

Figure 3.1 The agent-environment interaction in reinforcement learning.

[SOURCE: SUTTON & BARTO *Reinforcement Learning: An Introduction*]
LINK ON WEBSITE

FAGG, BARTO, HOOK:

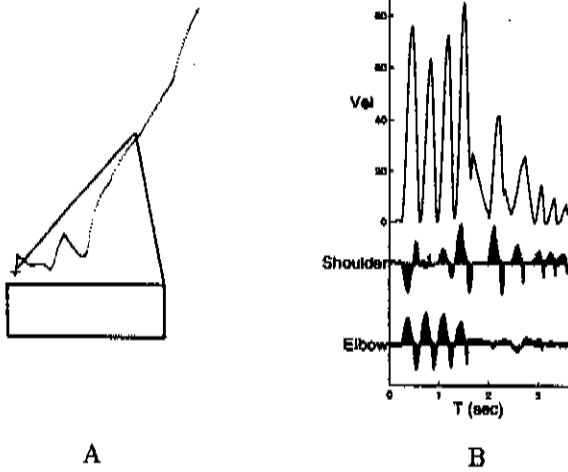


Figure 4: Behavior of the Arm Prior to Learning.

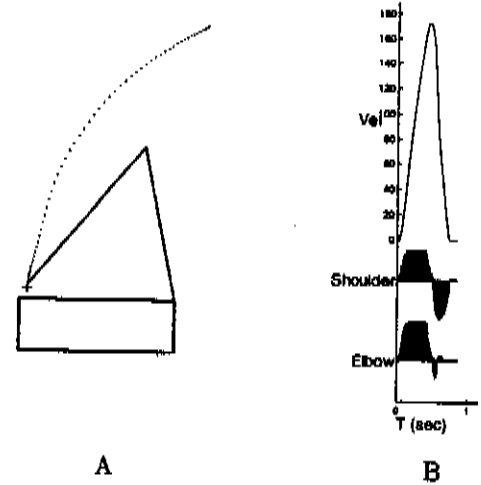


Figure 5: Behavior of the Arm After Learning.

ROSENSTEIN, BARTO:

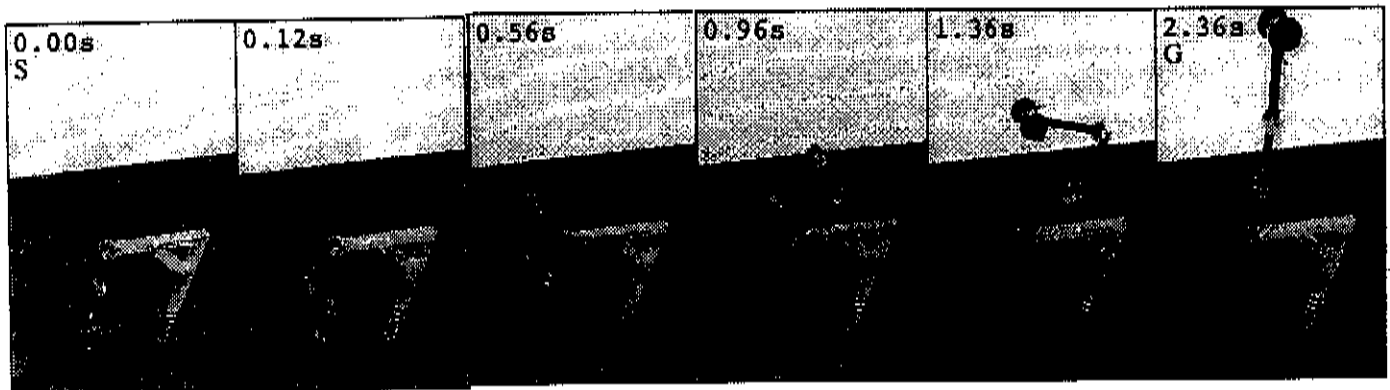
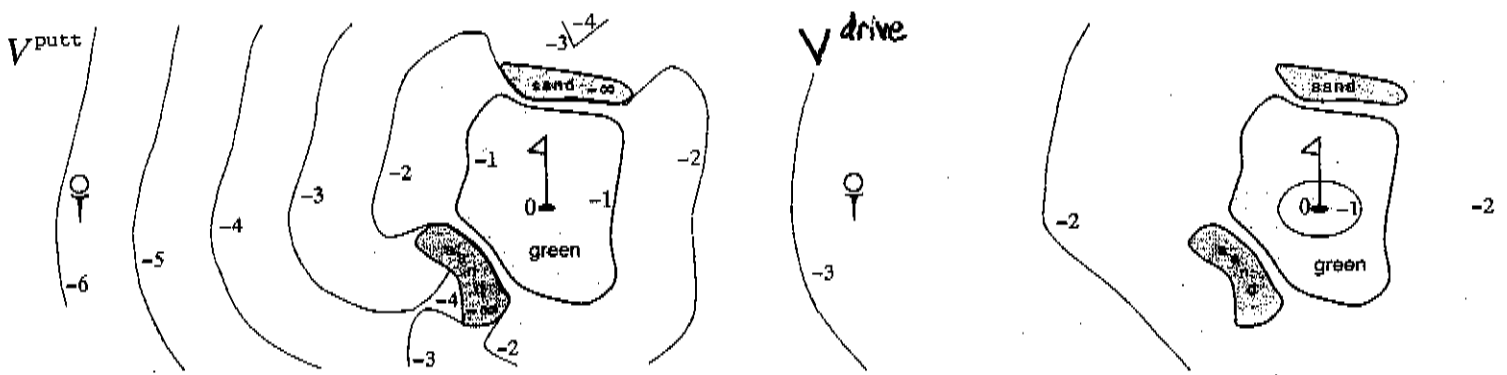


Figure 4: The reversal solution with an 9.25 kg payload.



BLACKJACK:

"STICK ONLY ON 20/21"

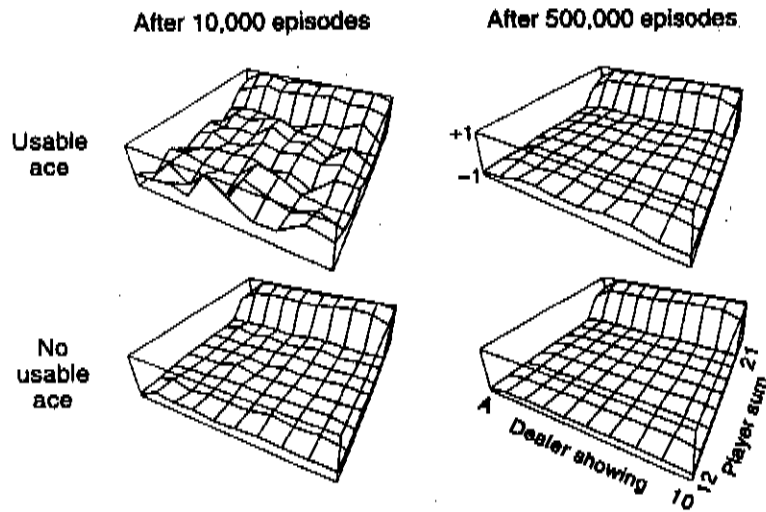


Figure 5.2 Approximate state-value functions for the blackjack policy that sticks only on 20 or 21, computed by Monte Carlo policy evaluation.

OPTIMAL POLICY

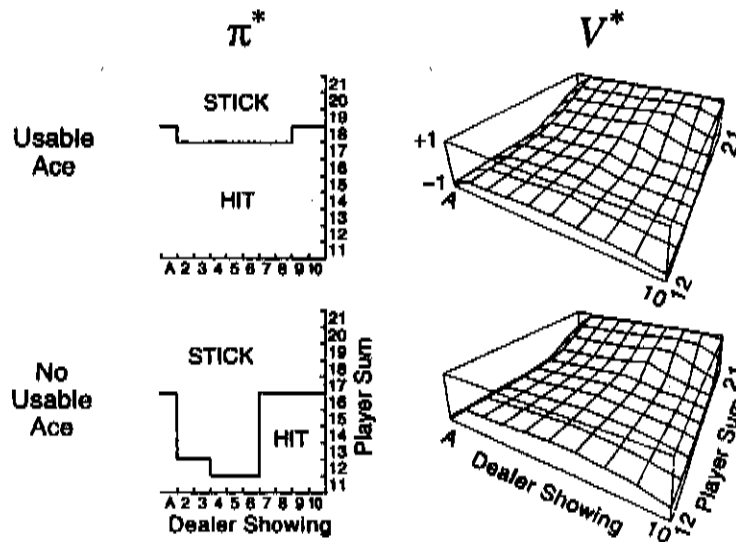


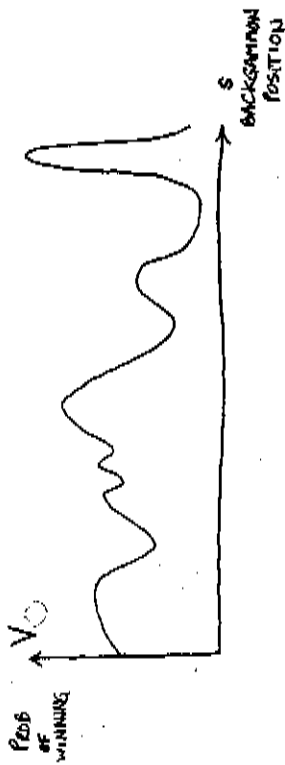
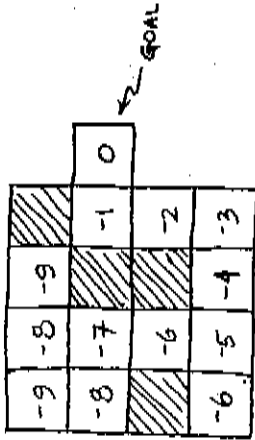
Figure 5.5 The optimal policy and state-value function for blackjack, found by Monte Carlo ES (Figure 5.4). The state-value function shown was computed from the action-value function found by Monte Carlo ES.

[SOURCE: Sutton and Barto]

ACTING WITH A VALUE FUNCTION

"CHOOSE ACTION THAT LEADS TO A STATE OF HIGHEST VALUE"

"FOLLOW GRADIENT"



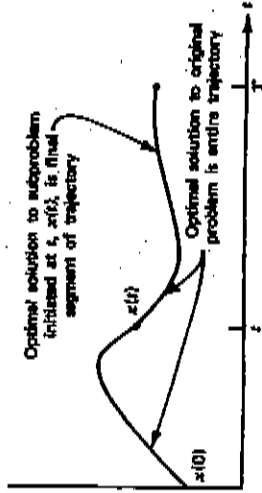
"ACTING GREEDILY WITH RESPECT TO VALUE FUNCTION"

DYNAMIC STRUCTURE OF VALUE FUNCTIONS

OPTIMAL PLAN FROM STAGE K = OPTIMAL CHOICE AT STAGE K FOLLOWED BY OPTIMAL PLAN FROM STAGE K+1

RICHARD BELLMAN:

The Principle of Optimality. An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.



FOR ANY POLICY:

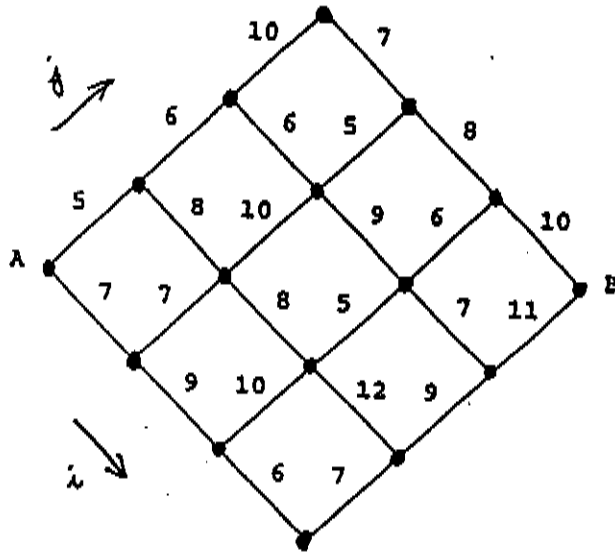
$$V(s) = r(s, \pi(s)) + \gamma V(s', \pi(s'))$$

OPTIMAL POLICY:

$$V^*(s) = \max_{a \in A(s)} \{ r(s, a) + \gamma V^*(s', \pi(s, a)) \}$$

DYNAMIC PROGRAMMING

- A METHOD TO SOLVE OPTIMAL CONTROL PROBLEMS.
- E.G., TRAVEL FROM A TO B WITH MINIMAL COST:



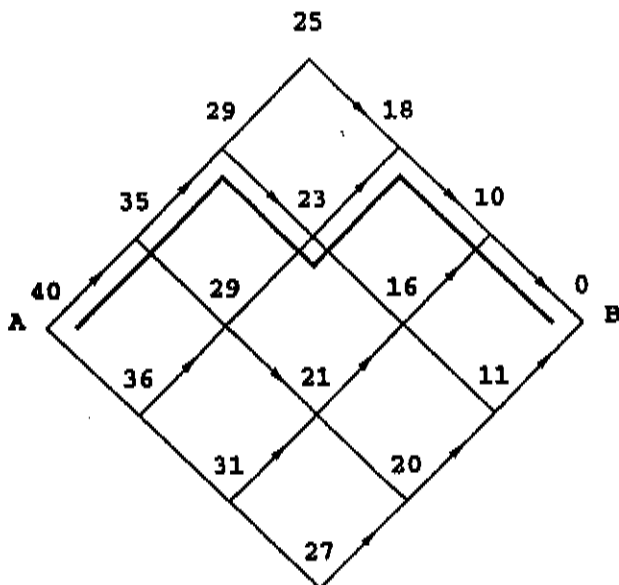
→ EDGE WEIGHTS ARE COSTS OF TRAVESING ARC

→ CAN ONLY MOVE NE + SE

- ONE WAY TO SOLVE:

- i. CALCULATE TOTAL COST OF ALL POSSIBLE PATHS (20 OF THEM)
- ii. RANK THEM
- iii. RETURN BEST

OPTIMAL SOLN FROM A TO B



NOTES: 20 PATHS
15 NUMBERS

$N \times N \rightarrow N^2 - 1$ VS ~~$[2(N-1)! / ((N-1)!)]$~~

$N=8$ 63 ~~3452~~

GO FROM CONSIDERING # OF PATHS AT EACH PT TO # OF ALT. ALTERNATIVES AT EACH PT.

VALUE ITERATION ALGORITHM

0. START WITH AN INITIAL GUESS OF THE VALUE FCN $V_0(s)$; SET $k=0$;

1. REPEAT

$$V_{k+1}(s) := \max_{a \in A(s)} \left\{ r(s,a) + \gamma \sum_{s'} T(s,a,s') V_k(s') \right\}; \quad k++;$$

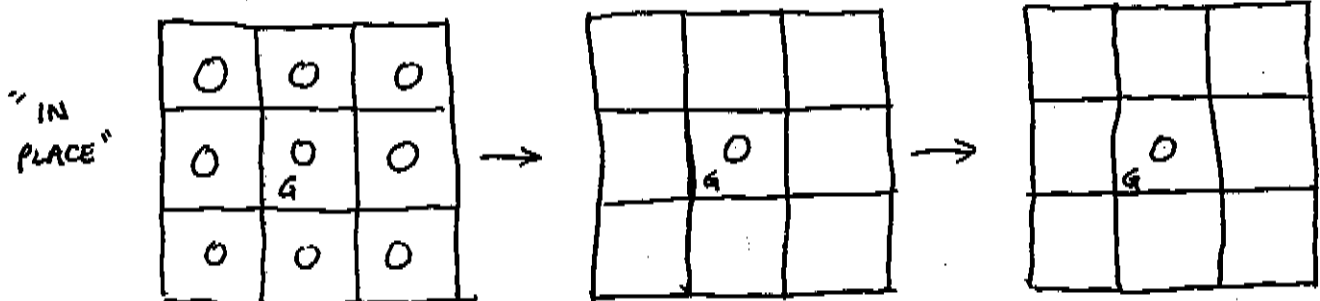
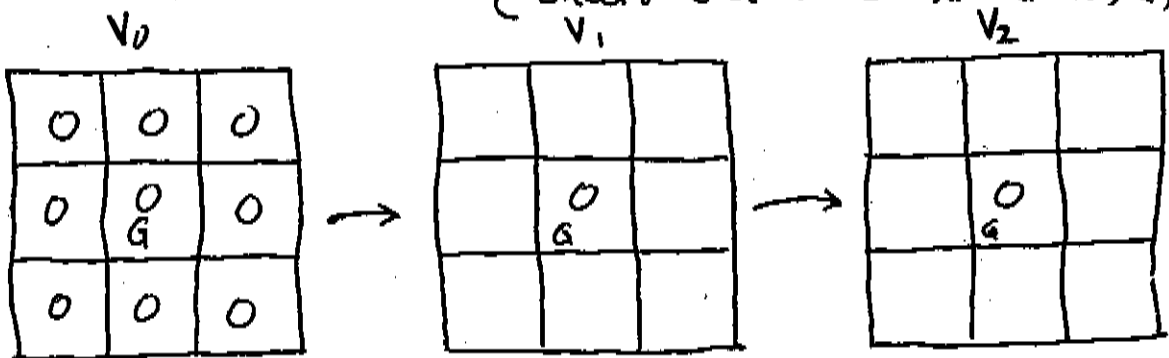
UNTIL Close Enough ($V_k(\cdot), V_{k+1}(\cdot)$)

→ THIS ALGORITHM CONVERGES TO V^* FOR ANY GUESS $V_0(\cdot)$ ~ USUALLY $V_0(s) \equiv 0$

→ GOING BACK TO EXAMPLE, WE HAD

$$V_k(s) \equiv V(s, 6-k)$$

→ ANOTHER EXAMPLE $\left(\begin{array}{l} -1 \text{ FOR N, W, S, E MOVES} \\ \text{EXCEPT COST IS 0 AT GOAL, G} \end{array} \right)$



POLICY ITERATION ALGORITHM

0. INITIALIZATION

CHOOSE ANY ADMISSIBLE POLICY $\pi_0(s)$; $k=0$;

1. REPEAT

$V^{\pi_k} := \text{POLICY-EVALUATION}(\pi_k)$;

% POLICY IMPROVEMENT

unchanged := true;

FOR EACH $s \in \mathcal{S}$

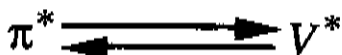
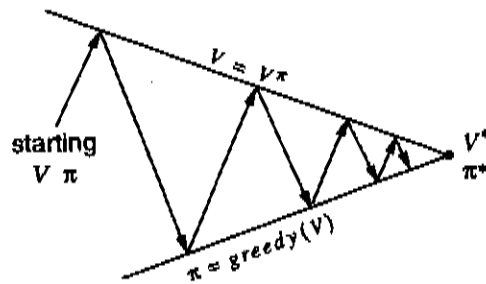
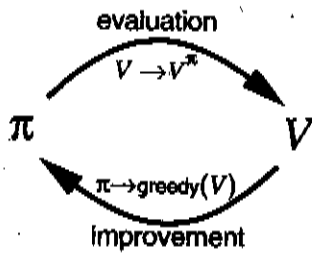
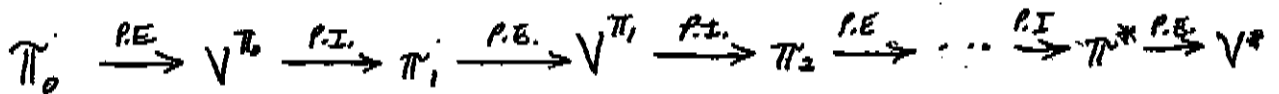
$b := \pi_k(s)$

$\pi_{k+1}(s) := \arg \max_{a \in A(s)} [r(s,a) + \gamma \sum_{s'} T(s,a,s') V^{\pi_k}(s')]$

IF $\pi_{k+1}(s) \neq b$, unchanged := false;

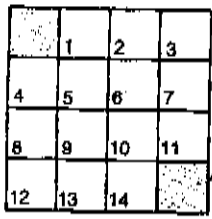
END FOR

UNTIL unchanged



[Figs FROM Sutton & Barto]

Example 4.1



TERMINAL STATE

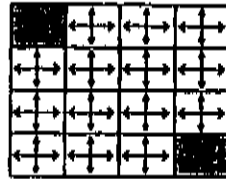
V_k for the random policy

Greedy policy w.r.t. V_k

$r = -1$ on all transitions

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



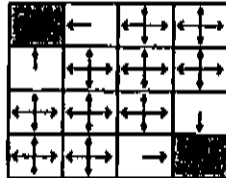
random policy



actions

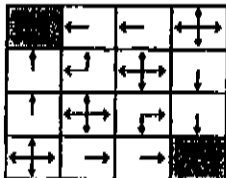
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



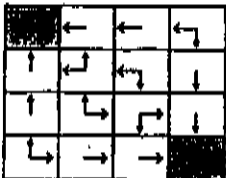
$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



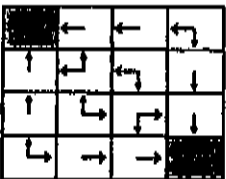
$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0



$k = 10$

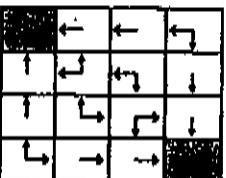
0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0



optimal policy

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



Input π , the policy to be evaluated
 Initialize $V(s) = 0$, for all $s \in \mathcal{S}^+$
 Repeat
 $\Delta \leftarrow 0$
 For each $s \in \mathcal{S}$:
 $v \leftarrow V(s)$
 $V(s) \leftarrow \sum_a \pi(s, a) \left\{ r(s, a) + \sum_{s'} T(s', a, s) \gamma V(s') \right\}$
 $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
 until $\Delta < \theta$ (a small positive number)
 Output $V \approx V^\pi$

Figure 4.1 Iterative policy evaluation.

Figure 4.2 Convergence of iterative policy evaluation on a small gridworld. The left column is the sequence of approximations of the state-value function for the random policy (all actions equal). The right column is the sequence of greedy policies corresponding to the value function estimates (arrows are shown for all actions achieving the maximum). The last policy is guaranteed only to be an improvement over the random policy, but in this case it, and all policies after the third iteration, are optimal.

[SOURCE: Sutton & Barto]

5.1 Monte Carlo Policy Evaluation

We begin by considering Monte Carlo methods for learning the state-value function for a given policy. Recall that the value of a state is the expected return—expected cumulative future discounted reward—starting from that state. An obvious way to estimate it from experience, then, is simply to average the returns observed after visits to that state. As more returns are observed, the average should converge to the expected value. This idea underlies all Monte Carlo methods.

In particular, suppose we wish to estimate $V^\pi(s)$, the value of a state s under policy π , given a set of episodes obtained by following π and passing through s . Each occurrence of state s in an episode is called a *visit* to s . The *every-visit MC method* estimates $V^\pi(s)$ as the average of the returns following all the visits to s in a set of episodes. Within a given episode, the first time s is visited is called the *first visit* to s . The *first-visit MC method* averages just the returns following first visits to s . These two Monte Carlo methods are very similar but have slightly different theoretical properties. First-visit MC has been most widely studied, dating back to the 1940s, and is the one we focus on in this chapter. We reconsider every-visit MC in Chapter 7. First-visit MC is shown in procedural form in Figure 5.1.

Both first-visit MC and every-visit MC converge to $V^\pi(s)$ as the number of visits (or first visits) to s goes to infinity. This is easy to see for the case of first-visit MC. In this case each return is an independent, identically distributed estimate of $V^\pi(s)$. By the law of large numbers the sequence of averages of these estimates converges to their expected value. Each average is itself an unbiased estimator, and the standard deviation of its error falls as $1/\sqrt{n}$, where n is the number of returns averaged. Every-visit MC is less straightforward, but its estimates also converge asymptotically to $V^\pi(s)$.

The use of Monte Carlo methods is best illustrated through an example.

Example 5.1 *Blackjack* is a popular casino card game. The object is to obtain cards the sum of whose numerical values is as great as possible without exceeding 21. All face cards count as 10, and the ace can count either as 1 or as 11. We consider the version in which each player competes independently against the dealer. The game begins with two cards dealt to both dealer and player. One of the dealer's cards is faceup and the other is facedown. If the player has 21 immediately (an ace and a 10-card), it is called a *natural*. He then wins unless the dealer also has a natural, in which case the game is a draw. If the player does not have a natural, then he can request additional cards, one by one (*hits*), until he either stops (*sticks*) or exceeds 21 (*goes bust*). If he goes bust, he loses; if he sticks, then it becomes the dealer's turn. The dealer hits or sticks according to a fixed strategy without choice: he sticks on any sum of 17 or greater, and hits otherwise. If the dealer goes bust, then the player wins; otherwise, the outcome—win, lose, or draw—is determined by whose final sum is closer to 21.

Initialize:

$\pi \leftarrow$ policy to be evaluated
 $V \leftarrow$ an arbitrary state-value function
 $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

- (a) Generate an episode using π
- (b) For each state s appearing in the episode:
 - $R \leftarrow$ return following the first occurrence of s
 - Append R to $Returns(s)$
 - $V(s) \leftarrow \text{average}(Returns(s))$

Figure 5.1 First-visit MC method for estimating V^π .

Playing blackjack is naturally formulated as an episodic finite MDP. Each game of blackjack is an episode. Rewards of +1, -1, and 0 are given for winning, losing, and drawing, respectively. All rewards within a game are zero, and we do not discount ($\gamma = 1$); therefore these terminal rewards are also the returns. The player's actions are to hit or to stick. The states depend on the player's cards and the dealer's showing card. We assume that cards are dealt from an infinite deck (i.e., with replacement) so that there is no advantage to keeping track of the cards already dealt. If the player holds an ace that he could count as 11 without going bust, then the ace is said to be *usable*. In this case it is always counted as 11 because counting it as 1 would make the sum 11 or less, in which case there is no decision to be made because, obviously, the player should always hit. Thus, the player makes decisions on the basis of three variables: his current sum (12–21), the dealer's one showing card (ace–10), and whether or not he holds a usable ace. This makes for a total of 200 states.

Consider the policy that sticks if the player's sum is 20 or 21, and otherwise hits. To find the state-value function for this policy by a Monte Carlo approach, one simulates many blackjack games using the policy and averages the returns following each state. Note that in this task the same state never recurs within one episode, so there is no difference between first-visit and every-visit MC methods. In this way, we obtained the estimates of the state-value function shown in Figure 5.2. The estimates for states with a usable ace are less certain and less regular because these states are less common. In any event, after 500,000 games the value function is very well approximated.

[Sutton & Barto, pp. 112–4.]