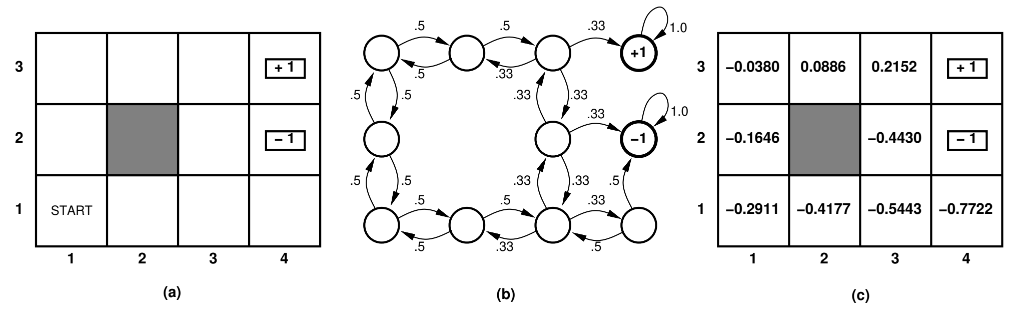

Figure 20.1 (a) A simple stochastic environment. State (1,1) is the start state. (b) Each state transitions to a neighboring state with equal probability among all neighboring states. State (4,2) is terminal with reward –1, and state (4,3) is terminal with reward +1. (c) The exact utility values.

REWARD-TO-GO

function is **additive** in the sense defined on page 502. We define the **reward-to-go** of a state as the sum of the rewards from that state until a terminal state is reached. Given this definition, it is easy to see that *the expected utility of a state is the expected reward-to-go of that state.*

The generic agent design for passive reinforcement learning of utilities is shown in Figure 20.2. The agent keeps an estimate U of the utilities of the states, a table N of counts of how many times each state was seen, and a table M of transition probabilities from state to state. We assume that each percept e is enough to determine the STATE (i.e., the state is accessible), the agent can determine the REWARD component of a percept, and the agent can tell if a percept indicates a TERMINAL? state. In general, an agent can update its current estimated utilities after each observed transition. The key to reinforcement learning lies in the algorithm for updating the utility values given the training sequences. The following subsections discuss three possible approaches to UPDATE.

Naïve updating

ADAPTIVE CONTROL THEORY

A simple method for updating utility estimates was invented in the late 1950s in the area of **adaptive control theory** by Widrow and Hoff (1960). We will call it the LMS (least mean squares) approach. In essence, it assumes that for each state in a training sequence, the *observed* reward-to-go on that sequence provides direct evidence of the actual expected reward-to-go. Thus, at the end of each sequence, the algorithm calculates the observed reward-to-go for each state and updates the estimated utility for that state accordingly. It can easily be shown (Exercise 20.1) that the LMS approach generates utility estimates that minimize the mean square error with respect to the observed data. When the utility function is represented by a table of values for each state, the update is simply done by maintaining a running average, as shown in Figure 20.3.

If we think of the utility function as a function, rather than just a table, then it is clear that the LMS approach is simply learning the utility function directly from examples. Each example has the state as input and the observed reward-to-go as output. This means that we have reduced reinforcement learning to a standard inductive learning problem, as discussed in Chapter 18. As