Grundlagen der künstlichen Intelligenz – Rational Decisions Over Time

Matthias Althoff

TU München

January 16, 2020

Organization

1. Markov Decision Processes
2. Value Iteration
3. Policy Iteration
4. Partially Observable Markov Decision Processes

The content is covered in the AI book by the section “Making Complex Decisions”.

Learning Outcomes

- You understand the concept of *Markov decision processes* (MPD) and *partially observable Markov decision processes* (POMDP) and can assess which concept to use for a given problem.
- You understand the concept of an *optimal policy* and how it differs from an *optimal sequence*.
- You can create utility functions of state sequences for finite and infinite time horizons.
- You know and understand the *Bellmann Principle of Optimality*.
- You can create and evaluate *Bellman equations* for a given stochastic optimization problem.
- You can solve Bellmann equations using *value iteration* and *policy iteration*.
- You can compute the belief state of a POMDP.
- You can convert a POMDP into an MDP.
- You understand how value iteration works for POMDPs.

Overview of Probabilistic Methods

This lecture focuses on actions in dynamic environments.

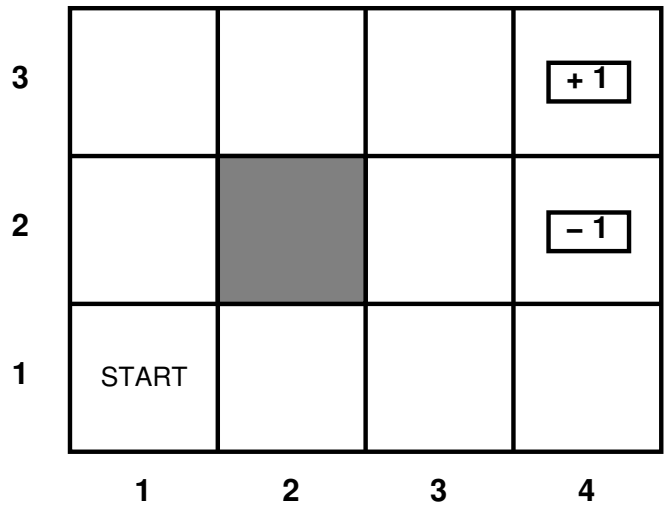
	Static environment	**Dynamic environment**
Without actions	Bayesian networks (lecture 9)	Hidden Markov models (lecture 10)
With actions	Decision networks (lecture 11)	Markov decision processes (lecture 12)

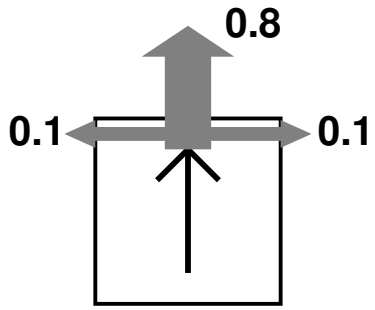
Variations of Markov Models

- Last lecture was on decision making of episodic environments.
- This lecture is about decision making in sequential environments, requiring us to consider dynamics.
- Previous models of sequential environments did not consider the possibility of taking actions:

		control over state transitions?	
		no	**yes**
states fully obser-vable?	**yes**	Markov chain	Markov Decision Process (MDP)
	no	Hidden Markov model (HMM)	Partially Observable Markov Decision Process (POMDP)

Example of a Markov Decision Process





- States $s \in S$, actions $a \in A = \{Up, Down, Left, Right\}$.
- **Model** $P(s'|s,a)$ = probability that a in s leads to s'.
- **Reward function (or utility function)** $R(s)$ (or $R(s,a)$, $R(s,a,s')$)
$$= \begin{cases} -0.04 & \text{(small penalty) for nonterminal states} \\ \pm 1 & \text{for terminal states} \end{cases}$$
- We assume the environment is fully observable: the agent always knows where it is.

Markov Decision Process (MDP)

A **Markov decision process** is a sequential decision problem for a fully observable, stochastic environment with a Markovian transition model and additive rewards.

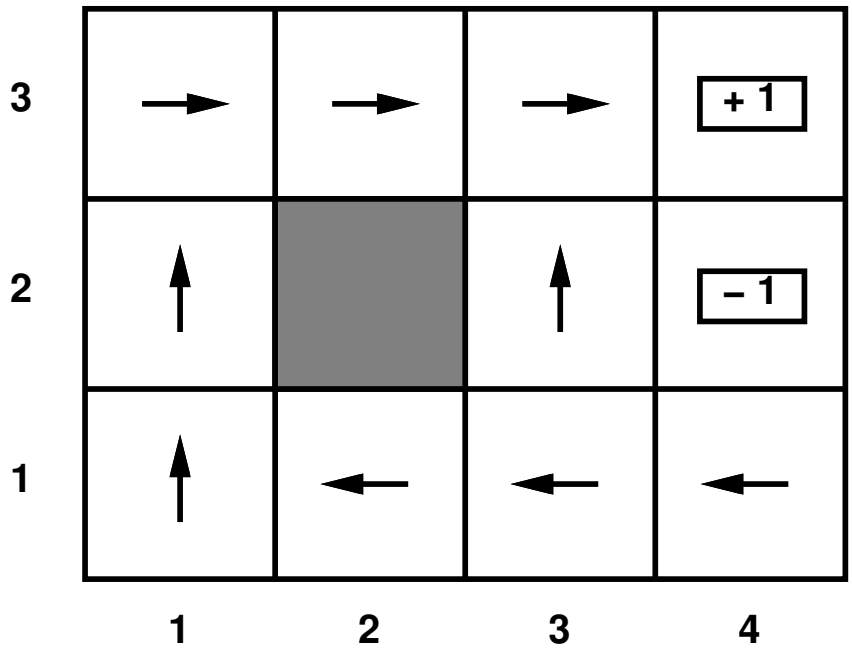
Formal definition

A Markov decision process is a 5-tuple (S, A, P, R, γ), where

- S is a finite set of states,
- A is a finite set of actions (alternatively, $A(s)$ is the finite set of actions available from state s),
- $P_a(s, s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t + 1$,
- $R(s, s')$ is the immediate reward (or expected immediate reward) received after transition to state s' from state s,
- $\gamma \in [0, 1]$ is the discount factor, which represents the difference in importance between future rewards and present rewards.

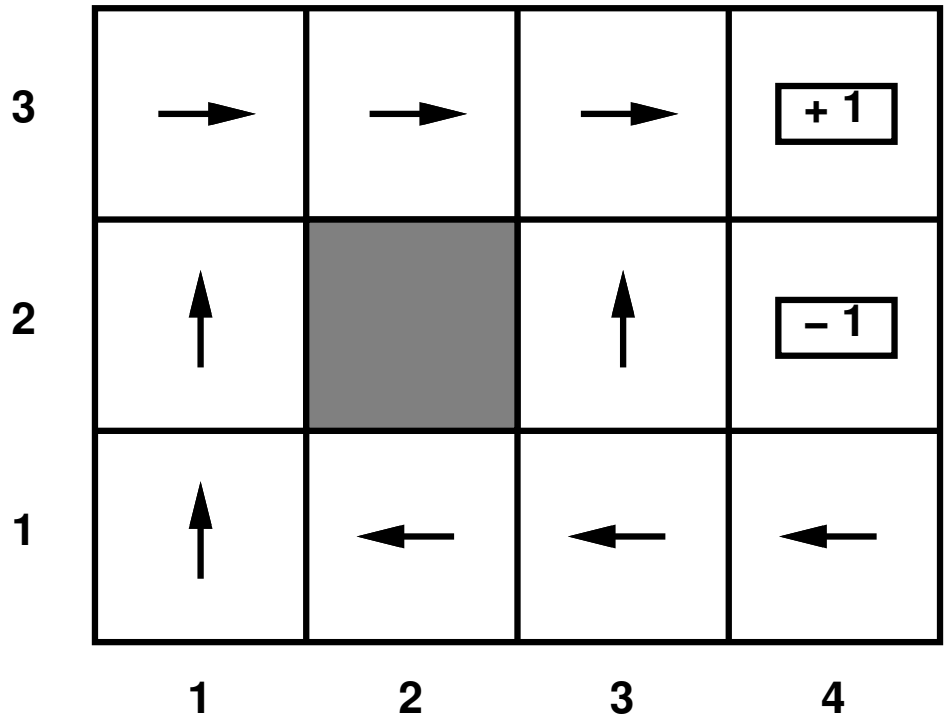
Solving MDPs

- In search problems, the aim is finding an optimal *sequence*.
- In MDPs, the aim is finding an optimal *policy* $\pi(s)$, i.e., the best action for every possible state s (because one can't predict where one will end up).
- The optimal policy maximizes the *expected utility*.
- Optimal policy when state penalty is $R(s, s') = R(s) = 0.04$:



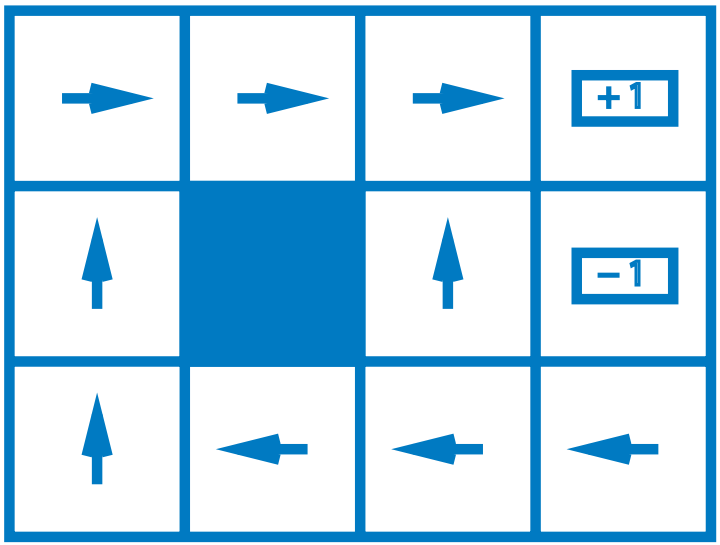
Tweedback Question

Does the optimal policy change when $R(s)$ is changed?

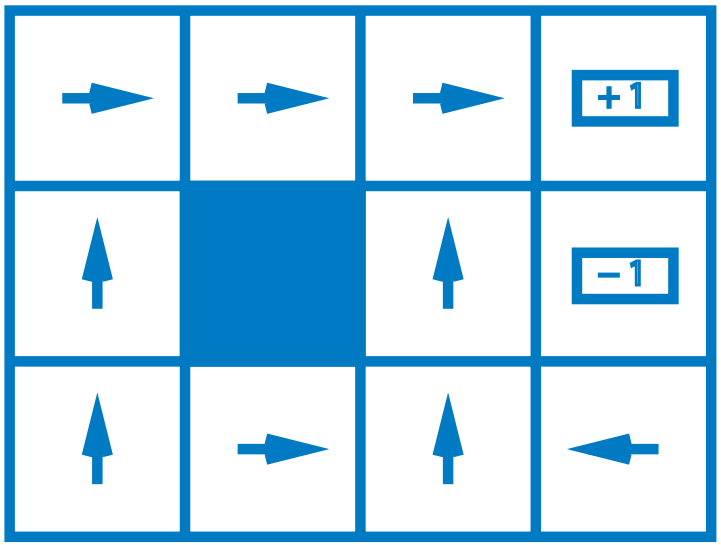


Tweedback Question

What is the missing reward value?



r = −0.04

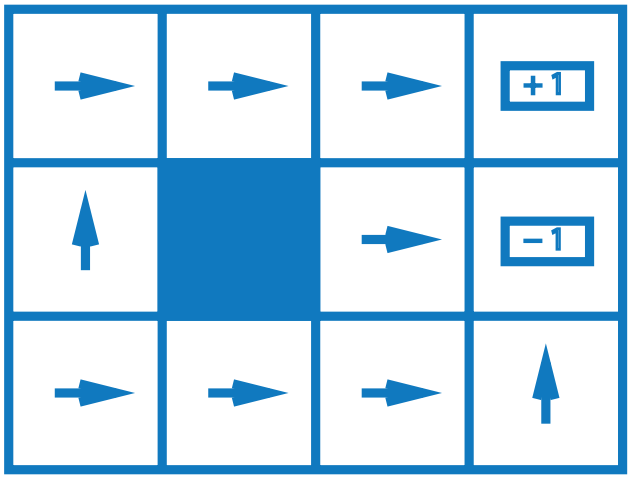


r = ?

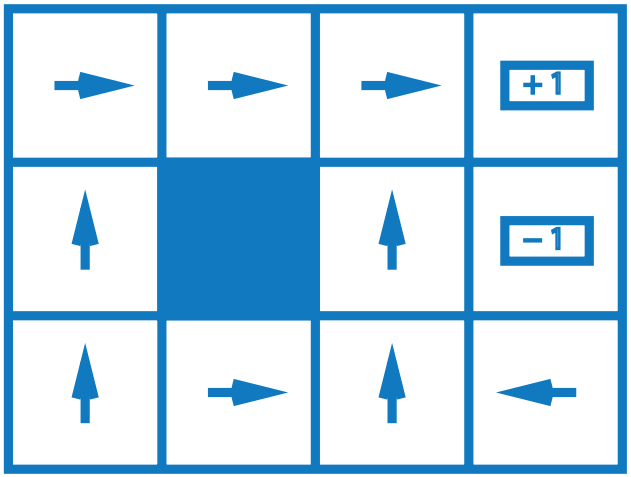
A −0.09,

B −0.02.

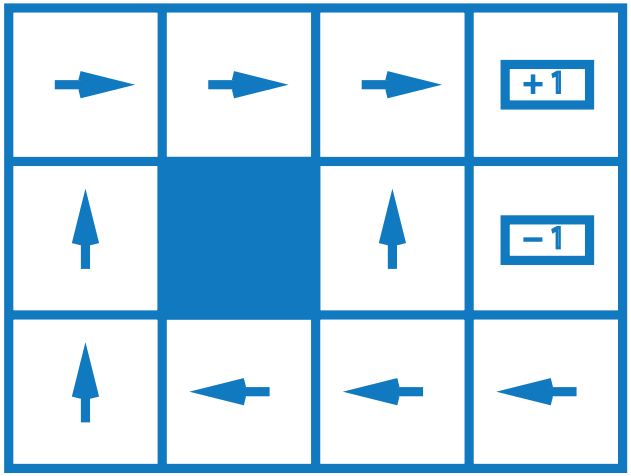
Optimal Policy for Different Ranges of $R(s)$



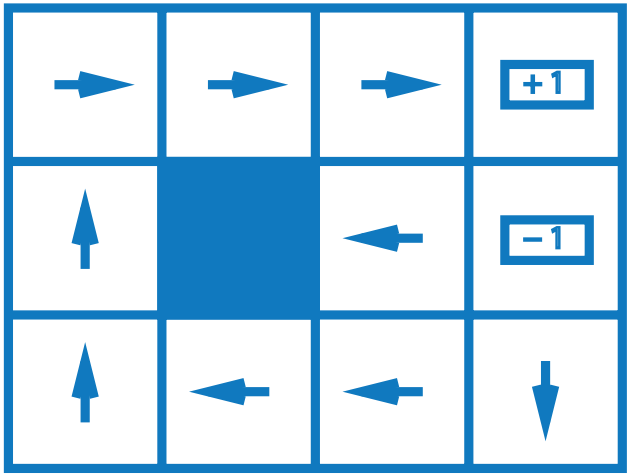
$r = [-\infty : -1.6284]$



$r = [-0.4278 : -0.0850]$



$r = [-0.0480 : -0.0274]$



$r = [-0.0218 : 0.0000]$

Motivating Examples

Some examples where MDPs have been applied:

- Agriculture
- Water resources
- Inspection, maintenance, and repair
- Purchasing, inventory, and production
- Finance and investment
- Queues
- Sales promotion
- Motor insurance claims
- Overbooking
- Epidemics
- Sports
- Patient admissions
- Design of experiments
- etc.

Utility of State Sequences

- We need to understand preferences between *sequences* of states.
- Typically consider **stationary preferences** on reward sequences:

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

Utility of sequences

There are only two coherent ways to combine rewards over time:

1. *Additive* utility function:
   $U([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \cdots$
2. *Discounted* utility function:
   $U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$
   where γ is the **discount factor**

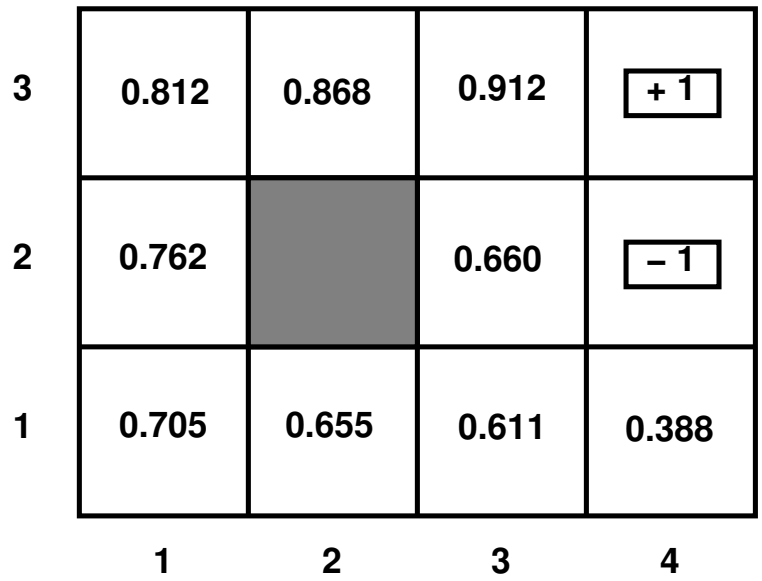
Utility of States

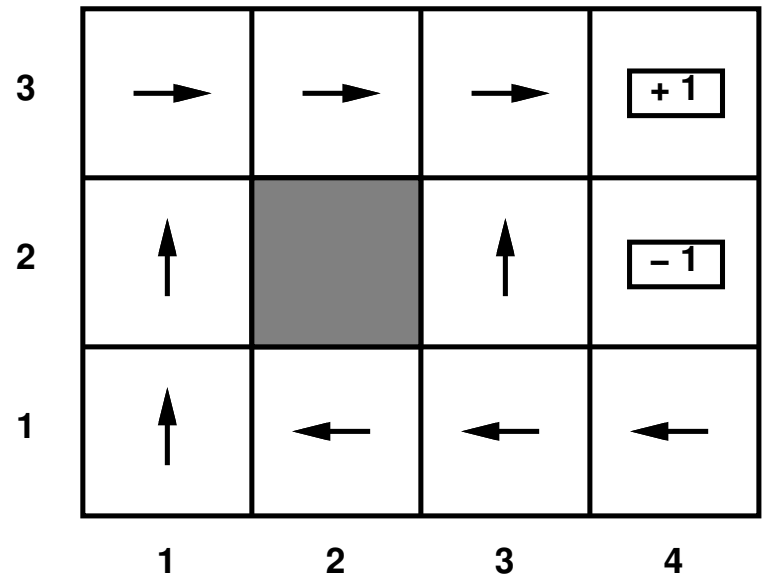
Utility of a *state* is defined to be

> $U(s) =$ expected (discounted) sum of rewards (until termination) assuming optimal actions.

Optimal policy of a state s and action space $A(s)$:

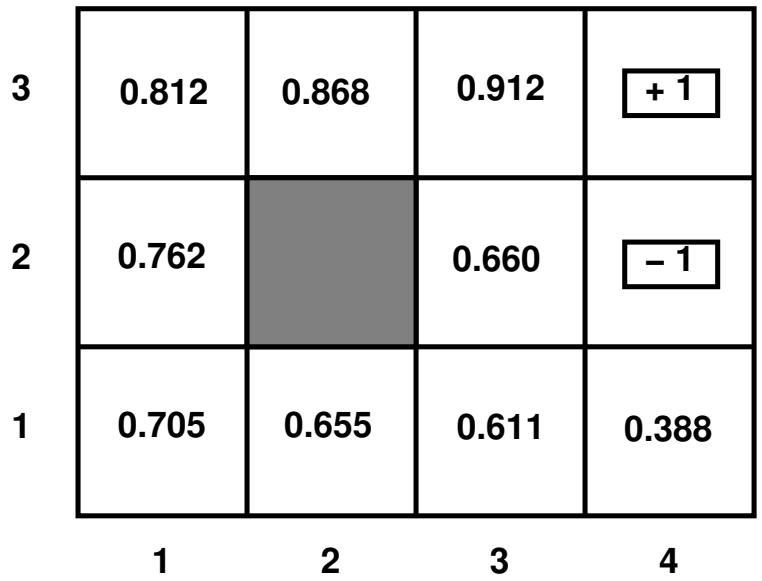
$$\pi^*(s) = \underset{a \in A(s)}{\texttt{argmax}} \sum_{s'} P(s'|s,a)U(s').$$

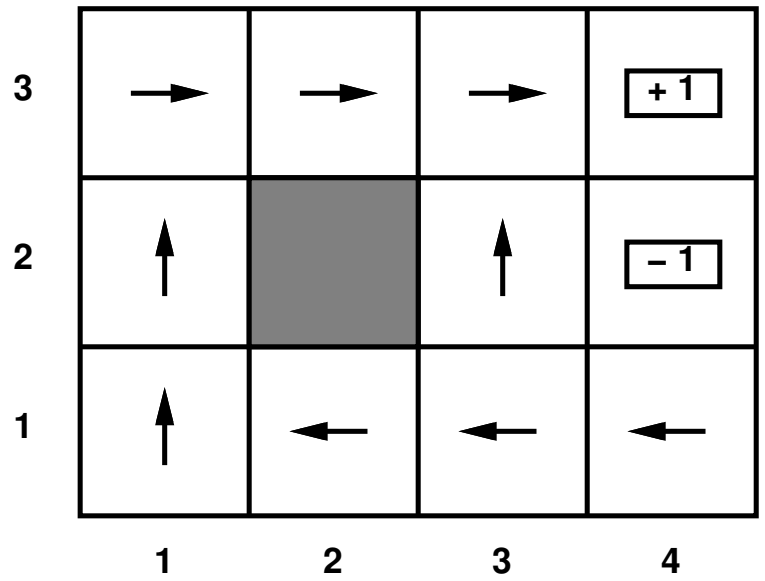




Optimal Policy for one State: Example

$$\pi^*(s) = \underset{a \in A(s)}{\mathtt{argmax}} \sum_{s'} P(s'|s,a)U(s').$$





Square (3,1):

up: $0.8 \cdot 0.660 + 0.1 \cdot 0.655 + 0.1 \cdot 0.388 \approx 0.632$

left: $0.8 \cdot 0.655 + 0.1 \cdot 0.660 + 0.1 \cdot 0.611 \approx 0.651$

down: $0.8 \cdot 0.611 + 0.1 \cdot 0.655 + 0.1 \cdot 0.388 \approx 0.593$

right: $0.8 \cdot 0.388 + 0.1 \cdot 0.660 + 0.1 \cdot 0.611 \approx 0.438$

Additive Utility For Infinite Horizons

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

1. **Finite horizon**: termination at a *fixed time* T
   $\Rightarrow$ **nonstationary** policy: $\pi(s)$ also depends on time left.
2. **Absorbing state(s)**: with probability 1, agent eventually "dies" for any π
   $\Rightarrow$ expected utility of every state is finite.
3. **Discounting**: assuming $\gamma < 1$, $R(s) \leq R_{\max}$,

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

   Smaller $\gamma \Rightarrow$ shorter horizon.

4. **Maximize system gain**: System gain = Average reward per time step
   Infinite sequences can be compared in terms of **average reward**.
   The analysis of average-reward algorithms is beyond the scope of this lecture.

In sum, discounting is often the most useful technique.

Bellmann Principle of Optimality

Bellmann Principle of Optimality

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. (Bellman, 1957)

Example:

- To optimally get from Garching Forschungszentrum to Olympiazentrum with the subway for a party at Olydisco, you have to pass Studentenstadt.
- The optimal path from Studentenstadt to Olympiazentrum must be a partial optimal path from Garching Forschungszentrum to Olympiazentrum.

Bellman Equation

Definition of utility of states leads to a simple relationship among utilities of neighboring states based on the *Bellmann Principle of Optimality*:

Bellman equation (1957)

expected sum of rewards = current reward + $\gamma \times$ expected sum of rewards after taking best action:

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

Example: Bellman equation of the 4×3 world for state $(1,1)$:

$$\begin{aligned} U(1,1) = -0.04 + \gamma \max[&0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && (up) \\ &0.9U(1,1) + 0.1U(1,2), && (left) \\ &0.9U(1,1) + 0.1U(2,1), && (down) \\ &0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)] && (right) \end{aligned}$$

Using the values from slide 14, one obtains that *up* is the best action.

Value Iteration Algorithm

Problem: Solving the Bellman equation requires solving n **nonlinear** equations in n unknowns.

Idea:

- Start with arbitrary utility values
- Update to make them **locally consistent** with Bellman eqn.
- Everywhere locally consistent $\Rightarrow$ global optimality.

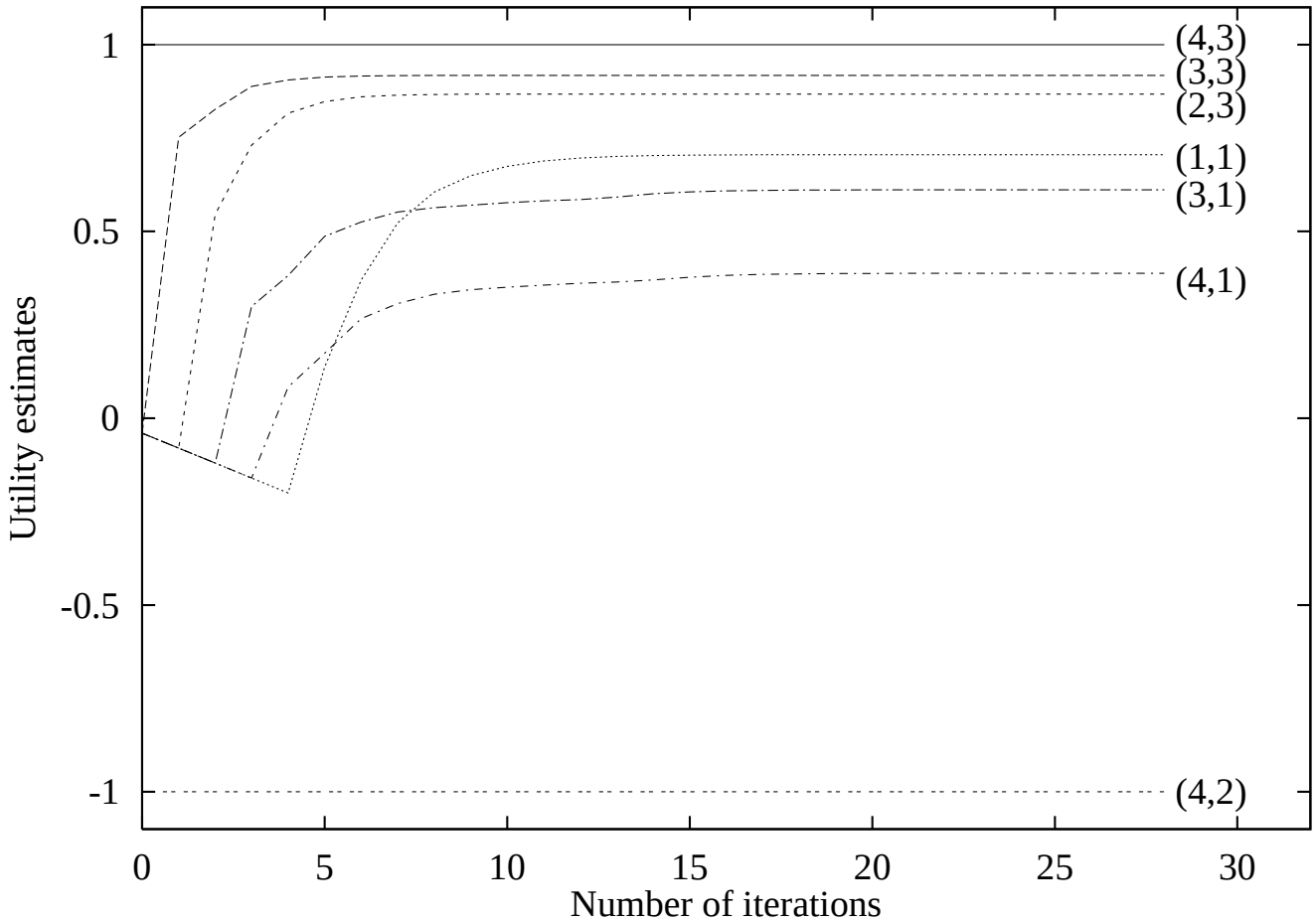
Algorithm:

- terminal state: $U(s) = R(s)$
- other states: Repeat for every s simultaneously until "no change"

$$U(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s') \qquad \text{for all } s$$

Value Iteration Algorithm: Example

Results for 4×3 world:



Convergence of Value Iteration

Max norm

We define the max-norm as $\|U\| = \max_s |U(s)|$
(often referred to as infinity norm $\|U\|_\infty$).

Consequently, $\|U - V\|$ = maximum difference between U and V.

Let U^t and U^{t+1} be successive approximations to the true utility U.

Theorem

For any two approximations U^t and V^t (see exercise 6 of AI book)

$$\|U^{t+1} - V^{t+1}\| \leq \gamma \|U^t - V^t\|$$

Any approximations must get closer to each other and thus closer to the true U.

Theorem

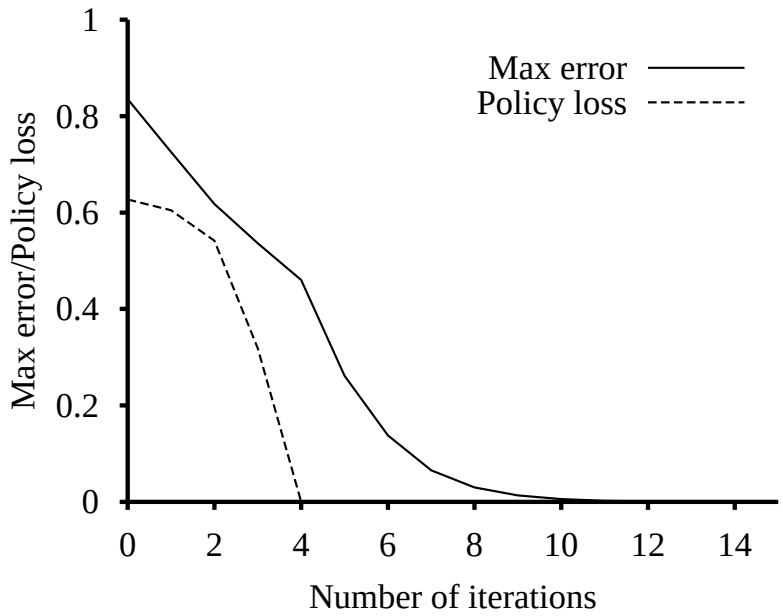
If $\|U^{t+1} - U^t\| < \epsilon$, then $\|U^{t+1} - U\| < 2\epsilon\gamma/(1-\gamma)$
I.e., once the change in U^t becomes small, we are almost done.

Policy Iteration: Basic Idea

It is possible to get an optimal policy even when the utility function estimate is inaccurate.

Right figure: Maximum error $\|U_i - U\|$ plotted with policy loss $\|U^{\pi_i} - U\|$ of the 4×3 world.

This motivates an alternative to find optimal policies:



Policy iteration

- **Policy evaluation**: Given a policy π_i, calculate $U_i = U^{\pi_i}$, the utility of each state if π_i were to be executed.
- **Policy improvement**: Calculate a new policy π_{i+1} using a one-step look-ahead based on U_i using $\pi_{i+1}(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s')$.

The algorithm terminates when the policy improvement yields no utility change.

Policy-Iteration Algorithm

function Policy-Iteration (*mdp*) **returns** a policy

inputs: *mdp*, a Markov decision process with states S, actions $A(s)$, transition model $P(s'|s,a)$

local variables: U, a vector of utilities for states in S, initially 0

π, a policy vector indexed by state, initially random

repeat

$U \leftarrow \texttt{Policy} - \texttt{Evaluation}(\pi, U, mdp)$

$unchanged \leftarrow true$

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

if $\max_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s') > \sum_{s'} P(s'|s,\pi(s))U(s')$ **do**

$\pi(s) \leftarrow \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a)U(s')$

$unchanged \leftarrow false$

until *unchanged*

return π

Policy Evaluation

- The policy improvement step in the previous algorithm is straightforward.
- How do we implement the `Policy-Evaluation` routine?
- It is actually easier than solving the standard Bellman equations, since the action in each state is fixed by the policy:

$$U_i(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) U_i(s').$$

- For instance, for the policy on slide 8 we have $\pi(1,1) = up$, $\pi(1,2) = up$, and so on, so that the simplified Bellman equations are

$$\begin{aligned} U_i(1,1) &= -0.04 + 0.8U_i(1,2) + 0.1U_i(1,1) + 0.1U_i(2,1), \\ U_i(1,2) &= -0.04 + 0.8U_i(1,3) + 0.2U_i(1,2), \\ &\vdots \end{aligned}$$

  These equations are linear! (since max() operator is removed)
- The linear equations can be solved in $\mathcal{O}(n^3)$ time, where n is the number of unknowns.

Decision making in Partially Observable Environments

- A Markov decision process (MDP) can be viewed as

$$\boxed{\text{MDP} = \text{Markov chain} + \text{actions} + \text{rewards}\,.}$$

- In reality, the agent does not know exactly in what state it is when the environment is **partially observable**.
- Consequently, the agent cannot execute a policy of the form $\pi(s)$, since s is not exactly known.
- After introducing a sensor model, we obtain **partially observable Markov decision processes** (POMDPs), which can be viewed as

$$\boxed{\text{POMDP} = \text{hidden Markov model} + \text{actions} + \text{rewards}\,.}$$

Partially Observable Markov Decision Processes (POMDP)

A POMDP is a generalization of an MDP without direct state observation.

Formal definition

A POMDP is a 7-tuple $(S, A, E, P, \Omega, R, \gamma)$, where

- S is a finite set of states,
- A is a finite set of actions,
- E is a finite set of evidences,
- $P_a(s, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t + 1$,
- $\Omega_a(e, s') = P(e_t = e | s_{t+1} = s', a_t = a)$ is the probability that observation e is made under action a when the next state is s',
- $R(s, s')$ is the immediate reward (or expected immediate reward) received after transition to state s' from state s,
- $\gamma \in [0, 1]$ is the discount factor.

POMDP vs. MDP

MDP

- ⊕ Tractable to solve.
- ⊕ Relatively easy to specify.
- ⊖ Assumes perfect knowledge of the state.

POMDP

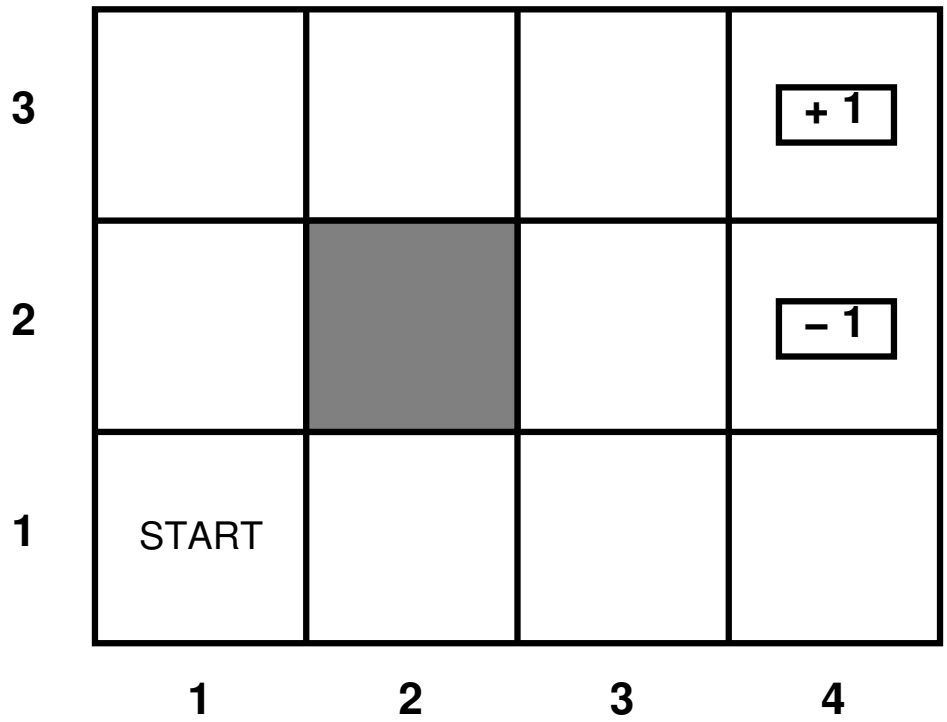
- ⊕ Treats all sources of uncertainty uniformly.
- ⊕ Allows for information gathering actions.
- ⊖ Hugely intractable to solve optimally.

Belief State

- Since the state is not exactly known, we introduce a belief state b.
- The belief state is not a classical state, but a probability distribution over s.
- We write $b(s)$ for the probability $P(s)$.
- Example: We could initialize the belief state of the 4×3 world uniformly among all non-target states as $b = \langle \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, 0, 0 \rangle$.
- From now on we assume $P(e_t = e | s_{t+1} = s', a_t = a) = P(e_t = e | s_{t+1} = s')$.

Tweedback Question

Most likely state when the agent plans to move left and senses one adjacent wall?



A x: 2, y: 1,
B x: 3, y: 3,
C x: 3, y: 1.

Belief State Update

Given an action a and an evidence e, we can compute the belief state update as

$$
\begin{aligned}
b'(s') &= P(s') && \text{(definition)} \\
&= P(s'|e,a,b) && (e,a,b \text{ are known}) \\
&= \frac{P(e|s',a,b)P(s'|a,b)}{P(e|a,b)} && \text{(Bayes' rule)} \\
&= \alpha\, P(e|s',a,b)P(s'|a,b) && \text{(normalization)} \\
&= \alpha\, P(e|s')P(s'|a,b) && (P(e|s',a,b) = P(e|s')) \\
&= \alpha\, P(e|s') \sum_s P(s'|a,b,s)P(s|a,b) && \text{(rule of total probability)} \\
&= \alpha\, P(e|s') \sum_s P(s'|s,a)P(s) && (a,b \text{ are known}) \\
&= \alpha\, P(e|s') \sum_s P(s'|s,a)b(s), && (b(s) = P(s))
\end{aligned}
$$

Due to the analogy with filtering, we write $b' = \texttt{Forward}(b,a,e)$.

Conversion to an MDP (1)

The goal is to reformulate a POMDP as an MDP. As a first step, we perform the following conversions:

$$
\begin{aligned}
P(e|a,b) &= \sum_{s'} P(e|s',a,b)P(s'|a,b) && \text{(rule of total probability)} \\
&= \sum_{s'} P(e|s')P(s'|a,b) && (P(e|s',a,b) = P(e|s')) \\
&= \sum_{s'} P(e|s')P(s'|a) && (a,b \text{ are known} \Rightarrow P(s'|a,b) = P(s'|a) = P(s')) \\
&= \sum_{s'} P(e|s') \sum_{s} P(s'|s,a)P(s|a) && \text{(rule of total probability)} \\
&= \sum_{s'} P(e|s') \sum_{s} P(s'|s,a)P(s) && (a \text{ is known}) \\
&= \sum_{s'} P(e|s') \sum_{s} P(s'|s,a)b(s) && (b(s) = P(s)) \qquad (1)
\end{aligned}
$$

This auxiliary result is continued on the next slide.

Conversion to an MDP (2)

The idea is to interpret the probability distribution over the states b as a state, which is updated as the known state for an MDP ($P(s'|s,a)$):

$$P(b'|b,a) = \sum_e P(b'|e,a,b)P(e|a,b) \qquad \text{(rule of total probability)}$$
$$= \sum_e P(b'|e,a,b) \sum_{s'} P(e|s') \sum_s P(s'|s,a)b(s) \qquad \text{(using (1))},$$

where

$$P(b'|e,a,b) = \begin{cases} 1, & \text{if } b' = \texttt{Forward}(b,a,e) \\ 0, & \text{otherwise.} \end{cases}$$

Next, we define a reward function for belief states

$$\rho(b) = \sum_s b(s)R(s).$$

$P(b'|b,a)$ and $\rho(b)$ define an **observable** MDP on the space of belief states.

Optimal Policy

The conversion of the POMDP to an MDP results in the following theorem:

Theorem (Astrom, 1965)

The optimal policy in a POMDP is a function $\pi(b)$ where b is the **belief state** (probability distribution over states).

Decision cycle of a POMDP:

1. Given the current belief state b, execute the action $a = \pi^*(b)$.
2. Receive percept e.
3. Set the current belief state to $b' = \texttt{Forward}(b, a, e)$ and repeat.

Value Iteration for POMDPs

Difficulty

For MDPs, we computed one utility value for each state.
Now we have infinitely many belief states since probability values are continuous!

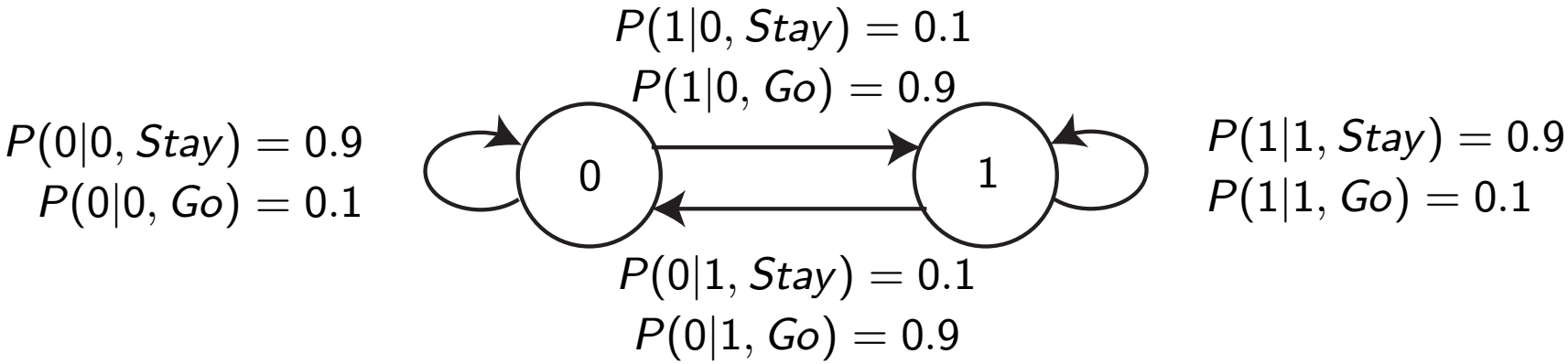
Observations

1. The utility of executing a **fixed** conditional plan p starting in state s is denoted by $\alpha_p(s)$. Then, the expected utility is $\sum_s b(s)\alpha_p(s)$, i.e., linear with b spanning a hyperplane (plane in n dimensions; you cannot geometrically imagine that) in the belief state.
2. For any belief state b, the optimal policy chooses the plan with the highest expected utility:

$$U(b) = U^{\pi^*}(b) = \max_p \mathbf{b}^T \boldsymbol{\alpha}_p$$

Simple Two-State World (1)

- States: 0 and 1
- Rewards: $R(0) = 0$ and $R(1) = 1$
- Discount factor: $\gamma = 1$
- Actions: *Stay* and *Go*
- Model: Action *Stay* results in the same state with probability 0.9, and *Go* changes the state with probability 0.9 (see figure)



- Sensor model: $P(percept = 0|0) = 0.6$, $P(percept = 0|1) = 0.4$,
  $P(percept = 1|0) = 0.4$, $P(percept = 1|1) = 0.6$

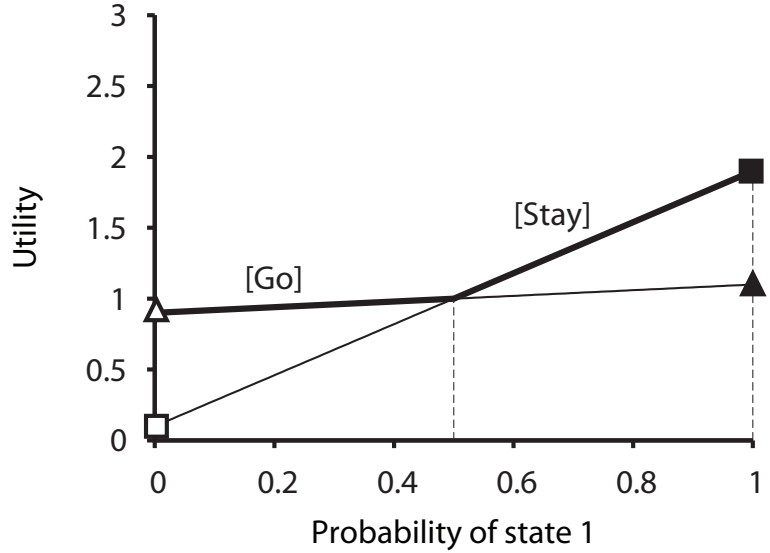
Simple Two-State World (2)

In the two-state world the belief state can be viewed as one-dimensional (probabilities sum up to 1).

Consider the one-step plans [*Stay*] and [*Go*], whose rewards consist of the current reward and the reward for the next state:

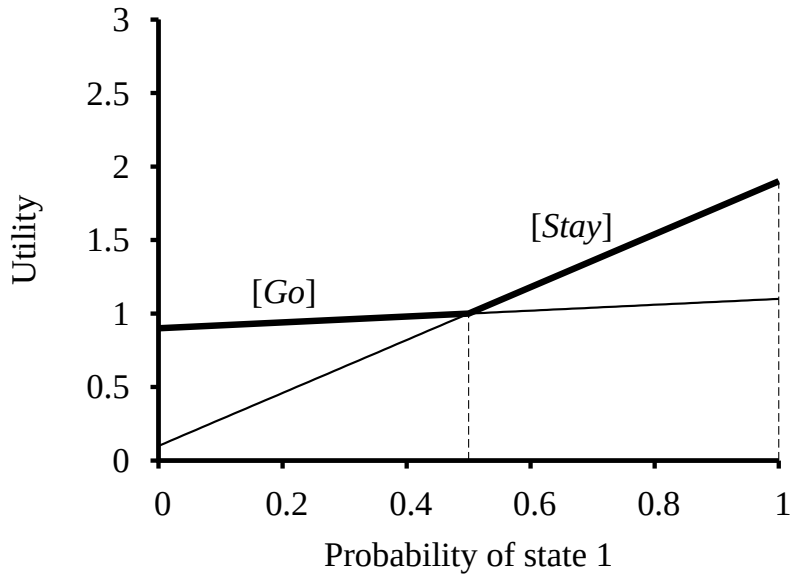
$$\alpha_{[Stay]}(0) = R(0) + \gamma(0.9R(0) + 0.1R(1)) = 0.1 \quad (\square)$$
$$\alpha_{[Stay]}(1) = R(1) + \gamma(0.9R(1) + 0.1R(0)) = 1.9 \quad (\blacksquare)$$
$$\alpha_{[Go]}(0) = R(0) + \gamma(0.9R(1) + 0.1R(0)) = 0.9 \quad (\triangle)$$
$$\alpha_{[Go]}(1) = R(1) + \gamma(0.9R(0) + 0.1R(1)) = 1.1 \quad (\blacktriangle)$$

Hyperplanes for $b(1)\,\alpha_{[Stay]}(1)$ and $b(1)\,\alpha_{[Go]}(1)$:



Simple Two-State World (3)

- The bold line represents the utility function for just one action.
- The optimal one-step policy is to *Stay* when $b(1) > 0.5$ and *Go* otherwise.



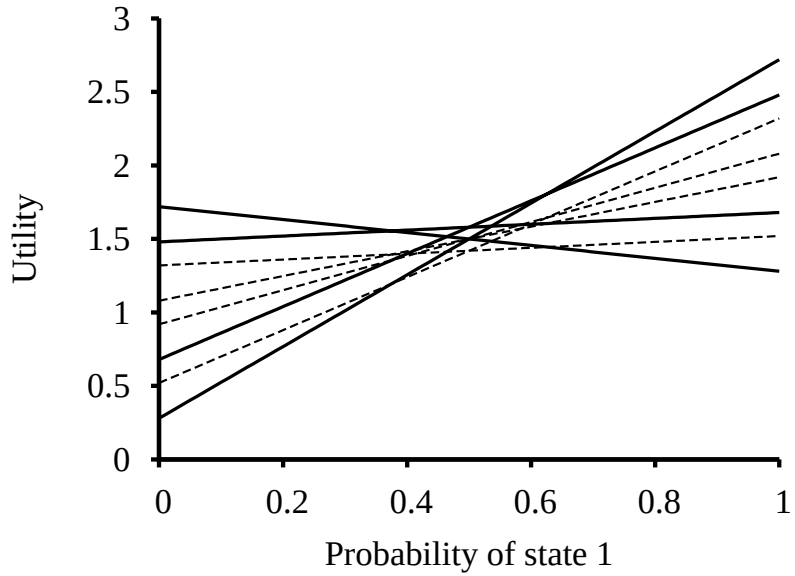
Simple Two-State World (4)

Next, we look at depth-2 plans (i.e., two actions are allowed).

This includes a first action, a subsequent percept, and then each way of choosing a depth-1 plan for each percept:

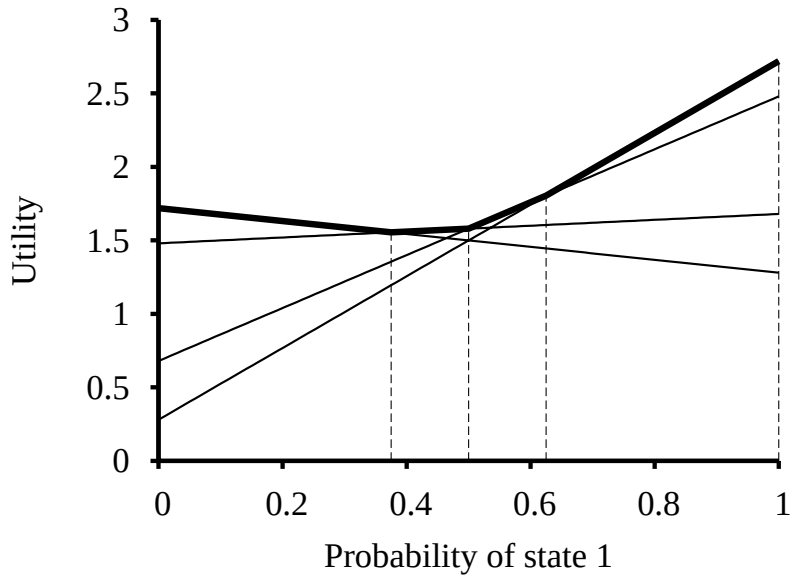
$$[\mathit{Stay};\ \textbf{if}\ \mathit{Percept} = 0\ \textbf{then}\ \mathit{Stay}\ \textbf{else}\ \mathit{Stay}]$$
$$[\mathit{Stay};\ \textbf{if}\ \mathit{Percept} = 0\ \textbf{then}\ \mathit{Stay}\ \textbf{else}\ \mathit{Go}]\ldots$$

There are 8 depth-2 plans, and their utilities are illustrated on the right. The 4 dashed lines are suboptimal across the entire belief space: they are **dominated** and need not be considered any longer.



Simple Two-State World (5)

The 4 remaining undominated plans are optimal in specific regions:



We repeat the process for depth 3, and so on. In general, let p be a depth-d conditional plan whose initial action is a and whose depth-$(d-1)$ subplan for percept e is $p.e$ so that

$$\alpha_p(s) = R(s) + \gamma \left(\sum_{s'} P(s'|s,a) \sum_{e} P(e|s') \alpha_{p.e}(s') \right) . \qquad (2)$$

The recursion naturally gives us a value iteration algorithm (see next slide).

POMDP-Value-Iteration Algorithm

function POMDP-Value-Iteration (*pomdp*,ϵ) **returns** a utility function

inputs: *pomdp*, a POMDP with states S, actions $A(s)$,
transition model $P(s'|s,a)$, sensor model $P(e|s)$,
reward $R(s)$, discount γ
ϵ, the maximum error allowed in the utility of any state

local variables: U, U', sets of plans p with associated utility vectors α_p

$U' \leftarrow$ a set containing just the empty plan [], with $\alpha_{[]}(s) = R(s)$

repeat

$U \leftarrow U'$

$U' \leftarrow$ the set of all plans consisting of an action and, for each possible next percept, a plan in U with utility vectors computed according to eq. (2)

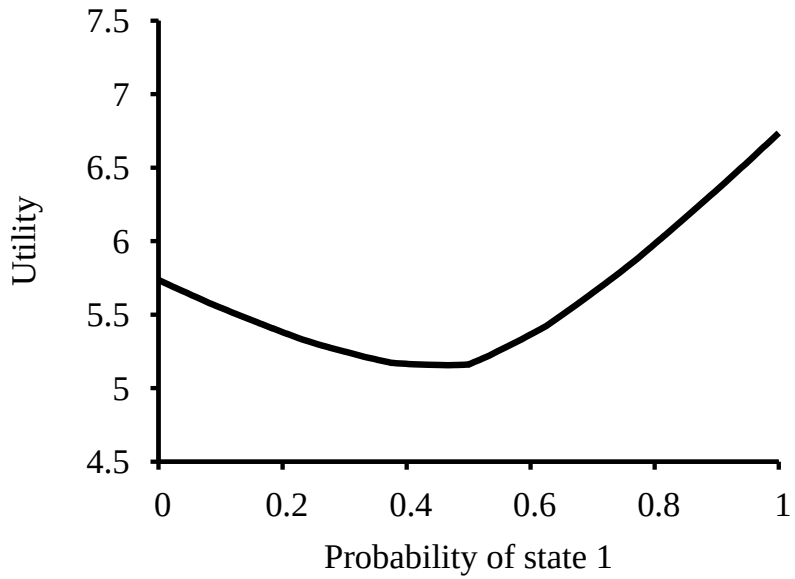
$U' \leftarrow$ `Remove-Dominated-Plans`(U')

until `Max-Difference` $(U,U') < \epsilon(1-\gamma)/\gamma$

return U

POMDP-Value-Iteration Algorithm: Complexity

- The complexity depends primarily on how many plans are generated.
- Given $|A|$ actions and $|E|$ possible observations, one can show that there are $|A|^{\mathcal{O}(|E|^{d-1})}$ distinct depth-d plans.
- Even for the two-state world with $d = 8$, the exact number is 2^{255}.
- The elimination of dominated plans is essential: the number of undominated plans for $d = 8$ is just 144 (see figure below).
- There exist approximate methods that are much faster to compute.



Summary

- Sequential decision problems in uncertain discrete environments can be modeled as Markov decision processes.
- The utility of a state sequence is the sum of all the rewards over the sequence, possibly discounted over time.
- The optimal solution of an MDP is a **policy** that associates a decision with every state that the agent might reach. A solution can be obtained by **value iteration**.
- **Policy iteration** converges faster since a policy might already be optimal without knowing the exact utilities of each state.
- Partially observable MDPs, or POMDPs, are much more difficult to solve than MDPs. They can be solved by conversion to an MDP in the continuous space of belief states.