MDPs

Finite Markov Decision Processes

- 3.1 The Agent-Environment Interface
- 3.2 Goals and Rewards
- 3.3 Returns and Episodes
- 3.4 Unified Notation for Episodic and Continuing Tasks
- 3.5 Policies and Value Functions
- 3.6 Optimal Policies and Optimal Value Functions
- 3.7 Optimality and Approximation

3.1 The Agent-Environment Interface

- At time step $t \in \mathcal{N}$:
  - State $S_t \in \mathcal{S}$: representation of the environment
  - Action $A_t \in \mathcal{A}(S_t)$: action chosen
  - Reward $R_{t+1} \in \mathcal{R}$: instantaneous reward
  - New state S_{t+1}
- Finite MDP:
  - $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ are finite.
  - Dynamic entirely defined by
$$\mathbb{P}\left(S_t = s', R_r = r | S_{t-1} = s, A_{t-1} = a\right) = p(s', r | s, a)$$

The Agent-Environment Interface

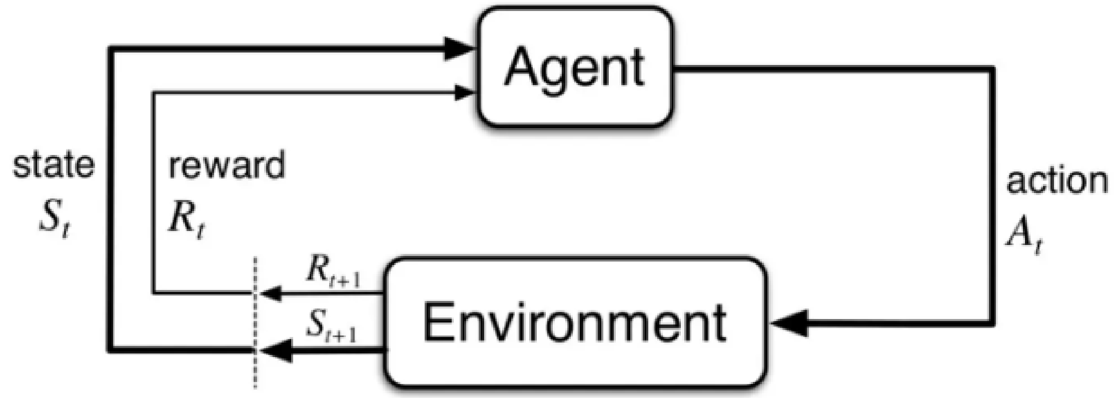


Figure 3.1: The agent–environment interaction in a Markov decision process.

Agent-Environment Interface

- p specifies a probability distribution for each choice of s and a, that is, that

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s).$$

- The state must include information about all aspects of the past agent–environment interaction that make a difference for the future. If it does, then the state is said to have the Markov property.

The Agent-Environment Interface

From the four-argument dynamics function, p, one can compute anything else one might want to know about the environment, such as the *state-transition probabilities* (which we denote, with a slight abuse of notation, as a three-argument function $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$),

- State-transition probabilities:
  $$p(s'|s, a) = \mathbb{P}\left(S_t = s' | S_{t-1} = s, A_{t-1} = a\right) = \sum_r p(s', r|s, a)$$
- Expected reward for a state-action:
  $$r(s, a) = \mathbb{E}\left[R_t | S_{t-1} = s, A_{t-1} = a\right] = \sum_r r \sum_{s'} p(s', r|s, a)$$
- Expected reward for a state-action-state:
  $$r(s, a, s') = \mathbb{E}\left[R_t | S_{t-1} = s, A_{t-1} = a, S_t = s'\right] = \sum_r r \frac{p(s', r|s, a)}{p(s'|s, a)}$$

Agent-Environment Interface

- The MDP framework is abstract and flexible and can be applied to many different problems in many different ways. For example, the time steps need not refer to fixed intervals of real time; they can refer to arbitrary successive stages of decision making and acting.
- Some of what makes up a state could be based on memory of past sensations or even be entirely mental or subjective.
- some actions might be totally mental or computational. For example, some actions might control what an agent chooses to think about, or where it focuses its attention.

Agent-Environment Interface

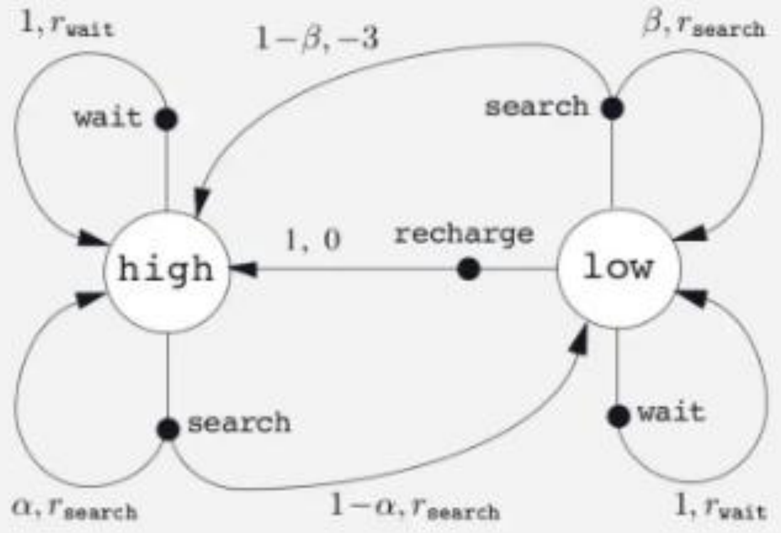
- The boundary between agent and environment is typically not the same as the physical boundary of a robot's or animal's body.
- The general rule we follow is that anything that cannot be changed arbitrarily by the agent is considered to be outside of it and thus part of its environment.
- We do not assume that everything in the environment is unknown to the agent.
- In fact, in some cases the agent may know everything about how its environment works and still face a difficult reinforcement learning task

MDP as Goal Directed Learning

- The agent–environment boundary can be located at di↲erent places for different purposes
- The MDP framework is a considerable abstraction of the problem of goal-directed learning from interaction.
- Goal-directed behavior can be reduced to three signals passing back and forth between an agent and its environment

The Agent-Environment Interface

s	a	s'	$p(s' \mid s,a)$	$r(s,a,s')$
high	search	high	α	$r_{\texttt{search}}$
high	search	low	$1-\alpha$	$r_{\texttt{search}}$
low	search	high	$1-\beta$	-3
low	search	low	β	$r_{\texttt{search}}$
high	wait	high	1	$r_{\texttt{wait}}$
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	$r_{\texttt{wait}}$
low	recharge	high	1	0
low	recharge	low	0	-



- Examples:
  - Bioreactor
  - Pick-and-Place Robots
  - Recycling Robot

Example1

- Bioreactor: (Actions) target temperatures and target stirring rates
- (States) thermocouple and other sensory readings, perhaps filtered and delayed, plus symbolic inputs representing the ingredients in the vat and the target chemical
- (Reward) moment-by-moment measures of the rate at which the useful chemical is produced by the bioreactor

Example 2

- Pick and Place Robots: (Action) voltages applied to each motor at each joint
- (States) latest readings of joint angles and velocities
- (Rewards) +1 for each object successfully picked up and placed

Recycling Robot

- High-level decisions about how to search for cans are made by a reinforcement learning agent based on the current charge level of the battery.
- a small state set S = {high, low}
- Agent can decide whether to (1) actively search for a can for a certain period of time, (2) remain stationary and wait for someone to bring it a can, or (3) head back to its home base to recharge its battery. When the energy level is high, recharging would always be foolish, so we do not include it in the action set for this state. The action sets are then A(high) = {search, wait} and A(low) = {search, wait, recharge}.

3.2 Goals and Rewards

That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).

- The reward signal is your way of communicating to the robot what you want it to achieve, not how you want it achieved.

3.3 Returns ans Episodes

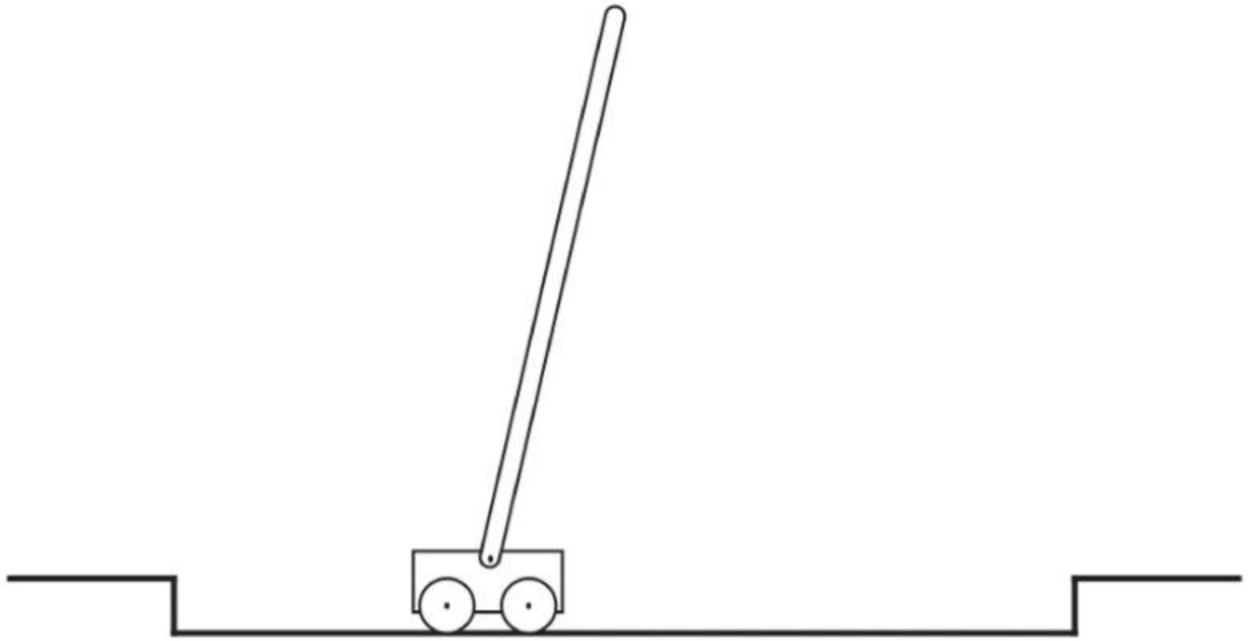
- Episodic: Final time step T and
$$G_t = \sum_{t'=t+1}^{T} R_{t'}$$
- Continuous tasks: undiscounted reward
$$G_t = \sum_{t'=t+1}^{+\infty} R_{t'} \quad \text{may not exist!}$$
- Continuous tasks: discounted reward
$$G_t = \sum_{0}^{+\infty} \gamma^k R_{t+1+k}$$

  with $0 \leq \gamma < 1$.

Returns ans Episodes

- Recursive property
$$G_t = R_{t+1} + \gamma G_{t+1}$$
- Finiteness if $|R| \leq M$
$$|G_t| \leq \begin{cases} (T - t - +1)M & \text{if } T < \infty \\ M\frac{1}{1-\gamma} & \text{otherwise} \end{cases}$$

Returns ans Episodes - Example 3.4



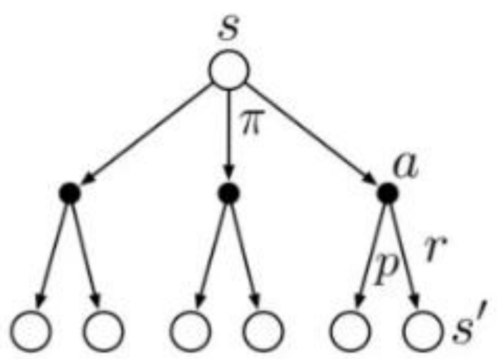
Unified Notation for Episodic and Continuing Tasks

- Episodic case: several episodes instead of a single trajectory $(S_{t,i}\ldots)$
- Absorbing state $\tilde{s}$ such $p(\tilde{s}|\tilde{s},a) = 1$ and $R_t = 0$ when $S_t = \tilde{s}$.
- Convert episodic case into a continuing one.
- Alternative: notation
$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$
- Undefined if $T = \infty$ and $\gamma = 1\ldots$

3.5 Policies and Value Functions

- Policy: $\pi(a|s)$
- Value function:
$$v_\pi(s) = \mathbb{E}_\pi\left[G_t | S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right]$$
- Action value function:
$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}_\pi\left[G_t | S_t = s, A_t = a\right] \\ &= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right] \end{aligned}$$
- *Implicit stationary assumption on* π*!*

3.5 Policies and Value Functions

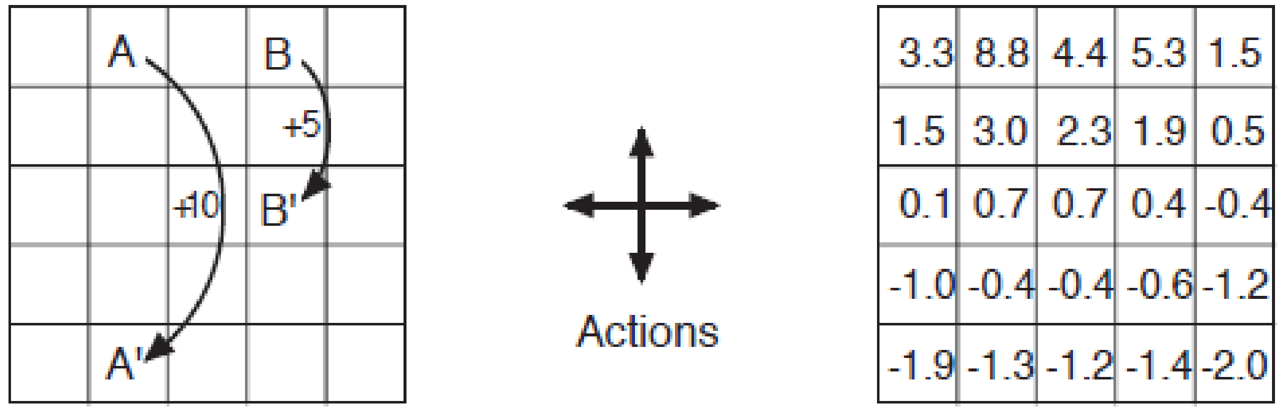


Backup diagram for v_π

- Bellman Equation

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi\left[G_t | S_t = s\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} + \gamma G_{t+1} | S_t = s\right] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a)\left[r + \gamma \mathbb{E}_\pi\left[G_{t+1} | S_{t+1} = s'\right]\right] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a)\left[r + \gamma v_\pi(s')\right]
\end{aligned}
$$

Example 3.5: Gridworld Figure 3.2 (left) shows a rectangular gridworld representation of a simple finite MDP. The cells of the grid correspond to the states of the environment. At each cell, four actions are possible: `north`, `south`, `east`, and `west`, which deterministically cause the agent to move one cell in the respective direction on the grid. Actions that would take the agent off the grid leave its location unchanged, but also result in a reward of −1. Other actions result in a reward of 0, except those that move the agent out of the special states A and B. From state A, all four actions yield a reward of +10 and take the agent to A′. From state B, all actions yield a reward of +5 and take the agent to B′.



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

Figure 3.2: Gridworld example: exceptional reward dynamics (left) and state-value function for the equiprobable random policy (right).

3.5 Policies and Value Functions - Figure 3.3

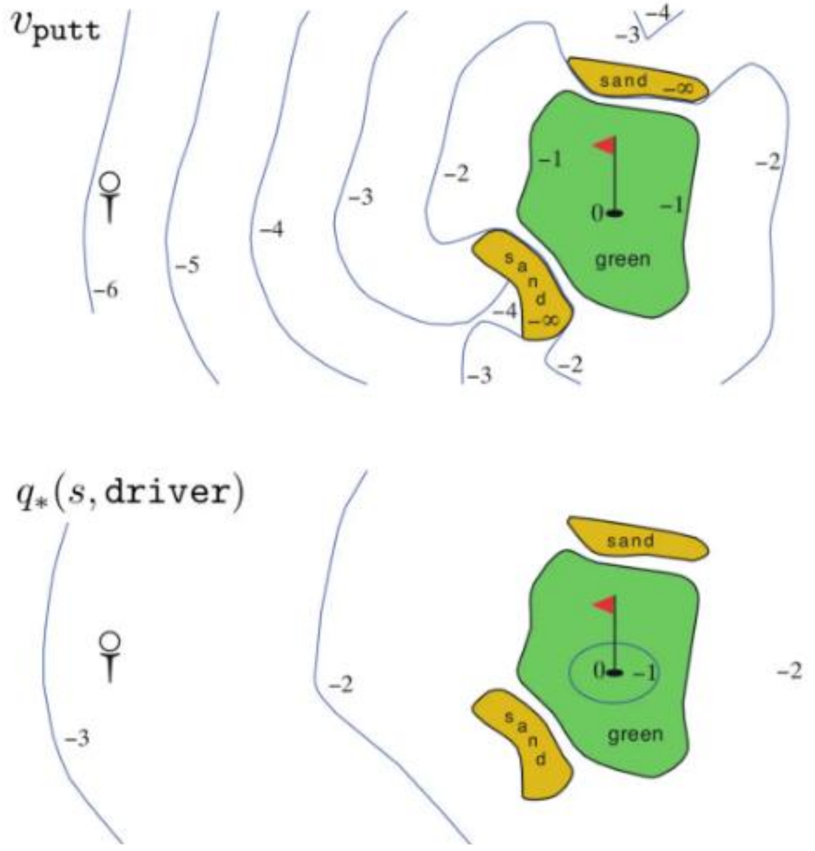


Figure 3.3: A golf example: the state-value function for putting (upper) and the optimal action-value function for using the driver (lower). ■

3.6 Optimal Policies and Optimal Value Functions

- Optimal policies: $v_{\pi_*}(s) \geq v_\pi(s)$ (not necessarily unique)
- Optimal state-value function :

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

- Optimal state-action-value function:

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

- Link:

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a\right]$$

3.6 Optimal Policies and Optimal Value Functions

- Bellman optimality equation:

$$\begin{aligned} v_*(s) &= \max_a q_*(s,a) \\ &= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a\right] \\ &= \max_a \sum_{s',r} p(s',r|s,a)\left(r + \gamma v_*(s')\right) \end{aligned}$$

- Bellman optimality equation for q:

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1},a') \middle| S_t = s, A_t = a\right] \\ &= \sum_{s',r} p(s',r|s,a)\left(r + \max_{a'} \gamma q_*(s',a')\right) \end{aligned}$$

3.6 Optimal Policies and Optimal Value Functions - Figure 3.4

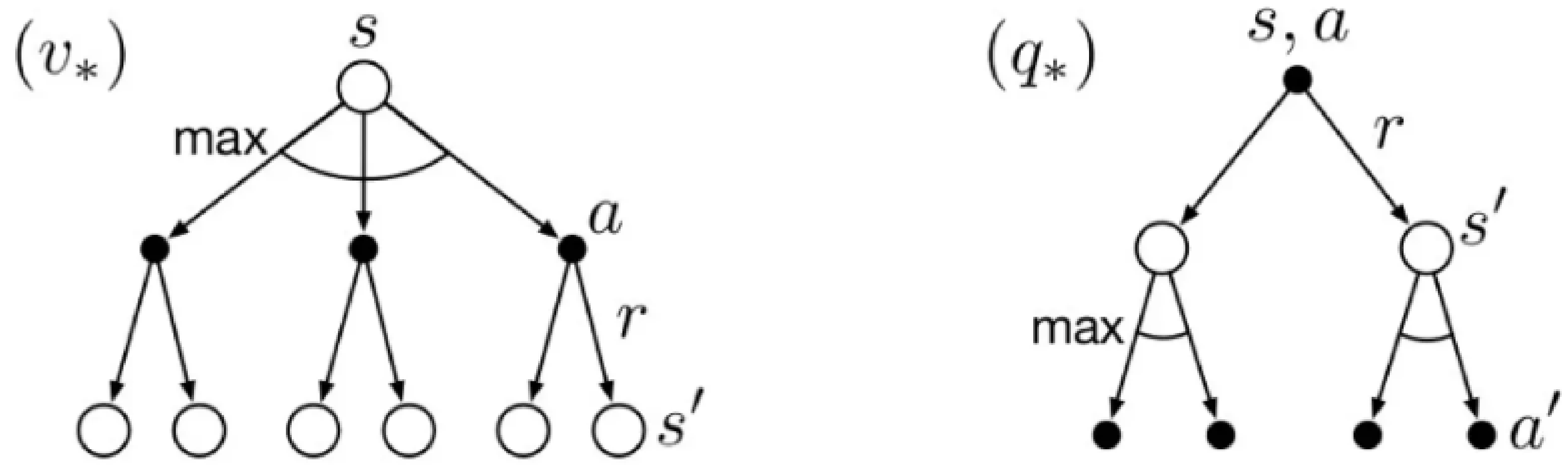
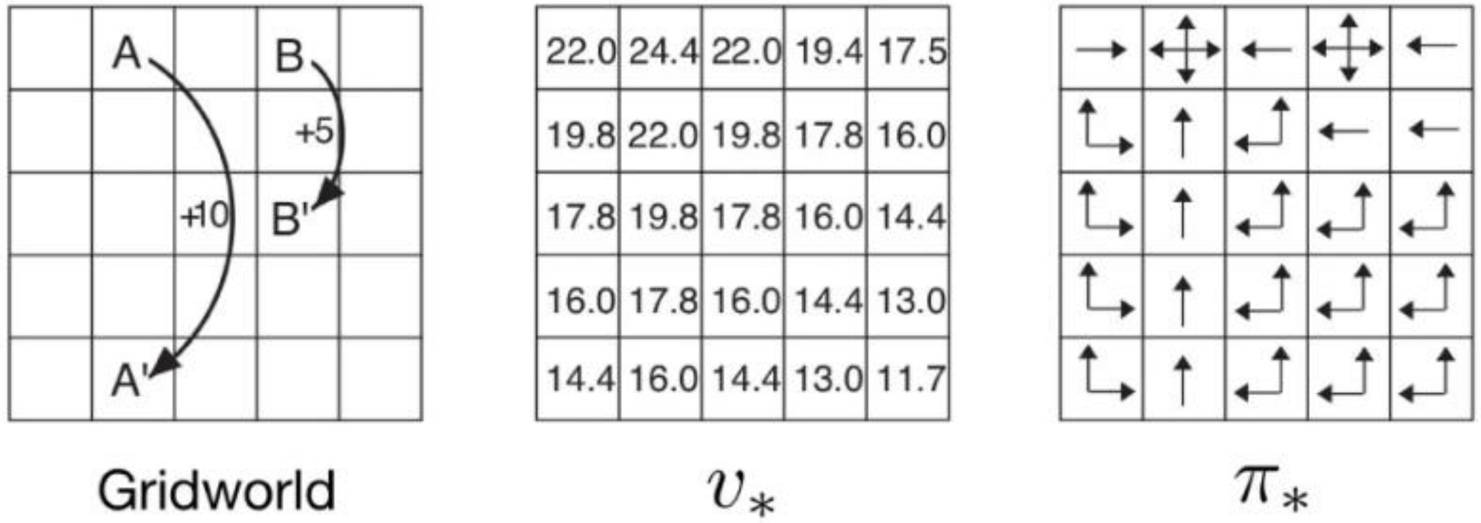


Figure 3.4: Backup diagrams for v_* and q_*

3.6 Optimal Policies and Optimal Value Functions - Figure 3.5



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

Figure 3.5: Optimal solutions to the gridworld example.

3.7 Optimality and Approximation

- Very difficult to learn the optimal policy.
- Knowing the environment helps but is not sufficient. (Chap. 4)
- Computational challenges even in the finite case! (Chap. 5-8)
- Need to resort to approximation! (Chap. 9-12)