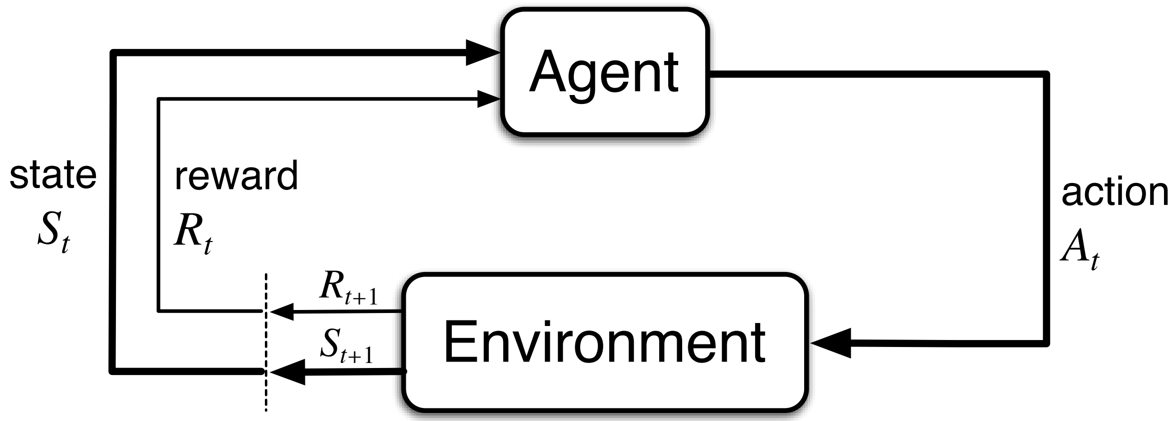
Evaluating Value Fcts:
Dynamic Programming,
Monte-Carlo,
Temporal Difference Learning

Recall: Agent-Environment Interface



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: Return

Agent wants to maximize it's 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

4 value functions

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

- All theoretical objects, expected values
- Distinct from their estimates: $V_t(s)$ $\quad Q_t(s, a)$

Today: Algorithms to Estimate v, q

- DP: Dynamic Programming
- MC: Monte-Carlo
- TD: Temporal Difference Learning

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 :

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

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:

$$
\begin{aligned}
G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\
&= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right) \\
&= R_{t+1} + \gamma G_{t+1}
\end{aligned}
$$

So:

$$
\boxed{
\begin{aligned}
v_\pi(s) &= E_\pi \left\{ G_t \,\middle|\, S_t = s \right\} \\
&= E_\pi \left\{ R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) \,\middle|\, S_t = s \right\}
\end{aligned}
}
$$

Or, without the expectation operator:

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

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*.

* In the usual case where the system of equations is invertible, but in the current context you would really need to work hard to make it non-invertible.

Q-Function

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

Iterative Methods

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

a "sweep"

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

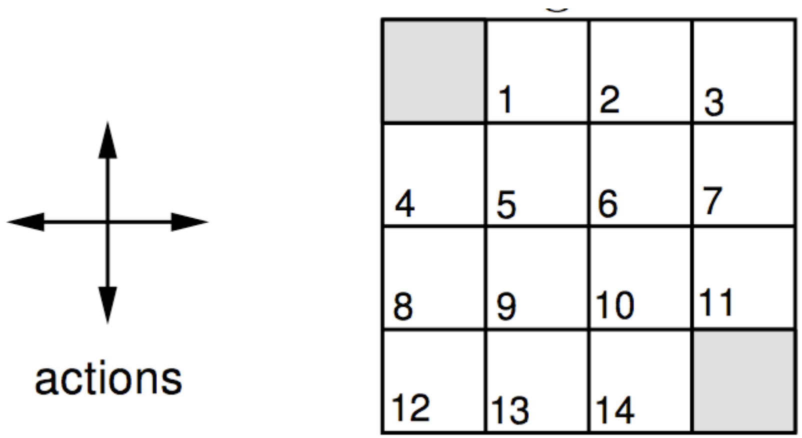
A **full policy-evaluation backup**:

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

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

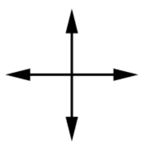
$\gamma = 1$

- 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

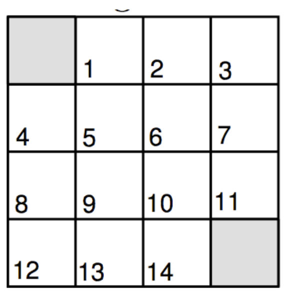
Iterative Policy Eval for the Small Gridworld

V_k for the Random Policy

$\pi =$ equiprobable random action choices



actions



$R = -1$
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

$k = 0$

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

$k = 1$

$k = 2$

$k = 3$

$k = 10$

$k = \infty$

Iterative Policy Eval for the Small Gridworld

V_k for the Random Policy

$k = 0$

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

$k = 1$

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

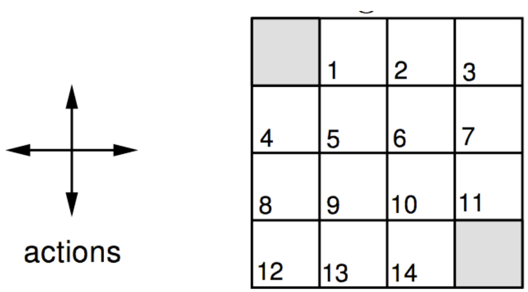
$k = 2$

$k = 3$

$k = 10$

$k = \infty$

$\pi =$ equiprobable random action choices



$R = -1$
on all transitions

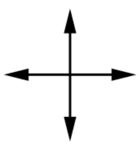
$\gamma = 1$

- 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

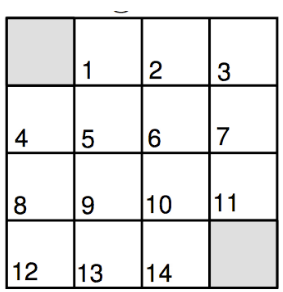
Iterative Policy Eval for the Small Gridworld

V_k for the Random Policy

$\pi =$ equiprobable random action choices



actions



$R = -1$
on all transitions

$\gamma = 1$

$k = 0$

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

$k = 1$

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

$k = 3$

- 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

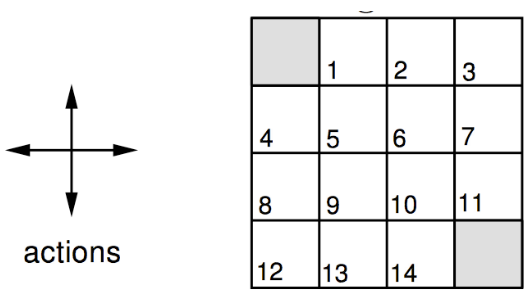
$k = 10$

$k = \infty$

Iterative Policy Eval for the Small Gridworld

V_k for the Random Policy

$\pi =$ 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

$k = 0$

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

$k = 1$

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

$k = 3$

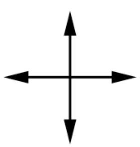
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

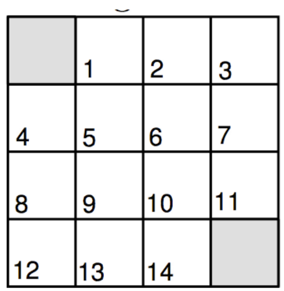
$k = \infty$

Iterative Policy Eval for the Small Gridworld

$\pi =$ equiprobable random action choices



actions



$R = -1$
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

$k = 0$

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

$k = 1$

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

$k = \infty$

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

Bellman Optimality Eqn

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

Bellman Optimality Eqn

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

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s, a)$$

Bellman Optimality Eqn

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

$$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s, a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \end{aligned}$$

Bellman Optimality Eqn

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

$$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \end{aligned}$$

Bellman Optimality Eqn

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

$$\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]
\end{aligned}$$

Bellman Optimality Eqn

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

$$\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma v_*(s')\big].
\end{aligned}$$

Bellman Optimality Eqn

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

$$v_*(s) = \max_a \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma v_*(s')\big]$$

Also as many equations as unknowns (non-linear, this time though).

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

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

Policy Improvement Cont.

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

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

What if the policy is unchanged by this?

Then the policy must be optimal!

Policy Iteration

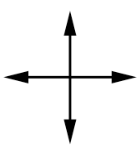
$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

policy evaluation

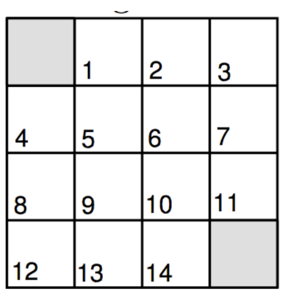
policy improvement
"greedification"

Greedy Policies for the Small Gridworld

$\pi =$ equiprobable random action choices



actions



$R = -1$
on all transitions

$\gamma = 1$

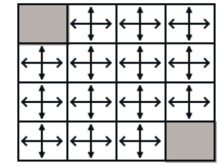
- An undiscounted episodic task
- Nonterminal states: 1, 2, . . ., 14;
- One terminal state (shown twice as shaded squares)
- Actions that would take agent off the grid leave state unchanged
- Reward is –1 until the terminal state is reached

V_k for the Random Policy

Greedy Policy w.r.t. V_k

$k = 0$

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

$k = 1$

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

$k = 10$

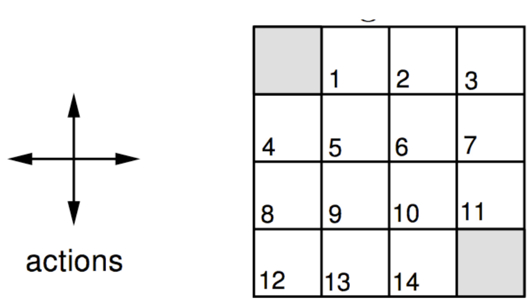
0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

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

$k = 0$

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

$k = 1$

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

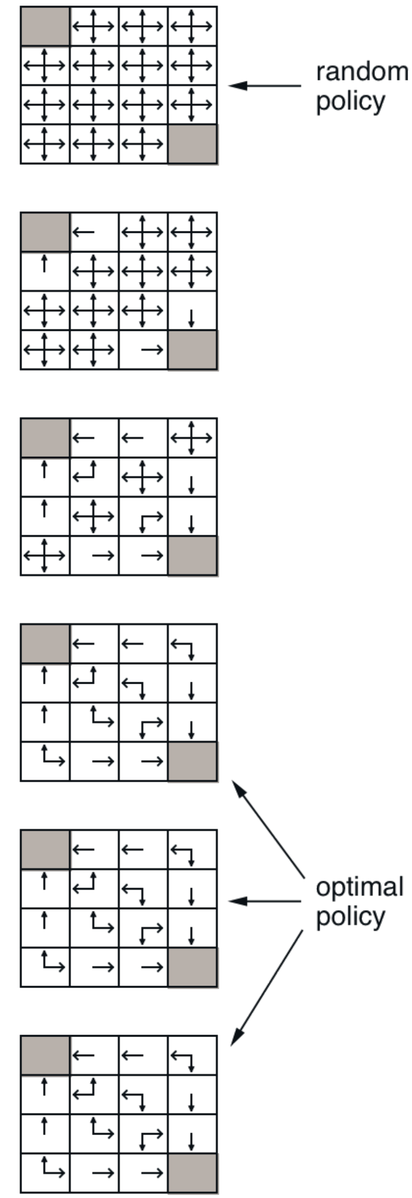
$k = 10$

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

$k = \infty$

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

Greedy Policy w.r.t. V_k



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
   - $\Delta \leftarrow 0$
   - For each $s \in \mathcal{S}$:
     - $v \leftarrow V(s)$
     - $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\big[r + \gamma V(s')\big]$
     - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

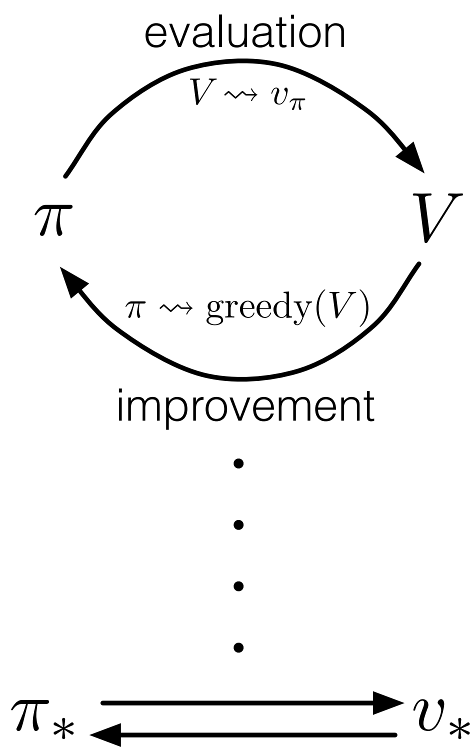
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   - $a \leftarrow \pi(s)$
   - $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
   - If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ $false$

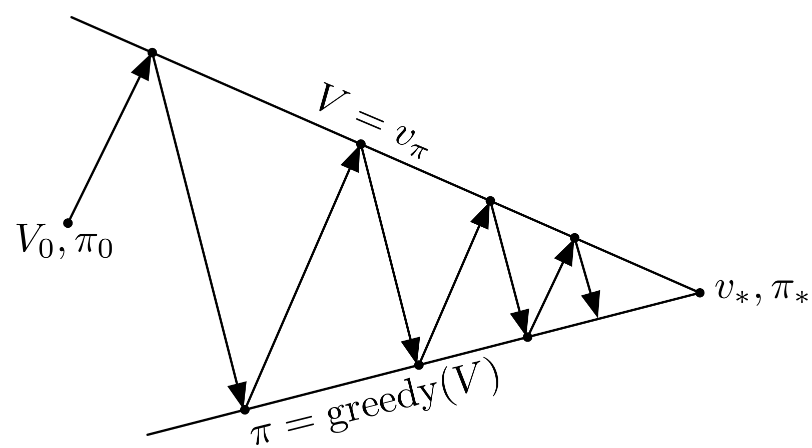
   If *policy-stable*, then stop and return V and π; else go to 2

Generalized Policy Iteration

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



A geometric metaphor for convergence of GPI:



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

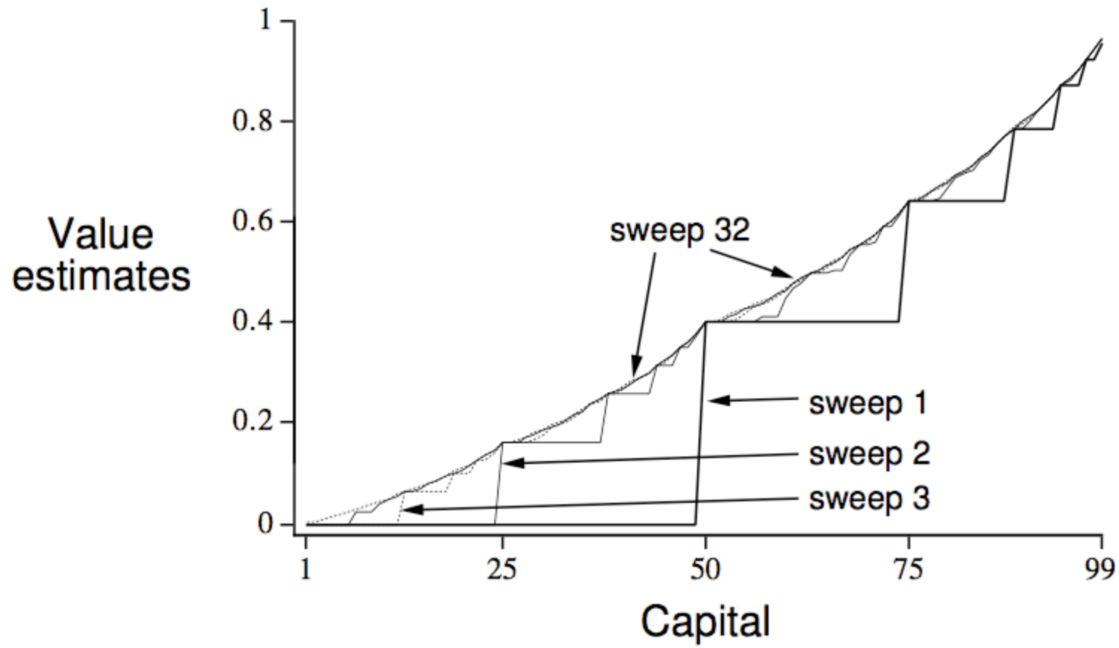
Output a deterministic policy, π, such that
$\quad \pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

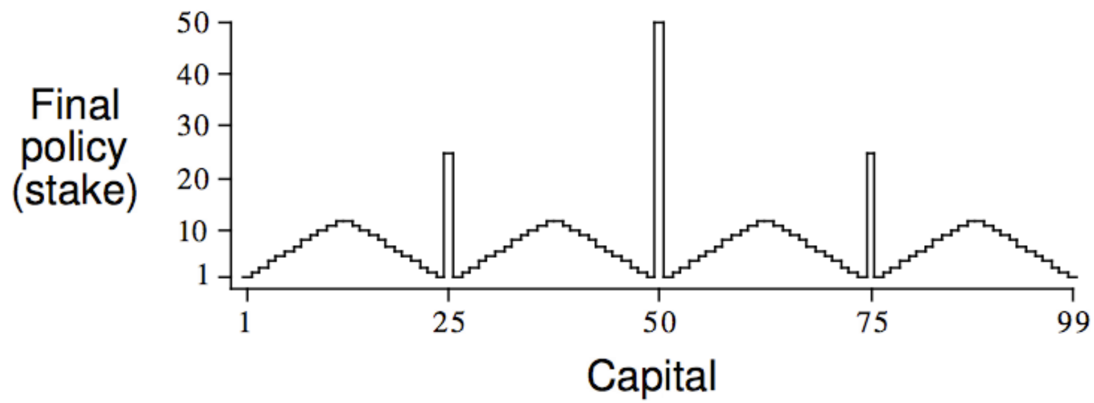
Gambler's Problem

- 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 $p = .4$

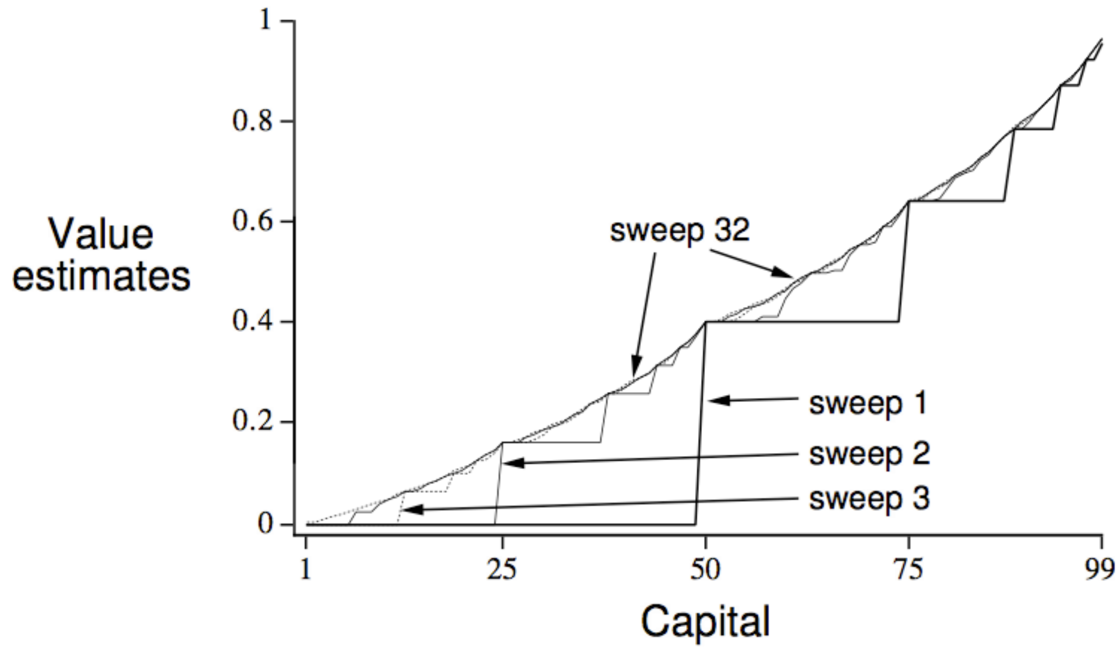
- States, Actions, Rewards? Discounting?

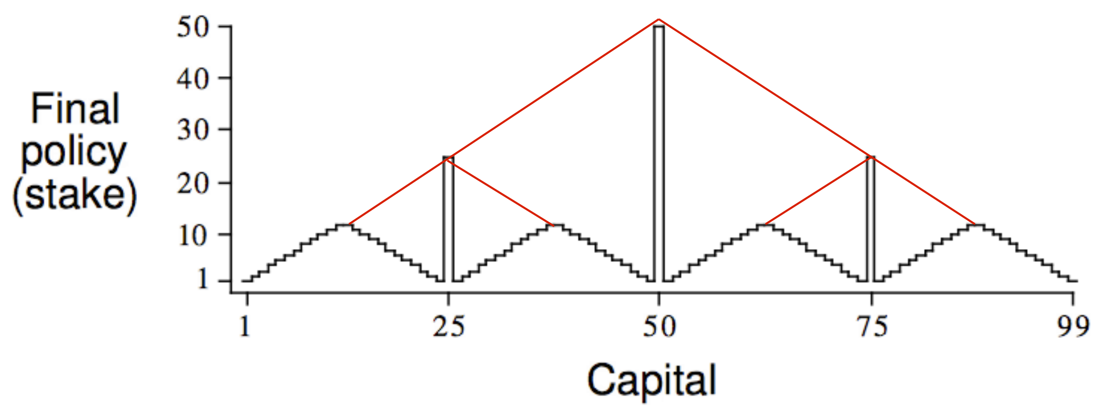
Gambler's Problem Solution





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

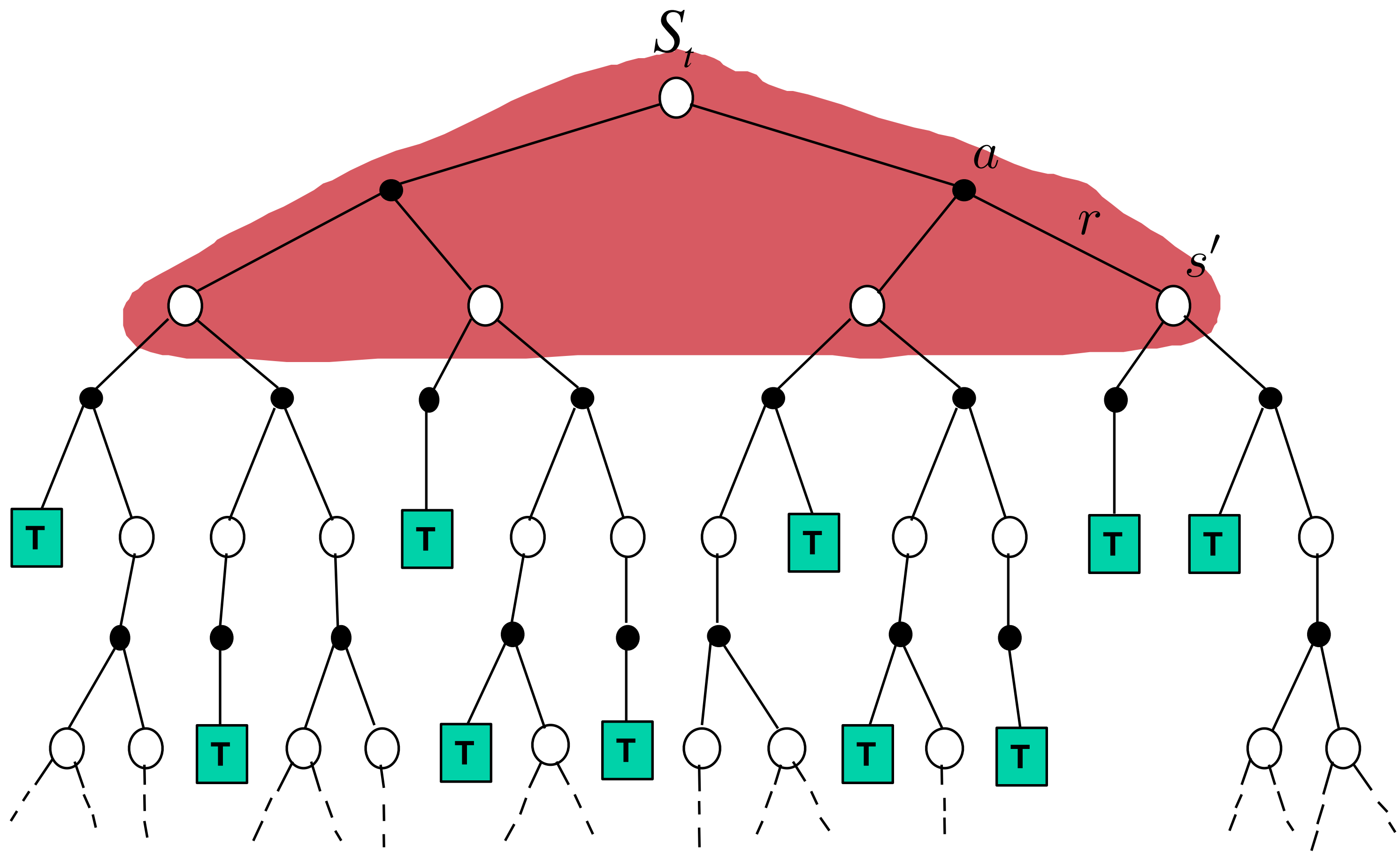
- To find an optimal policy is polynomial in the number of states…
- BUT, the number of states is often astronomical, e.g., often growing exponentially with the number of state variables (what Bellman called "the curse of dimensionality").
- In practice, classical DP can be applied to problems with a 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)

Dynamic Programming Policy Evaluation

$$V(S_t) \leftarrow E_\pi\Big[R_{t+1} + \gamma V(S_{t+1})\Big] = \sum_a \pi(a|S_t) \sum_{s',r} p(s',r|S_t,a)[r + \gamma V(s')]$$

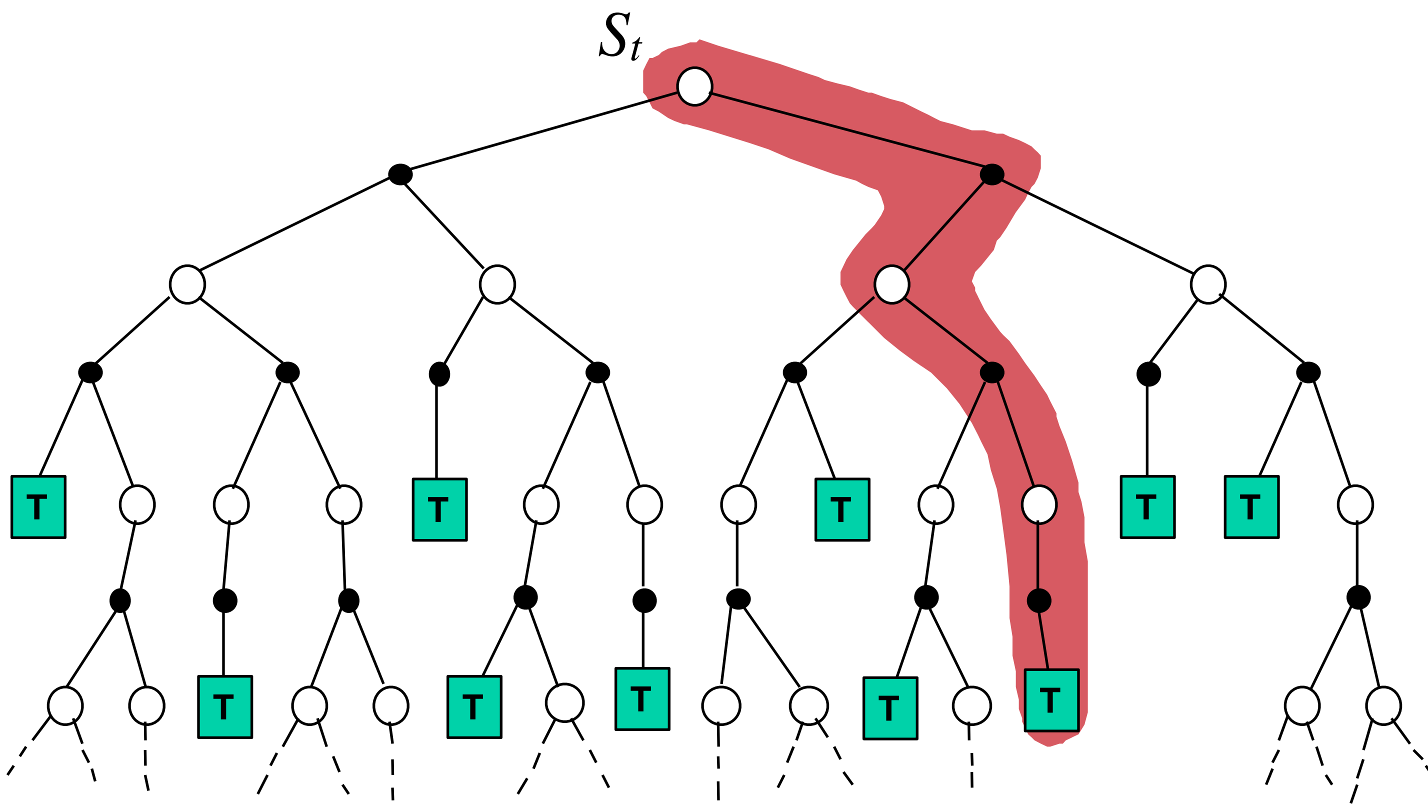


From Planning to Learning

- DP requires a *probability model* (as opposed to a generative or simulation model)
- We can interact with the world, learning a model (rewards and transitions) and then do DP
- This approach is called model-based RL
- Full probability model may hard to learn though
- Direct learning of the value function from interaction
- Still focusing on evaluating a fixed policy

Simple Monte Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha\left[G_t - V(S_t)\right]$$

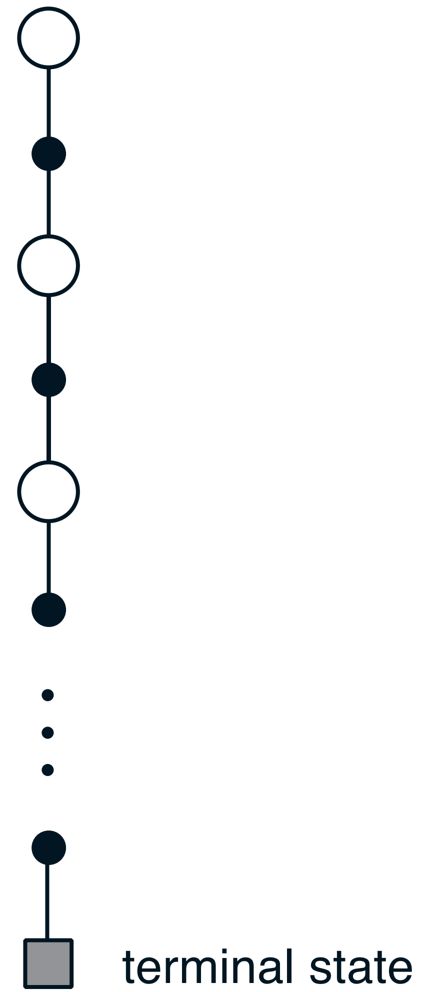


Monte Carlo Methods

- Monte Carlo methods are learning methods
  Experience → values, policy
- Monte Carlo methods can be used in two ways:
  - *model-free:* No model necessary and still attains optimality
  - *simulated:* Needs only a simulation, not a *full* model
- Monte Carlo methods learn from *complete* sample returns
  - Defined for episodic tasks (in the book)
- Like an associative version of a bandit method

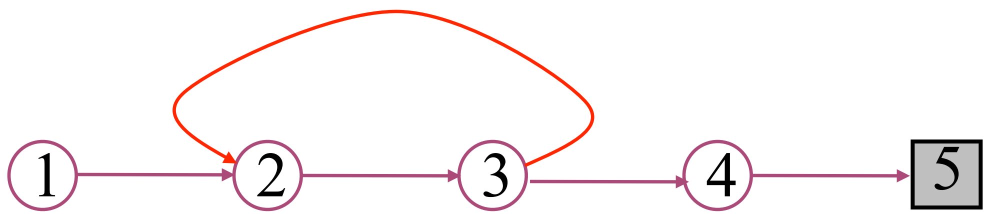
Backup diagram for Monte Carlo

- Entire rest of episode included
- Only one choice considered at each state (unlike DP)
  - thus, there will be an explore/exploit dilemma
- Does not bootstrap from successor states's values (unlike DP)
- Time required to estimate one state does not depend on the total number of states



Monte Carlo Policy Evaluation

- *Goal:* learn $v_\pi(s)$
- *Given:* some number of episodes under π which contain *s*
- *Idea:* Average returns observed after visits to s



- *Every-Visit MC:* average returns for *every* time *s* is visited in an episode
- *First-visit MC:* average returns only for *first* time *s* is visited in an episode
- Both converge asymptotically

First-visit Monte Carlo policy evaluation

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:

  Generate an episode using π

  For each state s appearing in the episode:

    $G \leftarrow$ return following the first occurrence of s

    Append G to $Returns(s)$

    $V(s) \leftarrow \text{average}(Returns(s))$

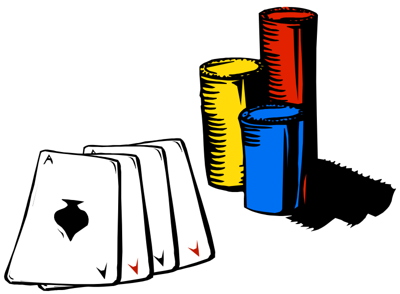
MC vs supervised regression

- Target returns can be viewed as a supervised label (true value we want to fit)
- State is the input
- We can use any function approximator to fit a function from states to returns! Neural nets, linear, nonparametric…

- *Unlike supervised learning: there is strong correlation between inputs and between outputs!*
- Due to the lack of iid assumptions, theoretical results from supervised learning cannot be directly applied

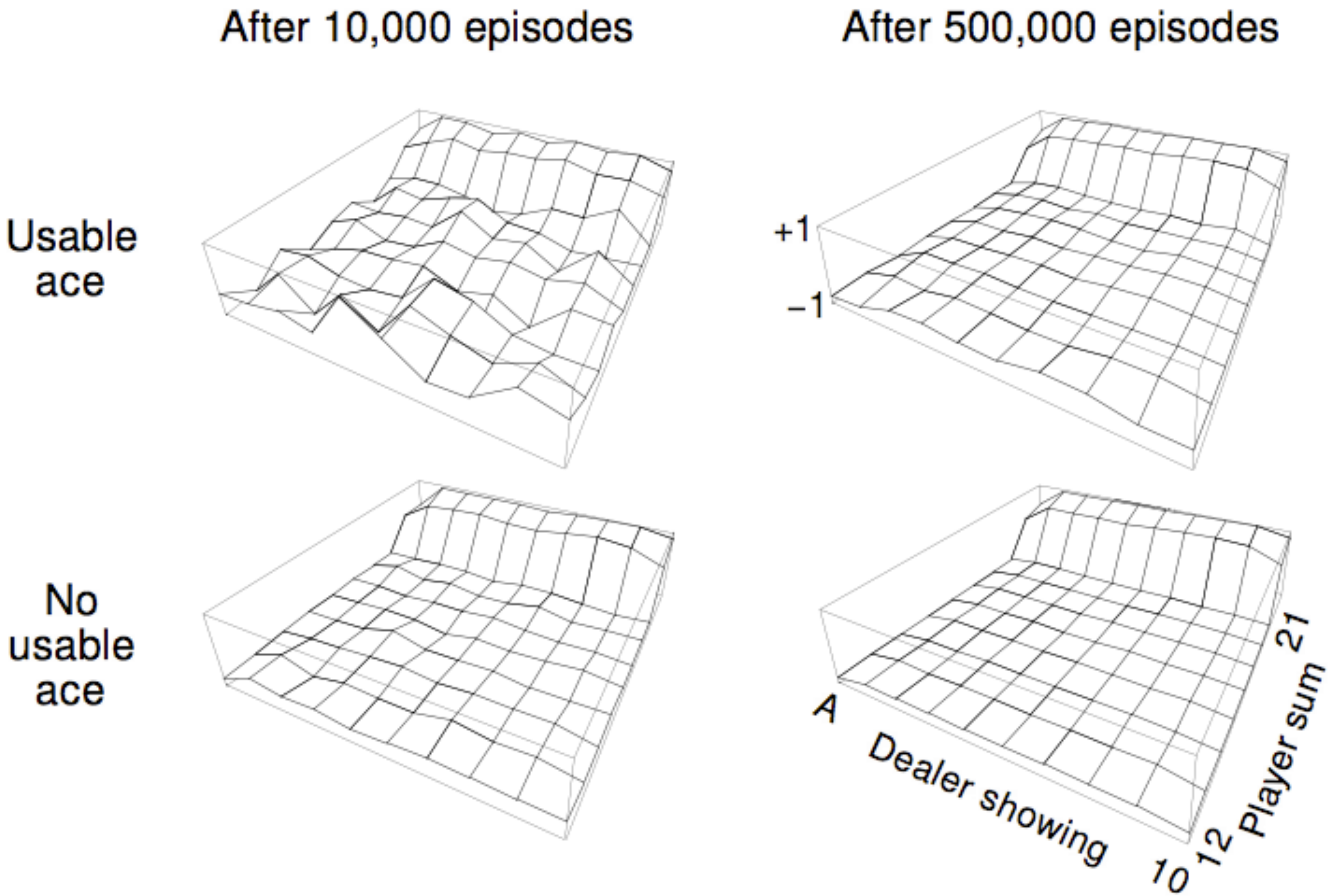
Blackjack example

- *Object:* Have your card sum be greater than the dealer's without exceeding 21.
- *States* (200 of them):
  - current sum (12-21)
  - dealer's showing card (ace-10)
  - do I have a useable ace?



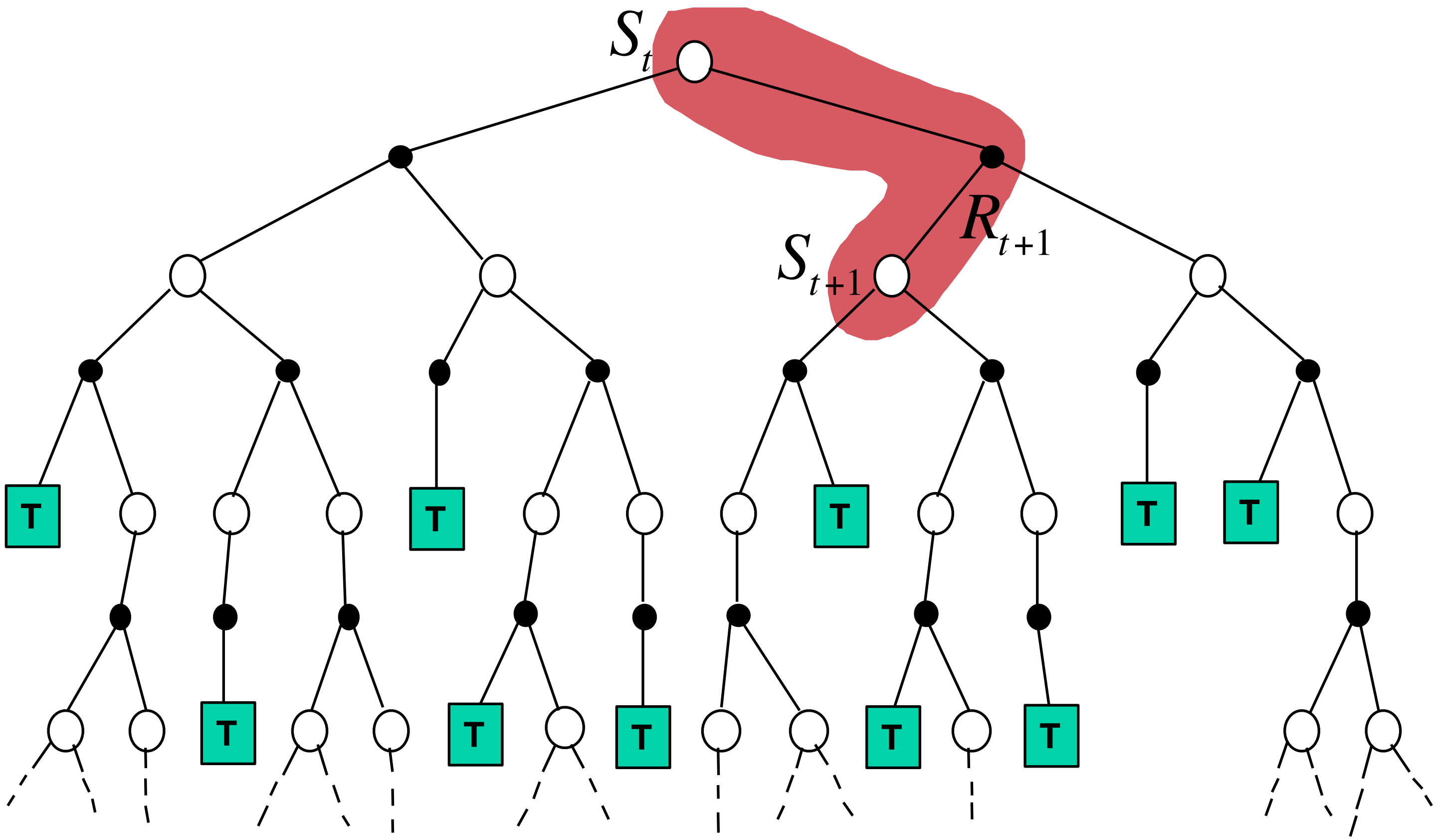
- *Reward:* +1 for winning, 0 for a draw, -1 for losing
- *Actions:* stick (stop receiving cards), hit (receive another card)
- *Policy:* Stick if my sum is 20 or 21, else hit
- No discounting ($\gamma = 1$)

Learned blackjack state-value functions



Simplest TD Method

$$V(S_t) \leftarrow V(S_t) + \alpha\left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]$$



TD methods bootstrap and sample

- Bootstrapping: update involves an *estimate*
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- Sampling: update does not involve an *expected value*
  - MC samples
  - DP does not sample
  - TD samples

TD Prediction

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

Recall: Simple every-visit Monte Carlo method:

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[G_t - V(S_t)\Big]$$

target: the actual return after time *t*

The simplest temporal-difference method TD(0):

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[\underbrace{R_{t+1} + \gamma V(S_{t+1})} - V(S_t)\Big]$$

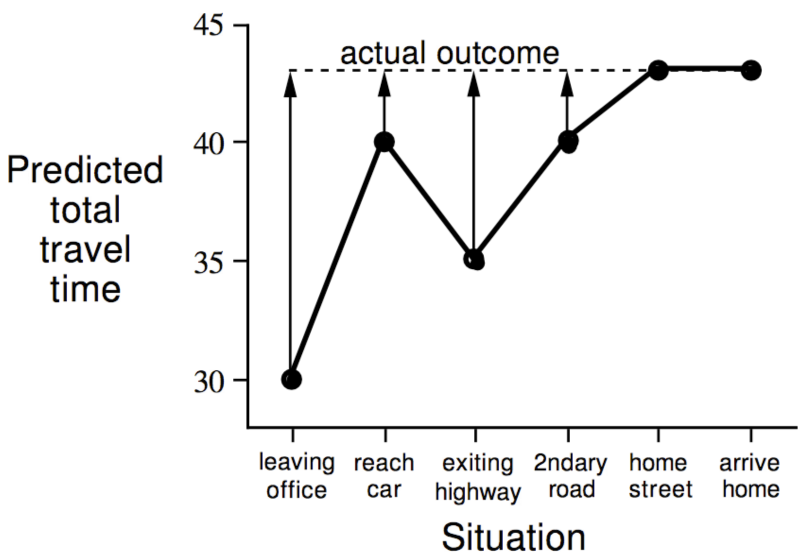
target: an estimate of the return

Example: Driving Home

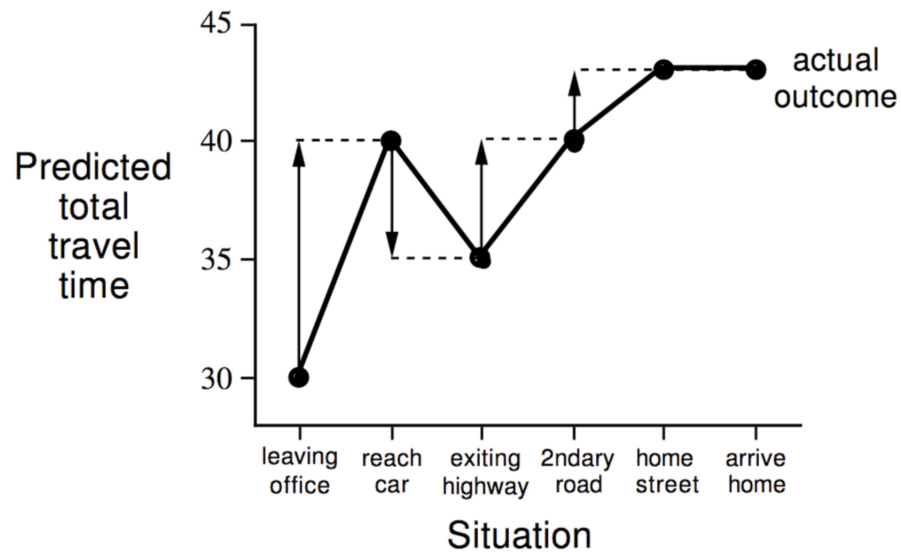
State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

Driving Home

Changes recommended by Monte Carlo methods (α=1)



Changes recommended by TD methods (α=1)



Advantages of TD Learning

- TD methods do not require a model of the environment, only experience
- TD, but not MC, methods can be fully incremental
  - You can learn before knowing the final outcome
    - Less memory
    - Less peak computation
  - You can learn without the final outcome
    - From incomplete sequences
- Both MC and TD converge (under certain assumptions to be detailed later), but which is faster? - Answer next time!