Reinforcement learning

Reinforcement learning

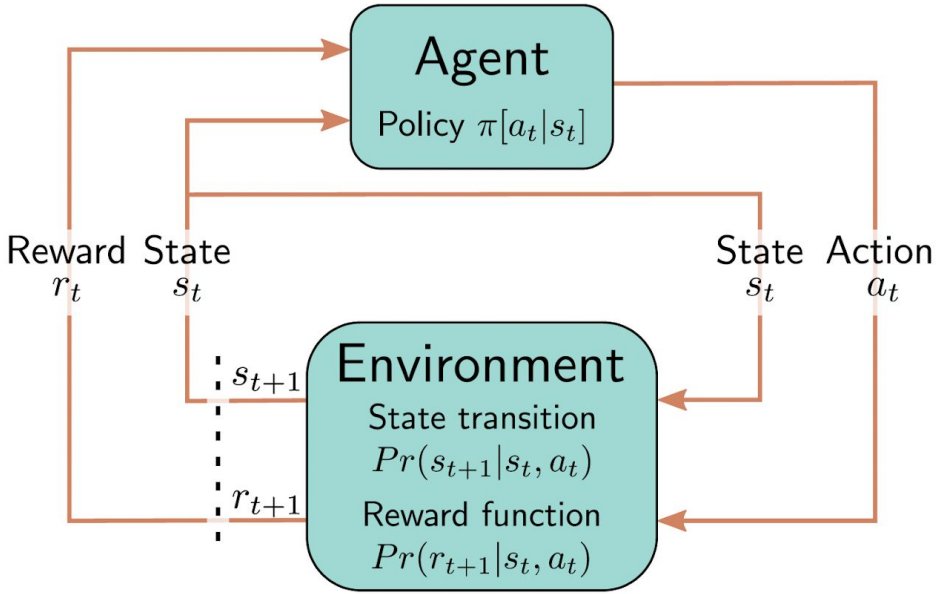
- Reinforcement learning (RL) is concerned with solving sequential decision-making problems.
- Many real-world problems—playing video games, sports, driving, robotic control—can be framed in this way.
- For example we can formulate the game of Tic-tac-toe in RL.

Reinforcement learning

5 Main components

- **Agent, Environment, State, Action, Reward**

Figure 19.6 Reinforcement learning loop. The agent takes an action a_t at time t based on the state s_t, according to the policy $\pi[a_t|s_t]$. This triggers the generation of a new state s_{t+1} (via the state transition function) and a reward r_{t+1} (via the reward function). Both are passed back to the agent, which then chooses a new action.



Trajectory

- In RL, a trajectory is an observed sequence of states, rewards, and actions.
- A rollout is a simulated trajectory. An episode is a trajectory that starts in an initial state and ends in a terminal state.
- E.g., a full game of chess starting from the standard opening position and ending in a win, lose, or draw.

Expected return

- In MDP, an agent carry out actions according to a policy.
- We want to choose a policy that maximizes the expected return.
- The agent will pass through a sequence of states, taking actions and receiving rewards.
- This sequence differs every time the agent starts in the same place since.

State and action values

- We assign a value to each state s_t and state-action pair $\{s_t, a_t\}$.
- The return G_t depends on the state s_t and the policy $\pi[a \mid s]$.
- In general, the policy $\pi\left[a_t \mid s_t\right]$, the state transitions $\Pr\left(s_{t+1} \mid s_t, a_t\right)$, and the rewards issued $\Pr\left(r_{t+1} \mid s_t, a_t\right)$ are all stochastic.
- We can characterize how "good" a state is under a given policy π by considering the expected return $v\left[s_t \mid \pi\right]$.

State and action values

- Expected return is the return that would be received on average from sequences that start from this state and is termed the state value or state-value function (figure 19.7a):

$$v\left[s_t \mid \pi\right] = \mathbb{E}\left[G_t \mid s_t, \pi\right]$$

- Informally, the state value tells us the long-term reward we can expect on average if we start in this state and follow the specified policy thereafter.

- It is highest for states where it's probable that subsequent transitions will bring large rewards soon (assuming the discount factor γ is less than one).

a)

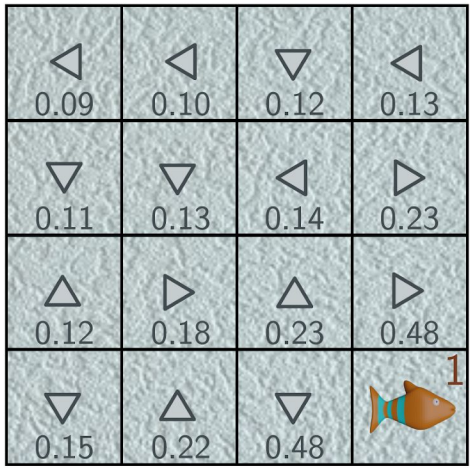


Figure 19.7 State and action values. a) The value $v[s_t|\pi]$ of a state s_t (number at each position) is the expected return for this state for a given policy π (gray arrows). It is the average sum of discounted rewards received over many trajectories started from this state. Here, states closer to the fish are more valuable. b) The

State and action values

- Similarly, the action value or state-action value function $q[s_t, a_t \mid \pi]$ is the expected return from executing action a_t in state s_t (figure 19.7b)

$$q[s_t, a_t \mid \pi] = \mathbb{E}[G_t \mid s_t, a_t, \pi].$$

- The action value tells us the long-term reward we can expect on average if we start in this state, take this action, and follow the specified policy thereafter.
- Through action value, RL algorithms connect future rewards to current actions.

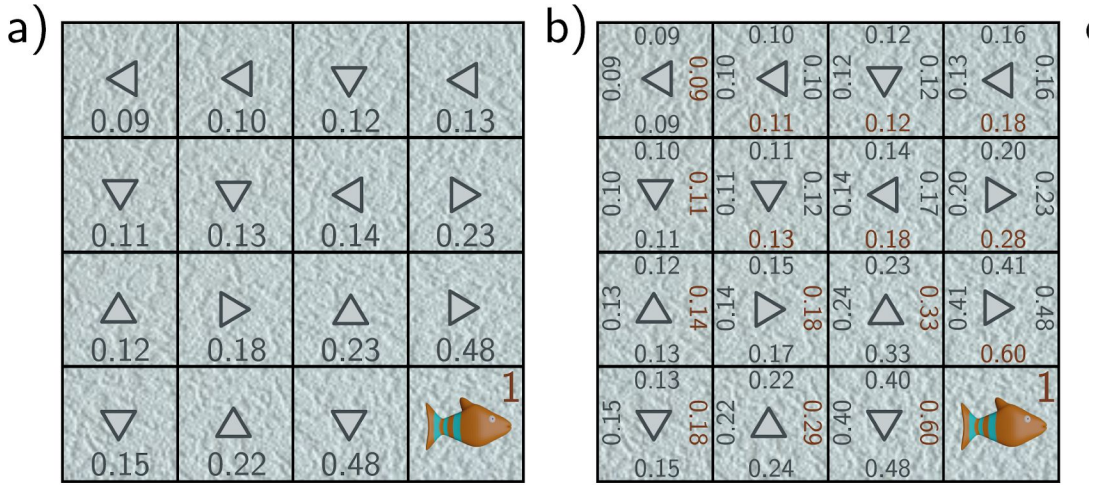


Figure 19.7 State and action values. a) The value $v[s_t|\pi]$ of a state s_t (number at each position) is the expected return for this state for a given policy π (gray arrows). It is the average sum of discounted rewards received over many trajectories started from this state. Here, states closer to the fish are more valuable. b) The value $q[s_t, a_t, \pi]$ of an action a_t in state s_t (four numbers at each position/state corresponding to four actions) is the expected return given that this particular action is taken in this state. In this case, it gets larger as we get closer to the fish and is larger for actions that head in the direction of the fish. c) If we know the

Optimal policy

- We want a policy that maximizes the expected return.
- For MDPs (but not POMDPs), there is always a deterministic, stationary policy that maximizes the value of every state.
- The optimal state-value function $v^* [s_t]$ given optimal policy is:

$$v^* [s_t] = \max_{\pi} [\mathbb{E} [G_t \mid s_t, \pi]] .$$

Optimal policy

- We want a policy that maximizes the expected return.
- For MDPs (but not POMDPs), there is always a deterministic, stationary policy that maximizes the value of every state.
- The optimal state-value function $v^*[s_t]$ given optimal policy is:

$$v^*[s_t] = \max_{\pi} [\mathbb{E}[G_t \mid s_t, \pi]].$$

- Similarly, given optimal policy, optimal state-action value function is:

$$q^*[s_t, a_t] = \max_{\pi} [\mathbb{E}[G_t \mid s_t, a_t, \pi]].$$

Optimal policy

- If we knew the optimal action-values $q^*[s_t, a_t]$, then we can derive the optimal policy by choosing the action a_t with the highest value as (figure 19.7c)

$$\pi[a_t \mid s_t] \leftarrow \underset{a_t}{\text{argmax}}\left[q^*[s_t, a_t]\right].$$

- The notation $\pi[a_t \mid s_t] \leftarrow a$ in equations means set $\pi[a_t \mid s]$ to one for action a and $\pi[a_t \mid s]$ to zero for other actions.

- Indeed, some reinforcement learning algorithms are based on alternately estimating the action values and the policy (see section 19.3).

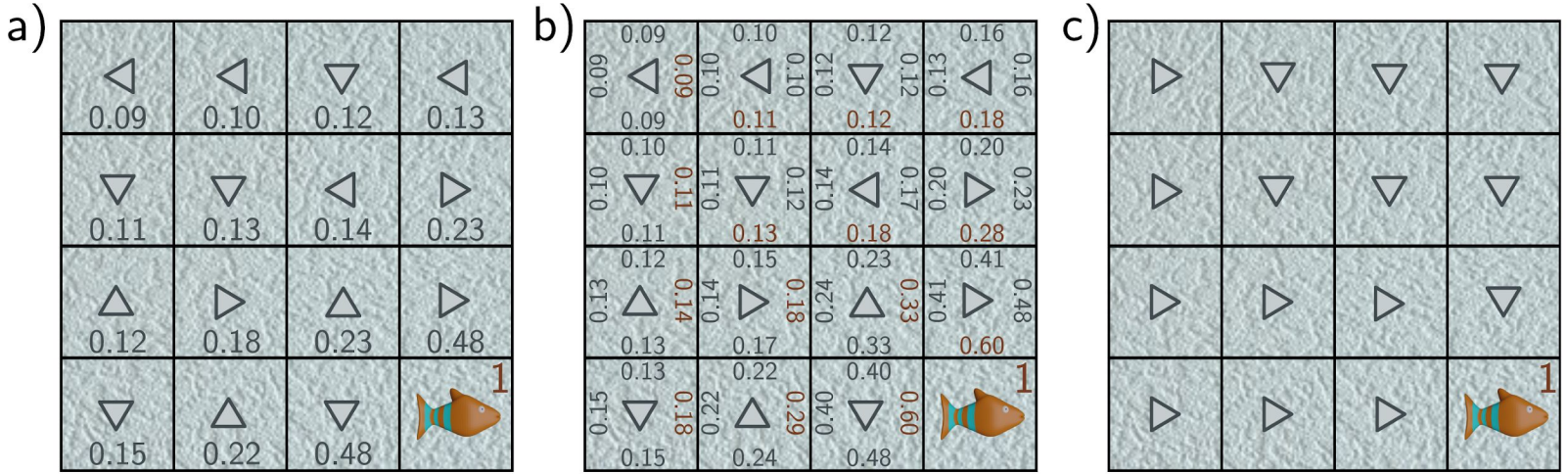


Figure 19.7 State and action values. a) The value $v[s_t|\pi]$ of a state s_t (number at each position) is the expected return for this state for a given policy π (gray arrows). It is the average sum of discounted rewards received over many trajectories started from this state. Here, states closer to the fish are more valuable. b) The value $q[s_t, a_t, \pi]$ of an action a_t in state s_t (four numbers at each position/state corresponding to four actions) is the expected return given that this particular action is taken in this state. In this case, it gets larger as we get closer to the fish and is larger for actions that head in the direction of the fish. c) If we know the action values at a state, then the policy can be modified so that it chooses the maximum of these values (red numbers in panel b).

Bellman equations

- For simplicity, lets write $v\left[s_t\right]$ and $q\left[s_t, a_t\right]$ instead of $v\left[s_t \mid \pi\right]$ and $q\left[s_t, a_t \mid \pi\right]$.
- We may not know the state values $v\left[s_t\right]$ or action values $q\left[s_t, a_t\right]$ for any policy.
- However, we know that they must be consistent with one another (figure 19.8). That is:

$$v\left[s_t\right] = \sum_{a_t} \pi\left[a_t \mid s_t\right] q\left[s_t, a_t\right]$$

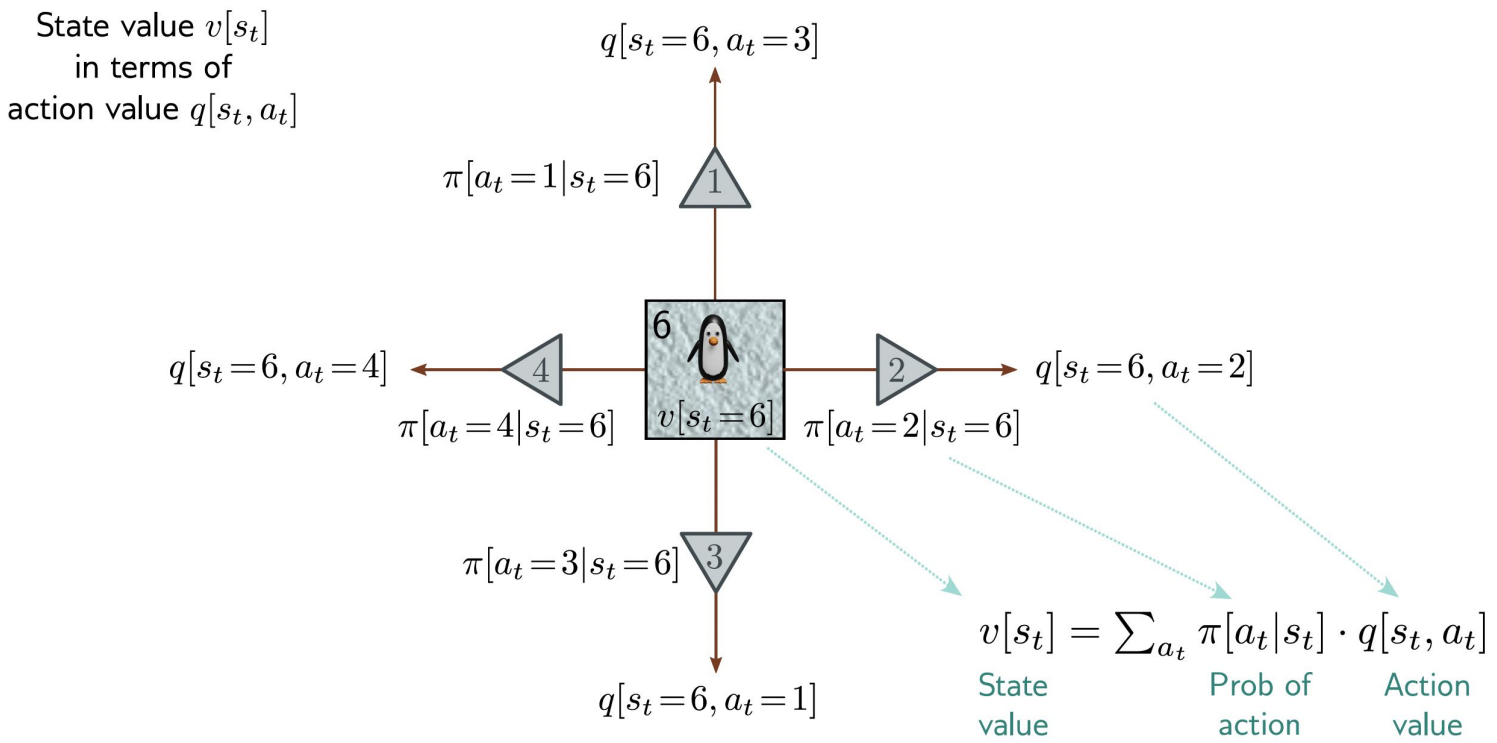


Figure 19.8 Relationship between state values and action values. The value of state six $v[s_t=6]$ is a weighted sum of the action values $q[s_t=6, a_t]$ at state six, where the weights are the policy probabilities $\pi[a_t|s_t=6]$ of taking that action.

Bellman equations

- Similarly, the value of an action can be written as (figure 19.9)

$$q\left[s_t, a_t\right] = r\left[s_t, a_t\right] + \gamma \cdot \sum_{s_{t+1}} \Pr\left(s_{t+1} \mid s_t, a_t\right) v\left[s_{t+1}\right].$$

- Here, we assume rewards are deterministic and can be written as $r\left[s_t, a_t\right]$.
- Note that the assignment of s_{t+1} is not deterministic. Therefore, we weight the values $v\left[s_{t+1}\right]$ according to the transition probabilities in above equation.

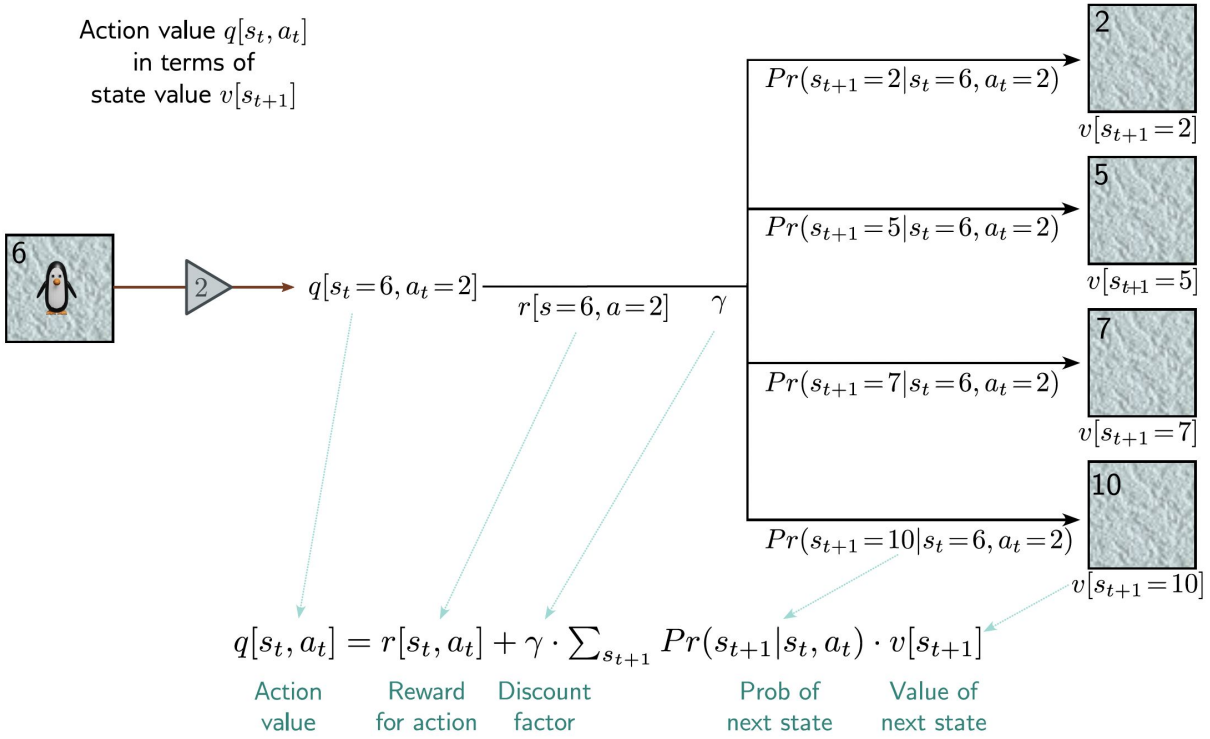


Figure 19.9 Relationship between action values and state values. The value $q[s_t = 6, a_t = 2]$ of taking action two in state six is the reward $r[s_t = 6, a_t = 2]$ from taking that action plus a weighted sum of the discounted values $v[s_{t+1}]$ of being in successor states, where the weights are the transition probabilities $Pr(s_{t+1}|s_t = 6, a_t = 2)$. The Bellman equations chain this relation with that of figure 19.8 to link the current and next (i) state values and (ii) action values.

Bellman equations

- Substituting for $q\left[s_t, a_t\right]$ in equation of $v\left[s_t\right]$, provides a relation between the state value at time t and $t+1$:

$$v\left[s_t\right] = \sum_{a_t} \pi\left[a_t \mid s_t\right] \left(r\left[s_t, a_t\right] + \gamma \cdot \sum_{s_{t+1}} \Pr\left(s_{t+1} \mid s_t, a_t\right) v\left[s_{t+1}\right] \right).$$

Bellman equations

- Substituting for $q[s_t, a_t]$ in equation of $v[s_t]$, provides a relation between the state value at time t and $t+1$:

$$v[s_t] = \sum_{a_t} \pi[a_t \mid s_t] \left(r[s_t, a_t] + \gamma \cdot \sum_{s_{t+1}} \Pr(s_{t+1} \mid s_t, a_t)\, v[s_{t+1}] \right).$$

- Similarly, substituting $v[s_t]$ into equation of $q[s_t, a_t]$ provides a relation between the action value at time t and $t+1$:

$$q[s_t, a_t] = r[s_t, a_t] + \gamma \cdot \sum_{s_{t+1}} \Pr(s_{t+1} \mid s_t, a_t) \left(\sum_{a_{t+1}} \pi[a_{t+1} \mid s_{t+1}]\, q[s_{t+1}, a_{t+1}] \right).$$

Bellman equations

$$v\left[s_t\right] = \sum_{a_t} \pi\left[a_t \mid s_t\right] \left(r\left[s_t, a_t\right] + \gamma \cdot \sum_{s_{t+1}} \Pr\left(s_{t+1} \mid s_t, a_t\right) v\left[s_{t+1}\right] \right).$$

$$q\left[s_t, a_t\right] = r\left[s_t, a_t\right] + \gamma \cdot \sum_{s_{t+1}} \Pr\left(s_{t+1} \mid s_t, a_t\right) \left(\sum_{a_{t+1}} \pi\left[a_{t+1} \mid s_{t+1}\right] q\left[s_{t+1}, a_{t+1}\right] \right).$$

- Above two equations are Bellman equations (backbone of many RL methods). They say that the state (action) values have to be self-consistent.

Bellman equations

$$v\left[s_t\right] = \sum_{a_t} \pi\left[a_t \mid s_t\right] \left(r\left[s_t, a_t\right] + \gamma \cdot \sum_{s_{t+1}} \Pr\left(s_{t+1} \mid s_t, a_t\right) v\left[s_{t+1}\right] \right).$$

$$q\left[s_t, a_t\right] = r\left[s_t, a_t\right] + \gamma \cdot \sum_{s_{t+1}} \Pr\left(s_{t+1} \mid s_t, a_t\right) \left(\sum_{a_{t+1}} \pi\left[a_{t+1} \mid s_{t+1}\right] q\left[s_{t+1}, a_{t+1}\right] \right).$$

- Above two equations are Bellman equations (backbone of many RL methods). They say that the state (action) values have to be self-consistent.
- Bellman equation helps us formulate the relationship between the value of a state or state-action pair and the expected cumulative reward.
- Note that when we update an estimate of one state (action) value, this will have a ripple effect that causes modifications to all the others.

Tabular reinforcement learning

- In Tabular RL algorithms, the state and action spaces are relatively small and can be represented in a tabular form
- Tabular RL algorithms don't rely on function approximation and divided into: model-based and model-free methods.
- Model-based methods: learning a transition model that predicts the next state and reward for each state-action pair.
- Model-free methods: estimating the policy or value function directly without building an explicit model of the environment.

Model-based methods

- Model-based methods use the MDP structure explicitly and find the best policy from the transition matrix $\Pr(s_{t+1} \mid s_t, a_t)$ and reward structure $r[s, a]$.
- If these are known, this is a straightforward optimization problem that can be tackled using dynamic programming.
- If they are unknown, they must first be estimated from observed MDP trajectories.
- Dynamic programming methods assume complete knowledge of the environment and are used for small-scale problems with discrete state and action spaces. Example, Policy Iteration and Value Iteration.

Dynamic programming (recall)

- It assume we have perfect knowledge of the transition and reward structure.
- It is difference from most RL algorithms that observe agent interacting with the environment to gather information about these quantities indirectly.
- The state values $v[s]$ are initialized arbitrarily (usually to zero).
- The deterministic policy $\pi[a \mid s]$ is also initialized (e.g., by choosing a random action for each state).
- The algorithm then alternates between iteratively computing the state values for the current policy (policy evaluation) and improving that policy (policy improvement).

Monte Carlo methods

- Unlike dynamic programming algorithms, Monte Carlo methods don't assume knowledge of the MDP's transition probabilities and reward structure.
- They gain experience by repeatedly sampling trajectories from the MDP and observing the rewards.
- They alternate between computing the action values (based on this experience) and updating the policy (based on the action values).

Monte Carlo methods

- To estimate the action values $q[s, a]$, a series of episodes are run.
- Each starts with a given state and action and thereafter follows the current policy, producing a series of actions, states, and returns (figure 19.11a).
- The action value for a given state-action pair under the current policy is estimated as the average of the empirical returns that follow after each time this pair is observed (figure 19.11b).
- Then the policy is updated by choosing the action with the maximum value at every state (figure 19.11c):

$$\pi[a \mid s] \leftarrow \underset{a}{\text{argmax}}[q[s, a]].$$

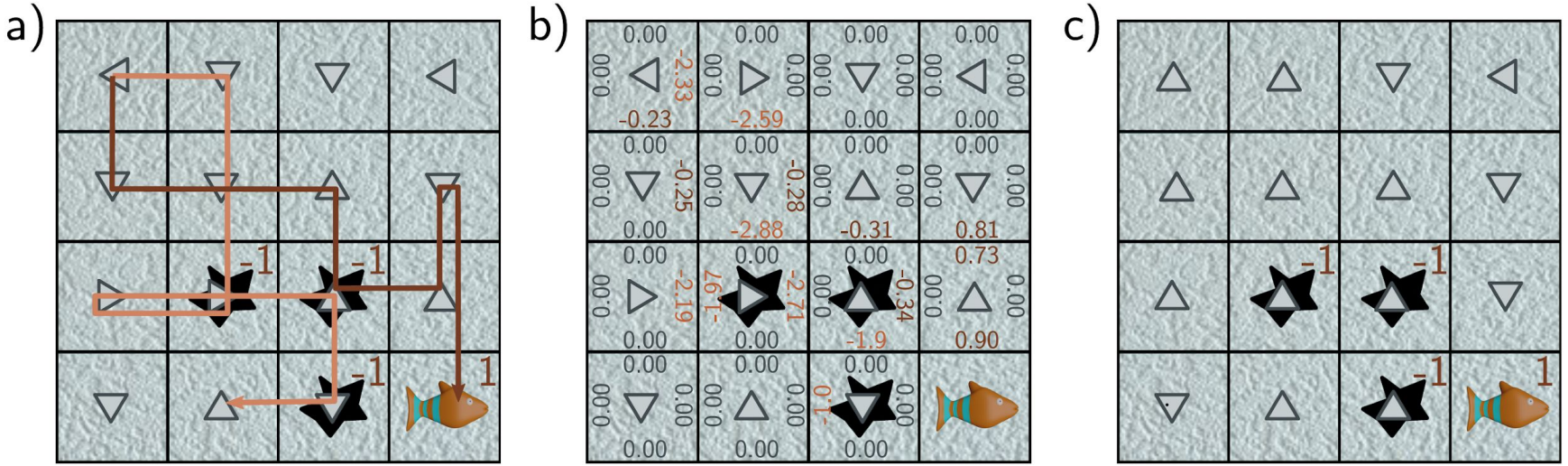


Figure 19.11 Monte Carlo methods. a) The policy (arrows) is initialized randomly. The MDP is repeatedly simulated, and the trajectories of these episodes are stored (orange and brown paths represent two trajectories). b) The action values are empirically estimated based on the observed returns averaged over these trajectories. In this case, the action values were all initially zero and have been updated where an action was observed. c) The policy can then be updated according to the action which received the best (or least bad) reward.