

Markov Decision Processes

CMU 10-703

Katerina Fragkiadaki



Supervision for learning goal-seeking behaviors

1. Learning from expert demonstrations (last lecture)

Instructive feedback: the expert directly suggests correct actions, e.g., your (oracle) advisor directly suggests to you ideas that are worth pursuing

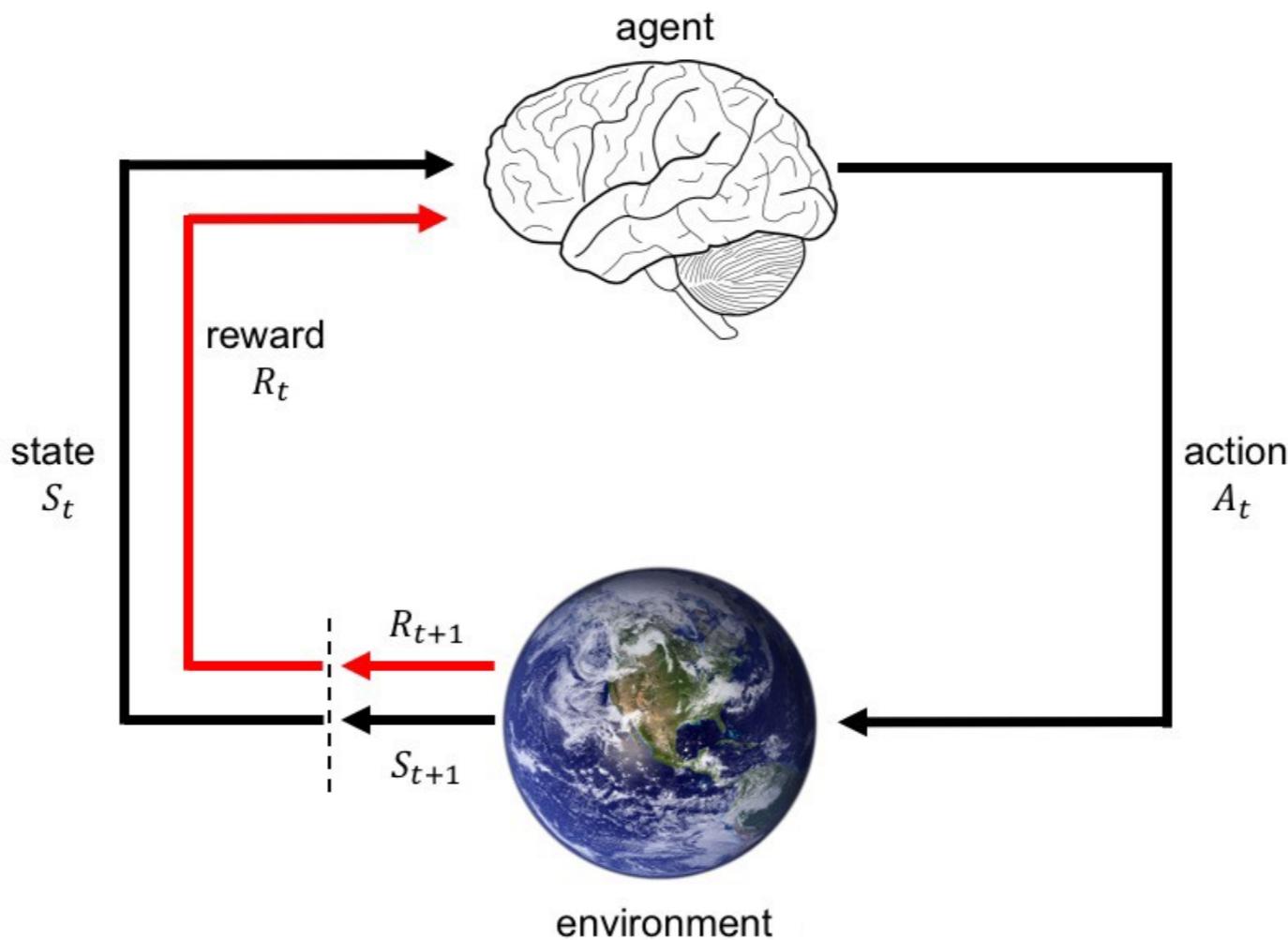
2. Learning from rewards while interacting with the environment

Evaluative feedback: the environment provides signal whether actions are good or bad. E.g., your advisor tells you if your research ideas are worth pursuing

Note: Evaluative feedback depends on the current policy the agent has: if you never suggest good ideas, you will never have the chance to know they are worthwhile. Instructive feedback is independent of the agent's policy.

Reinforcement learning

Learning behaviours from rewards while interacting with the environment



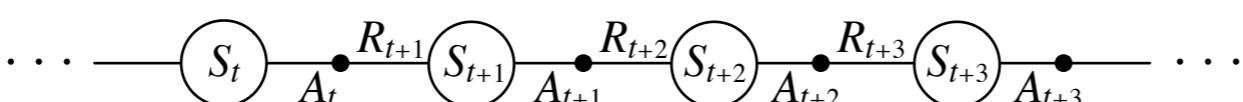
Agent and environment interact at discrete time steps: $t = 0, 1, 2, 3, \dots$

Agent observes state at step t : $S_t \in \mathcal{S}$

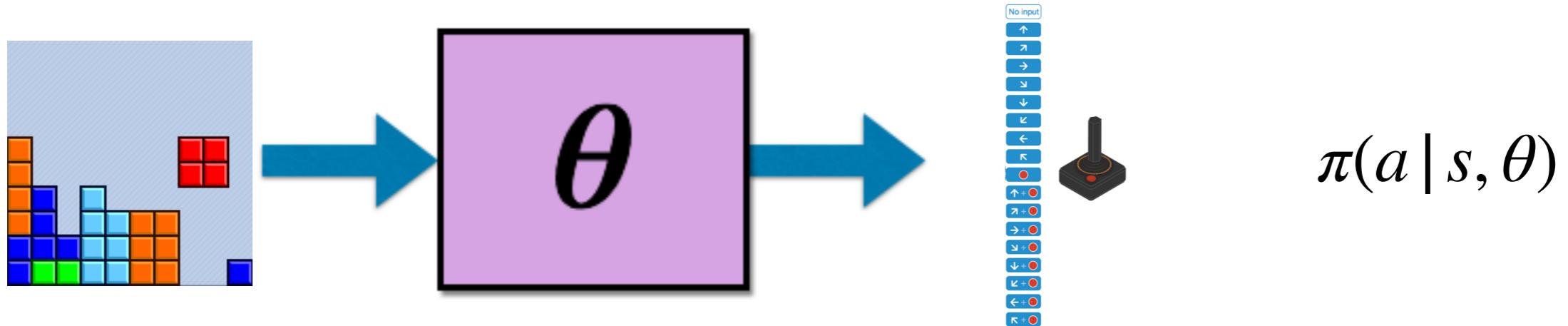
produces action at step t : $A_t \in \mathcal{A}(S_t)$

gets resulting reward: $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$

and resulting next state: $S_{t+1} \in \mathcal{S}^+$



Q: How can we learn the weights?



- Use Markov Design Process (MDP) formulation!
- Intuitively, the world is structured, it is comprised of states, reward is decomposed over states, states transition to one another with some transition probabilities (dynamics), etc..

Finite Markov Decision Process

A **Finite Markov Decision Process** is a tuple $(\mathcal{S}, \mathcal{A}, T, r, \gamma)$

- \mathcal{S} is a finite set of states
- \mathcal{A} is a finite set of actions
- p is one step dynamics function
- r is a reward function
- γ is a discount factor $\gamma \in [0, 1]$

Markovian States

- A state captures whatever information is available to the agent at step t about its environment.
- The state can include immediate “sensations,” highly processed sensations, and structures built up over time from sequences of sensations, memories etc.
- A state should summarize past sensations so as to retain all “essential” information, i.e., it should have the **Markov Property**:

$$\mathbb{P}[R_{t+1} = r, S_{t+1} = s' | S_0, A_0, R_1, \dots, S_{t-1}, A_{t-1}, R_t, S_t, A_t] = \mathbb{P}[R_{t+1} = r, S_{t+1} = s' | S_t, A_t]$$

for all $s' \in \mathcal{S}$, $r \in \mathcal{R}$, and all histories

- We should be able to throw away the history once state is known

The agent learns a Policy

Definition: A policy is a distribution over actions given states,

$$\pi(a | s) = \Pr(A_t = a | S_t = s), \forall t$$

- A policy fully defines the behavior of an agent
- The policy is stationary (time-independent)
- During learning, the agent changes his policy as a result of experience

Special case: deterministic policies

$\pi(s) = \text{the action taken with prob} = 1 \text{ when } S_t = s$

Definitions

Agent: an entity that is equipped with **sensors**, in order to sense the environment, and **end-effectors** in order to act in the environment, and goals that he wants to achieve

Policy: a mapping function from observations (sensations, inputs of the sensors) to actions of the end effectors.

Model: the mapping function from states/observations and actions to future states/observations

Planning: unrolling a model forward in time and selecting the best action sequence that satisfies a specific goal

Plan: a sequence of actions

The recycling robot MDP

- At each step, robot has to decide whether it should (1) actively search for a can, (2) wait for someone to bring it a can, or (3) go to home base and recharge.
- Searching is better but runs down the battery; if runs out of power while searching, has to be rescued (which is bad).
- Decisions made on basis of current energy level: high, low.
- Reward = number of cans collected

The recycling robot MDP

$$\mathcal{S} = \{\text{high}, \text{low}\}$$

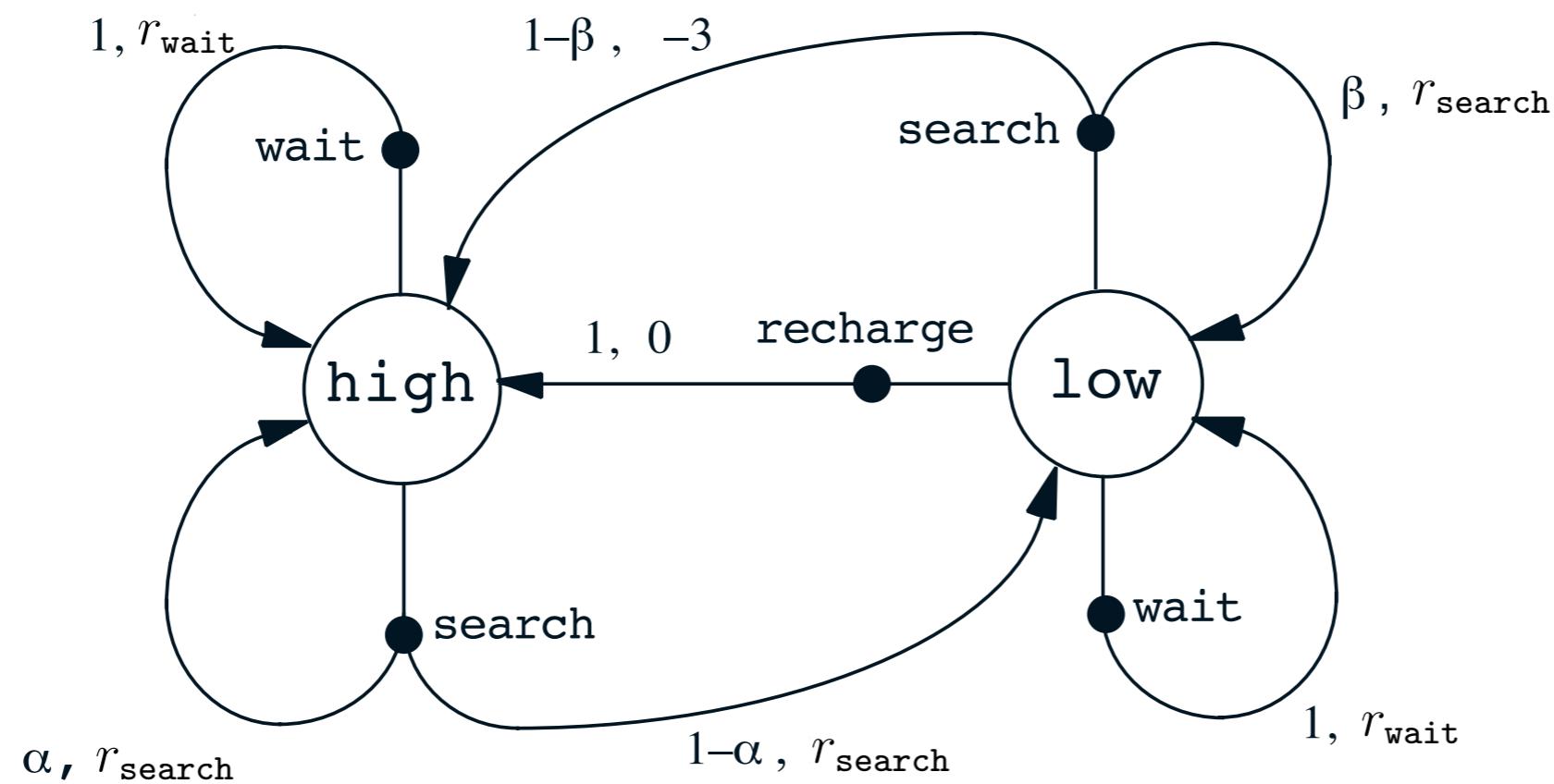
$$\mathcal{A}(\text{high}) = \{\text{search}, \text{wait}\}$$

$$\mathcal{A}(\text{low}) = \{\text{search}, \text{wait}, \text{recharge}\}$$

r_{search} = expected no. of cans while searching

r_{wait} = expected no. of cans while waiting

$$r_{\text{search}} > r_{\text{wait}}$$



Q: what the robot will do does it depend on the number of cans he has collected thus far?

Rewards reflect goals

Rewards are scalar values provided by the environment to the agent that indicate whether goals have been achieved, e.g., 1 if goal is achieved, 0 otherwise, or -1 for overtime step the goal is not achieved

- Goals specify **what** the agent needs to achieve, not **how** to achieve it.
- The simplest and cheapest form of supervision, and surprisingly general:
All of what we mean by goals and purposes can be well thought of as the maximization of the cumulative sum of a received scalar signal (reward)

$$r(s, a) = \mathbb{E}[R_{t+1} | S_t = s, A_t = a] = \sum_{r \in \mathbb{R}} r \sum_{s' \in S} p(s', r | s, a)$$

- Goal seeking behaviour, achieving purposes and expectations can be formulated mathematically as maximizing expected cumulative sum of scalar values...

Returns G_t - Episodic tasks

Episodic tasks: interaction breaks naturally into episodes, e.g., plays of a game, trips through a maze.

There is no memory across episodes.

In episodic tasks, we almost always use simple *total reward*:

$$G_t = R_{t+1} + R_{t+2} + \dots + R_T$$

where T is a final time step at which a **terminal state** is reached, ending an episode.

Returns G_t - Continuing tasks

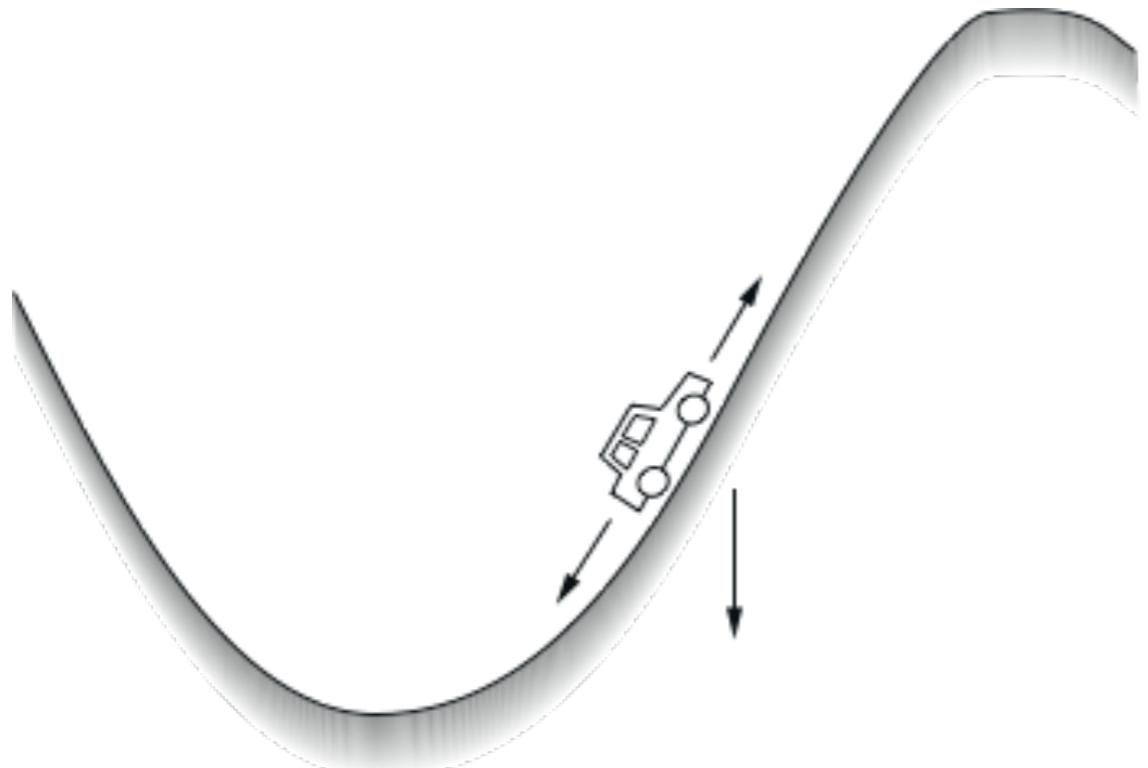
Continuing tasks: interaction does not have natural episodes, but just goes on and on...just like real life

In continuing tasks, we often use simple *total discounted reward*:

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Why temporal discounting? A sequence of interactions based on which the reward will be judged at the end is called **episode**. Episodes can have finite or infinite length. For infinite length, the uncounted sum blows up, thus we add discounting $\gamma < 1$ to prevent this, and treat both cases in a similar manner.

Mountain car



Get to the top of the hill
as quickly as possible.

reward = -1 for each step where **not** at top of hill

⇒ return = - number of steps before reaching top of hill

Return is maximized by minimizing
number of steps to reach the top of the hill.

Value Functions are Expected Returns

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

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

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

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

Optimal Value Functions are Best Achievable Expected Returns

- **Definition:** The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

$$v_*(s) = \max_{\pi} v_{\pi}(s)$$

- The *optimal action-value function* $q_*(s, a)$ is the maximum action-value function over all policies

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

Value Functions

- Value functions measure the goodness of a particular state or state/action pair: how good is for the agent to be in a particular state or execute a particular action at a particular state, **for a given policy**.
- Optimal value functions measure **the best possible** goodness of states or state/action pairs *under all possible policies*.

	state values	action values
prediction	v_π	q_π
control	v_*	q_*

Solving MDPs

- **Prediction:** Given an MDP $(\mathcal{S}, \mathcal{A}, T, r, \gamma)$ and a policy

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

find the state and action value functions.

- **Optimal control:** given an MDP $(\mathcal{S}, \mathcal{A}, T, r, \gamma)$, find the optimal policy (aka the planning problem). Compare with the learning problem with missing information about rewards/dynamics.

Why Value Functions are useful

Value functions capture the knowledge of the agent regarding how good is each state for the goal he is trying to achieve.

We communicate our value functions to one another.

“...knowledge is represented as a large number of approximate value functions learned in parallel...”

Horde: A Scalable Real-time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction, Sutton et al.

Why Value Functions are useful

An optimal policy can be found by maximizing over $q_*(s, a)$:

$$\pi_*(a | s) = \begin{cases} 1, & \text{if } a = \operatorname{argmax}_{a \in \mathcal{A}} q^*(s, a) \\ 0, & \text{otherwise} \end{cases}$$

An optimal policy can be found from $v_*(s)$ and the model dynamics using one step look ahead:

$$\pi_*(a | s) = \begin{cases} 1, & \text{if } a = \operatorname{argmax}_{a \in \mathcal{A}} \left(\sum_{s',r} p(s', r | s, a) (r + \gamma v_*(s')) \right) \\ 0, & \text{otherwise} \end{cases}$$

- **If we know $q^*(s,a)$ we immediately have the optimal policy, we do not need the dynamics!**
- If we know $v^*(s)$, we need the dynamics to do one step lookahead, to choose the optimal action

Value Functions are Expected Returns

- The value of a state, given a policy:

$$v_\pi(s) = \mathbb{E}\{G_t \mid S_t = s, A_{t:\infty} \sim \pi\} \quad v_\pi : \mathcal{S} \rightarrow \mathbb{R}$$

- The value of a state-action pair, given a policy:

$$q_\pi(s, a) = \mathbb{E}\{G_t \mid S_t = s, A_t = a, A_{t+1:\infty} \sim \pi\} \quad q_\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$$

- The optimal value of a state:

$$v_*(s) = \max_\pi v_\pi(s) \quad v_* : \mathcal{S} \rightarrow \mathbb{R}$$

- The optimal value of a state-action pair:

$$q_*(s, a) = \max_\pi q_\pi(s, a) \quad q_* : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$$

- Optimal policy: π_* is an optimal policy if and only if

$$\pi_*(a|s) > 0 \text{ only where } q_*(s, a) = \max_b q_*(s, b) \quad \forall s \in \mathcal{S}$$

- in other words, π_* is optimal iff it is *greedy* wrt q_*

Q: What are the expectations over (what is stochastic)?

Bellman Expectation Equation

$$\begin{aligned}G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} L \\&= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} L \right) \\&= R_{t+1} + \gamma G_{t+1}\end{aligned}$$

Bellman Expectation Equation

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} \mathbb{L} \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} \mathbb{L} \right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

So by taking expectations:

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

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

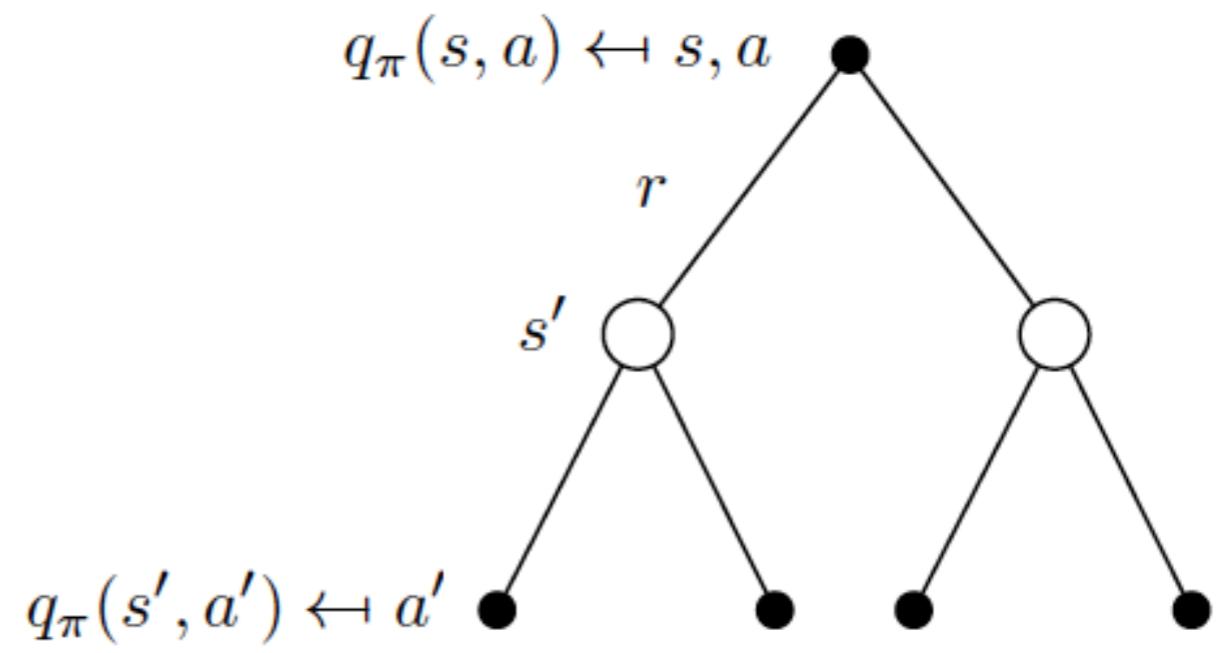
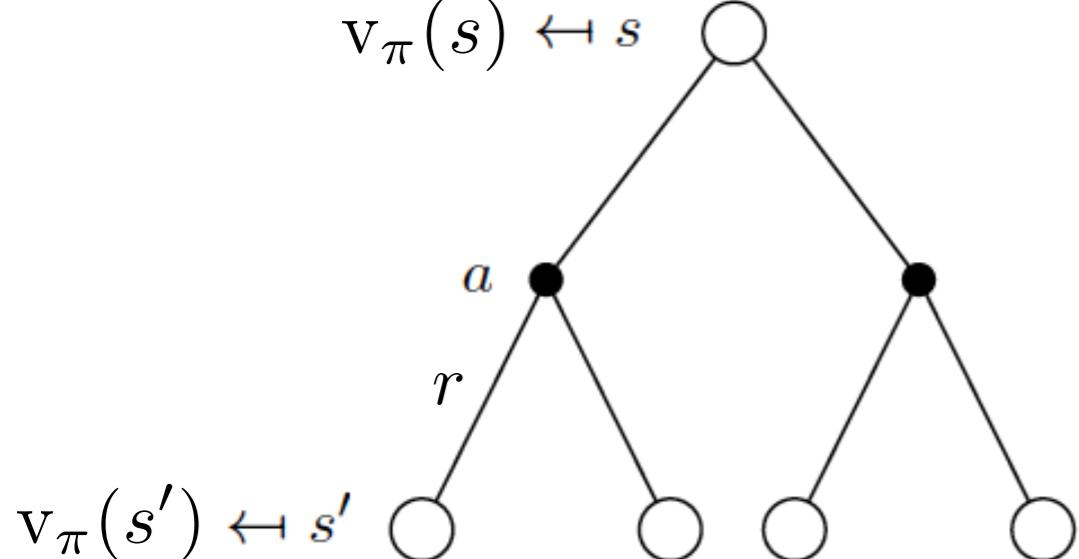
Or, without the expectation operator:

$$\begin{aligned} v_\pi(s) &= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) \left[r + \gamma v_\pi(s') \right] \\ q_\pi(s, a) &= \sum_{r, s'} p(s, r' | s, a) \left(r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a') \right) \end{aligned}$$

This is a set of linear equations, one for each state.
The value function for π is its unique solution.

Back-up diagram for value functions

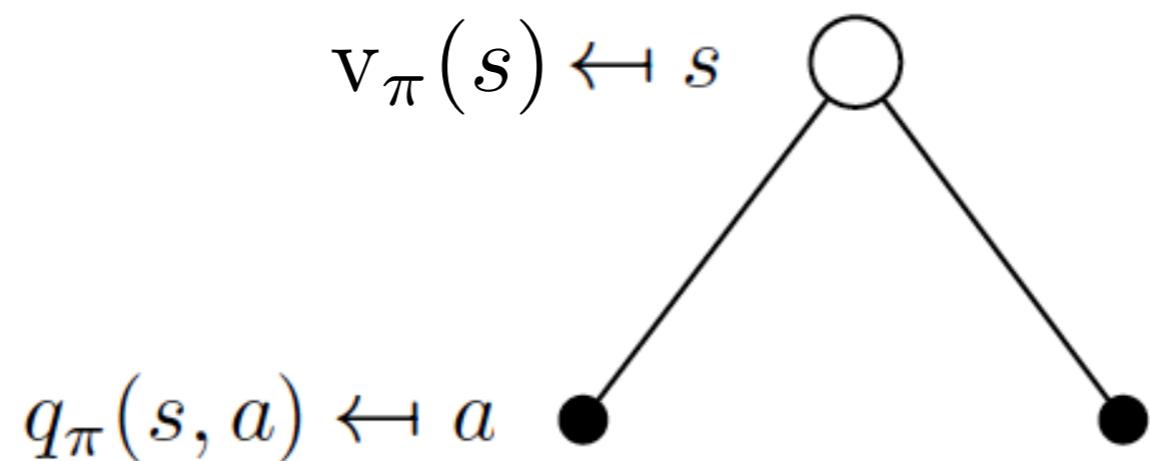
The probabilities of landing on each of the leaves sum to 1



$$v_{\pi}(s) = \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) [r + \gamma v_{\pi}(s')]$$

$$q_{\pi}(s, a) = \sum_{r, s'} p(s', r|s, a) \left(r + \gamma \sum_{a'} \pi(a'|s') q_{\pi}(s', a') \right)$$

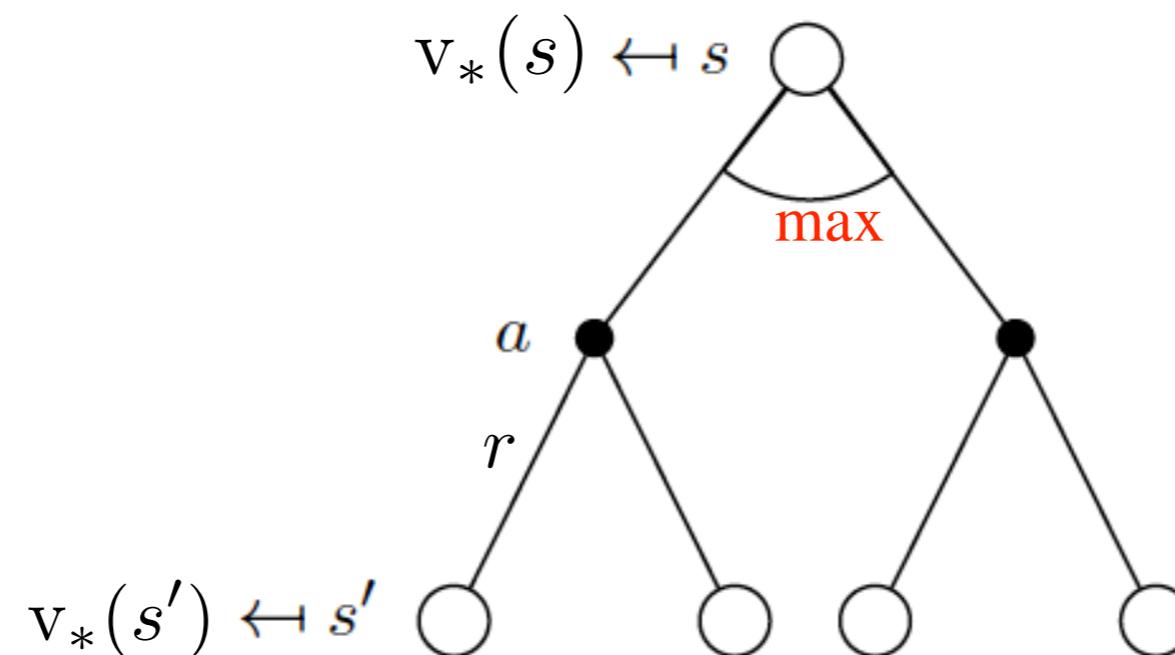
Relating state and state/action value functions



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

Bellman Optimality Equations for \mathcal{V}_*

The value of a state under an **optimal** policy must equal the expected return for the **best** action from that state

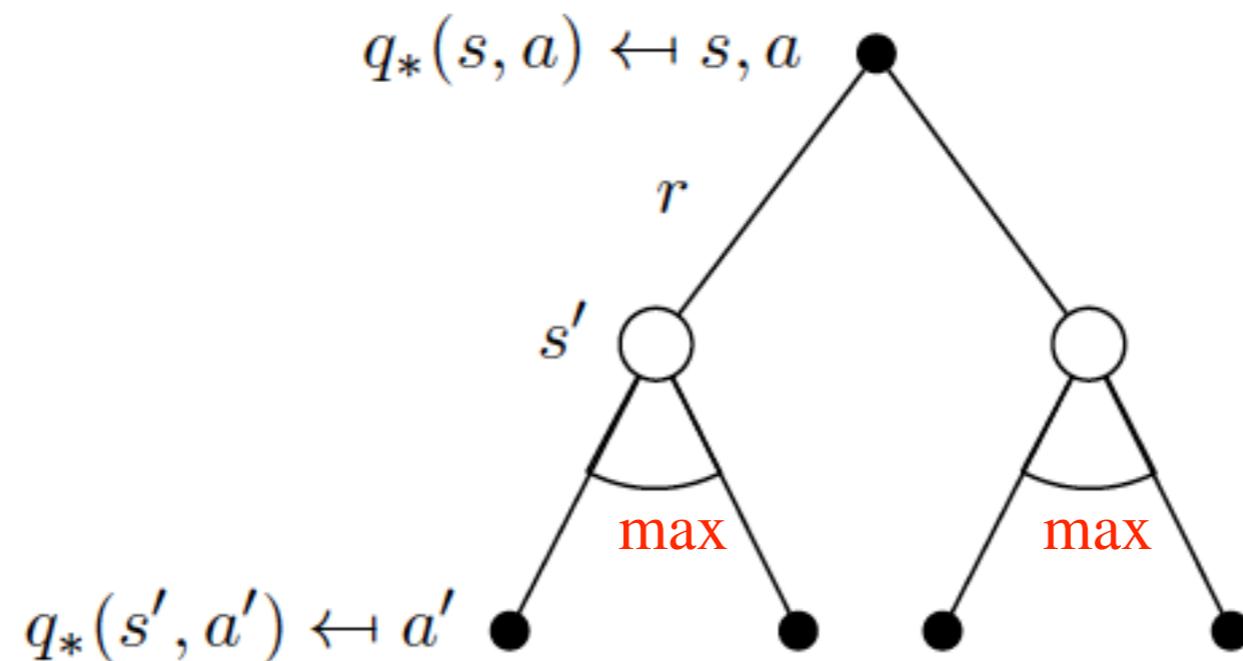


$$v_*(s) = \max_{a \in \mathcal{A}} \left(\sum_{s',r} p(s',r | s, a) (r + \gamma v_*(s')) \right)$$

For the Bellman expectation equations we sum over all the leaves, here we choose only the best action branch!

v^* is the unique solution of this system of nonlinear equations

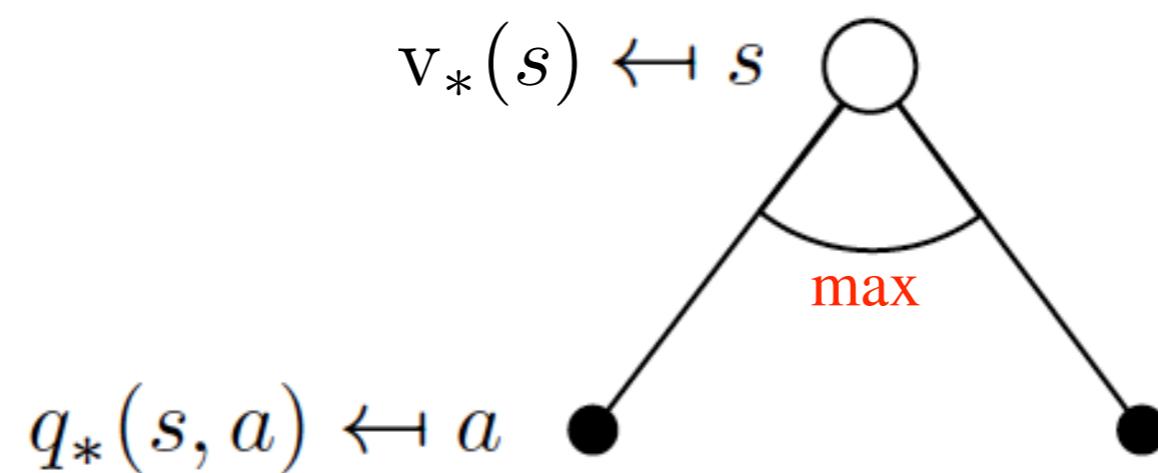
Bellman Optimality Equations for q_*



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

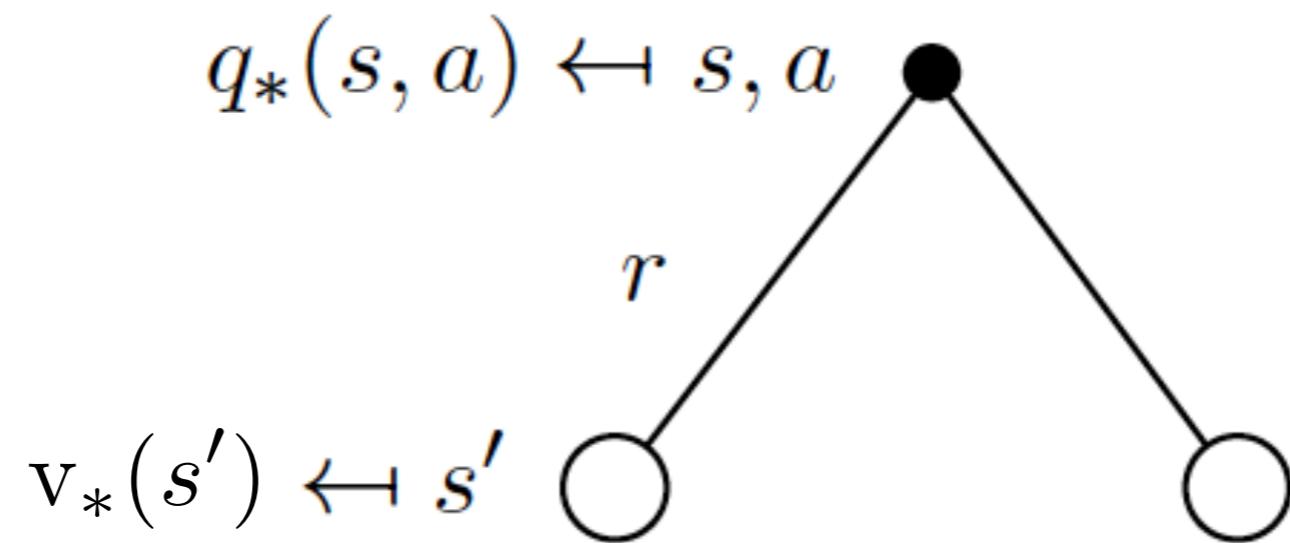
q^* is the unique solution of this system of nonlinear equations

Relating Optimal State and Action Value Functions



$$v_*(s) = \max_a q_*(s, a)$$

Relating Optimal State and Action Value Functions

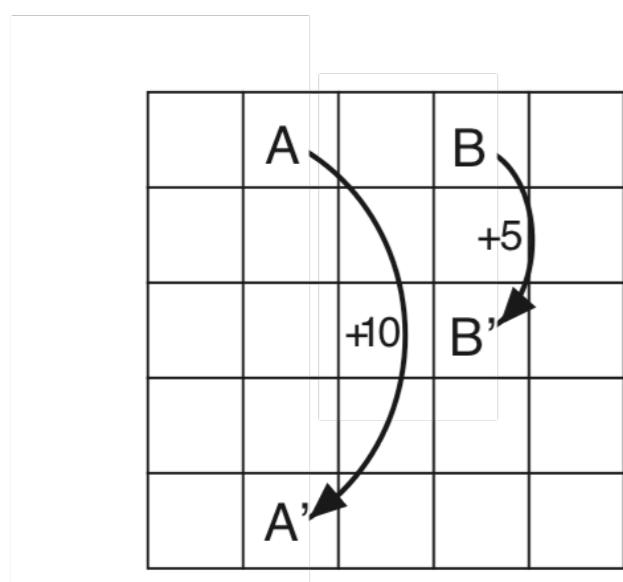


$$q_*(s, a) = \sum_{s', r} p(s', r | s, a)(r + \gamma v_*(s'))$$

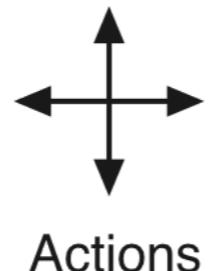
Gridworld-value function

- Actions: north, south, east, west; deterministic.
- If would take agent off the grid: no move but reward = -1
- Other actions produce reward = 0, except actions that move agent out of special states A and B as shown.

$$8.83 = 10 + 0.9 * (-1.3)$$



(a)



Actions

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

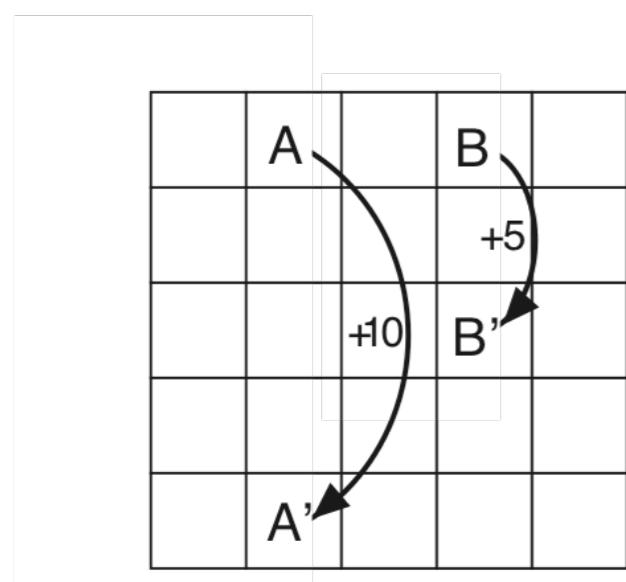
(b)

State-value function
for **equiprobable**
random policy;
 $\gamma = 0.9$

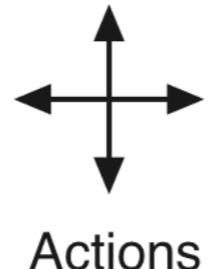
Gridworld-value function

- Actions: north, south, east, west; deterministic.
- If would take agent off the grid: no move but reward = -1
- Other actions produce reward = 0, except actions that move agent out of special states A and B as shown.

$$4.43 = 0.25 * (0 + 0.9 * 5.3 + 0 + 0.9 * 2.3 + 0 + 0.9 * 8.8 + -1 + 0.9 * 4.4)$$



(a)



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(b)

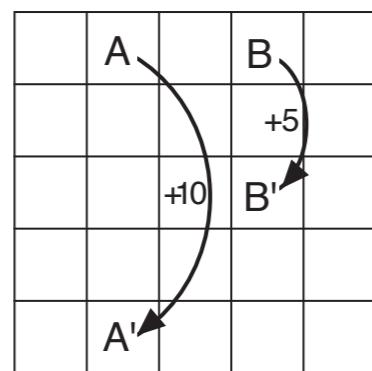
State-value function
for **equiprobable**
random policy;
 $\gamma = 0.9$

Gridworld - optimal value function

Any policy that is greedy with respect to ν_* is an optimal policy.

Therefore, given ν_* , one-step-ahead search produces the long-term optimal actions.

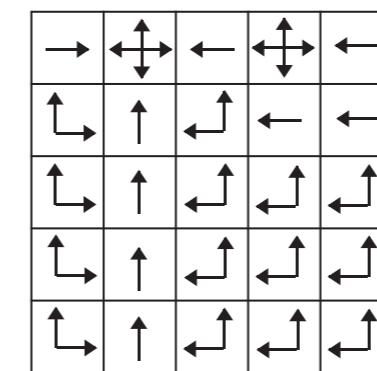
$$24.4 = 10 + 0.9 * (16.0)$$



a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) ν_*



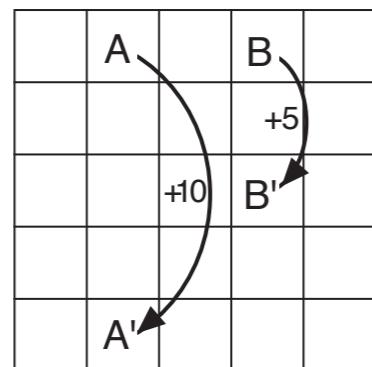
c) π_*

Gridworld - optimal value function

Any policy that is greedy with respect to v_* is an optimal policy.

Therefore, given v_* , one-step-ahead search produces the long-term optimal actions.

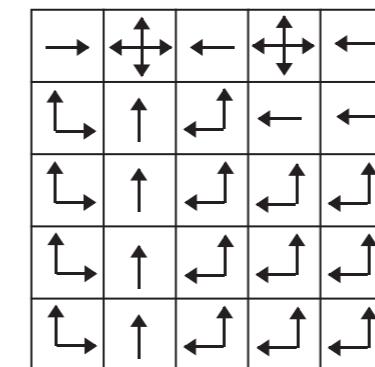
$$22.0 = \max(0+0.9 * 19.4, \\ 0+0.9 * 19.8, \\ 0+0.9 * 24.4, \\ -1+0.9 * 22.0)$$



a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) v_*



c) π_*

Optimal Policy

Define a partial ordering over policies

$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

Theorem: For any Markov Decision Process

- There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi, \forall \pi$
- All optimal policies achieve the optimal value function, $v_{\pi_*}(s) = v_*(s)$
- All optimal policies achieve the optimal action-value function, $q_{\pi_*}(s, a) = q_*(s, a)$

Solving the Bellman Equations

MDPs to MRPs

MDP under a **fixed** policy becomes **Markov Reward Process (MRP)**

$$\begin{aligned} v_\pi(s) &= \sum_{a \in \mathcal{A}} \pi(a|s) \left(r(s, a) + \gamma \sum_{s' \in \mathcal{S}} T(s'|s, a) v_\pi(s') \right) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) r(s, a) + \gamma \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} T(s'|s, a) v_\pi(s') \\ &= r_s^\pi + \gamma \sum_{s' \in \mathcal{S}} T_{s's}^\pi v_\pi(s') \end{aligned}$$

where $r_s^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) r(s, a)$ and $T_{s's}^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) T(s'|s, a)$

Matrix Form

The Bellman expectation equation can be written concisely using the induced MRP as

$$v_\pi = r^\pi + \gamma T^\pi v_\pi$$

with direct solution

$$v_\pi = (I - \gamma T^\pi)^{-1} r^\pi$$

of complexity $O(N^3)$

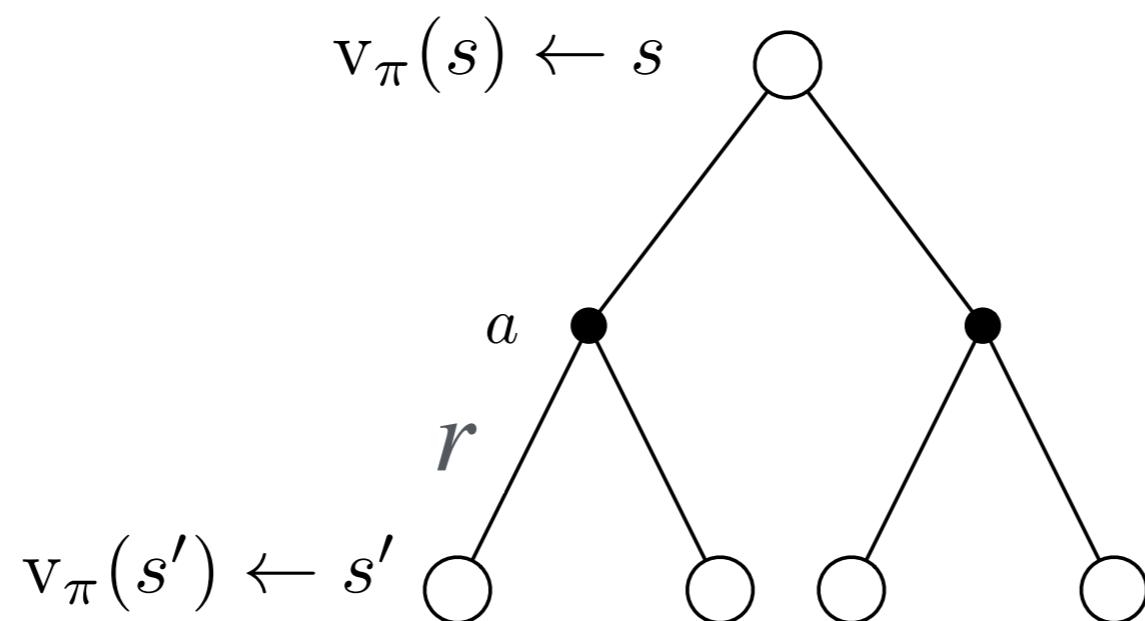
here T^π is an $|S| \times |S|$ matrix, whose (j,k) entry gives $P(s_k | s_j, a=\pi(s_j))$
 r^π is an $|S|$ -dim vector whose j^{th} entry gives $E[r | s_j, a=\pi(s_j)]$
 v_π is an $|S|$ -dim vector whose j^{th} entry gives $V_\pi(s_j)$

where $|S|$ is the number of distinct states

Iterative Methods: Recall the Bellman Equation

$$v_{\pi}(s) = \sum_a \pi(a | s) \sum_{r,s'} p(s', r | s, a) (r + \gamma v_{\pi}(s'))$$

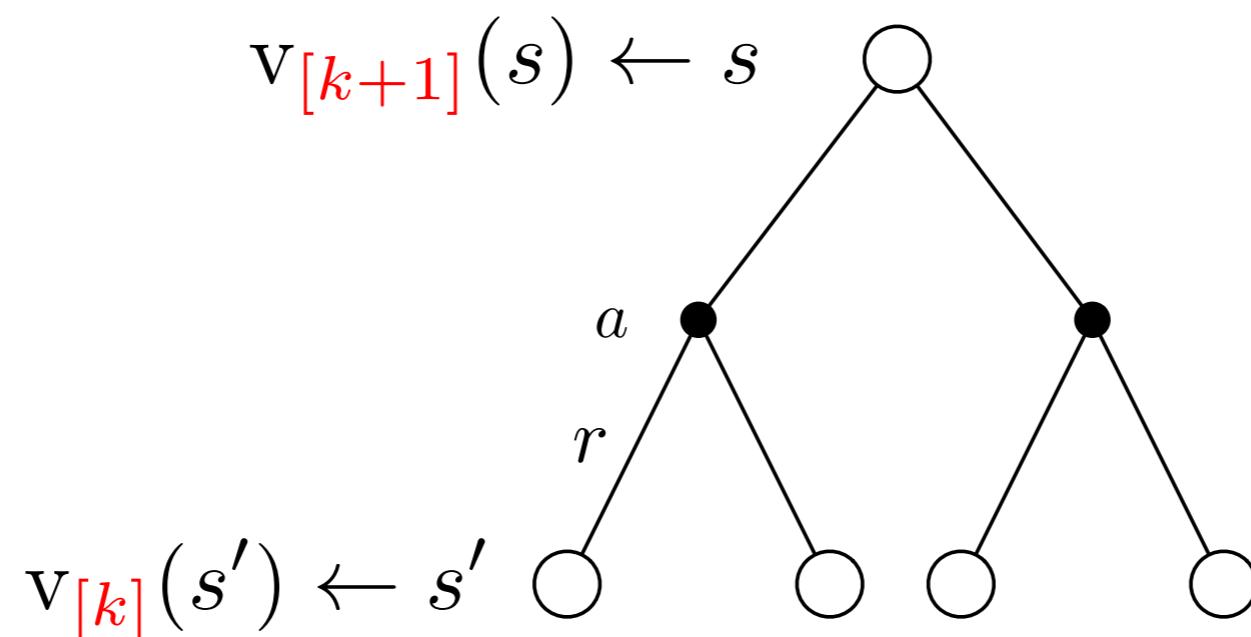
$$v_{\pi}(s) = \sum_a \pi(a | s) \left(r(s, a) + \gamma \sum_{r,s'} p(s' | s, a) v_{\pi}(s') \right)$$



Iterative Methods: Backup Operation

Given an expected value function at iteration k , we back up the expected value function at iteration $k+1$:

$$v_{[k+1]}(s) = \sum_a \pi(a | s) \left(r(s, a) + \gamma \sum_{r, s'} p(s' | s, a) v_{[k]}(s') \right)$$



Iterative Methods: Sweep

A **sweep** consists of applying the backup operation $v \rightarrow v'$ for **all the states** in \mathcal{S}

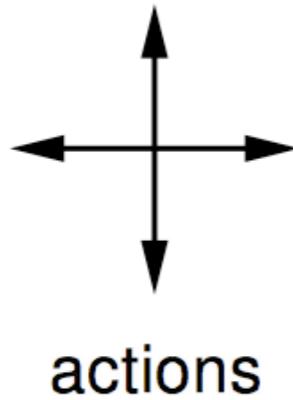
Applying the back up operator iteratively

$$v[0] \rightarrow v[1] \rightarrow v[2] \rightarrow \dots v_\pi$$

A full policy evaluation backup:

$$v_{[k+1]}(s) = \sum_a \pi(a | s) \left(r(s, a) + \gamma \sum_{r, s'} p(s' | s, a) v_{[k]}(s') \right), \forall s$$

A Small-Grid World



	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R = -1$
on all transitions

$$\gamma = 1$$

- An undiscounted episodic task
- Nonterminal states: 1, 2, ..., 14
- Terminal state: one, shown in shaded square
- Actions that would take the agent off the grid leave the state unchanged
- Reward is -1 until the terminal state is reached

Iterative Policy Evaluation

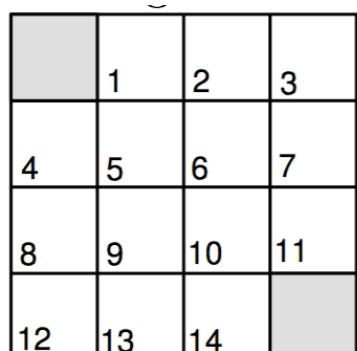
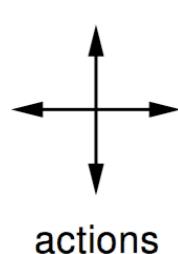
$V[k]$ for the random policy

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

$k = 0$

Policy π , an equiprobable random action

$k = 1$



$k = 2$

$k = 3$

- An undiscounted episodic task
- Nonterminal states: 1, 2, ..., 14
- Terminal state: one, shown in shaded square
- Actions that would take the agent off the grid leave the state unchanged
- Reward is -1 until the terminal state is reached

$k = 10$

$k = \infty$

Iterative Policy Evaluation

$V[k]$ for the random policy

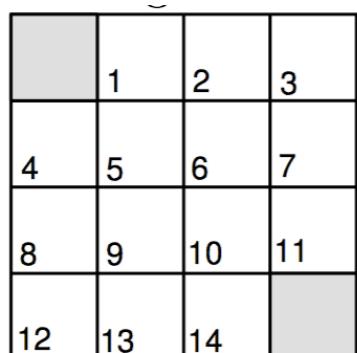
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 0$

Policy π , an equiprobable random action

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

$k = 1$



$k = 2$

- An undiscounted episodic task
- Nonterminal states: 1, 2, ..., 14
- Terminal state: one, shown in shaded square
- Actions that would take the agent off the grid leave the state unchanged
- Reward is -1 until the terminal state is reached

$k = 3$

$k = 10$

$k = \infty$

Iterative Policy Evaluation

$V[k]$ for the random policy

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

$k = 0$

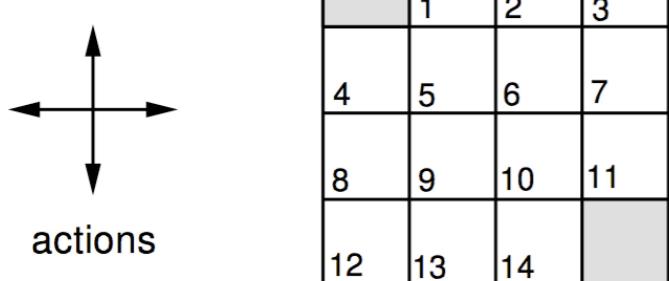
Policy π , an equiprobable random action

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

$k = 1$

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

$k = 2$



$k = 3$

- An undiscounted episodic task
- Nonterminal states: 1, 2, ..., 14
- Terminal state: one, shown in shaded square
- Actions that would take the agent off the grid leave the state unchanged
- Reward is -1 until the terminal state is reached

$k = 10$

$k = \infty$

Iterative Policy Evaluation

$V[k]$ for the random policy

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

$k = 0$

Policy π , an equiprobable random action

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

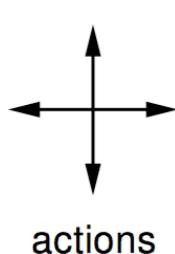
$k = 1$

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

$k = 2$

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

$k = 3$



	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

- An undiscounted episodic task
- Nonterminal states: 1, 2, ..., 14
- Terminal state: one, shown in shaded square
- Actions that would take the agent off the grid leave the state unchanged
- Reward is -1 until the terminal state is reached

$k = 10$

$k = \infty$

Iterative Policy Evaluation

$V[k]$ for the random policy

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

$k = 0$

Policy π , an equiprobable random action

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

$k = 1$

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

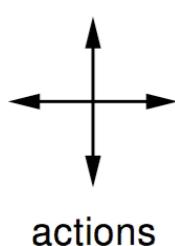
$k = 2$

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

$k = 3$

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

$k = 10$

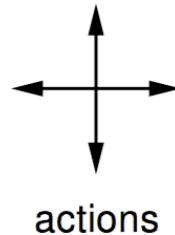


- An undiscounted episodic task
- Nonterminal states: 1, 2, ..., 14
- Terminal state: one, shown in shaded square
- Actions that would take the agent off the grid leave the state unchanged
- Reward is -1 until the terminal state is reached

$k = \infty$

Iterative Policy Evaluation

Policy π , an equiprobable random action



	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

- An undiscounted episodic task
- Nonterminal states: 1, 2, ..., 14
- Terminal state: one, shown in shaded square
- Actions that would take the agent off the grid leave the state unchanged
- Reward is -1 until the terminal state is reached

$V[k]$ for the random policy

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

$k = 0$

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

$k = 1$

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

$k = 2$

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

$k = 3$

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

$k = 10$

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

$k = \infty$

Iterative Policy Evaluation

Input π , the policy to be evaluated

Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat

$$\Delta \leftarrow 0$$

For each $s \in \mathcal{S}$:

$$v \leftarrow V(s)$$

$$V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a) [r + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

until $\Delta < \theta$ (a small positive number)

Output $V \approx v_\pi$

Contraction Mapping Theorem

An operator F on a normed vector space \mathcal{X} is a **γ -contraction**, for $0 < \gamma < 1$, provided for all $x, y \in \mathcal{X}$

$$\|T(x) - T(y)\| \leq \gamma \|x - y\|$$

Theorem (Contraction mapping)

For a γ -contraction F in a complete normed vector space \mathcal{X}

- F converges to a unique fixed point in \mathcal{X}
- at a linear convergence rate γ

Remark. In general we only need metric (vs normed) space

Value Function Space

- Consider the vector space V over value functions
- There are $|\mathcal{S}|$ dimensions
- Each point in this space fully specifies a value function $v(s)$
- Bellman backup brings value functions closer in this space
- And therefore the backup must converge to a unique solution

Value Function ∞ -Norm

- We will measure distance between state-value functions u and v by the ∞ -norm
- i.e. the largest difference between state values,

$$\|u - v\|_\infty = \max_{s \in \mathcal{S}} |u(s) - v(s)|$$

Bellman Expectation Backup is a Contraction

- Define the Bellman expectation backup operator

$$F^\pi(v) = r^\pi + \gamma T^\pi v$$

- This operator is a γ -contraction, i.e. it makes value functions closer by at least γ ,

$$\begin{aligned}\|F^\pi(u) - F^\pi(v)\|_\infty &= \|(r^\pi + \gamma T^\pi u) - (r^\pi + \gamma T^\pi v)\|_\infty \\&= \|\gamma T^\pi(u - v)\|_\infty \\&\leq \|\gamma T^\pi(\mathbf{1}\|(u - v)\|_\infty)\|_\infty \\&= \|\gamma(T^\pi\mathbf{1})\|u - v\|_\infty\|_\infty \\&= \|\gamma\mathbf{1}\|u - v\|_\infty\|_\infty \\&= \gamma\|u - v\|_\infty\end{aligned}$$