CS 5/7320
Artificial Intelligence

Reinforcement Learning
AIMA Chapter 17+22

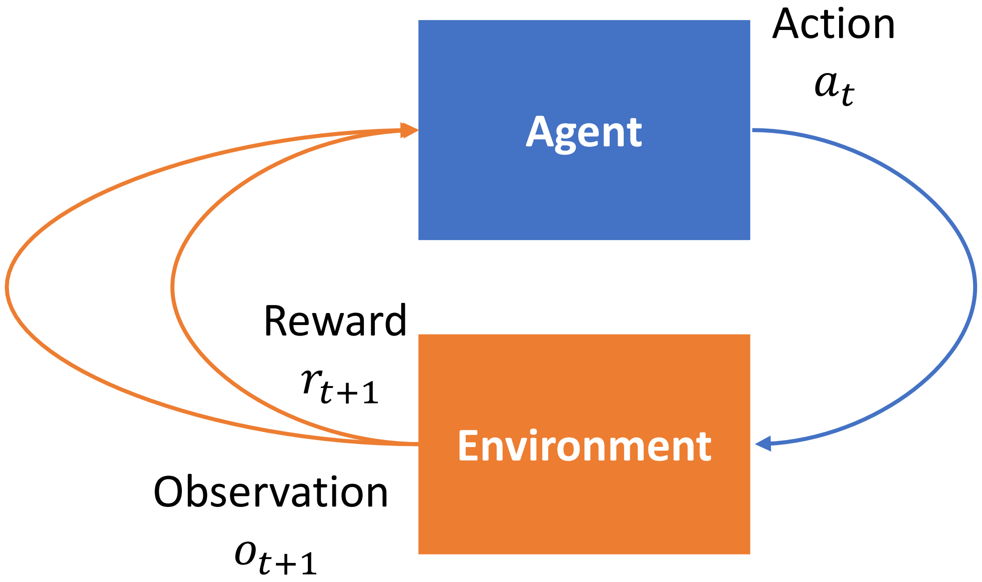
Slides by Michael Hahsler
with figures from the AIMA textbook.

Sequential Decision Problems

AIMA Chapter 17: Making Complex Decisions

Sequential Decision Problems

- **Utility-based agent**: The agent's utility depends on a sequence of decisions.
- Sequential decision problems incorporate utilities, uncertainty, and sensing.



Sequence: $r_0, o_0, a_0, r_1, o_1, a_1, r_2, o_2, a_2,, \ldots$

Observation and reward depend on the state of the system and the agent wants to maximize (discounted) expected reward over time

$$U = E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Markov Decision Process (MDP)

- Fully observable environment: The agent's observation is the state $o_t = s_t$.
- A MDP defines a sequential decision problem with
  - a finite set of states S (initial state s_0)
  - a set actions $ACTIONS(s)$ in each state s of actions
  - a transition model $P(s' \mid s, a)$ where $a \in ACTIONS(s)$
  - a reward function $R(s)$

- The goal is to find an **optimal policy** $\boldsymbol{\pi}^*$ that prescribes for each state the optimal action $\pi(s)$ to maximize the expected utility over time.

Example: 4x3 Grid World

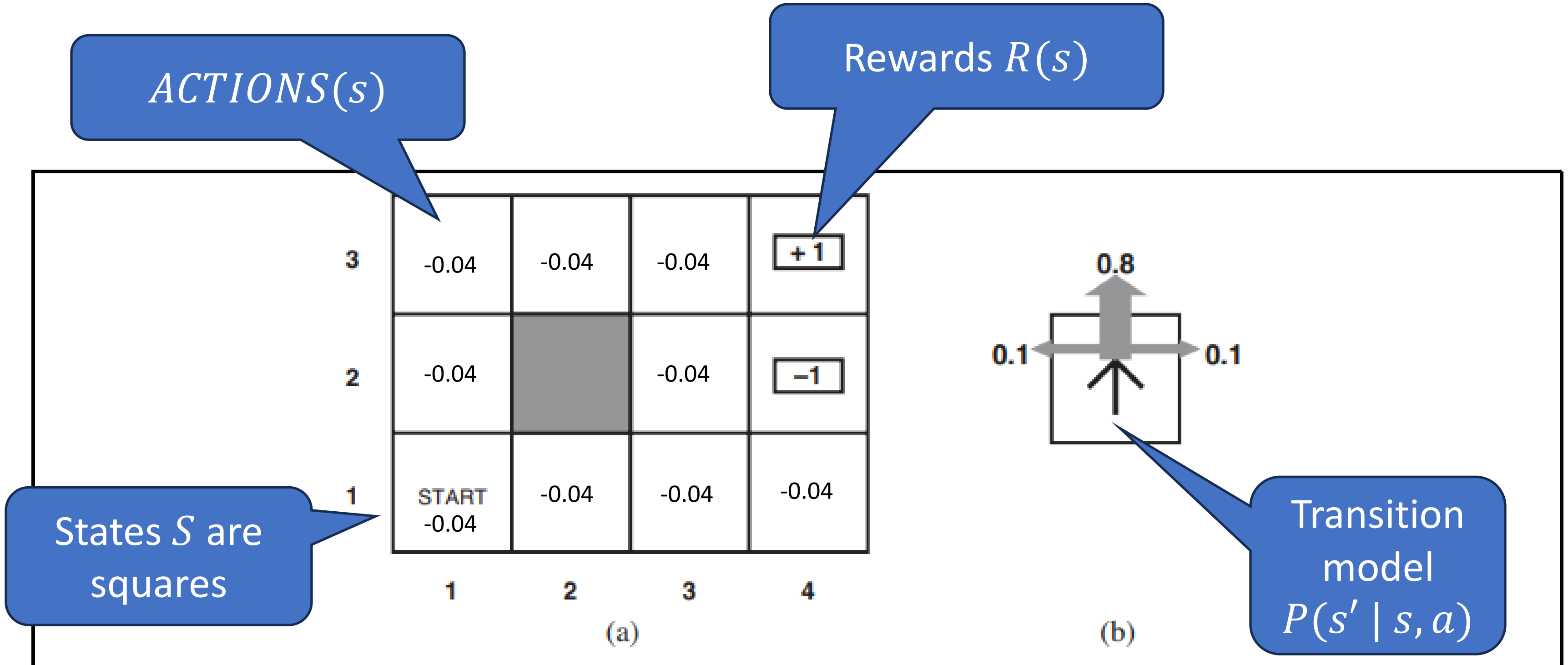


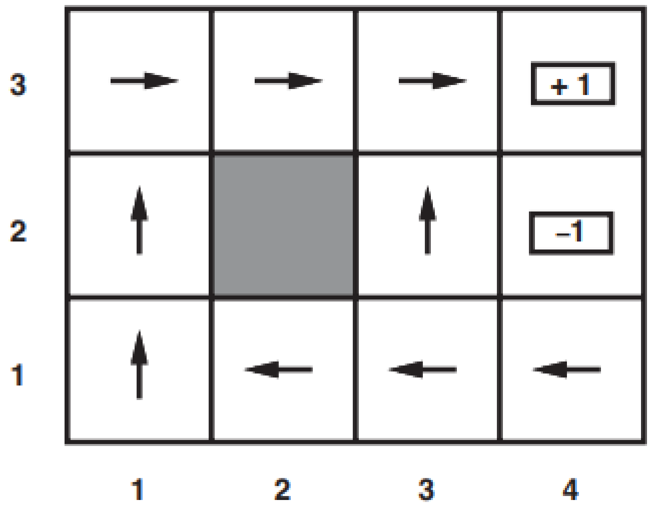
Figure 17.1 (a) A simple 4×3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. The two terminal states have reward +1 and –1, respectively, and all other states have a reward of –0.04.

Goal: What direction should we go in each square?

$$\pi(s)$$

Solution: 4x3 Grid World

Optimal action in each state (policy π^*)



Always move to higher utility states

Value of being in a state $U(s)$ (given that we will follow π^*)

	1	2	3	4
3	0.812	0.868	0.918	+1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388

$\gamma = 1$

Question: How to we find the optimal value function/optimal policy?

Value Iteration

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
rewards $R(s)$, discount γ
ϵ, the maximum error allowed in the utility of any state
local variables: U, U', vectors of utilities for states in S, initially zero
δ, the maximum change in the utility of any state in an iteration

repeat
$U \leftarrow U'$; $\delta \leftarrow 0$
for each state s **in** S **do**
$$U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']$$
if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
until $\delta < \epsilon(1-\gamma)/\gamma$
return U

Bellman update

U converges to U^{π^*}

Policy Iteration

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      if $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s'] > \sum_{s'} P(s' \mid s, \pi[s])\ U[s']$ **then do**
        $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
        $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

Calculate U given current policy
(eighter solve an LP or iterative solution)

Policy
Improvement

U converges to U^{π^*}
and π converges to π^*

Partially Observable Markov Decision Model (POMDP)

- If the environment is partially observable then the model is expanded by
  - a sensor model $P(o \mid s)$ for receiving observation o given being in state s.

- This makes things a lot more complicated and we have to work with **belief states**. A belief state is a distribution over states.
  Example: For a problem with three states, the belief state $b \; =< .2, .8, 0 >$ means the agent beliefs that is 20% in state 1 and 80% in state 2.

- This leads to a **belief MDP** that has an infinite number of states (the belief states).
- The solution of a POMDP is a policy with the optimal action for a set of belief states.
- For all but tiny problems, POMDPs can only be solved **approximately**.

Reinforcement Learning

AIMA Chapter 22

Reinforcement Learning

- The basis of reinforcement learning are MDPs.

- What if we do not have a transition model $P(s' \mid s, a)$?
- Now we cannot solve the MDP (estimate the state utility function/policy) because we cannot predict future states!

- The agent needs to explore (try actions) and use the reward signal to update its belief about the utility of states and actions (i.e., this is also called learning or estimation)

Q-Learning

- Q-Learning learns the state-action value function $Q(s, a)$.
- Relationship with the state value function:

$$U(s) = \max_a Q(s, a) \ .$$

function Q-LEARNING-AGENT(*percept*) **returns** an action
  inputs: *percept*, a percept indicating the current state s' and reward signal r'
  persistent: Q, a table of action values indexed by state and action, initially zero
    N_{sa}, a table of frequencies for state–action pairs, initially zero
    s, a, r, the previous state, action, and reward, initially null

  if TERMINAL?(s) **then** $Q[s, None] \leftarrow r'$
  if s is not null **then**
    increment $N_{sa}[s, a]$
    $Q[s, a] \leftarrow Q[s, a] + \alpha(N_{sa}[s, a])(r + \gamma \max_{a'} Q[s', a'] - Q[s, a])$
  $s, a, r \leftarrow s', \text{argmax}_{a'} f(Q[s', a'], N_{sa}[s', a']), r'$
  return a

Make $Q[s, a]$ a little more similar to the received reward + the best Q-value of the successor state.

f is the exploration function and decides on the next action. As N increases it can exploit good actions more.

Function Approximation

- U (Q) needs to store and estimate one entry for each state (state/action combination)!
- Issues and solutions
  - Too many entries to store → lossy compression
  - Many combinations are rarely seen → generalize to unseen entries

- **Idea**: Estimate the state value using a function approximator $\hat{U}(s) = g_\theta(s)$ that learns g_{theta} based on features of s.

- 4x3 Grid World Example: Use a linear combination of state features (x, y) and learn $\boldsymbol{\theta}$ from observed data.

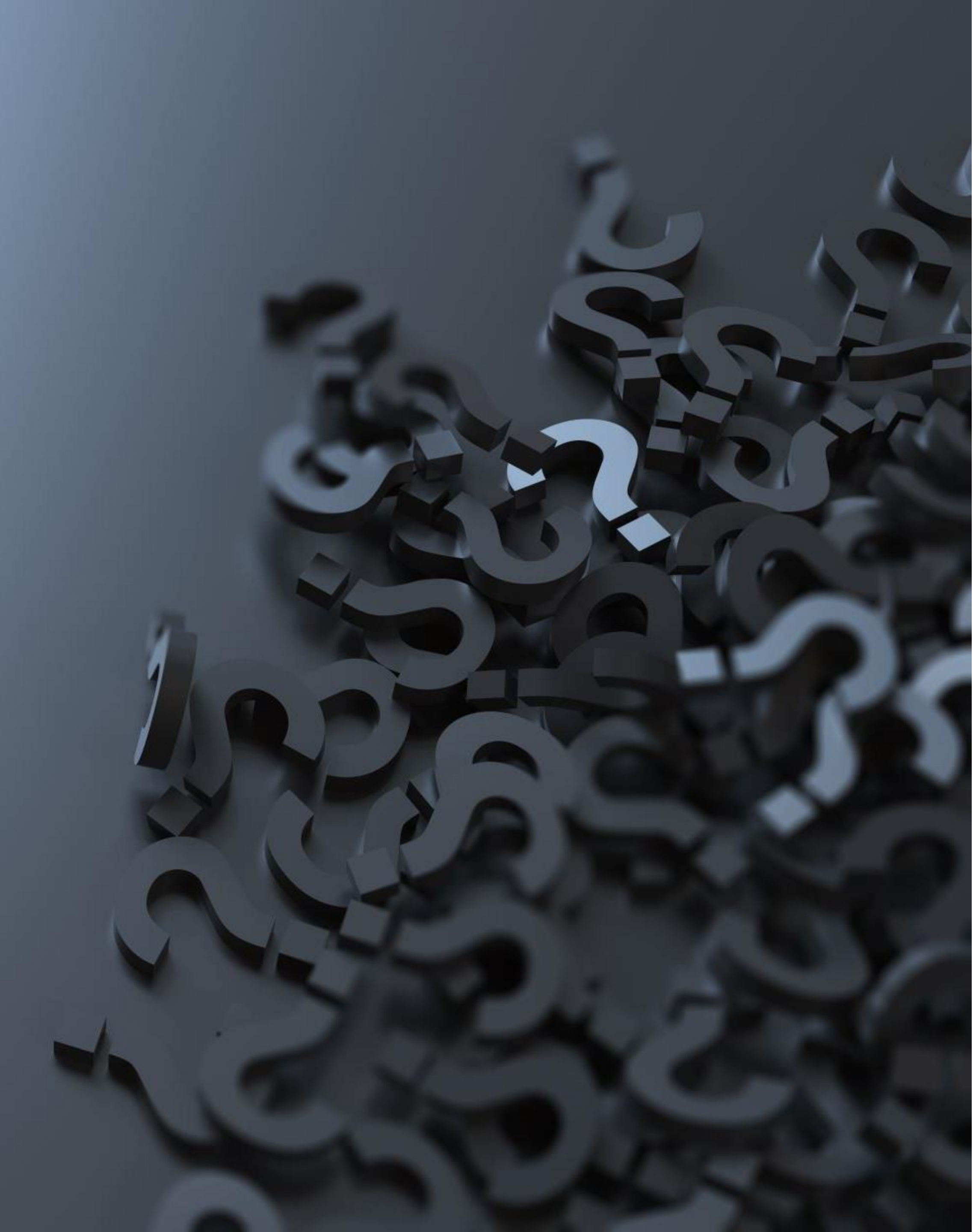
$$\hat{U}_{theta}(x, y) = \theta_0 + \theta_1 x + \theta_2 y$$

Learn $\boldsymbol{\theta}$ from observed interactions with the environment to approximate $U(s)$

$U(s)$

	1	2	3	4
3	0.812	0.868	0.918	+1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388

Notes:
We can also approximate the state-value function Q.
We typically need non-linear approximators that can be incrementally updated (online learning). → Deep ANNs



Summary

- Agents can learn the value of being in a state from **reward signals**.
- Rewards can be delayed (e.g., at the end of a game).
- Not being able to fully **observe the state** makes the problem more difficult (POMDP).
- **Unknown transition models** lead to the need of exploration by trying actions (model free methods like Q-Learning).
- All these problems are computationally very expensive and often can only be solved by **approximation**.
- All functions (U, Q, etc.) can be approximated. State of the art is to use deep artificial neural networks.