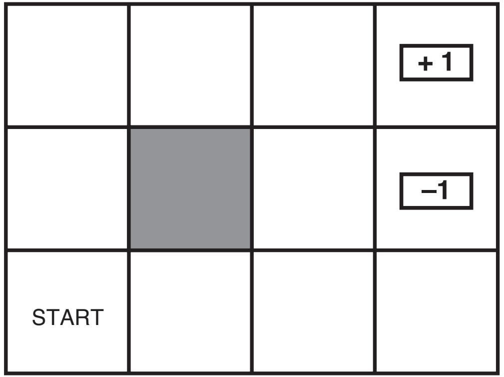
# Sequential decisions under uncertainty
# Markov Decision Processes (MDP)

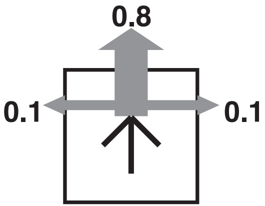
Tomáš Svoboda

Department of Cybernetics, Vision for Robotics and Autonomous Systems,
Center for Machine Perception (CMP)

May 28, 2018

# Unreliable actions in observable grid world




States $s \in S$, actions $a \in A$

Model $T(s, a, s') \equiv P(s'|s, a) =$ probability that $a$ in $s$ leads to $s'$

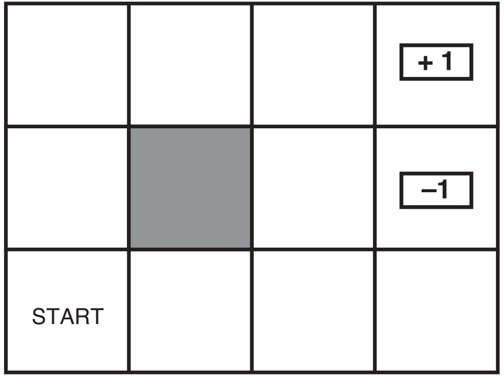


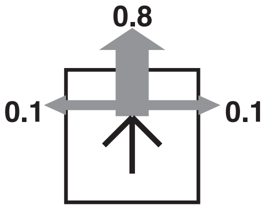
Observable - agent knows where it is. However, it does not always obey the command.

There is a treasure but there is also some danger.

The danger state - think about a mountaineous are with safer but longer and shorter but more dangerous paths - a dangerous node may represent a chasm.

# Unreliable actions in observable grid world




States $s \in S$, actions $a \in A$

Model $T(s, a, s') \equiv P(s'|s, a)$ = probability that $a$ in $s$ leads to $s'$



Observable - agent knows where it is. However, it does not always obey the command.

There is a treasure but there is also some danger.

The danger state - think about a mountaineous are with safer but longer and shorter but more dangerous paths - a dangerous node may represent a chasm.

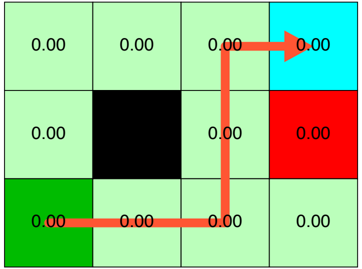
# Unreliable actions



Actions: go over a glacier bridge or around?

# Plan? Policy

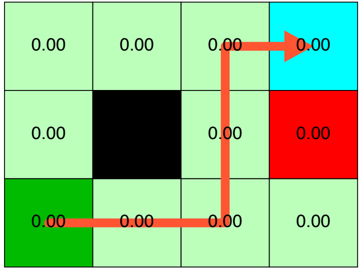
- In deterministic world: Plan – sequence of actions from Start to Goal.
- MDPs, we need a policy $\pi : S \to A$.
- An action for each possible state.
- What is the best policy?

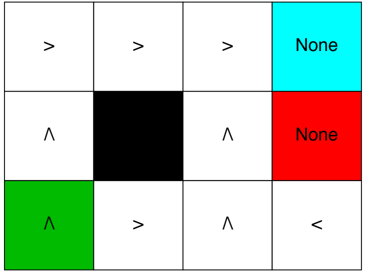



What is the best policy, we will come to that in a minute, ...

# Plan? Policy

- In deterministic world: Plan – sequence of actions from Start to Goal.
- MDPs, we need a policy $\pi : S \to A$.
- An action for each possible state.
- What is the best policy?

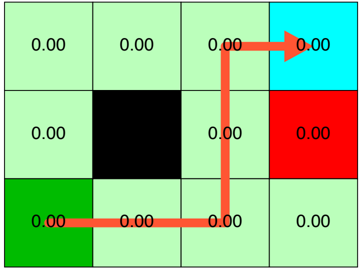


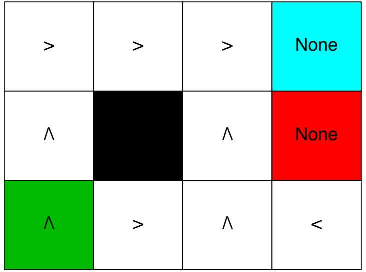




What is the best policy, we will come to that in a minute, ...

# Plan? Policy

- In deterministic world: Plan – sequence of actions from Start to Goal.
- MDPs, we need a policy $\pi : S \to A$.
- An action for each possible state.
- What is the best policy?




What is the best policy, we will come to that in a minute, ...

# Rewards

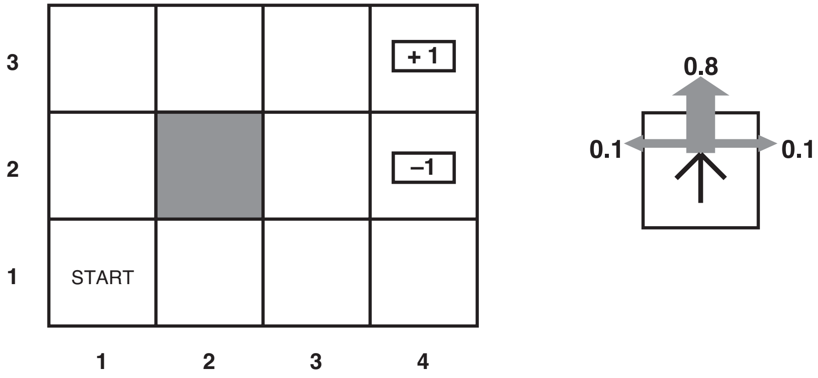
| | 0 | 1 | 2 | 3 | |
|---|---|---|---|---|---|
| 0 | -0.04 | -0.04 | -0.04 | 1.00 | 0 |
| 1 | -0.04 | | -0.04 | -1.00 | 1 |
| 2 | -0.04 | -0.04 | -0.04 | -0.04 | 2 |
| | 0 | 1 | 2 | 3 | |

Reward function $R(s)$ (or $R(s, a)$, $R(s, a, s')$)

$$= \begin{cases} -0.04 & \text{(small penalty) for nonterminal states} \\ \pm 1 & \text{for terminal states} \end{cases}$$

What do the rewards express? Reward to an agent to be/dwell in that state? Obviously we want the robot to go to the goal and do not stay too long in the maze.

# Markov Decision Processes (MDPs)




(a) (b)

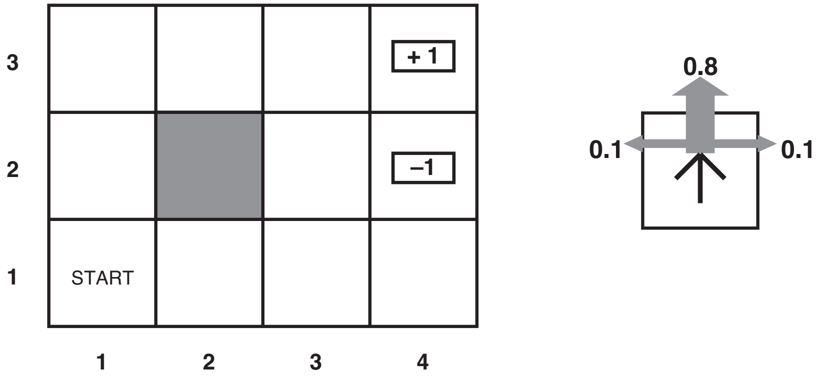
States $s \in S$, actions $a \in A$

Model $T(s, a, s') \equiv P(s'|s, a)$ = probability that $a$ in $s$ leads to $s'$

Reward function $R(s)$ (or $R(s, a)$, $R(s, a, s')$)

$$= \begin{cases} -0.04 & \text{(small penalty) for nonterminal states} \\ \pm 1 & \text{for terminal states} \end{cases}$$

# Markov Decision Processes (MDPs)




(a) (b)

States $s \in S$, actions $a \in A$

Model $T(s, a, s') \equiv P(s'|s, a)$ = probability that $a$ in $s$ leads to $s'$

Reward function $R(s)$ (or $R(s, a)$, $R(s, a, s')$)

$$= \begin{cases} -0.04 & \text{(small penalty) for nonterminal states} \\ \pm 1 & \text{for terminal states} \end{cases}$$

# Markovian property

- Given the present state, the future and the past are independent.
- MDP: Markov means action depends only on the current state.
- In search: successor function depends on the current state only.

# Optimal policies

On-line demos.

- $R(S) = \{-0.04, 1, -1\}$
- $R(S) = \{-2, 1, -1\}$
- $R(S) = \{-0.01, 1, -1\}$

How to measure quality of a policy?

# Optimal policies

On-line demos.

- $R(S) = \{-0.04, 1, -1\}$
- $R(S) = \{-2, 1, -1\}$
- $R(S) = \{-0.01, 1, -1\}$

How to measure quality of a policy?

# Optimal policies

On-line demos.

- $R(S) = \{-0.04, 1, -1\}$
- $R(S) = \{-2, 1, -1\}$
- $R(S) = \{-0.01, 1, -1\}$

How to measure quality of a policy?

# Optimal policies

On-line demos.

- $R(S) = \{-0.04, 1, -1\}$
- $R(S) = \{-2, 1, -1\}$
- $R(S) = \{-0.01, 1, -1\}$

How to measure quality of a policy?

# What is the quality of a policy?

- Executing policy - sequence of states.
- Utility of a state sequence.

# What is the quality of a policy?

- Executing policy - sequence of states.
- Utility of a state sequence.

# Utilities of sequences

- State reward $R(s)$
- State sequence $[s_0, s_1, s_2, \ldots,]$

Typically, consider stationary preferences on reward sequences:

$$[r, r_0, r_1, r_2, \ldots] \succ [r, r_0', r_1', r_2', \ldots] \Leftrightarrow [r_0, r_1, r_2, \ldots] \succ [r_0', r_1', r_2', \ldots]$$

If stationary preferences:
Utility ($h$-history)
$U_h([s_0, s_1, s_2, \ldots,]) = R(s_0) + R(s_1) + R(s_2) + \cdots$

Discounted utility, discount factor $0 \leq \gamma \leq 1$:
$U_h([s_0, s_1, s_2, \ldots,]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$

# Utilities of sequences

- State reward $R(s)$
- State sequence $[s_0, s_1, s_2, \dots, ]$

Typically, consider stationary preferences on reward sequences:

$$[r, r_0, r_1, r_2, \dots] \succ [r, r'_0, r'_1, r'_2, \dots] \Leftrightarrow [r_0, r_1, r_2, \dots] \succ [r'_0, r'_1, r'_2, \dots]$$

If stationary preferences:
Utility ($h$-history)
$U_h([s_0, s_1, s_2, \dots, ]) = R(s_0) + R(s_1) + R(s_2) + \cdots$

Discounted utility, discount factor $0 \le \gamma \le 1$:
$U_h([s_0, s_1, s_2, \dots, ]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$

# Utilities of sequences

- State reward $R(s)$
- State sequence $[s_0, s_1, s_2, \ldots,]$

Typically, consider stationary preferences on reward sequences:

$$[r, r_0, r_1, r_2, \ldots] \succ [r, r'_0, r'_1, r'_2, \ldots] \Leftrightarrow [r_0, r_1, r_2, \ldots] \succ [r'_0, r'_1, r'_2, \ldots]$$

If stationary preferences:
Utility ($h$-history)
$U_h([s_0, s_1, s_2, \ldots,]) = R(s_0) + R(s_1) + R(s_2) + \cdots$

Discounted utility, discount factor $0 \leq \gamma \leq 1$:
$U_h([s_0, s_1, s_2, \ldots,]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$

# Comparing policies; Finite vs infinite horizon

Problem: Infinite lifetime $\Rightarrow$ additive utilities are infinite.

- Finite horizon: termination at a fixed time $\Rightarrow$ nonstationary policy, $\pi(s)$ depends on the time left.
- Discounting, $\gamma < 1$, $R(s) \leq R_{\max}$

$$U([s_0, s_1, s_2, \ldots, s_\infty]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \frac{R_{\max}}{1-\gamma}$$

- Absorbing state.

# Comparing policies; Finite vs infinite horizon

Problem: Infinite lifetime $\Rightarrow$ additive utilities are infinite.

- Finite horizon: termination at a fixed time $\Rightarrow$ nonstationary policy, $\pi(s)$ depends on the time left.
- Discounting, $\gamma < 1, R(s) \leq R_{\text{max}}$

$$U([s_0, s_1, s_2, \ldots, s_\infty]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \frac{R_{\text{max}}}{1-\gamma}$$

- Absorbing state.

# Comparing policies; Finite vs infinite horizon

Problem: Infinite lifetime $\Rightarrow$ additive utilities are infinite.

- Finite horizon: termination at a fixed time $\Rightarrow$ nonstationary policy, $\pi(s)$ depends on the time left.
- Discounting, $\gamma < 1, R(s) \leq R_{\max}$

$$U([s_0, s_1, s_2, \dots, s_\infty]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \frac{R_{\max}}{1-\gamma}$$

- Absorbing state.

# MDPs recap

Markov decition processes (MDPs):

- Set of states $S$
- Set of actions $A$
- Transitions $P(s'|s,a)$ or $T(s,a,s')$
- Rewards $R(s)$; and discount $\gamma$

MDP quantities:

- Policy $\pi(s)$ – choice of action for each state
- Utility of a sequence – sum of (discounted) rewards.

# MDPs recap

Markov decition processes (MDPs):

- Set of states $S$
- Set of actions $A$
- Transitions $P(s'|s,a)$ or $T(s,a,s')$
- Rewards $R(s)$; and discount $\gamma$

MDP quantities:

- Policy $\pi(s)$ – choice of action for each state
- Utility of a sequence – sum of (discounted) rewards.

# Solving MDPs: Finding the best policy

- Executing policy - sequence of states.
- Utility of a state sequence.
- But actions are unreliable - environment is stochastic.
- Expected utility of a policy.

$$U^{\pi} = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)\right]$$

Best policy $\pi^*$ maximizes above.

# Solving MDPs: Finding the best policy

- Executing policy - sequence of states.
- Utility of a state sequence.
- But actions are unreliable - environment is stochastic.
- Expected utility of a policy.

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

Best policy $\pi^*$ maximizes above.

# Solving MDPs: Finding the best policy

- Executing policy - sequence of states.
- Utility of a state sequence.
- But actions are unreliable - environment is stochastic.
- Expected utility of a policy.

$$U^{\pi} = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^{t} R(s_t)\right]$$

Best policy $\pi^*$ maximizes above.

# Solving MDPs: Finding the best policy

- Executing policy - sequence of states.
- Utility of a state sequence.
- But actions are unreliable - environment is stochastic.
- Expected utility of a policy.

$$U^{\pi} = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^{t} R(s_t)\right]$$

Best policy $\pi^*$ maximizes above.

# Utility of a state - State value

$V(s)$ = expected (discounted) sum of rewards (until termination) assuming *optimal* actions. Hence:

$$V(s) = U^{\pi^*}(s)$$

Given $V(s')$ choosing the best action for $s$ is MEU:

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a)V(s')$$



It is not “Go to the higher value”!

## Utility of a state - State value

$V(s)$ = expected (discounted) sum of rewards (until termination) assuming *optimal* actions. Hence:

$V(s) = U^{\pi^*}(s)$

Given $V(s')$ choosing the best action for $s$ is MEU:

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a) V(s')$$



It is not "Go to the higher value"!

# Utility of a state - State value

$V(s)$ = expected (discounted) sum of rewards (until termination) assuming *optimal* actions. Hence:

$$V(s) = U^{\pi^*}(s)$$

Given $V(s')$ choosing the best action for $s$ is MEU:

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a)V(s')$$

| | 0 | 1 | 2 | 3 | |
|---|---|---|---|---|---|
| 0 | 0.81 | 0.87 | 0.92 | 1.00 | 0 |
| 1 | 0.76 | | 0.66 | -1.00 | 1 |
| 2 | 0.70 | 0.65 | 0.61 | 0.39 | 2 |
| | 0 | 1 | 2 | 3 | |

| | 0 | 1 | 2 | 3 | |
|---|---|---|---|---|---|
| 0 | > | > | > | 1.00 | 0 |
| 1 | ^ | | ^ | -1.00 | 1 |
| 2 | ^ | < | < | < | 2 |
| | 0 | 1 | 2 | 3 | |

It is not “Go to the higher value”!

# Utility of a state - State value

$V(s)$ = expected (discounted) sum of rewards (until termination) assuming *optimal* actions. Hence:

$$V(s) = U^{\pi^*}(s)$$

Given $V(s')$ choosing the best action for $s$ is MEU:

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a)V(s')$$

| | 0 | 1 | 2 | 3 | |
|---|---|---|---|---|---|
| 0 | 0.95 | 0.96 | 0.98 | 1.00 | 0 |
| 1 | 0.94 | | 0.89 | -1.00 | 1 |
| 2 | 0.92 | 0.91 | 0.90 | 0.80 | 2 |
| | 0 | 1 | 2 | 3 | |

| | 0 | 1 | 2 | 3 | |
|---|---|---|---|---|---|
| 0 | > | > | > | 1.00 | 0 |
| 1 | ∧ | | < | -1.00 | 1 |
| 2 | ∧ | < | < | ∨ | 2 |
| | 0 | 1 | 2 | 3 | |

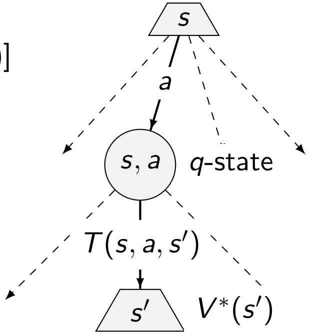
It is not “Go to the higher value”!

# MDP search tree

The value of a $q$-state $(s, a)$:
$Q^*(s,a) = \sum_{s'} T(s,a,s')\,[R(s,a,s') + \gamma\, V^*(s')]$

The value of a state $s$:

$$\begin{aligned} V^*(s) &= R(s) + \gamma \max_a \sum_{s'} T(s,a,s') V^*(s') \\ &= \max_a Q^*(s,a) \end{aligned}$$





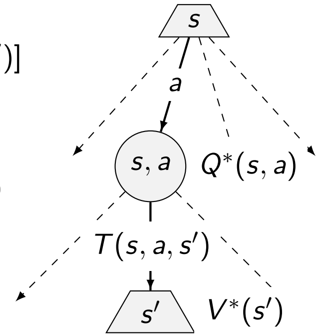

How to compute $V(s)$? Well, we could solve the expectimax search - but it grows quickly. We can see $R(s)$ as the price for leaving the state $s$ just anyhow.

# MDP search tree

The value of a $q$-state $(s, a)$:
$Q^*(s, a) = \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma\, V^*(s')\right]$

The value of a state $s$:

$$\begin{aligned} V^*(s) &= R(s) + \gamma \max_a \sum_{s'} T(s, a, s') V^*(s') \\ &= \max_a Q^*(s, a) \end{aligned}$$



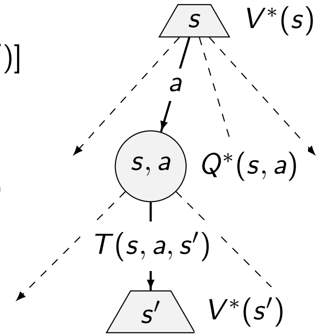

How to compute $V(s)$? Well, we could solve the expectimax search - but it grows quickly. We can see $R(s)$ as the price for leaving the state $s$ just anyhow.

## MDP search tree

The value of a $q$-state $(s,a)$:
$Q^*(s,a) = \sum_{s'} T(s,a,s')\,[R(s,a,s') + \gamma\, V^*(s')]$

The value of a state $s$:

$$
\begin{aligned}
V^*(s) &= R(s) + \gamma \max_a \sum_{s'} T(s,a,s') V^*(s') \\
&= \max_a Q^*(s,a)
\end{aligned}
$$

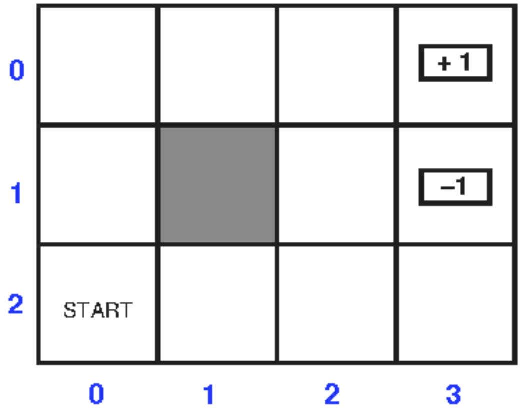


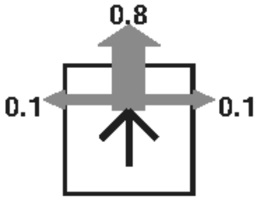



How to compute $V(s)$? Well, we could solve the expectimax search - but it grows quickly. We can see $R(s)$ as the price for leaving the state $s$ just anyhow.

# Bellman equation for state values

$$V(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V(s')$$






$V(2,0)$ computation on the table - one row for each action. We got $n$ equations for $n$ unknown - $n$ states. But max is a non-linear operator!

# Value iteration

- Start with arbitrary $V_0(s)$
- Compute Bellman update (one ply of expectimax from each state)

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

- Repeat until convergence

The idea: Bellman update makes local consistency with the Bellmann equation. Everywhere locally consistent $\Rightarrow$ globally optimal.



What is the complexity of each iteration? $O(S^2 A)$

# Value iteration

- Start with arbitrary $V_0(s)$
- Compute Bellman update (one ply of expectimax from each state)

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

- Repeat until convergence

The idea: Bellman update makes local consistency with the Bellmann equation. Everywhere locally consistent $\Rightarrow$ globally optimal.



What is the complexity of each iteration? $O(S^2 A)$

# Value iteration

- Start with arbitrary $V_0(s)$
- Compute Bellman update (one ply of expectimax from each state)

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

- Repeat until convergence

The idea: Bellman update makes local consistency with the Bellmann equation. Everywhere locally consistent $\Rightarrow$ globally optimal.



What is the complexity of each iteration? $O(S^2 A)$

# Value iteration

- Start with arbitrary $V_0(s)$
- Compute Bellman update (one ply of expectimax from each state)

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

- Repeat until convergence

The idea: Bellman update makes local consistency with the Bellmann equation. Everywhere locally consistent $\Rightarrow$ globally optimal.



What is the complexity of each iteration? $O(S^2 A)$

## Convergence

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

$$\gamma < 1$$

$$-R_{\max} \leq R(s) \leq R_{\max}$$

Max norm:

$$\|V\| = \max_s |V(s)|$$

$$U([s_0, s_1, s_2, \ldots, s_\infty]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \frac{R_{\max}}{1-\gamma}$$

# Convergence

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

$$\gamma < 1$$

$$-R_{\max} \leq R(s) \leq R_{\max}$$

Max norm:

$$\|V\| = \max_s |V(s)|$$

$$U([s_0, s_1, s_2, \ldots, s_\infty]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \frac{R_{\max}}{1-\gamma}$$

# Convergence cont'd

$V_{k+1} \leftarrow BV_k$
$\|BV_k - BV'_k\| \leq \gamma \|V_k - V'_k\|$
$\|BV_k - V_{\text{true}}\| \leq \gamma \|V_k - V_{\text{true}}\|$
Rewards are bounded, at the beginning then Value error is
$\|V_0 - V_{true}\| \leq \frac{2R_{\max}}{1-\gamma}$
We run $N$ iterations and reduce the error by factor $\gamma$ in each and want to stop the error is below $\epsilon$:
$\gamma^N 2R_{\max}/(1-\gamma) \leq \epsilon$ Taking logs, we find: $N \geq \frac{\log(2R_{\max}/\epsilon(1-\gamma))}{\log(1/\gamma)}$
To stop the iteration we want to find a bound relating the error to the size of *one* Bellman update for any given iteration.
We stop if

$$\|V_{k+1} - V_k\| \leq \frac{\epsilon(1-\gamma)}{\gamma}$$

then also: $\|V_{k+1} - V_{\text{true}}\| \leq \epsilon$ Proof on the next slide

# Convergence cont'd

$\|V_{k+1} - V_{\text{true}}\| \leq \epsilon$ is the same as $\|V_{k+1} - V_{\infty}\| \leq \epsilon$
Assume $\|V_{k+1} - V_k\| = \text{err}$
In each of the following iteration steps we reduce the error by the factor $\gamma$.
Till $\infty$, the total sum of reduced errors is:

$$\text{total} = \gamma\text{err} + \gamma^2\text{err} + \gamma^3\text{err} + \gamma^4\text{err} + \cdots = \frac{\gamma\text{err}}{(1-\gamma)}$$

We want to have total $< \epsilon$.

$$\frac{\gamma\text{err}}{(1-\gamma)} < \epsilon$$

From it follows that

$$\text{err} < \frac{\epsilon(1-\gamma)}{\gamma}$$

Hence we can stop if $\|V_{k+1} - V_k\| < \epsilon(1-\gamma)/\gamma$

# Value iteration demo

$$V_{k+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) V_k(s')$$

| | 0 | 1 | 2 | 3 | |
|---|---|---|---|---|---|
| 0 | 0.81 | 0.87 | 0.92 | 1.00 | 0 |
| 1 | 0.76 | | 0.66 | -1.00 | 1 |
| 2 | 0.70 | 0.65 | 0.61 | 0.39 | 2 |
| | 0 | 1 | 2 | 3 | |

Run mdp_agents.py and try to compute next state value in advance.
Remind the $R(s) = -0.04$ and $\gamma = 1$ in order to simplify computation.
Then discuss the course of the Values.

# Value iteration algorithm

**function** VALUE-ITERATION(env,$\epsilon$) **returns:** state values $V$
  **input:** env - MDP problem, $\epsilon$
  $V' \leftarrow 0$ in all states
  **repeat** ▷ iterate values until convergence
    $V \leftarrow V'$ ▷ keep the last known values
    $\delta \leftarrow 0$ ▷ reset the max difference
    **for each** state $s$ **in** $S$ **do**
      $V'[s] \leftarrow R(s) + \gamma \max\limits_{a \in A(s)} \sum_{s'} P(s'|s,a)V(s')$
      **if** $|V'[s] - V[s]| > \delta$ **then** $\delta \leftarrow |V'[s] - V[s]|$
    **end for**
  **until** $\delta < \epsilon(1-\gamma)/\gamma$
**end function**

# Value iteration algorithm

**function** VALUE-ITERATION(env,$\epsilon$) **returns:** state values $V$

  **input:** env - MDP problem, $\epsilon$

  $V' \leftarrow 0$ in all states

  **repeat** ▷ iterate values until convergence

    $V \leftarrow V'$ ▷ keep the last known values

    $\delta \leftarrow 0$ ▷ reset the max difference

    **for each** state $s$ **in** $S$ **do**

      $V'[s] \leftarrow R(s) + \gamma \max\limits_{a \in A(s)} \sum_{s'} P(s'|s,a)V(s')$

      **if** $|V'[s] - V[s]| > \delta$ **then** $\delta \leftarrow |V'[s] - V[s]|$

    **end for**

  **until** $\delta < \epsilon(1-\gamma)/\gamma$

**end function**

# Value iteration algorithm

**function** VALUE-ITERATION(env,$\epsilon$) **returns:** state values $V$

  **input:** env - MDP problem, $\epsilon$

  $V' \leftarrow 0$ in all states

  **repeat** ▷ iterate values until convergence

    $V \leftarrow V'$ ▷ keep the last known values

    $\delta \leftarrow 0$ ▷ reset the max difference

    **for each** state $s$ **in** $S$ **do**

      $V'[s] \leftarrow R(s) + \gamma \max\limits_{a \in A(s)} \sum_{s'} P(s'|s,a)V(s')$

      **if** $|V'[s] - V[s]| > \delta$ **then** $\delta \leftarrow |V'[s] - V[s]|$

    **end for**

  **until** $\delta < \epsilon(1-\gamma)/\gamma$

**end function**

# Value iteration algorithm

**function** VALUE-ITERATION(env,$\epsilon$) **returns:** state values $V$

    **input:** env - MDP problem, $\epsilon$

    $V' \leftarrow 0$ in all states

    **repeat** $\triangleright$ iterate values until convergence

        $V \leftarrow V'$ $\triangleright$ keep the last known values

        $\delta \leftarrow 0$ $\triangleright$ reset the max difference

        **for each** state $s$ **in** $S$ **do**

            $V'[s] \leftarrow R(s) + \gamma \max\limits_{a \in A(s)} \sum_{s'} P(s'|s,a) V(s')$

            **if** $|V'[s] - V[s]| > \delta$ **then** $\delta \leftarrow |V'[s] - V[s]|$

        **end for**

    **until** $\delta < \epsilon(1-\gamma)/\gamma$

**end function**

# Value iteration algorithm

**function** VALUE-ITERATION(env,$\epsilon$) **returns:** state values $V$
  **input:** env - MDP problem, $\epsilon$
  $V' \leftarrow 0$ in all states
  **repeat** ▷ iterate values until convergence
    $V \leftarrow V'$ ▷ keep the last known values
    $\delta \leftarrow 0$ ▷ reset the max difference
    **for each** state $s$ **in** $S$ **do**
      $V'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)V(s')$
      **if** $|V'[s] - V[s]| > \delta$ **then** $\delta \leftarrow |V'[s] - V[s]|$
    **end for**
  **until** $\delta < \epsilon(1-\gamma)/\gamma$
**end function**

# References

Some figures from [1].

[1] Stuart Russell and Peter Norvig.
*Artificial Intelligence: A Modern Approach*.
Prentice Hall, 3rd edition, 2010.
http://aima.cs.berkeley.edu/.