CMU 15-781

Lecture 11: Markov Decision Processes II

Teacher:
Gianni A. Di Caro

Recap: Defining MDPs

- Markov decision processes:
  - Set of states S
  - Start state s_0
  - Set of actions A
  - Transitions $\mathbf{P}(s'|s,a)$ (or $\mathbf{T}(s,a,s')$)
  - Rewards $R(s,a,s')$ (and discount γ)
- MDP quantities so far:
  - Policy π = Choice of action for each state
  - Utility/Value = sum of (discounted) rewards
  - Optimal policy π^* = Best choice, that max Utility

Utility and Policy selection

- *Utility of a state sequence (its return)*: sum of the discounted rewards obtained during the state sequence

$$U(s_t) = U([s_{t+1}, \ldots, s_\infty]) = \sum_{k=0}^{\infty} \gamma^k R(s_{t+k+1})$$

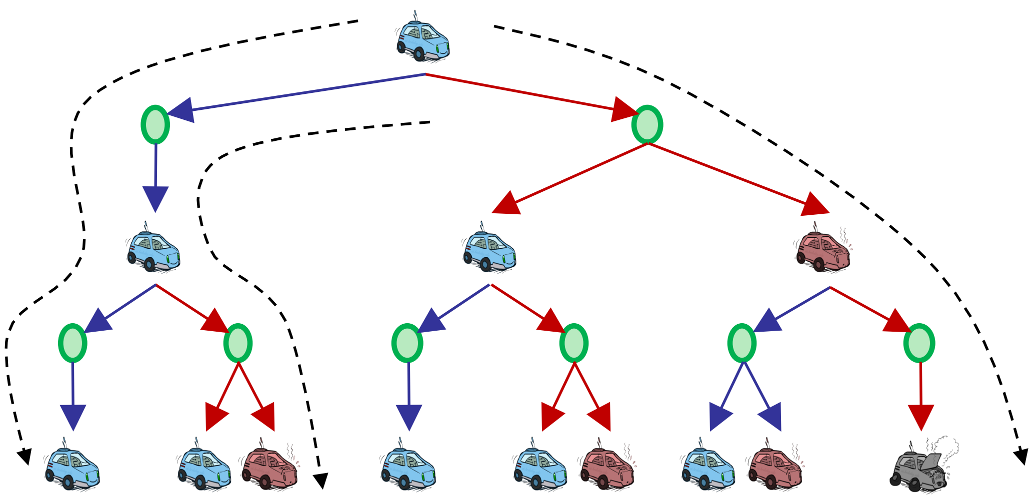
- Utility/return of the state sequence from current state s_t

$$U(s_t) = U([s_{t+1}, \ldots, s_\infty]) = \sum_{k=0}^{\infty} \gamma^t R(s_{t+k+1})$$

- The *rational agent* tries to select actions so that the sum of the discounted rewards it receives over the future is maximized (i.e., its utility is maximized)

Utility and Policy selection

- State/reward sequences depend on applied policy π, and effects of probabilistic transitions $P(s'| s, \pi(s))$ on actions
- $\rightarrow$ Rational agent aims to find the action policy π^* that maximizes the *expected value* of the utility for all $s_0 \in S$



VALUE FUNCTION AND Q-FUNCTION

- The **value $V^{\pi}(s)$ of a state** s under the policy π is the expected value of its return, the utility of all state sequences starting in s and applying π

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid s_0 = s\right]$$

State Value-function

- The **value $Q^{\pi}(s,a)$ of taking an action a in state s under policy π** is the expected return starting from s, taking action a, and thereafter following π:

$$Q^{\pi}(s,a) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid s_0 = s, a_0 = a\right]$$

Action Value-function

Value function

$$
\begin{aligned}
V^\pi(s) &= E\left[\sum_{t=0}^{\infty}\gamma^t R(s_{t+1}) \mid s_0 = s\right] \\
&\rightarrow E\left[\sum_{k=0}^{\infty}\gamma^k R(s_{t+k+1}) \mid s_t = s\right] \\
&= E\left[R(s_{t+1}) + \gamma R(s_{t+2}) + \gamma^2 R(s_{t+3}) + \ldots\right] \\
&= E\left[R(s_{t+1}) + \gamma\sum_{k=0}^{\infty}\gamma^k R(s_{k+t+2}) \mid s_t = s\right] \\
&= E\left[R(s_{t+1}) + \gamma V^\pi(s_{t+1}) \mid s_t = s\right]
\end{aligned}
$$

Bellman equation for Value function

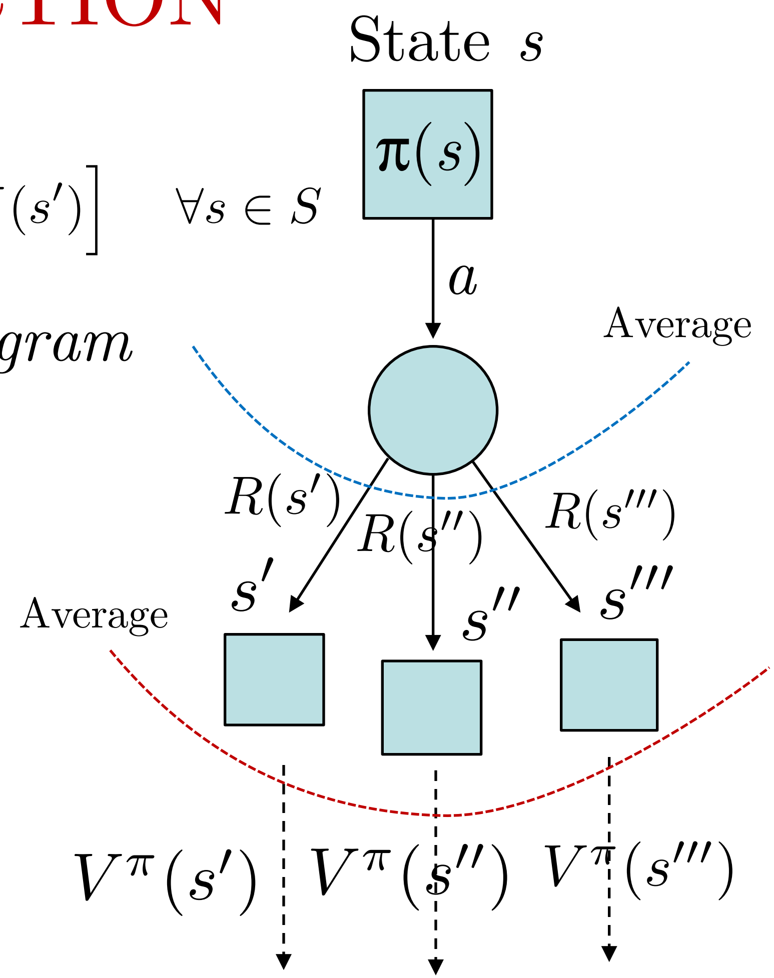
$$
\begin{aligned}
V^\pi(s) = & \quad E\left[R(s_{t+1}) + \gamma V^\pi(s_{t+1}) \mid s_t = s\right] \\
= & \quad \sum_{s' \in S} p\Big(s' \mid s, \pi(s)\Big)\Big[R(s, \pi(s), s') + \gamma V^\pi(s')\Big] \qquad \forall s \in S
\end{aligned}
$$

- **Expected immediate reward (short-term)** for taking action $\pi(s)$ prescribed by π for state s + **Expected future reward (long-term)** get after taking that action from that state and following π

Bellman equation for Value function

$$V^{\pi}(s) = \sum_{s' \in S} p\Big(s' \mid s, \pi(s)\Big)\Big[R(s, \pi(s), s') + \gamma V^{\pi}(s')\Big] \qquad \forall s \in S$$

Backup diagram



- Additivity of utility +
- Markov property

- Relation between the value of a state and that of its neighbors

- Recursive state equations that need to be mutually consistent

Bellman equation for Value function

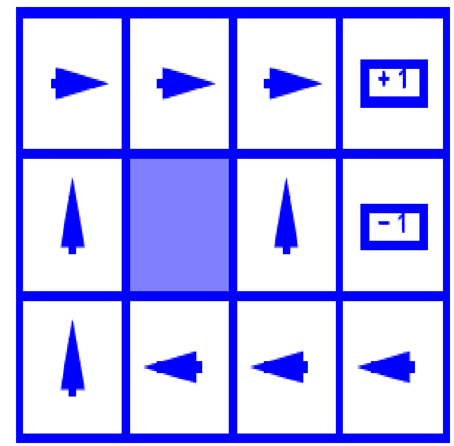
$$V^\pi(s) = \sum_{s' \in S} p\Big(s' \mid s, \pi(s)\Big)\Big[R(s, \pi(s), s') + \gamma V^\pi(s')\Big] \quad \forall s \in S$$

- How do we find V values for all states?
- $|S|$ linear equations in $|S|$ unknowns

Values for the grid world states

3	0.812	0.868	0.918	+ 1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388
	1	2	3	4

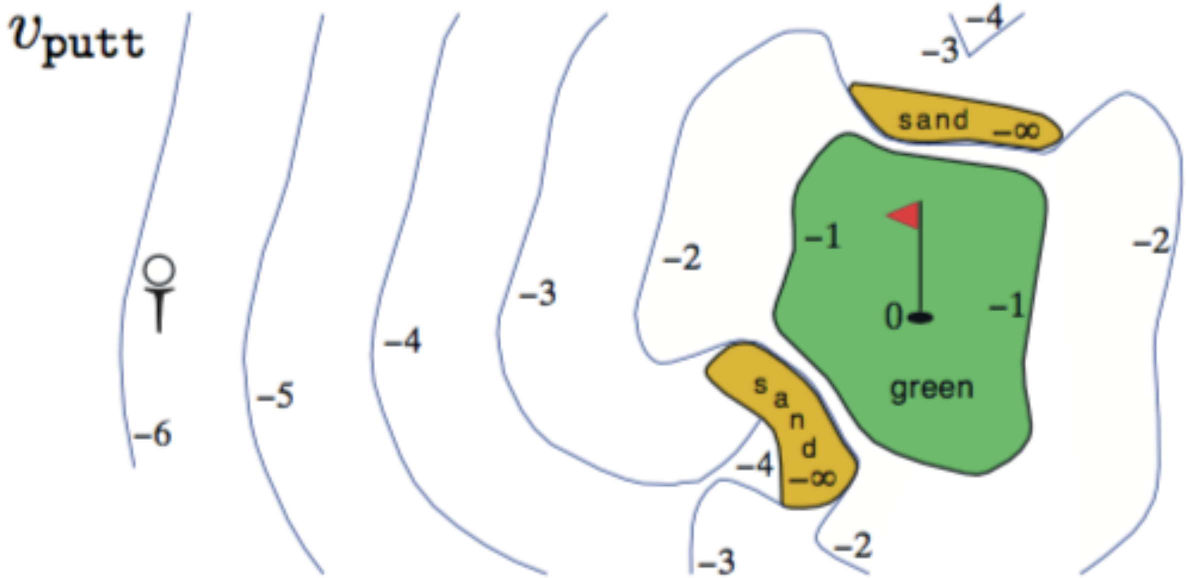
γ=1, R(s)=-00.4



π

(π is also optimal)

A golf club example



- *Value of a state*: negative of the number of strokes to the hole from that location
- *Actions*: which club to use $\{putter, driver\}$
- *Policy*: only use the *putter*

Optimal state and action value functions

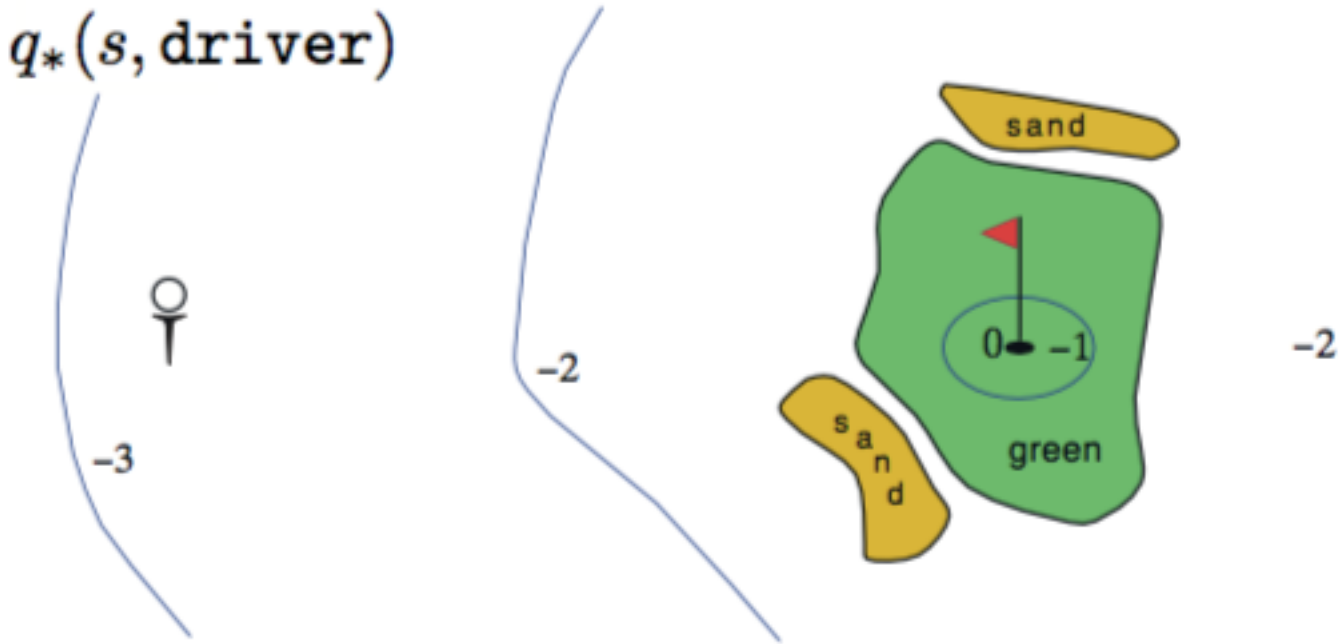
- $V^*(s)$ = Highest possible expected utility from s

$$V^*(s) = \max_{\pi} V^{\pi}(s) \quad \forall s \in S$$

- Optimal action-value function:

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \quad \forall\ s \in S, a \in A$$

Optimal action-value example



- Optimal action-values for choosing club=*driver*, and afterward select either the *driver* or the *putter*, whichever is better.

BELLMAN *OPTIMALITY* EQUATIONS FOR V

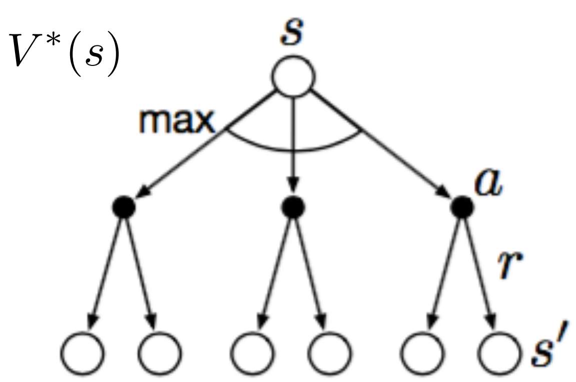
- The value $V^*(s) = V^{\pi^*}(s)$ of a state s under the *optimal policy* π^* must equal the expected utility for the *best action* from that state $\rightarrow$

$$
\begin{aligned}
V^*(s) &= \max_{a \in A(s)} Q^{\pi^*}(s,a) \\
&= \max_{a \in A(s)} E\Big[R(s_{t+1}) + \gamma V^*(s_{t+1}) \;\Big|\; s_t = s, a_t = a\Big] \\
&= \max_{a \in A(s)} \sum_{s' \in S} p(s'|s,a)\big[R(s,a,s') + \gamma V^*(s')\big]
\end{aligned}
$$

Bellman optimality equations for V

$$V^*(s) = \max_{a \in A(s)} \sum_{s' \in S} p(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right] \quad \forall s \in S$$

- $|S|$ *non-linear* equations in $|S|$ unknowns
- The vector $\boldsymbol{V}^*$ is the unique solution to the system

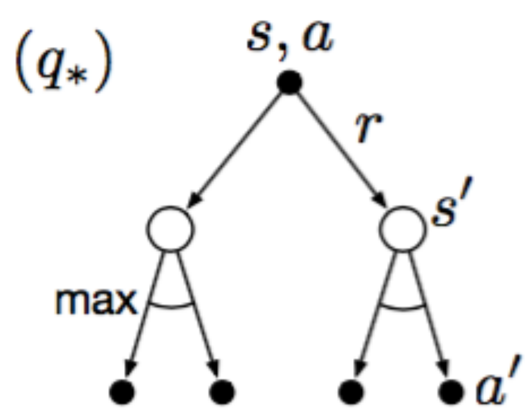


BELLMAN OPTIMALITY EQUATIONS FOR Q

$$
\begin{aligned}
Q^*(s,a) = \quad & E\Big[R(s_{t+1}) + \gamma \max_{a'} Q^*(s_{t+1},a') \mid s_t = s, a_t = a\Big] \\
= \quad & \sum_{s' \in S} p\Big(s' \mid s,a\Big)\Big[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')\Big]
\end{aligned}
$$

$$\forall s \in S, a \in A$$

- $|S| \times |A(s)|$ *non-linear* equations
- The vector $\boldsymbol{Q}^*$ is the unique solution to the system



Finding the optimal policy

- If we have computed $\boldsymbol{V^*} \rightarrow$

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} p\Big(s' \mid s, a\Big)\Big[R(s,a,s') + \gamma V^*(s')\Big]$$

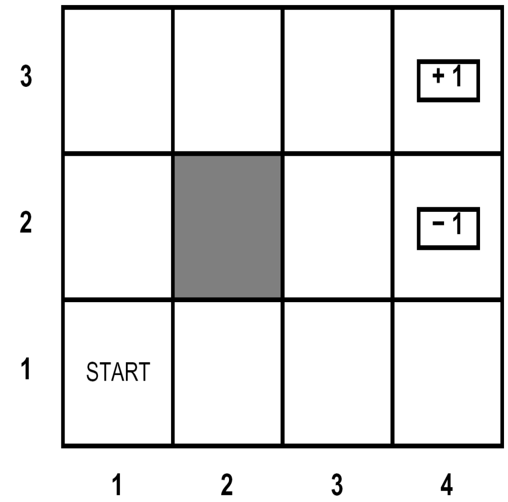
It's *one-step ahead* search
$\rightarrow$ *Greedy policy* with respect to V^*

- If we have computed $\boldsymbol{Q^*} \rightarrow$

$$\begin{aligned} \pi^*(s) = & \arg\max_{a \in A(s)} Q^*(s,a) \\ = & \arg\max_{a \in A(s)} \sum_{s' \in S} p\Big(s' \mid s, a\Big)\Big[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')\Big] \end{aligned}$$

Optimal V* for the grid world

$$V^*(s) = \max_{a \in A(s)} \sum_{s' \in S} p(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$



For our grid world (find that *up* is the best),
Let's omit the rewards, assuming R=0:

- $V^*(1,1) = \gamma \max\{u, l, d, r\}$ [

  $\{0.8V^*(1,2) + 0.1V^*(2,1) + 0.1V^*(1,1)\}$, *up*

  $\{0.9V^*(1,1) + 0.1V^*(1,2)\}$, *left*

  $\{0.9V^*(1,1) + 0.1V^*(2,1)\}$, *down*

  $\{0.8V^*(2,1) + 0.1V^*(1,2) + 0.1V^*(1,1)\}$ *right*

  ]

V* FOR RECYCLING ROBOT

Two states {high, low}

$$
\begin{aligned}
v_*(\mathtt{h}) &= \max\left\{\begin{array}{l} p(\mathtt{h}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{l}) + \gamma v_*(\mathtt{l})], \\ p(\mathtt{h}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{l}) + \gamma v_*(\mathtt{l})] \end{array}\right\} \\
&= \max\left\{\begin{array}{l} \alpha[r_{\mathtt{s}} + \gamma v_*(\mathtt{h})] + (1-\alpha)[r_{\mathtt{s}} + \gamma v_*(\mathtt{l})], \\ 1[r_{\mathtt{w}} + \gamma v_*(\mathtt{h})] + 0[r_{\mathtt{w}} + \gamma v_*(\mathtt{l})] \end{array}\right\} \\
&= \max\left\{\begin{array}{l} r_{\mathtt{s}} + \gamma[\alpha v_*(\mathtt{h}) + (1-\alpha)v_*(\mathtt{l})], \\ r_{\mathtt{w}} + \gamma v_*(\mathtt{h}) \end{array}\right\}.
\end{aligned}
$$

$$
v_*(\mathtt{l}) = \max\left\{\begin{array}{l} \beta r_{\mathtt{s}} - 3(1-\beta) + \gamma[(1-\beta)v_*(\mathtt{h}) + \beta v_*(\mathtt{l})] \\ r_{\mathtt{w}} + \gamma v_*(\mathtt{l}), \\ \gamma v_*(\mathtt{h}) \end{array}\right\}
$$

How do we solve the Bellman optimality equations?

$$V^*(s) = \max_{a \in A(s)} \sum_{s' \in S} p(s'|s,a)\big[R(s,a,s') + \gamma V^*(s')\big] \quad \forall s \in S$$

- |S| *non-linear* equations in |S| unknowns (because of max)
- Equations suggest an *iterative*, recursive update approach

$$V_{k+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s' \in S} p(s'|s,a)\big[R(s,a,s') + \gamma V_k(s')\big] \quad \forall s \in S$$

State Backup: $\boldsymbol{V}_{\mathrm{k+1}} = \boldsymbol{B}\,\boldsymbol{V}_{\mathrm{k}}$

VALUE ITERATION

1. **Initialization:**
   Initialize arbitrarily $V(s)\ \forall s \in S$ (e.g., $V(s) = 0$)

2. **Value Iteration:**

Repeat

$\Delta \leftarrow 0$

$k \leftarrow 0$

Foreach $s \in S$

Sweep state space

$v \leftarrow V(s)$

Bellman update / Backup operator

$$V_{k+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s' \in S} p(s'|s,a)\big[R(s,a,s') + \gamma V_k(s')\big]$$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

$k \leftarrow k + 1$

Until $\Delta < \theta$ (small positive number)

Or a criterion based on V estimation error (see later)

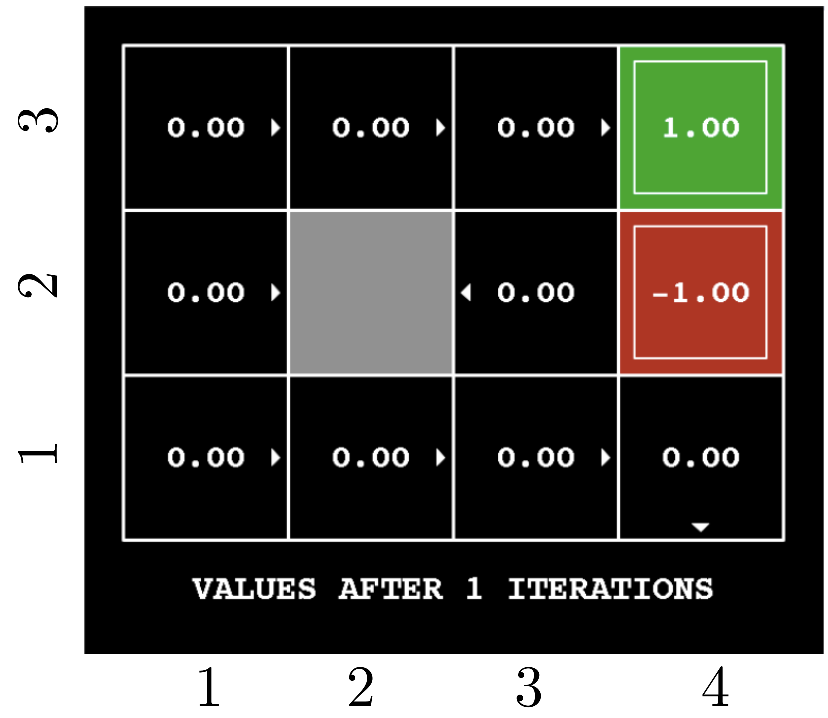
Output a deterministic policy $\pi \approx \pi^*$, such that

$$\pi(s) = \arg\max_{a \in A(s)} \sum_{s' \in S} p(s'|s,a)\big[R(s,a,s') + \gamma V_k(s')\big]$$

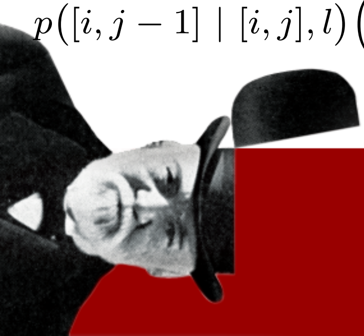
Value Iteration on Grid World

R(s)=0 everywhere except at the terminal states

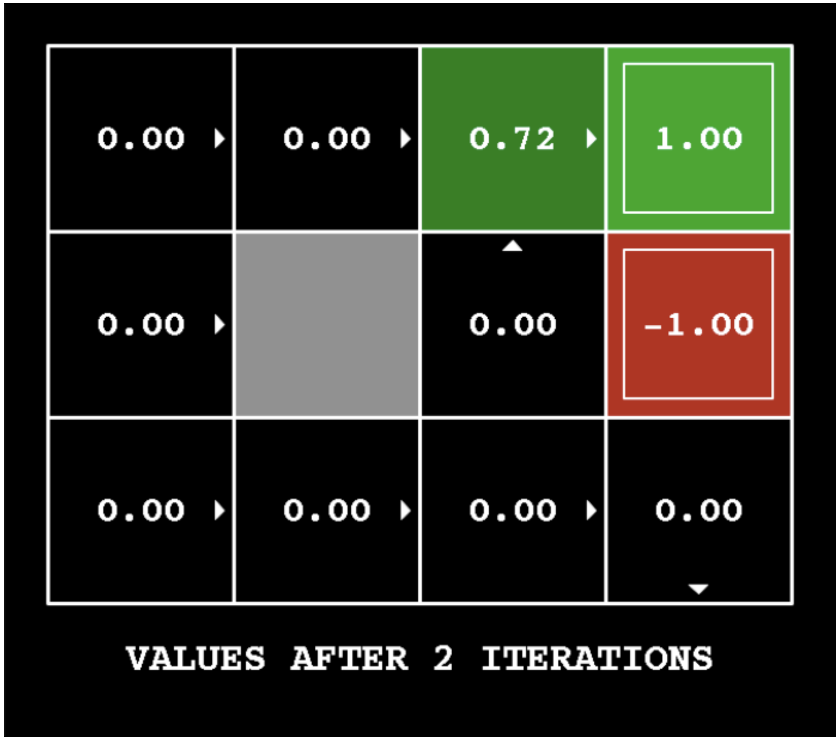
$V_k(s) = 0$ at $k=0$

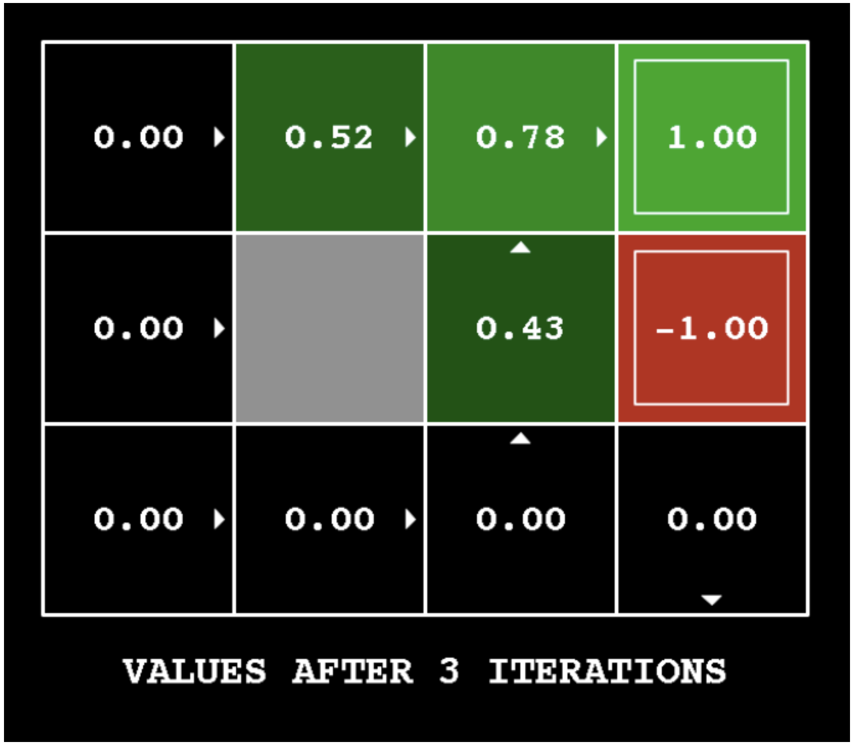


$$
\begin{aligned}
&V_{k+1}([i,j]) \leftarrow \max[\;] \\
&\max\Bigg[\bigg\{p\big([i+1,j]\mid[i,j],u\big)\Big(R([i+1,j])+\gamma V_k([i+1,j])\Big)+p\big([i-1,j]\mid[i,j],u\big)\Big(R([i-1,j])+\gamma V_k([i-1,j])\Big)+ \\
&p\big([i,j+1]\mid[i,j],u\big)\Big(R([i,j+1])+\gamma V_k([i,j+1])\Big)+p\big([i,j-1]\mid[i,j],u\big)\Big(R([i,j-1])+\gamma V_k([i,j-1])\Big)+ \\
&p\big([i,j]\mid[i,j],u\big)\Big(R([i,j])+\gamma V_k([i,j])\Big)\bigg\}_{up}, \\
&\bigg\{p\big([i+1,j]\mid[i,j],d\big)\Big(R([i+1,j])+\gamma V_k([i+1,j])\Big)+ \\
&p\big([i-1,j]\mid[i,j],d\big)\Big(R([i-1,j])+\gamma V_k([i-1,j])\Big)+p\big([i,j+1]\mid[i,j],d\big)\Big(R([i,j+1])+\gamma V_k([i,j+1])\Big)+ \\
&p\big([i,j-1]\mid[i,j],d\big)\Big(R([i,j-1])+\gamma V_k([i,j-1])\Big)+p\big([i,j]\mid[i,j],d\big)\Big(R([i,j])+\gamma V_k([i,j])\Big)\bigg\}_{down}, \\
&\bigg\{p\big([i+1,j]\mid[i,j],r\big)\Big(R([i+1,j])+\gamma V_k([i+1,j])\Big)+ \\
&p\big([i-1,j]\mid[i,j],r\big)\Big(R([i-1,j])+\gamma V_k([i-1,j])\Big)+p\big([i,j+1]\mid[i,j],r\big)\Big(R([i,j+1])+\gamma V_k([i,j+1])\Big)+ \\
&p\big([i,j-1]\mid[i,j],r\big)\Big(R([i,j-1])+\gamma V_k([i,j-1])\Big)+p\big([i,j]\mid[i,j],r\big)\Big(R([i,j])+\gamma V_k([i,j])\Big)\bigg\}_{right}, \\
&\bigg\{p\big([i+1,j]\mid[i,j],l\big)\Big(R([i+1,j])+\gamma V_k([i+1,j])\Big)+ \\
&p\big([i-1,j]\mid[i,j],l\big)\Big(R([i-1,j])+\gamma V_k([i-1,j])\Big)+p\big([i,j+1]\mid[i,j],l\big)\Big(R([i,j+1])+\gamma V_k([i,j+1])\Big)+ \\
&p\big([i,j-1]\mid[i,j],l\big)\Big(R([i,j-1])+\gamma V_k([i,j-1])\Big)+p\big([i,j]\mid[i,j],l\big)\Big(R([i,j])+\gamma V_k([i,j])\Big)\bigg\}_{left}\Bigg]
\end{aligned}
$$



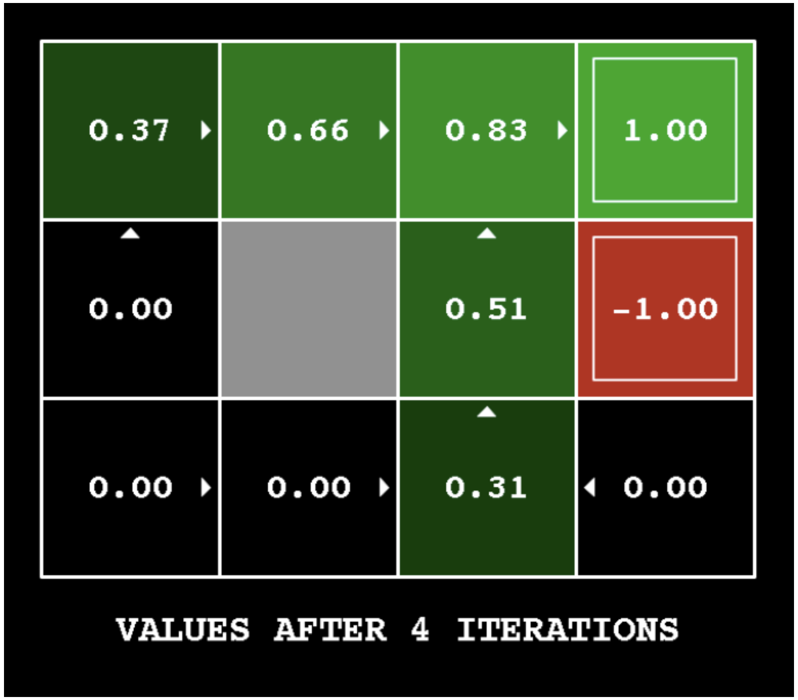
Value Iteration on Grid World

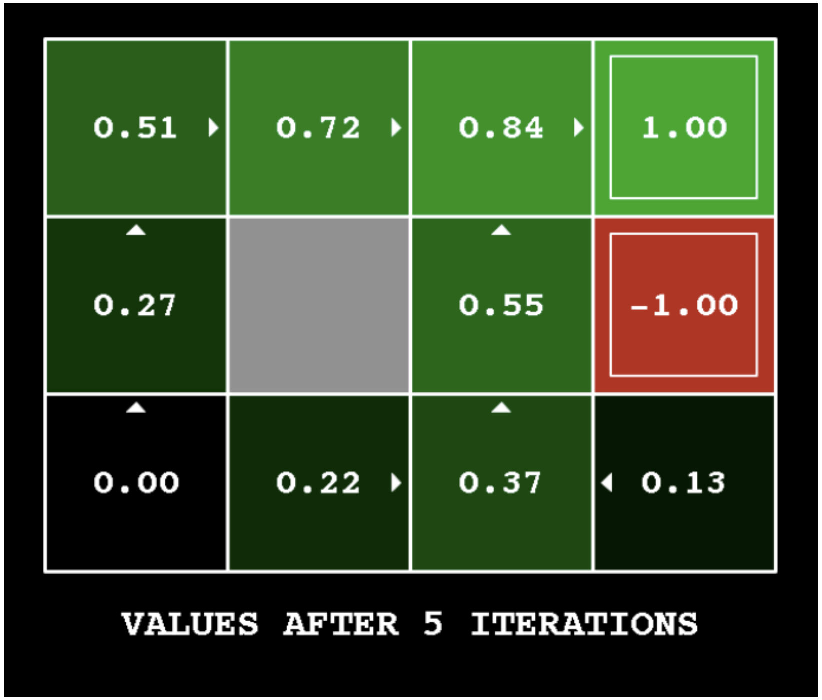




Example figures from P. Abbeel

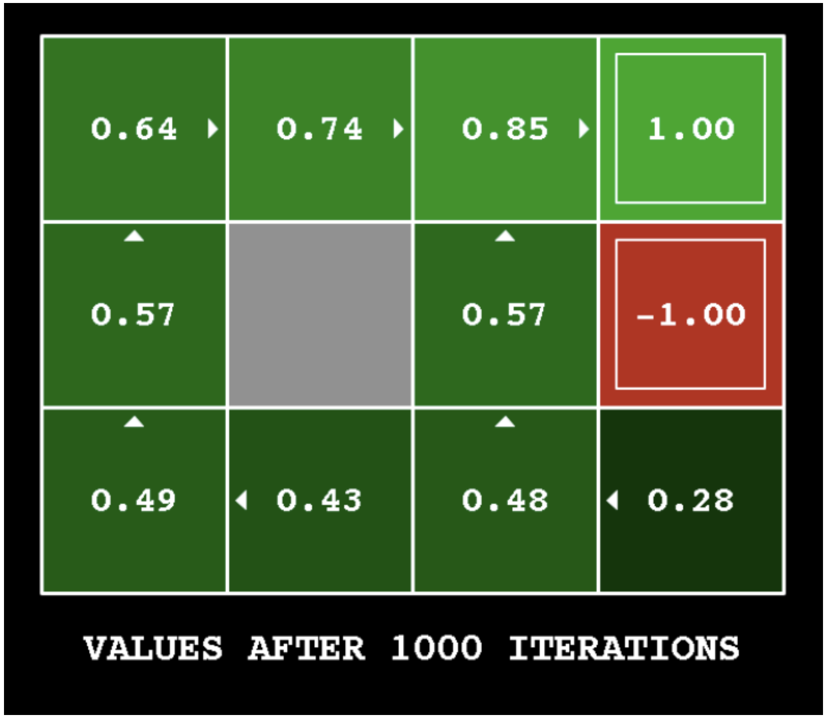
Value Iteration on Grid World





Example figures from P. Abbeel

Value Iteration on Grid World



Example figures from P. Abbeel

Computational Cost for 1 Update of $V(s)$ for all s in Value Iteration?

- For all states s

$$V_{k+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s' \in S} p(s'|s,a)\left[R(s,a,s') + \gamma V_k(s')\right]$$

$$|\mathbf{A}| \cdot |\mathbf{S}|^{\mathbf{2}}$$

Does Value Iteration converge?

- Yes, **it does converge** to a unique solution, $\boldsymbol{V}^*$
- It does because a backup operation **B** is a contraction by a factor γ on the space of the state value vectors:
- If apply **B** to two different value functions, their max norm distance shrinks

$$\text{max norm: } ||\boldsymbol{V}|| = \max_{s \in S} |V(s)|$$

$$||\boldsymbol{V} - \boldsymbol{V}'|| = \text{max difference between two corresponding states}$$

$$\Downarrow$$

$$||\boldsymbol{B}\boldsymbol{V}_k - \boldsymbol{B}\boldsymbol{V}'_k|| \leq \gamma ||\boldsymbol{V}_k - \boldsymbol{V}'_k||$$

Bellman Operator is a Contraction

|| V-V'|| = Infinity norm (find max difference over all states, e.g. max(s) |V(s) – V'(s) |

$$\left\|BV - BV'\right\| = \left\| \max_a \left[R(s,a) + \gamma \sum_{s_j \in S} p(s_j \mid s_i, a) V(s_j) \right] - \max_{a'} \left[R(s,a') - \gamma \sum_{s_j \in S} p(s_j \mid s_i, a') V'(s_j) \right] \right\|$$

$$\leq \max_a \left\| \left[R(s,a) + \gamma \sum_{s_j \in S} p(s_j \mid s_i, a) V(s_j) - R(s,a) + \gamma \sum_{s_j \in S} p(s_j \mid s_i, a) V'(s_j) \right] \right\|$$

$$\leq \gamma \max_a \left\| \left[\sum_{s_j \in S} p(s_j \mid s_i, a) V(s_j) - \sum_{s_j \in S} p(s_j \mid s_i, a) V'(s_j) \right] \right\|$$

$$= \gamma \max_a \left\| \left[\sum_{s_j \in S} p(s_j \mid s_i, a) (V(s_j) - V'(s_j)) \right] \right\|$$

$$\leq \gamma \max_{a, s_i} \sum_{s_j \in S} p(s_j \mid s_i, a) \left| V(s_j) - V'(s_j) \right|$$

$$\leq \gamma \max_{a, s_i} \sum_{s_j \in S} p(s_j \mid s_i, a) \left\| V - V' \right\|$$

$$= \gamma \left\| V - V' \right\|$$

Holds for $\gamma < 1$

Properties of Contraction

- Only has 1 fixed point (the point reach if apply a contraction operator many times)
  - If had two, then would not get closer when apply contraction function, violating definition of contraction
- When apply contraction function to any argument, value must get closer to fixed point
  - Fixed point doesn't move
  - Repeated function applications yield fixed point

Value Iteration Converges

- Value iteration converges to unique solution which is the optimal value function
- Proof: $$\lim_{k\to\infty} V_k = V^*$$

$$\|V_{k+1} - V^*\|_\infty = \|BV_k - V^*\|_\infty \le \gamma \|V_k - V^*\|_\infty \le \dots$$
$$\le \gamma^{k+1} \|V_0 - V^*\|_\infty \to 0$$

CONVERGENCE RATE?

- $\|V_k - V^*\|$ = *error* in the estimate V_k, therefore by using the previous relation:

$$\|V_{k+1} - V^*\|_\infty = \|BV_k - V^*\|_\infty \le \gamma \|V_k - V^*\|_\infty \le \ldots$$

$$\le \gamma^{k+1} \|V_0 - V^*\|_\infty \to 0$$

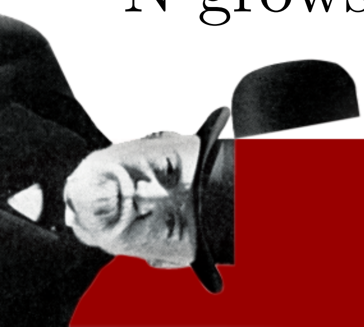
- $\to$ Error is reduced by a factor of at least γ on each iteration
  $\to$ Error decreases as γ^N after N iterations
  $\to$ *Exponentially fast convergence*

CONVERGENCE RATE?

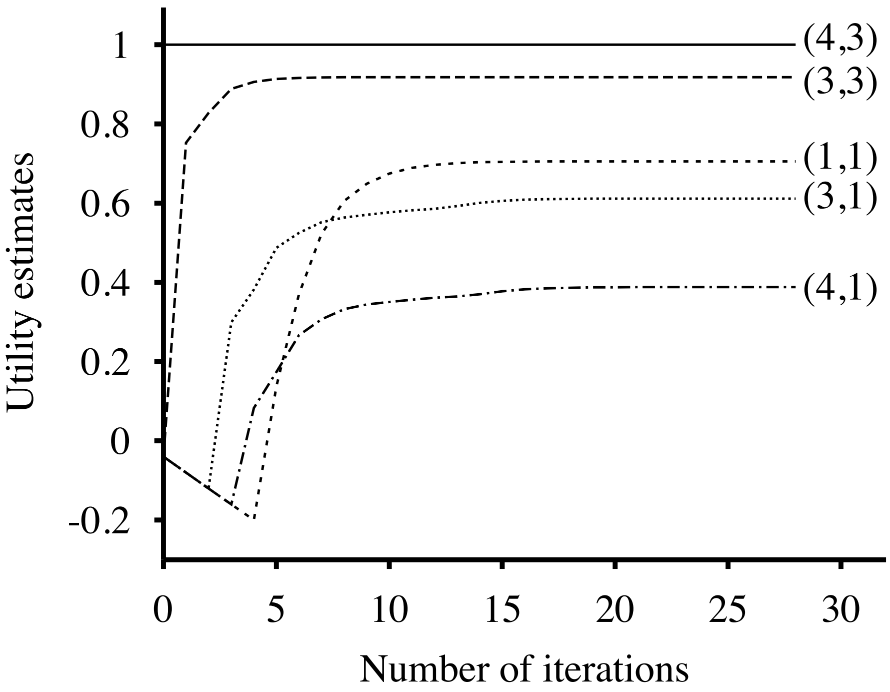
- #Iterations to reach an error bound ε?
  - Utilities of all states are bounded by $\pm R_{max}/(1-\gamma)$ (this is the sum of the geometric series representing the utility)
  - → Maximum initial error: $\varepsilon_0 = ||\boldsymbol{V}_0 - \boldsymbol{V}^*|| \leq 2R_{max}/(1-\gamma)$
  - After N iterations: $\varepsilon_N \leq \gamma^N 2R_{max}/(1-\gamma)$
- The #iterations to have an error of at most ε grows with γ:

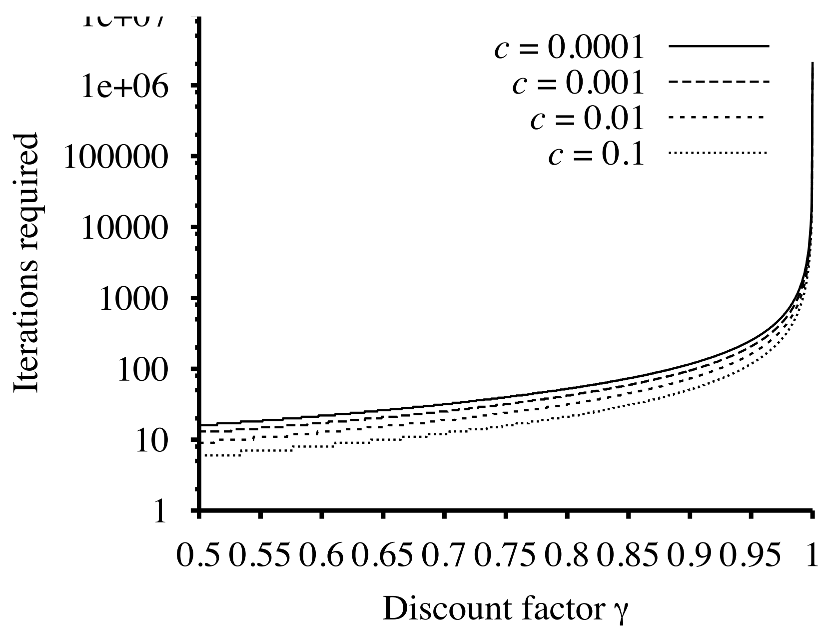
$$N = \lceil \log(2R_{max}/\epsilon(1-\gamma))/\log(1/\gamma) \rceil$$

- N grows rapidly as γ is selected to be close to 1



Convergence in the grid world





Bellman update error and θ

- **Question:** how do we set θ in the termination condition?
- Previous error bounds, that are quite conservative and might not be a good indicator on when to stop
- A better bound relates the error $\|\boldsymbol{V}_{k+1} - \boldsymbol{V}^*\|$ in $\boldsymbol{V}$ to the size $\|\boldsymbol{V}_{k+1} - \boldsymbol{V}_k\|$ of the Bellman update at each iteration:

$$\text{If } \|V_{k+1} - V_k\| < \varepsilon(1-\gamma)/\gamma \Rightarrow \|V_{k+1} - V^*\| < \varepsilon$$

(Ronald J. Williams and Leemon C. Baird III. Tight performance bounds on greedy policies based on imperfect value functions. Technical Report NU-CCS-93-14, 1993)

- $\Rightarrow$ A good choice is: $\theta = \varepsilon(1-\gamma)/\gamma$, based on desired ε, γ

Policy Loss

- **Question:** do we really need to wait for convergence in the value functions before to use the value functions to define a good (greedy) policy?

- $||V^{\pi(k)} - V^*|| =$ **Policy loss**: the max the agent can lose by executing $\pi(k)$ instead of π^* $\rightarrow$ *This is what matters!*

- $\pi(k)$ is the *greedy policy* obtained at iteration k from V_k and $V^{\pi(k)}(s)$ is value of state s applying greedy policy $\pi(k)$

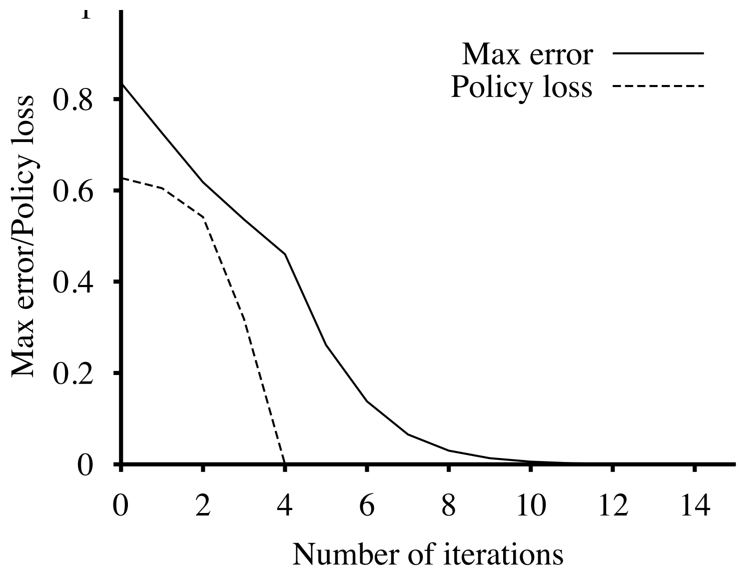
Policy Loss

- Using previous results for the bound, it can be shown that:

$$\text{If } \|V_k - V^*\| < \varepsilon \;\Rightarrow\; \|V^{\pi(k)} - V^*\| < 2\varepsilon\,\gamma/(1-\gamma)$$

- **In practice,** it often occurs that $\pi(k)$ becomes optimal long before V_k has converged!

Grid World: After k=4, the greedy policy is optimal, while the estimation error in V_k is still 0.46



Policy Iteration: Motivation

- Is the error in value function estimation really essential to extract the optimal policy?
  (which is what the agent needs)

- Not really, if one action (the optimal) gets really better than the others, the exact magnitude of the $V(s)$ doesn't really matter to select the action in the greedy policy (i.e., don't need "precise" V values), more important are relative proportions

Policy Iteration

- **Policy iteration, alternate:**
  - Policy evaluation: given a policy calculate the value of each state as that policy were executed
  - Policy improvement: Calculate a new policy according to the maximization of the utilities using one-step look-ahead based on current policy

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

Figure from Sutton and Barto

(Generalized) Policy Iteration

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

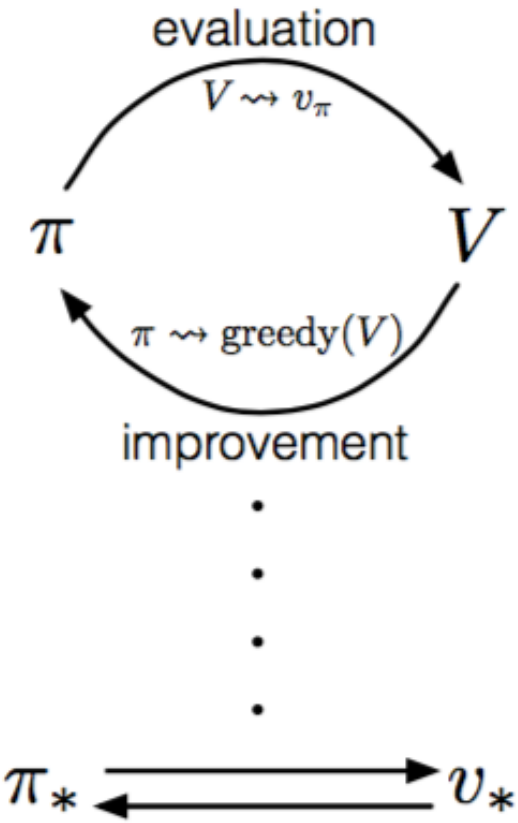


Figure from Sutton and Barto

(Iterative) Policy Evaluation

1. **Initialization:**
   Input π, the policy to be evaluated
   Initialize $V(s)\ \forall s \in S$ (e.g., $V(s) = 0$)

2. **Policy Evaluation:**
   Repeat
   $\quad \Delta \leftarrow 0$
   $\quad k \leftarrow 0$
   $\quad$ **Foreach** $s \in S$
   $\quad\quad v \leftarrow V(s)$
   $$V_{k+1}(s) \leftarrow \sum_{s' \in S} p(s'|s,a)\big[R(s,a,s') + \gamma V_k(s')\big]$$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   $\quad k \leftarrow k + 1$
   Until $\Delta < \theta$ (small positive number)

   Output $\boldsymbol{V} \approx \boldsymbol{V}^{\pi}$

Analytic Solution is also possible!

$$V^{\pi}(s) = \sum_{s' \in S} p(s' \mid s, \pi(s)) \left[R(s, \pi(s), s') + \gamma V^{\pi}(s') \right]$$

Let T^{π} be a S x S matrix where the (i,j) entry is:

$$T^{\pi}(s_i, s_j) = p(s_j \mid s_i, \pi(s_i))$$

$$\vec{V} = T^{\pi}\vec{R} + \gamma T^{\pi}\vec{V}$$

$$\vec{V} - \gamma T^{\pi}\vec{V} = T^{\pi}\vec{R}$$

$$\vec{V} = (1 - \gamma T^{\pi})^{-1} T^{\pi}\vec{R}$$

Requires taking an inverse of a S by S matrix $O(S^3)$

Policy Improvement

- Suppose we have computed V^π for a deterministic policy π
- For a given state s, is there any better action a, $a \neq \pi(s)$?
- The value of doing a in s can be computed with $Q^\pi(s,a)$

- If an $a \neq \pi(s)$ is found, such that $Q^\pi(s,a) > V^\pi(s)$, then it's better to switch to action a
- The same can be done for all states

Policy Improvement

- → A new policy π' can be obtained in this way by being **greedy** with respect to the current V^{π}

$$\pi'(s) = \arg\max_a Q^{\pi}(s,a) \quad \forall\ s \in S$$

- Performing the greedy operation ensures that $V^{\pi'} \geq V^{\pi}$
- → Monotonic policy improvement by being greedy wrt current value functions / policy

- If $V^{\pi'} = V^{\pi}$ then we are back to the Bellman equations, meaning that both policies are optimal, there is no further space for improvement

Monotonic Improvement in Policy

- For any two value functions V_1 and V_2, $V_1 \geq V_2$ $\leftrightarrow$ $V_1(s) \geq V_2(s)\ \forall s \in S$
- *Proposition:* $V^{\pi'} \geq V^{\pi}$ with strict inequality if π is suboptimal, where π' is the new policy we get from doing policy improvement (i.e., being one-step greedy)

PROOF

$$V^{\pi}(s) \le \max_a Q^{\pi}(s,a)$$

$$= \sum_{s' \in S} p(s' \mid s, \pi'(s)) \left[R(s, \pi'(s), s') + \gamma V^{\pi}(s') \right]$$

$$\le \sum_{s' \in S} p(s' \mid s, \pi'(s)) \left[R(s, \pi'(s), s') + \gamma \max_{a'} Q^{\pi}(s', a') \right]$$

$$= \sum_{s' \in S} p(s' \mid s, \pi'(s)) \begin{bmatrix} R(s, \pi'(s), s') + \\ \gamma \sum_{s' \in S} p(s'' \mid s', \pi'(s'))(R(s', \pi'(s'), s'' + \gamma V^{\pi}(s'')) \end{bmatrix}$$

$$\ldots \le V^{\pi'}(s)$$

Policy Iteration

1. **Initialization:**

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

2. **Policy Evaluation:**

   Repeat

   $\Delta \leftarrow 0$

   $k \leftarrow 0$

   Foreach $s \in S$

   $v \leftarrow V_k$

   $$V_{k+1} \leftarrow \sum_{s' \in S} p\big(s'|s, \pi(s)\big)\Big[R(s, \pi(s), s') + \ \gamma V_k(s')\Big]$$

   $\Delta \leftarrow \max(\Delta, |v - V_{k+1}(s)|)$

   $k \leftarrow k + 1$

   Until $\Delta < \theta$ (small positive number)

   Output $\boldsymbol{V} \approx \boldsymbol{V}^{\pi}$

Policy Iteration

3. **Policy Improvement:**

   policy-stable $\leftarrow$ *true*

   Foreach $s \in S$

   old-action $\leftarrow \pi(s)$

   $$\pi(s) \leftarrow \arg\max_{a \in A(s)} \sum_{s'} p\Big(s' \mid s, a\Big)\Big[R(s, a, s') + \gamma V(s')\Big]$$

   If old-action $\neq \pi(s)$

   policy-stable $\leftarrow$ *false*

   If policy-stable

   stop

   return $\boldsymbol{V} \approx \boldsymbol{V}^*$, $\pi \approx \pi^*$

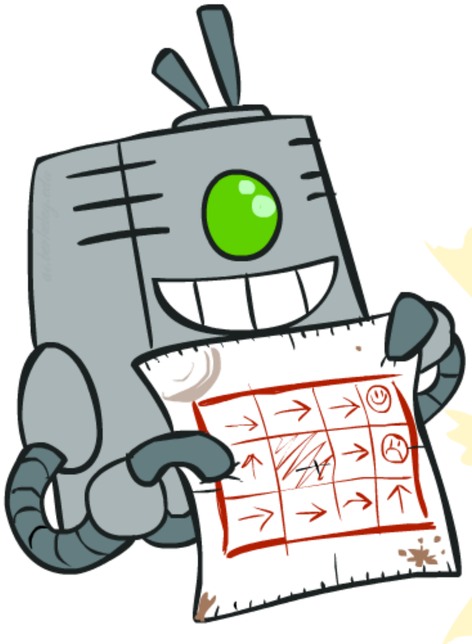
   Else

   Goto 2.

Policy Iteration

Maintain value of policy

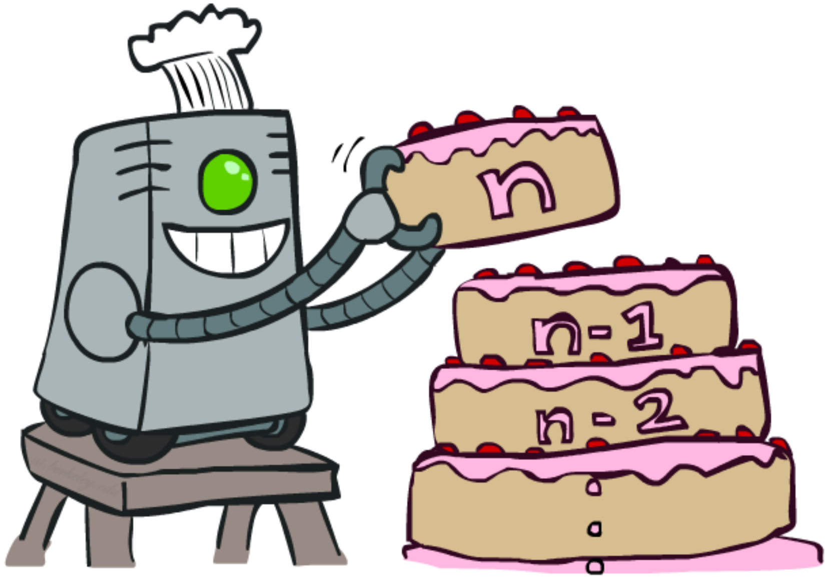
Improve policy



Value Iteration

Keep optimal value for

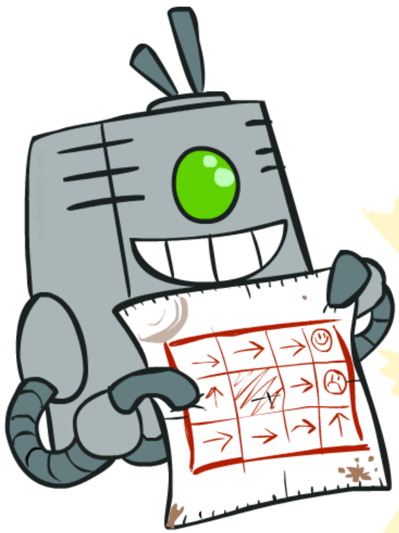
finite steps, increase steps



Policy Iteration

Fewer Iterations

More expensive per iteration



$O(|A|\cdot|S|^2)$ Improvement

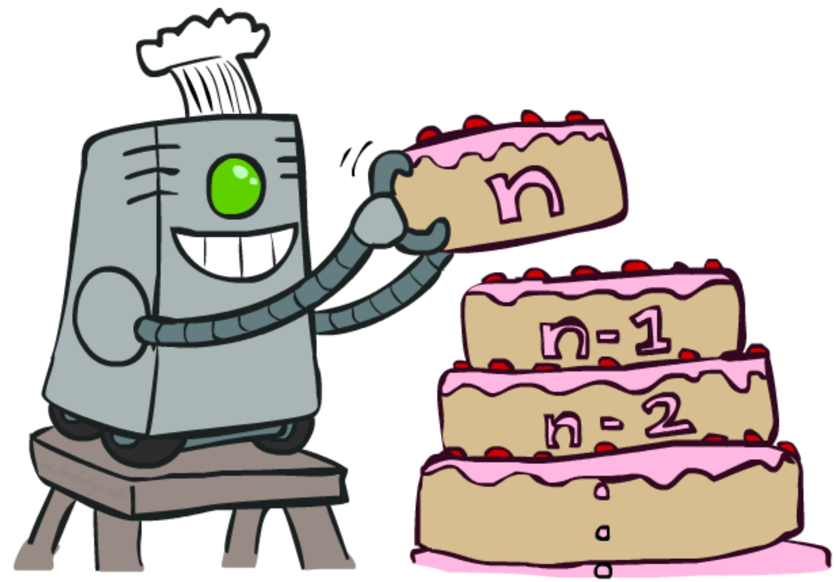
$O(||S|^3)$ Evaluation

Max $|A|^{|S|}$ possible policies to evaluate and improve

Value Iteration

More iterations

Cheaper per iteration



$O(|A|\cdot|S|^2)$ per iteration

In principle an exponential number of iterations to $\varepsilon \to 0$

Drawings by Ketrina Yim

Carnegie Mellon University

MDPs: What You Should Know

- Definition
- How to define for a problem
- Value iteration and policy iteration
  - How to implement
  - Convergence guarantees
  - Computational complexity