reinforcement learning

A supervised learning agent needs to be told the correct move for each position it encounters, but such feedback is seldom available.

In the absence of feedback from a teacher, an agent can learn a transition model for its own moves and can perhaps learn to predict the opponent's moves, but *without some feedback about what is good and what is bad, the agent will have no grounds for deciding which move to make.*

The agent needs to know that something good has happened when it (accidentally) checkmates the opponent, and that something bad has happened when it is checkmated—or vice versa, if the game is suicide chess. This kind of feedback is called a **reward**, or **reinforcement**.

In games like chess, the reinforcement is received only at the end of the game

Animals seem to be hardwired to recognize pain and hunger as negative rewards and pleasure and food intake as positive rewards. Reinforcement has been carefullystudied by animal psychologists for over 60 years

An optimal policy is a policy that maximizes the expected total reward. The task of **reinforcement learning** is to use observed rewards to learn an optimal (or nearly optimal) policy for the environment

In many complex domains, reinforcement learning is the only feasible way to train a program to perform at high levels. For example, in game playing, it is very hard for a human to provide accurate and consistent evaluations of large numbers of positions, which would be needed to train an evaluation function directly from examples. we will allow for probabilistic action outcomes.

Thus, the agent faces an unknown Markov decision process. We will consider three of the agent designs

- A **utility-based agent** learns a utility function on states and uses it to select actions that maximize the expected outcome utility.
- A **Q-learning** agent learns an **action-utility function**, or **Q-function**, giving the expected utility of taking a given action in a given state.
- A **reflex agent** learns a policy that maps directly from states to actions.

passive learning, where the agent's policy is fixed and the task is to learn the utilities of states (or state–action pairs)

active learning, where the agent must also learn what to do

The principal issue is **exploration**: an agent must experience as much aspossible of its environment in order to learn how to behave in it.

PASSIVE REINFORCEMENT LEARNING

we start with the case of a passive learning agent using a state-based representation in a fully observable environment. In passive learning, the agent's policy π is fixed: in state s, it always executes the action $\pi(s)$.

It s goal is simply to learn how good the policy is-that is, to learn the utility function $U^{\pi}(s)$.

We will use as our example the 4×3 world introduced in Chapter 17. Figure 21.1 shows a policy for that world and the corresponding utilities.

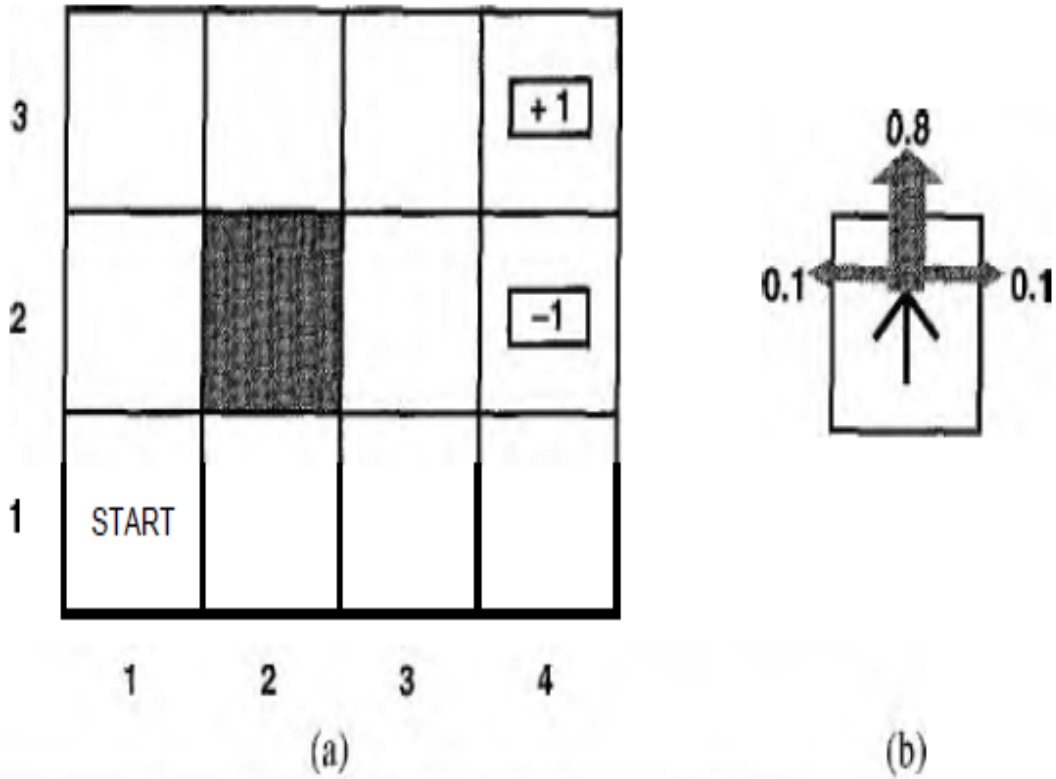

Figure 17.1 (a) A simple 4 x **3** environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. 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.

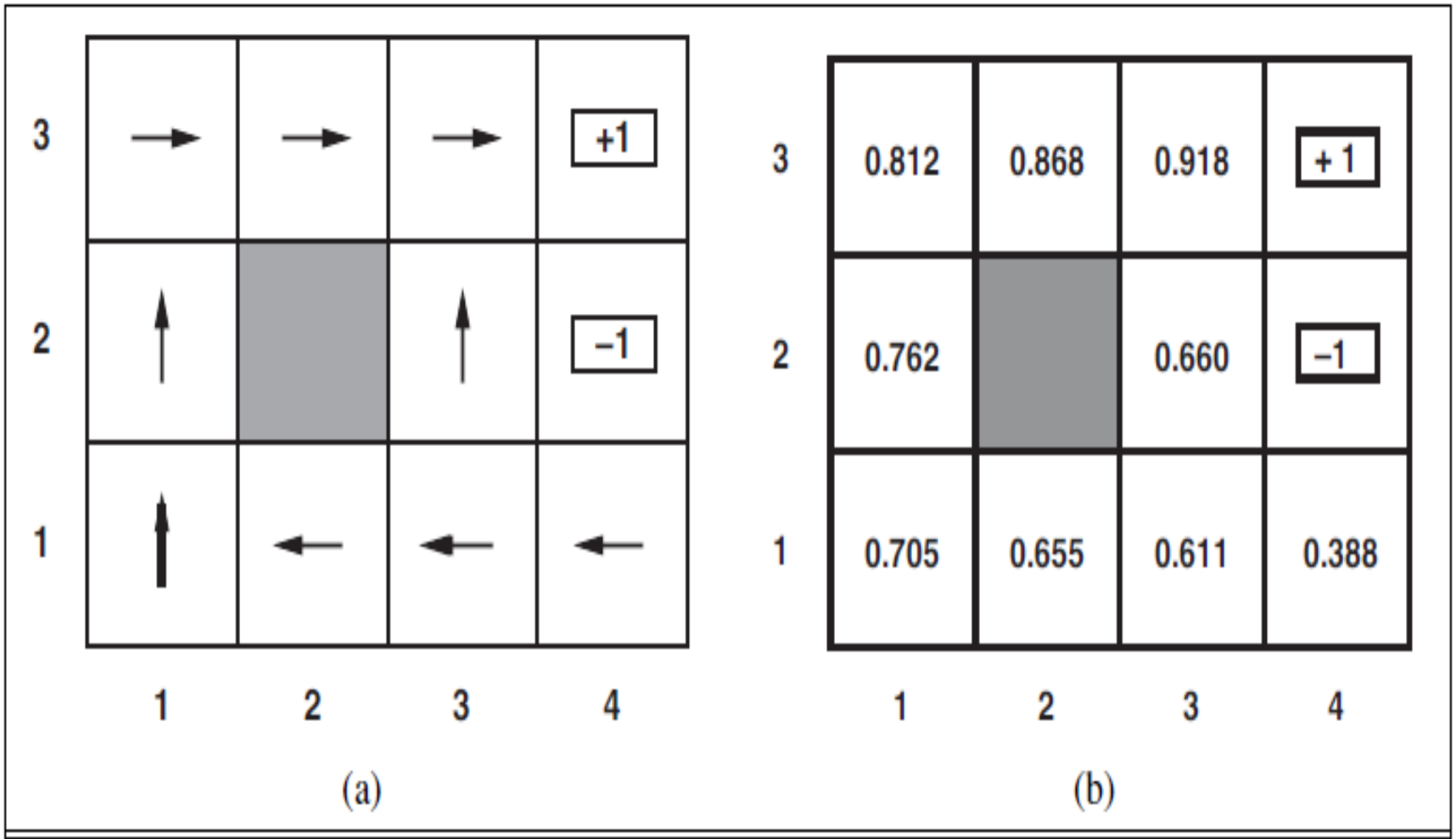

Figure 21.1 (a) A policy π for the 4×3 world; this policy happens to be optimal with rewards of $R(s) = -0.04$ in the nonterminal states and no discounting. (b) The utilities of the states in the 4×3 world, given policy π.

passive learning agent does not know the **transition model** P(s ' | s, a), which specifies the probability of reaching state s' from state s after doing action a; nor does it know the **reward function** R(s), which specifies the reward for each state.

The agent executes a set of **trials** in the environment using its policy π. In each trial, the agent starts in state (1,1) and experiences a sequence of state transitions until it reaches one of the terminal states, (4,2) or (4,3). Its percepts supply both the current state and the reward received in that state. Typical trials might look like this:

$$(1,1)_{-.04} \rightsquigarrow (1,2)_{-.04} \rightsquigarrow (1,3)_{-.04} \rightsquigarrow (1,2)_{-.04} \rightsquigarrow (1,3)_{-.04} \rightsquigarrow (2,3)_{-.04} \rightsquigarrow (3,3)_{-.04} \rightsquigarrow (4,3)_{+1}$$

$$(1,1)_{-.04} \rightsquigarrow (1,2)_{-.04} \rightsquigarrow (1,3)_{-.04} \rightsquigarrow (2,3)_{-.04} \rightsquigarrow (3,3)_{-.04} \rightsquigarrow (3,2)_{-.04} \rightsquigarrow (3,3)_{-.04} \rightsquigarrow (4,3)_{+1}$$

$$(1,1)_{-.04} \rightsquigarrow (2,1)_{-.04} \rightsquigarrow (3,1)_{-.04} \rightsquigarrow (3,2)_{-.04} \rightsquigarrow (4,2)_{-1} .$$

Note that each state percept is subscripted with the reward received. The object is to use the information about rewards to learn the expected utility $U^{\pi}(s)$ associated with each non terminal state s.

The utility is defined to be the expected sum of (discounted) rewards obtained if policy π is followed.

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right]$$

where $R(s)$ is the reward for a state, St (a random variable) is the state reached at time t when executing policy π, and $S_0 = s$. We will include a **discount factor** γ in all of our equations, but for the 4×3 world we will set $\gamma = 1$.