 points for stay
  2. 10 points for quit
- State Transition Matrix:

$$\mathcal{P}^{stay} = \begin{bmatrix} 0.67 & 0.33 \\ 0 & 0 \end{bmatrix}$$

$$\mathcal{P}^{quit} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$ deterministic



Search vs MDPs

1. The successor function $Succ(s, a)$ is a special case of state transition probability matrix:

$$\mathcal{P}(s, a, s') = \begin{cases} 1 & if\ s' = Succ(s, a) \\ 0 & otherwise \end{cases}$$

2. Another difference is that instead of minimizing costs (search), MDPs maximize rewards

3. In search, the solution is a path. In MDPs, it is a policy π that maps each state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$
   - Policy should maximize the total rewards
   - $\pi(a|s) = P[A_t = a | S_t = s]$

Evaluating a policy

- The total rewards is called the utility (AKA Return G_t) of a policy: the (discounted) sum of the rewards on the path (this is a random variable, so can't be maximized)

Path	Utility
[in; stay, 4, end]	4
[in; stay, 4, in; stay, 4, in; stay, 4, end]	12
[in; stay, 4, in; stay, 4, end]	8
[in; stay, 4, in; stay, 4, in; stay, 4, in; stay, 4, end]	16
…	…
(action, reward, new state)	

- Value (expected return): The value of a policy at a state is the expected return

Return

- Path: $s_0, \quad a_1 r_1 s_1, \; a_2 r_2 s_2, \; \ldots$
  - Sometimes written as $s_1 a_1 r_2, \; s_2 a_2 r_3, \; \ldots$
- Discount factor: reduces future rewards:
  - a reward today might be worth more than the same reward tomorrow
- $\gamma = 1$ (save for the future):

$$[stay, stay, stay, stay]: \; 4 + 4 + 4 + 4 = 16$$

- $\gamma = 0$ (live in the moment):

$$[stay, stay, stay, stay]: \; 4 + 0 \cdot (4 + \cdots) = 4$$

- $\gamma = 0.5$ (balanced life):

$$[stay, stay, stay, stay]: \left(\frac{1}{2}\right)^0 \cdot 4 + \left(\frac{1}{2}\right)^1 \cdot 4 + \left(\frac{1}{2}\right)^2 \cdot 4 + \left(\frac{1}{2}\right)^3 \cdot 4 = 7.5$$

Value and Q-Value

Value of a policy

- The state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π

$$v_\pi(s) = E_\pi[G_t \mid S_t = s]$$

Q-value of a policy

- The action-value function $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π

$$q_\pi(s, a) = E_\pi[G_t \mid S_t = s, A_t = a]$$

Value and Q-Value

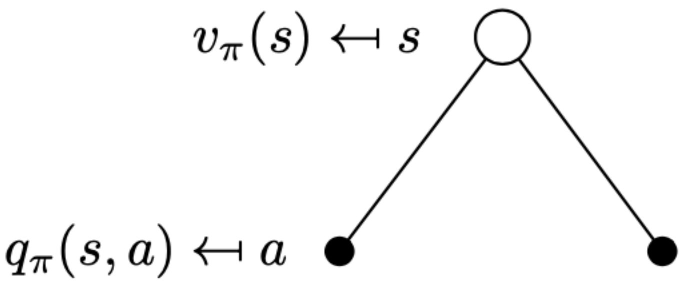
- The state-value function: immediate reward plus discounted value of successor state:

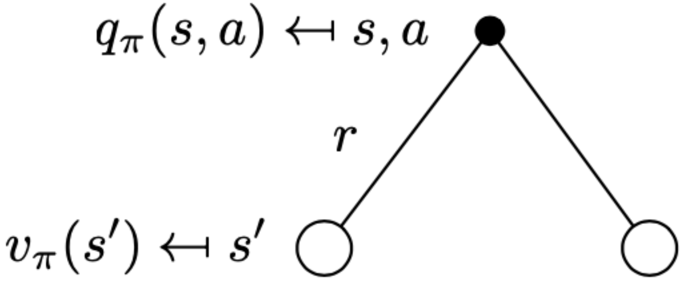
$$v_\pi(s) = E_\pi\left[R_{t+1} + \gamma\, v_\pi(S_{t+1}) | S_t = s\right]$$

- The action-value function: immediate reward plus discounted value of successor state:

$$q_\pi(s, a) = E_\pi[R_{t+1} + \gamma\, q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

Bellman Expectation Equation



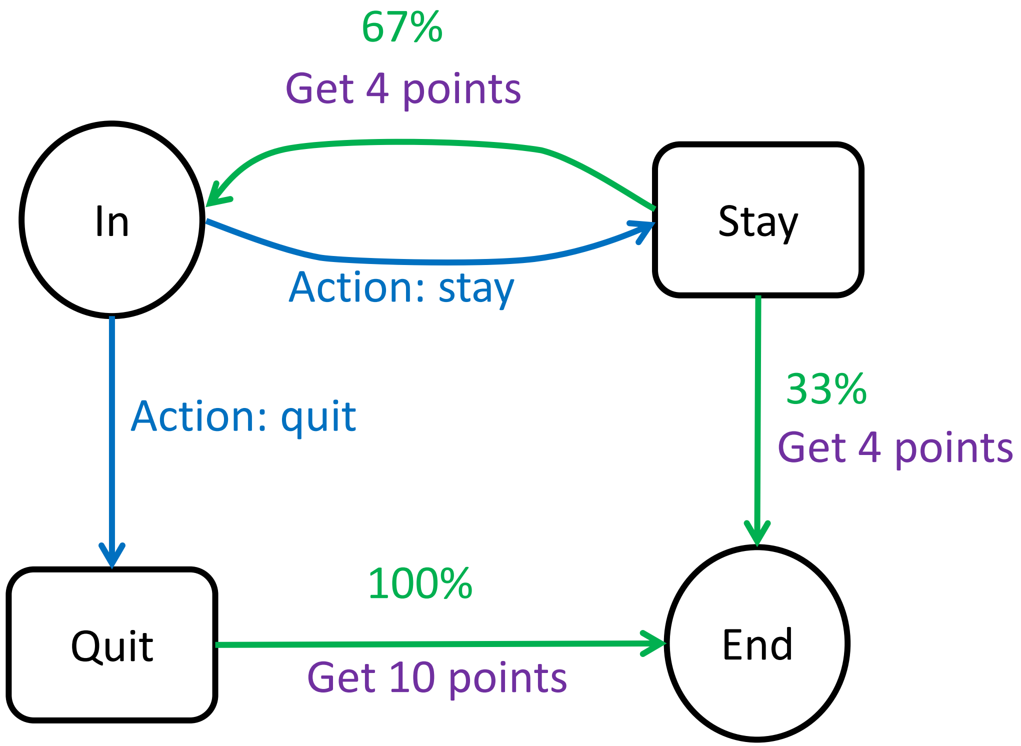


$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s)(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v_\pi(s'))$$

Example: Dice Game



Assume the policy is "stay":

$$\pi(a|s) = P[A_t = a|S_t = s]$$

$$\pi = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{matrix} \text{Stay} \\ \text{Quit} \end{matrix}$$

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s)\left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v_\pi(s')\right)$$

$$\gamma = 1$$

$$V_\pi(End) = 0$$

action quit: $V_\pi(In) = 0(\ \dots\ ) = 0$

action stay: $V_\pi(In) = 1\left[4 + 1\left(\frac{1}{3}V_\pi(End) + \frac{2}{3}V_\pi(In)\right)\right]$

$$V_\pi(In) = 4 + \left(\frac{1}{3} \times 0 + \frac{2}{3}V_\pi(In)\right)$$

$$V_\pi(In) = 4 + \frac{2}{3}V_\pi(In)$$

$$V_\pi(In) = 12$$

Policy evaluation

- The previous solution isn't always possible, so we use an algorithm called iterative policy evaluation

Initialize $V_{\pi}^{(0)}(s) \leftarrow 0$ for all states s

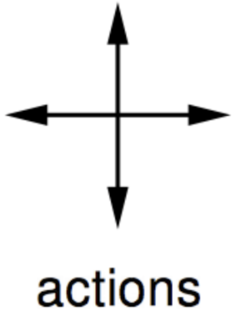
For iteration $t = 1, \dots, T$

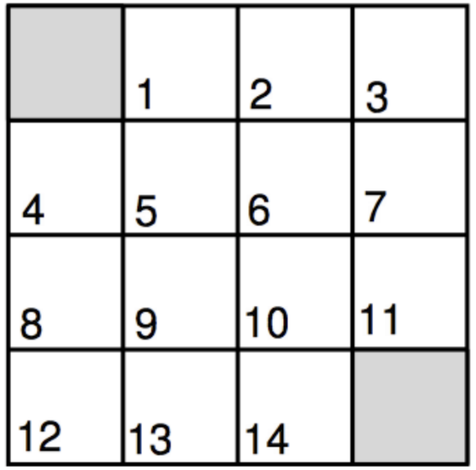
For each state s:

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s)\left(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v_k(s')\right)$$

Students

Gridworld Example





- Undiscounted episodic MDP ($\gamma = 1$)
- Nonterminal states 1, ..., 14
- One terminal state (shown twice as shaded squares)
- Actions leading out of the grid leave state unchanged
- Reward is −1 until the terminal state is reached
- Environment is deterministic
- Agent follows uniform random policy

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

Students

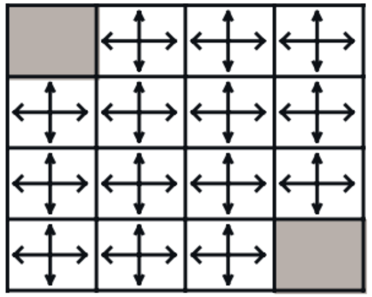
$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s)(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v_k(s'))$$

Gridworld Example

v_k for the random policy | greedy policy w.r.t. v_k

$k = 0$

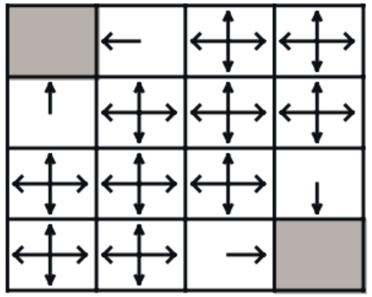
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



random policy

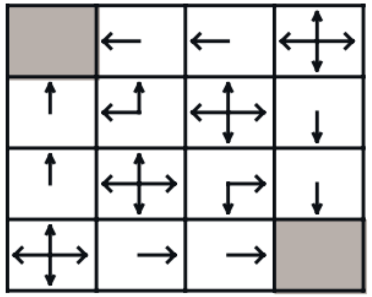
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



Students

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s)(R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a, v_k(s'))$$

Gridworld Example

v_k for the random policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

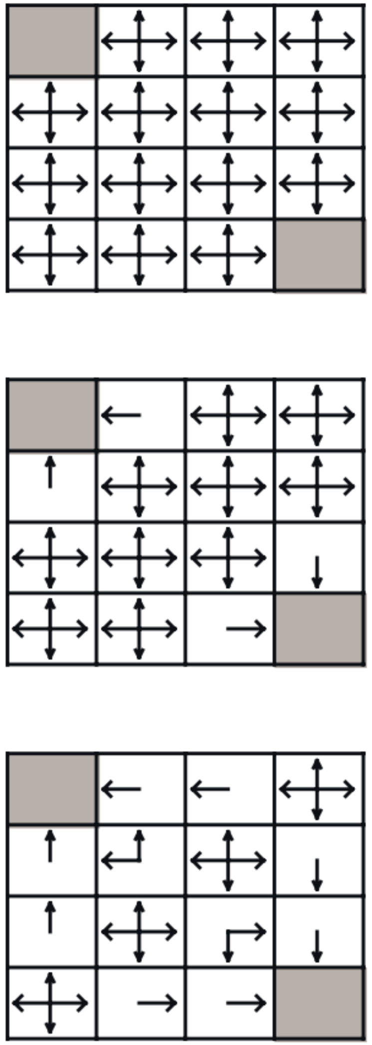
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

greedy policy w.r.t. v_k



random policy

Action: north / south / east / west

$\pi(a|s) = 0.25$

$R_s^a = -1$

$\gamma = 1$

$v_{k=0} = 0$

So, for each action:

=0.25[-1+(1)(0)] =-0.25

Then sum:

Total = -1

Students

Gridworld Example

$k = 3$

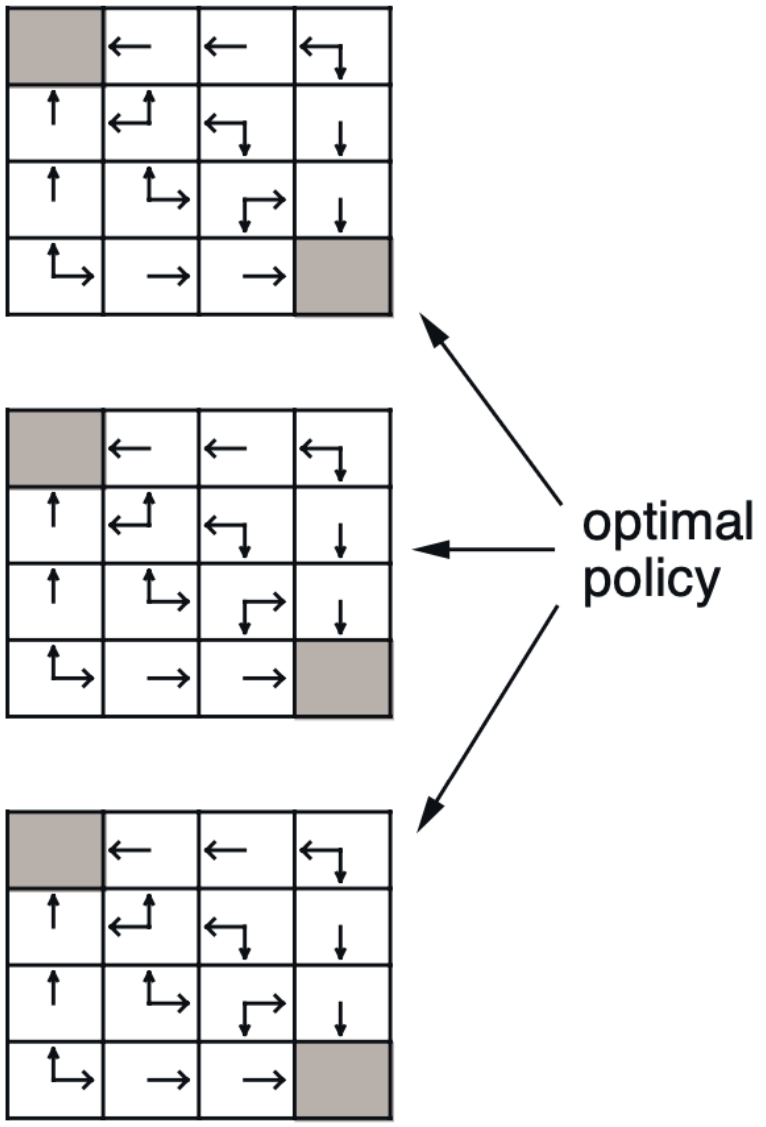
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

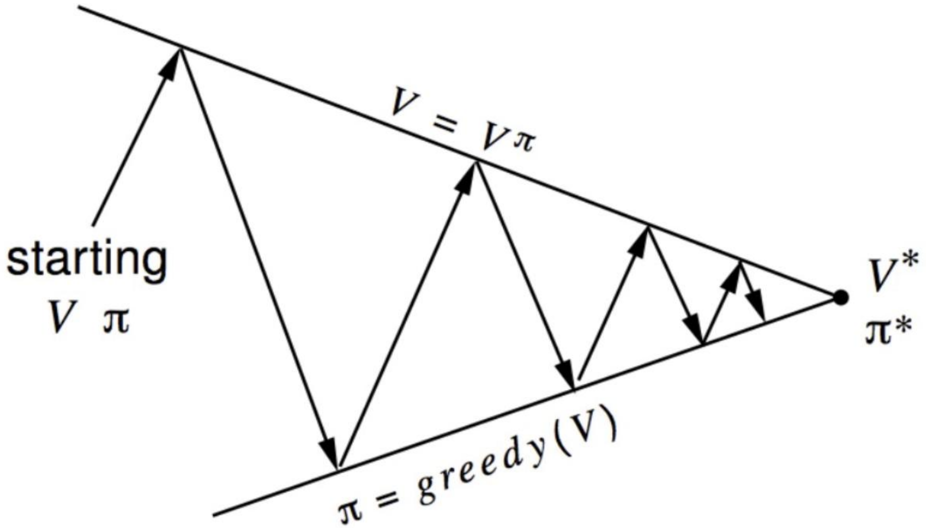
$k = \infty$

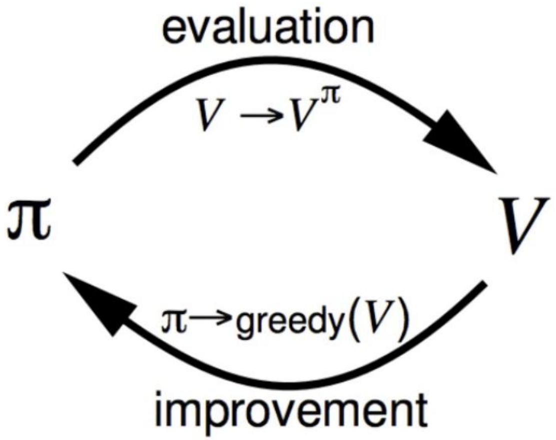
0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



How to Improve a Policy

- The policy is improved using Policy Iteration





- Policy evaluation: Estimate v_π using Iterative policy evaluation
- Policy improvement: Generate $\pi' \geq \pi$ using Greedy policy improvement