Sequential decision making
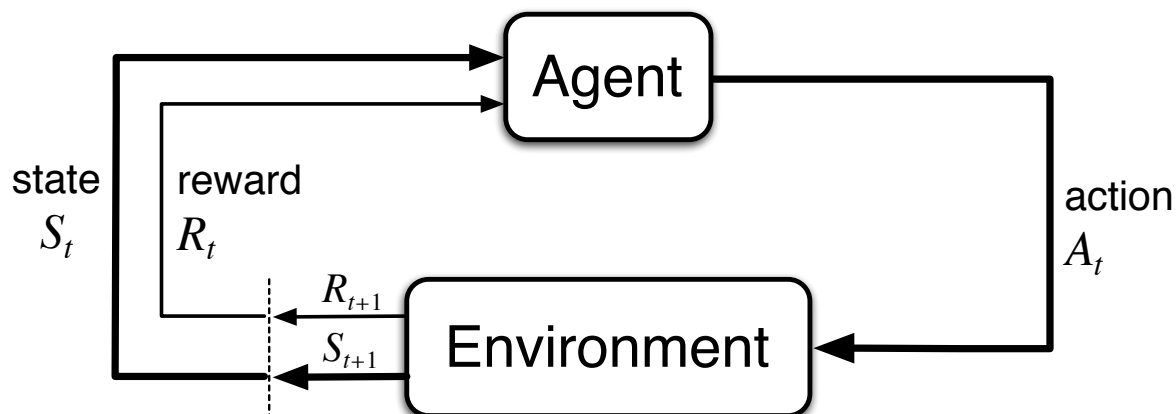# Markov Decision Processes
Dynamic Programming

Agent and environment interact at discrete time steps: $t = 0, 1, 2, 3, \ldots$

Agent observes state at step t: $S_t \in \mathcal{S}$

produces action at step t : $A_t \in \mathcal{A}(S_t)$

gets resulting reward: $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$

and resulting next state: $S_{t+1} \in \mathcal{S}^+$

Recall: Markov Decision Processes

❐ If a reinforcement learning task has the Markov Property, it is basically a **Markov Decision Process (MDP)**.

❐ If state and action sets are finite, it is a **finite MDP**.

❐ To define a finite MDP, you need to give:

- **state and action sets**

- one-step "dynamics"

$$p(s', r|s, a) = \mathbf{Pr}\{S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\}$$

$$p(s'|s, a) \doteq \Pr\{S_{t+1} = s' \mid S_t = s, A_t = a\} = \sum_{r \in \mathcal{R}} p(s', r|s, a)$$

$$r(s, a) \doteq \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a)$$

Recall: The Agent Learns a Policy

Policy at step t = π_t =

a mapping from states to action probabilities

$\pi_t(a \mid s)$ = probability that $A_t = a$ when $S_t = s$

Special case - *deterministic policies*:

$\pi_t(s)$ = the action taken with prob=1 when $S_t = s$

❑ Reinforcement learning methods specify how the agent changes its policy as a result of experience.

❑ Roughly, the agent's goal is to get as much reward as it can over the long run.

Recall: Return

Suppose the sequence of rewards after step t is:

$$R_{t+1}, R_{t+2}, R_{t+3}, \ldots$$

What do we want to maximize?

At least three cases, but in all of them,

we seek to maximize the **expected return**, $E\{G_t\}$, on each step t.

- <u>Total reward</u>, G_t = sum of all future reward in the episode
- <u>Discounted reward</u>, G_t = sum of all future *discounted* reward
- <u>Average reward</u>, G_t = average reward per time step

Discounted Return

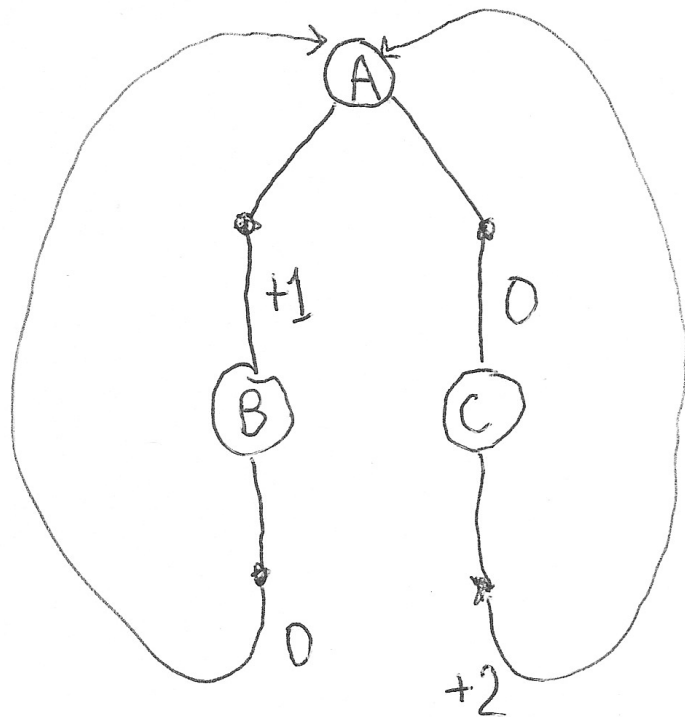**Continuing tasks**: interaction does not have natural episodes, but just goes on and on...

In this class, for continuing tasks we will always use *discounted return*:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

where γ, $0 \leq \gamma \leq 1$, is the **discount rate**.

shortsighted $0 \leftarrow \gamma \rightarrow 1$ farsighted

Typically, $\gamma = 0.9$

What policy is optimal?
  A: left
  B: Right   C: other

If $\gamma = 0$?

If $\gamma = .99$

If $\gamma = \frac{1}{2}$?

Episodic Tasks: Total Reward

Episodic tasks: interaction breaks naturally into episodes, e.g., plays of a game, trips through a maze
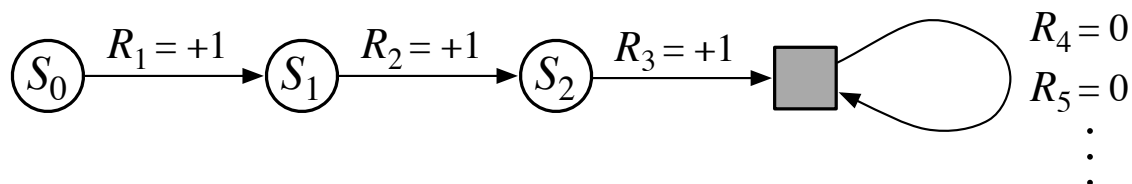
In episodic tasks, we almost always use simple *total reward*:

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T,$$

where T is a final time step at which a **terminal state** is reached, ending an episode.

A Trick to Unify Notation for Returns

❏ In episodic tasks, we number the time steps of each episode starting from zero.

❏ Think of each episode as ending in an absorbing state that always produces reward of zero:

❏ We can cover <u>all</u> cases by writing $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$,

where γ can be 1 only if a zero reward absorbing state is always reached.

Episodic and Continuing Tasks: Average Reward

In episodic tasks, we can also use *average reward*:

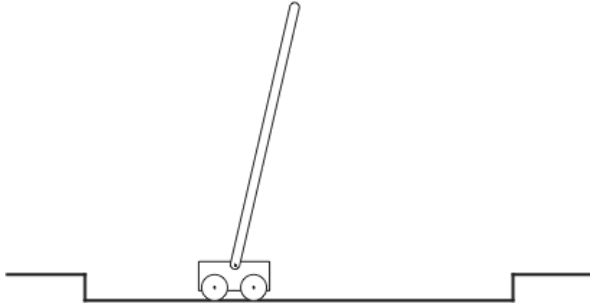$$G_0 = (\sum_{t=0}^{T} R_t)/T$$

where T is a final time step at which a **terminal state** is reached, ending an episode.

In continuing tasks, we can also define *average reward*:

$$G = \lim_{T \to \infty} \left((\sum_{t=0}^{T} R_t)/T \right)$$

More on this later!

An Example: Pole Balancing

Avoid **failure:** the pole falling beyond a critical angle or the cart hitting end of track

As an **episodic task** where episode ends upon failure:
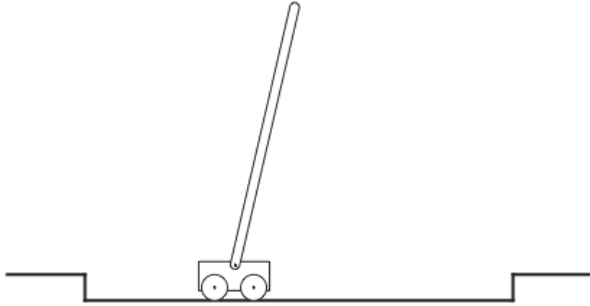
An Example: Pole Balancing
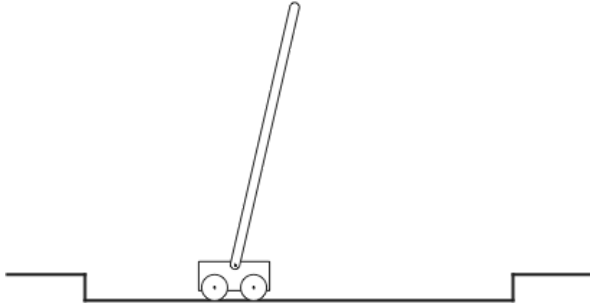

Avoid **failure:** the pole falling beyond a critical angle or the cart hitting end of track

As an **episodic task** where episode ends upon failure:

        reward   = +1 for each step before failure

        $\Rightarrow$ return = number of steps before failure

An Example: Pole Balancing

Avoid **failure:** the pole falling beyond a critical angle or the cart hitting end of track

As a **continuing task** with discounted return:
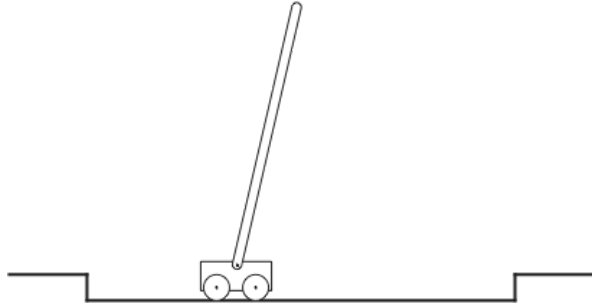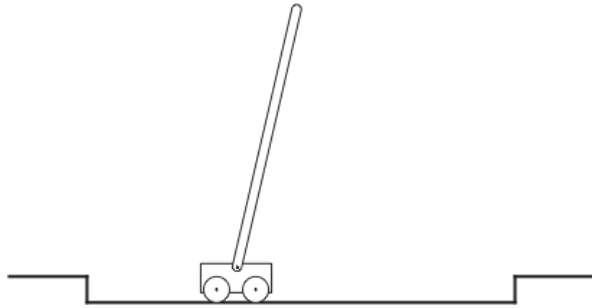
An Example: Pole Balancing

Avoid **failure:** the pole falling beyond a critical angle or the cart hitting end of track

As a **continuing task** with discounted return:

reward $= -1$ upon failure; 0 otherwise

$\Rightarrow$ return $= -\gamma^{k}$, for k steps before failure

An Example: Pole Balancing

Avoid **failure:** the pole falling beyond a critical angle or the cart hitting end of track

As an **episodic task** where episode ends upon failure:

reward   = +1 for each step before failure

$\Rightarrow$ return = number of steps before failure

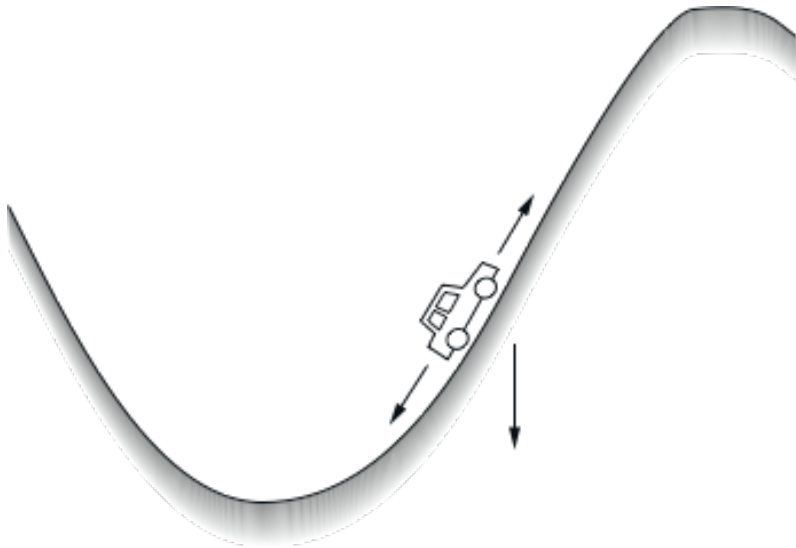As a **continuing task** with discounted return:

reward   = −1 upon failure; 0 otherwise

$\Rightarrow$ return = $-\gamma^{k}$, for k steps before failure

In either case, return is maximized by avoiding failure for as long as possible.

Another Example: Mountain Car

Get to the top of the hill
as quickly as possible.

Return is maximized by minimizing
number of steps to reach the top of the hill.

Mountain Car: Discounted
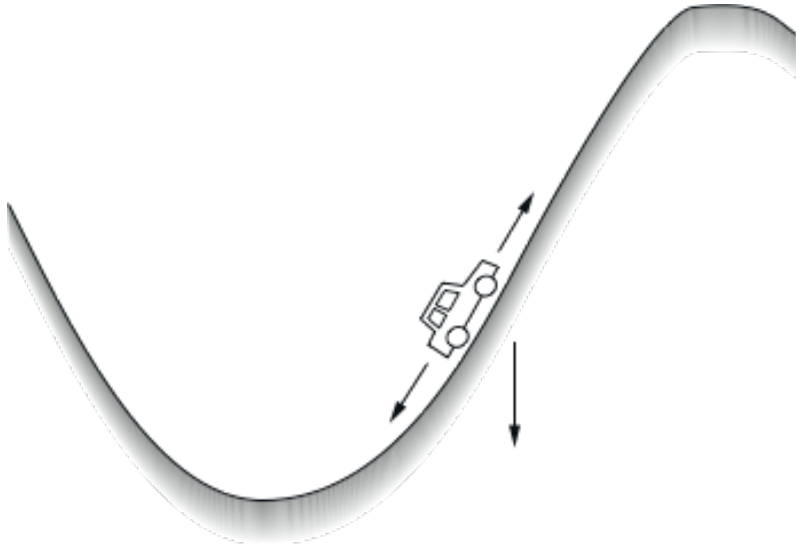
Get to the top of the hill
as quickly as possible.

Reward: 1 at the top of the hill, 0 otherwise
Return: if discount <1, k=number of time steps, so return is γ^k

Return is maximized by minimizing
number of steps to reach the top of the hill.

Mountain Car: Episodic
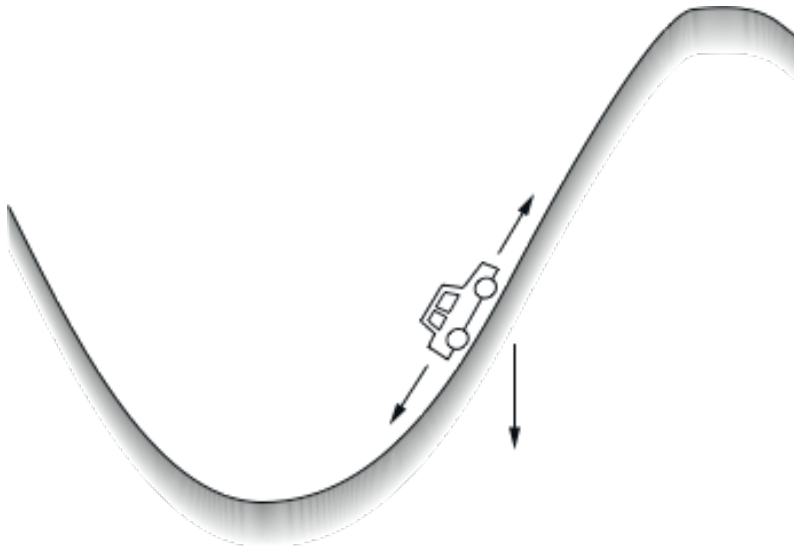
Get to the top of the hill
as quickly as possible.

reward $= -1$ for each step where **not** at top of hill

$\Rightarrow$ return $= -$ number of steps before reaching top of hill

Return is maximized by minimizing
number of steps to reach the top of the hill.

4 value functions

	state values	action values
prediction	v_π	q_π
control	v_*	q_*

- All theoretical objects, expected values

- Distinct from their estimates: $V_t(s)$ $Q_t(s, a)$

Values are *expected* returns

- The value of a state, given a policy:

$$v_\pi(s) = \mathbb{E}\{G_t \mid S_t = s, A_{t:\infty} \sim \pi\} \qquad v_\pi : \mathcal{S} \to \Re$$

- The value of a state-action pair, given a policy:

$$q_\pi(s, a) = \mathbb{E}\{G_t \mid S_t = s, A_t = a, A_{t+1:\infty} \sim \pi\} \qquad q_\pi : \mathcal{S} \times \mathcal{A} \to \Re$$

- The optimal value of a state:

$$v_*(s) = \max_\pi v_\pi(s) \qquad v_* : \mathcal{S} \to \Re$$

- The optimal value of a state-action pair:

$$q_*(s, a) = \max_\pi q_\pi(s, a) \qquad q_* : \mathcal{S} \times \mathcal{A} \to \Re$$

- Optimal policy: π_* is an optimal policy if and only if

$$\pi_*(a|s) > 0 \text{ only where } q_*(s, a) = \max_b q_*(s, b) \qquad \forall s \in \mathcal{S}$$

  - in other words, π_* is optimal iff it is *greedy* wrt q_*

Value Functions

❏ The **value of a state** is the expected return starting from that state; depends on the agent's policy:

State - value function for policy π :

$$v_\pi(s) = E_\pi \left\{ G_t \mid S_t = s \right\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right\}$$

❏ The **value of an action (in a state)** is the expected return starting after taking that action from that state; depends on the agent's policy:

Action - value function for policy π :

$$q_\pi(s,a) = E_\pi \left\{ G_t \mid S_t = s, A_t = a \right\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right\}$$

Gridworld

- ☐ Actions: `north`, `south`, `east`, `west`; deterministic.
- ☐ If would take agent off the grid: no move but reward = –1
- ☐ Other actions produce reward = 0, except actions that move agent out of special states A and B as shown.
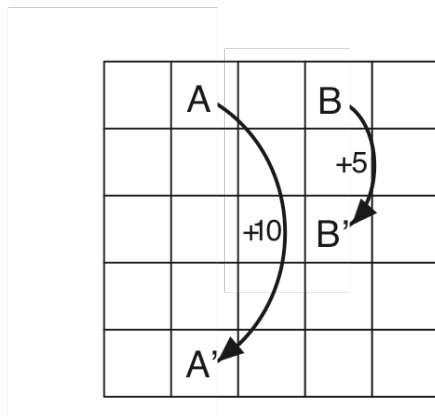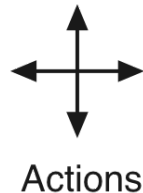
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(a)　　　　　　Actions　　　　　　(b)

State-value function for equiprobable random policy; $\gamma = 0.9$

Policy Evaluation

Policy Evaluation: for a given policy π, compute the state-value function v_π

Recall: **State-value function for policy π**

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \;\middle|\; S_t = s\right]$$

Bellman Equation for a Policy π

The basic idea:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots$$

$$= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right)$$

$$= R_{t+1} + \gamma G_{t+1}$$

So:

$$v_\pi(s) = E_\pi \left\{ G_t \mid S_t = s \right\}$$

$$= E_\pi \left\{ R_{t+1} + \gamma v_\pi \left(S_{t+1} \right) \mid S_t = s \right\}$$

Or, without the expectation operator:
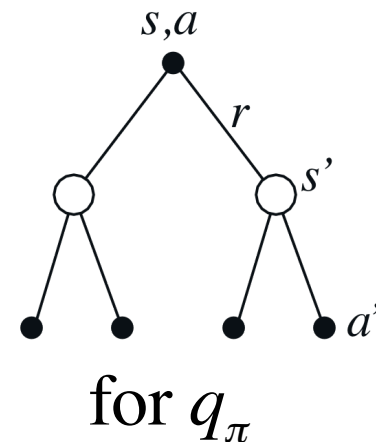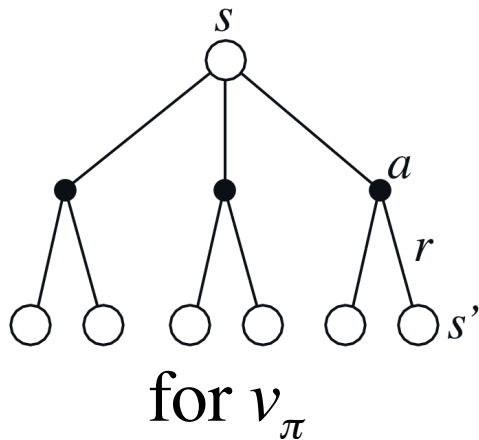
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma v_\pi(s') \right]$$

More on the Bellman Equation

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big]$$

This is a set of equations (in fact, linear), one for each state. The value function for π is its unique solution.

Backup diagrams:

for v_π for q_π

Policy Evaluation

Policy Evaluation: for a given policy π, compute the state-value function v_π

Recall: **State-value function for policy π**

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\Big|\, S_t = s\right]$$

Recall: **Bellman equation for v_π**

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big]$$

—a system of $|\mathcal{S}|$ simultaneous equations

Iterative Methods

$$v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_k \rightarrow v_{k+1} \rightarrow \cdots \rightarrow v_\pi$$

a "sweep"

A sweep consists of applying a **backup operation** to each state.

A **full policy-evaluation backup**:

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_k(s')\Big] \qquad \forall s \in \mathcal{S}$$

Iterative Policy Evaluation – One array version

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

A Small Gridworld
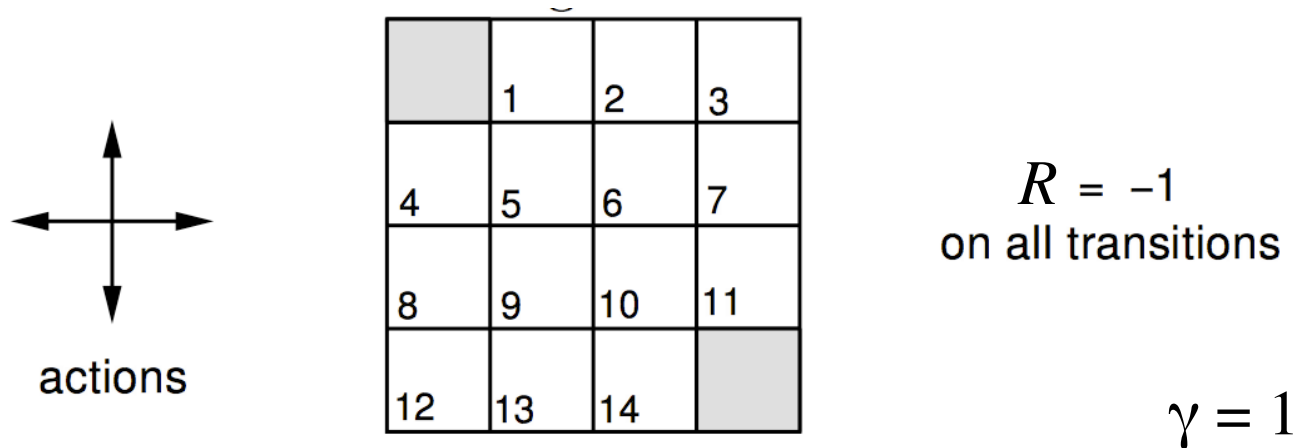
$R = -1$
on all transitions

$\gamma = 1$

❏ An undiscounted episodic task

❏ Nonterminal states: 1, 2, . . ., 14;

❏ One terminal state (shown twice as shaded squares)

❏ Actions that would take agent off the grid leave state unchanged

❏ Reward is –1 until the terminal state is reached

Iterative Policy Eval for the Small Gridworld

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

$k = 0$

$\pi =$ equiprobable random action choices

$k = 1$

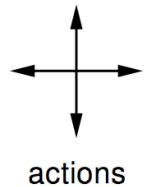|	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R = -1$
on all transitions

$k = 2$

actions

$\gamma = 1$

$k = 3$

❑ An undiscounted episodic task

❑ Nonterminal states: $1, 2, \ldots, 14$;

❑ One terminal state (shown twice as shaded squares)

$k = 10$

❑ Actions that would take agent off the grid leave state unchanged

❑ Reward is -1 until the terminal state is reached

$k = \infty$

Iterative Policy Eval
for the Small Gridworld

$\pi =$ equiprobable random action choices

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

$k = 0$

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

$k = 1$

$k = 2$

$R = -1$
on all transitions

$\gamma = 1$

$k = 3$

actions

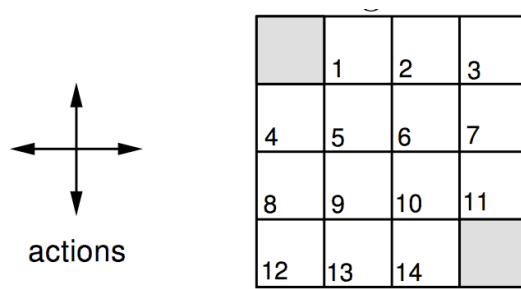|	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

☐ An undiscounted episodic task

☐ Nonterminal states: 1, 2, . . ., 14;

☐ One terminal state (shown twice as shaded squares)

$k = 10$

☐ Actions that would take agent off the grid leave state unchanged

☐ Reward is –1 until the terminal state is reached

$k = \infty$

Iterative Policy Eval
for the Small Gridworld

$\pi =$ equiprobable random action choices

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

$k = 0$

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

$k = 1$

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

actions

$R = -1$
on all transitions

$\gamma = 1$

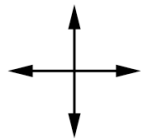| 0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$k = 2$

$k = 3$

❏ An undiscounted episodic task

❏ Nonterminal states: $1, 2, \ldots, 14$;

❏ One terminal state (shown twice as shaded squares)

$k = 10$

❏ Actions that would take agent off the grid leave state unchanged

❏ Reward is -1 until the terminal state is reached

$k = \infty$

Iterative Policy Eval
for the Small Gridworld

π = equiprobable random action choices

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

$k = 0$

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

actions

$R = -1$
on all transitions

$\gamma = 1$

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

$k = 1$

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

$k = 2$

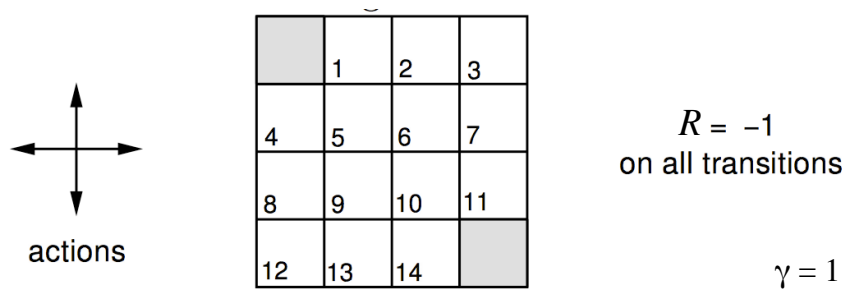| 0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 3$

❏ An undiscounted episodic task

❏ Nonterminal states: 1, 2, . . ., 14;

❏ One terminal state (shown twice as shaded squares)

$k = 10$

❏ Actions that would take agent off the grid leave state unchanged

❏ Reward is −1 until the terminal state is reached

$k = \infty$

Iterative Policy Eval
for the Small Gridworld

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

$k = 0$

$\pi =$ equiprobable random action choices

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

$k = 1$

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

actions

$R = -1$
on all transitions

$\gamma = 1$

$k = 2$

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

❐ An undiscounted episodic task

❐ Nonterminal states: 1, 2, . . ., 14;

❐ One terminal state (shown twice as shaded squares)

❐ Actions that would take agent off the grid leave state unchanged

❐ Reward is −1 until the terminal state is reached

$k = 10$

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

$k = \infty$

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

Policy Iteration

$$\pi_0 \xrightarrow{\ \text{E}\ } v_{\pi_0} \xrightarrow{\ \text{I}\ } \pi_1 \xrightarrow{\ \text{E}\ } v_{\pi_1} \xrightarrow{\ \text{I}\ } \pi_2 \xrightarrow{\ \text{E}\ } \cdots \xrightarrow{\ \text{I}\ } \pi_* \xrightarrow{\ \text{E}\ } v_*$$

policy evaluation   policy improvement
                          "greedification"

Policy Improvement

Suppose we have computed v_π for a deterministic policy π.

For a given state s,
would it be better to do an action $a \neq \pi(s)$?

It is better to switch to action a for state s if and only if

$$q_\pi(s,a) > v_\pi(s)$$

And, we can compute $q_\pi(s,a)$ from v_π by:

$$
\begin{aligned}
q_\pi(s, a) &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \sum_{s',r} p(s', r | s, a)\Big[r + \gamma v_\pi(s')\Big].
\end{aligned}
$$

Policy Improvement Cont.

Do this for all states to get a new policy $\pi' \geq \pi$ that is **greedy** with respect to v_π :

$$
\begin{aligned}
\pi'(s) &= \arg\max_a q_\pi(s, a) \\
&= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \arg\max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma v_\pi(s')\Big],
\end{aligned}
$$

What if the policy is unchanged by this?
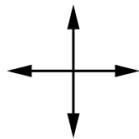 Then the policy must be optimal!

Policy Iteration

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

policy evaluation   policy improvement
"greedification"

Greedy Policies
for the Small Gridworld

Greedy Policy
w.r.t. V_k

$k = 0$

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

$\pi = $ equiprobable random action choices

$k = 1$

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

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R = -1$
on all transitions

$k = 2$

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

actions

$\gamma = 1$

$k = 3$

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

❑ An undiscounted episodic task

❑ Nonterminal states: 1, 2, . . ., 14;

❑ One terminal state (shown twice as shaded squares)

$k = 10$

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

❑ Actions that would take agent off the grid leave state unchanged

❑ Reward is −1 until the terminal state is reached

$k = \infty$

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

Greedy Policies for the Small Gridworld
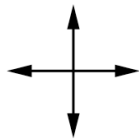
$\pi =$ equiprobable random action choices

$R = -1$
on all transitions

$\gamma = 1$

☐ An undiscounted episodic task

☐ Nonterminal states: 1, 2, . . ., 14;

☐ One terminal state (shown twice as shaded squares)

☐ Actions that would take agent off the grid leave state unchanged

☐ Reward is −1 until the terminal state is reached

V_k for the Random Policy

Greedy Policy w.r.t. V_k

$k = 0$ — random policy

$k = 1$

$k = 2$

$k = 3$

$k = 10$ — optimal policy

$k = \infty$

Policy Iteration – One array version (+ policy)

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\quad \Delta \leftarrow 0$
   $\quad$ For each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad a \leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
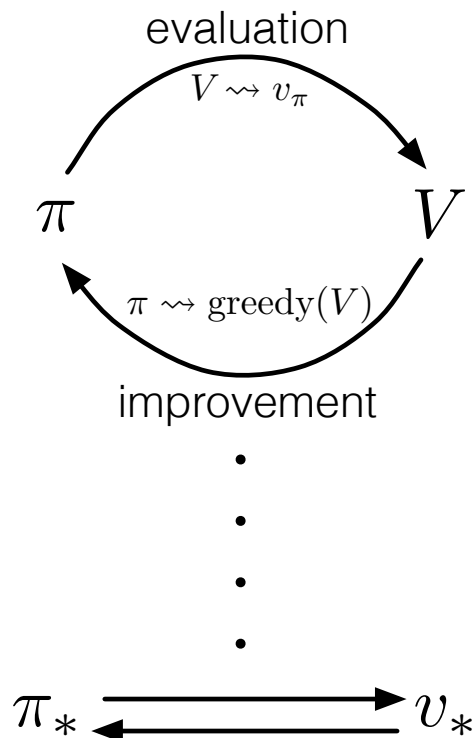 $\quad$ If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return V and π; else go to 2
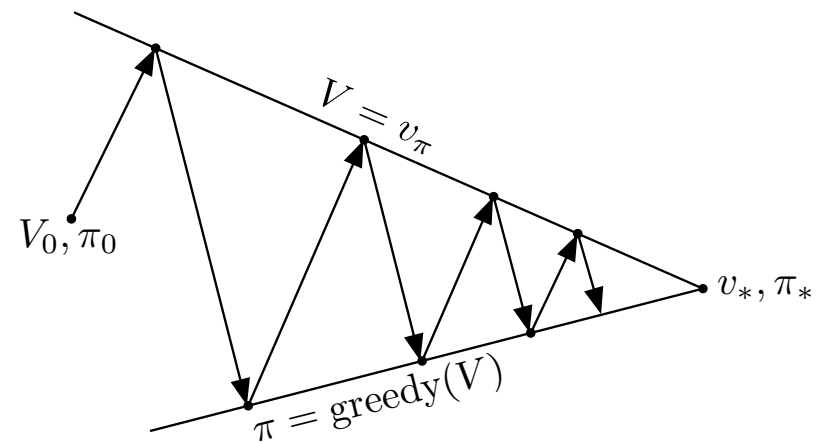
Generalized Policy Iteration

Generalized Policy Iteration (GPI):
any interaction of policy evaluation and policy improvement,
independent of their granularity.

evaluation

$V \rightsquigarrow v_\pi$

π V

$\pi \rightsquigarrow \text{greedy}(V)$

improvement

$\pi_* \longleftrightarrow v_*$

A geometric metaphor for
convergence of GPI:

$V = v_\pi$

V_0, π_0

v_*, π_*

$\pi = \text{greedy}(V)$

Value Iteration

Recall the **full policy-evaluation backup**:

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \Big[r + \gamma v_k(s') \Big] \qquad \forall s \in \mathcal{S}$$

Here is the **full value-iteration backup**:

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s',r|s,a) \Big[r + \gamma v_k(s') \Big] \qquad \forall s \in \mathcal{S}$$

Value Iteration – One array version

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s',r} p(s', r | s, a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
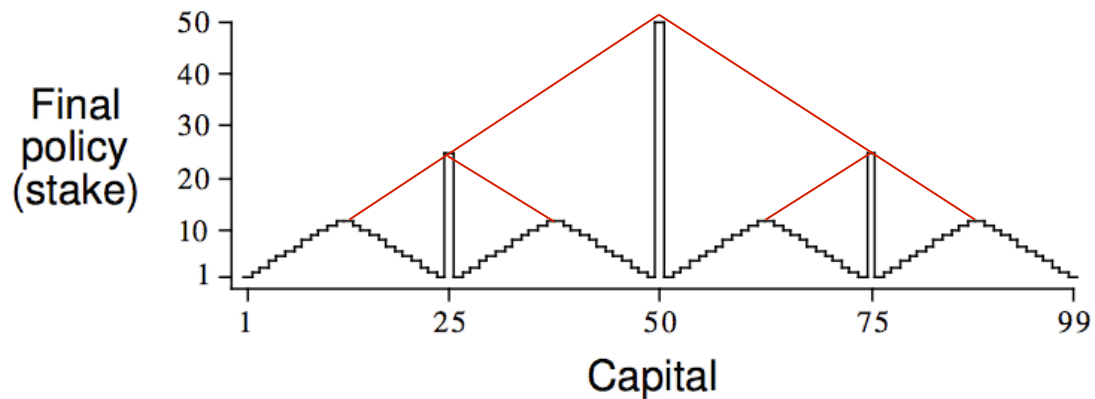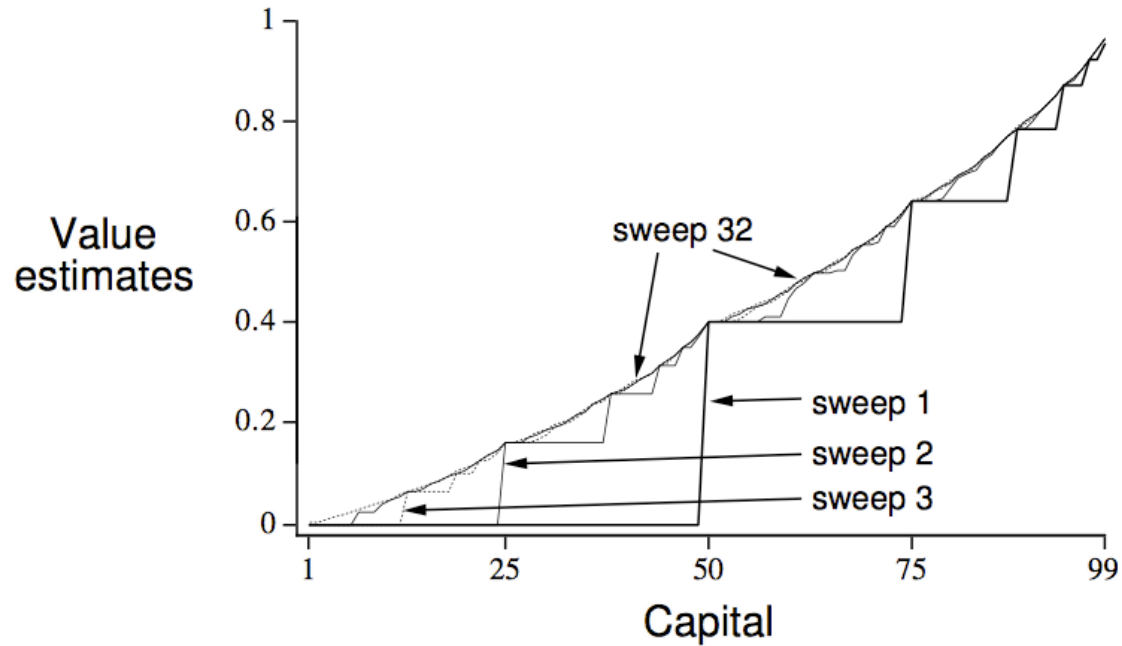 $\pi(s) = \arg\max_a \sum_{s',r} p(s', r | s, a)\big[r + \gamma V(s')\big]$

Gambler's Problem

❒ Gambler can repeatedly bet $ on a coin flip

❒ Heads he wins his stake, tails he loses it

❒ Initial capital $\in$ {$1, $2, ... $99}

❒ Gambler wins if his capital becomes $100
loses if it becomes $0

❒ Coin is unfair

▪ Heads (gambler wins) with probability $p = .4$

❒ States, Actions, Rewards? Discounting?

Gambler's Problem Solution
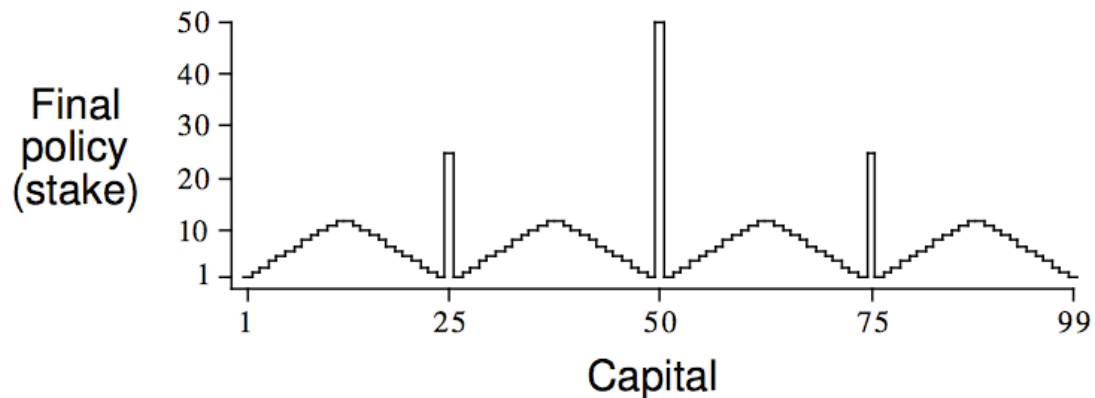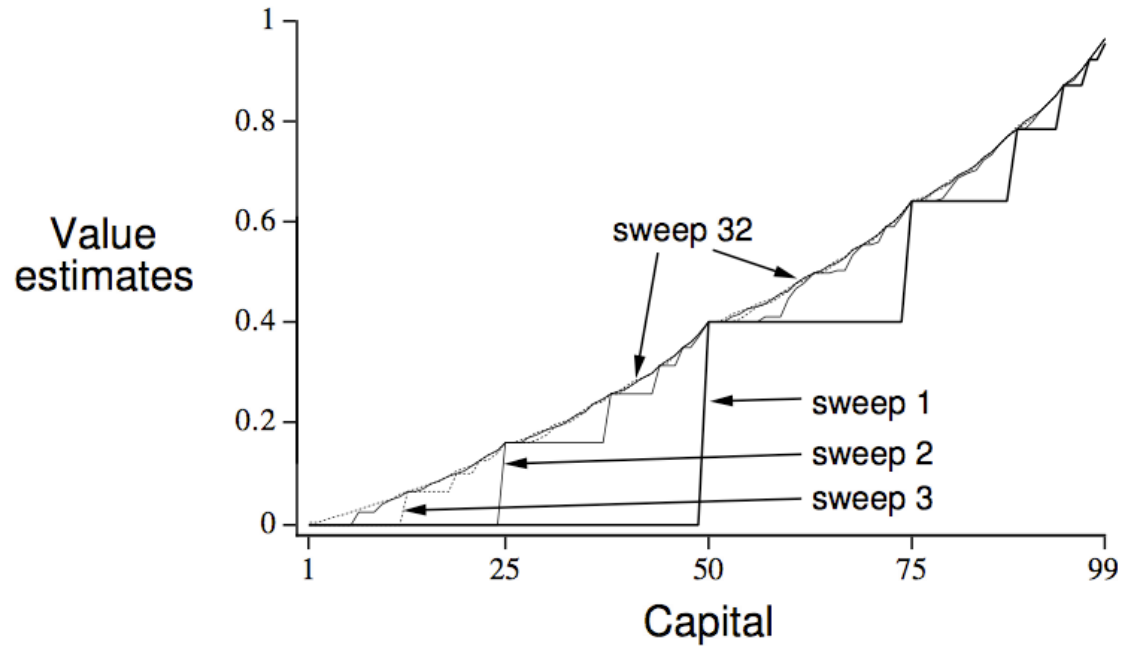
Gambler's Problem Solution

Asynchronous DP

❏ All the DP methods described so far require exhaustive sweeps of the entire state set.

❏ Asynchronous DP does not use sweeps. Instead it works like this:

- ▪ Repeat until convergence criterion is met:
  - – Pick a state at random and apply the appropriate backup

❏ Still need lots of computation, but does not get locked into hopelessly long sweeps

❏ Can you select states to backup intelligently? YES: an agent's experience can act as a guide.

Efficiency of DP

- ❐ To find an optimal policy is polynomial in the number of states…

- ❐ BUT, the number of states is often astronomical, e.g., often growing exponentially with the number of state variables (what Bellman called "the curse of dimensionality").

- ❐ In practice, classical DP can be applied to problems with a few millions of states.

- ❐ Asynchronous DP can be applied to larger problems, and is appropriate for parallel computation.

- ❐ It is surprisingly easy to come up with MDPs for which DP methods are not practical.

Summary

- ❑ Policy evaluation: backups without a max
- ❑ Policy improvement: form a greedy policy, if only locally
- ❑ Policy iteration: alternate the above two processes
- ❑ Value iteration: backups with a max
- ❑ Full backups (to be contrasted later with sample backups)
- ❑ Generalized Policy Iteration (GPI)
- ❑ Asynchronous DP: a way to avoid exhaustive sweeps
- ❑ **Bootstrapping**: updating estimates based on other estimates
- ❑ Biggest limitation of DP is that it requires a *probability model* (as opposed to a generative or simulation model)