Spezielle Themen der Künstlichen Intelligenz

11. Termin:

Markov Decision Problems (MDPs, POMDPs)

Decision-making under uncertainy

Principle of maximum expected utility (MEU)
An agent is rational **iff** it chooses the action that yields the highest expected utility, averaged over all possible outcomes of the action

Algorithm:

A non-deterministic action can have several outcomes $Result_i(A)$

Prior to executing A, the agent needs to...

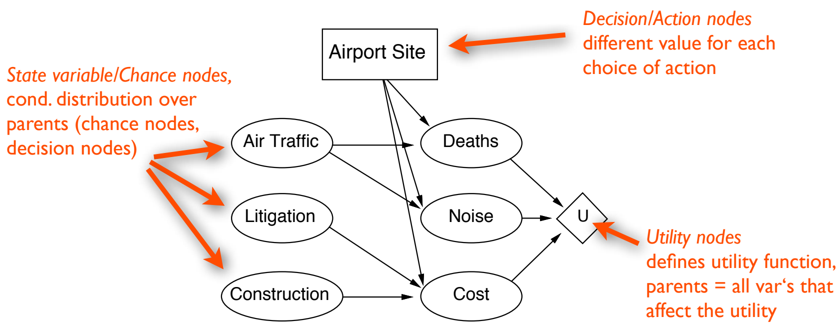
1. determine the probabilities $P(Result_i(A)|Do(A),E)$
2. calculate the expected utility of A, given evidence E:
   $EU(A|E)=\sum_i P(Result_i(A)|Do(A),E)\ U(Result_i(A))$
   with $U(S)$ utility function of state S
3. decide which action to take

Simple decisions for single actions only.

Bayesian Decision Networks

Extend Bayesian Networks to handle actions and utilities

- also called influence diagrams



Evaluating Decision Networks

Use Bayes Net inference methods to compute expected utilities.

Algorithm:

Set the evidence variables for current state

For *each possible value* of a decision node:

1. set decision node to that value
2. calculate the posterior probability of the parent nodes of the utility node, using BN inference
3. calculate the resulting (expected) utility for action

Pick the action with the highest utility

Question: what do I need to know to make a solid decision?

Value of information

Idea: compute value of acquiring each possible piece of evidence
Can be done **directly from decision network**

Example: buying oil drilling rights
Two blocks A and B, exactly one has oil, worth k
Prior probabilities 0.5 each, mutually exclusive
Current price of each block is $k/2$
"Consultant" offers accurate survey of A. Fair price?

Solution: compute expected value of information
= expected value of best action given the information
minus expected value of best action without information
Survey may say "oil in A" or "no oil in A", **prob. 0.5 each** (given!)
= [$0.5 \times$ value of "buy A" given "oil in A"
+ $0.5 \times$ value of "buy B" given "no oil in A"]
– 0
$= (0.5 \times k/2) + (0.5 \times k/2) - 0 = k/2$

General formula

Current evidence E, current best action α
Possible action outcomes S_i, potential new evidence E_j

$$EU(\alpha|E) = \max_a \Sigma_i \, U(S_i) \; P(S_i|E, a)$$

Value of current best action

Suppose we knew $E_j = e_{jk}$, then we would choose $\alpha_{e_{jk}}$ s.t.

$$EU(\alpha_{e_{jk}}|E, E_j = e_{jk}) = \max_a \Sigma_i \, U(S_i) \; P(S_i|E, a, E_j = e_{jk})$$

Value of new best action after evidence

E_j is a random variable whose value is *currently* unknown
$\Rightarrow$ must compute expected gain over all possible values:

$$VPI_E(E_j) = \left(\Sigma_k \, P(E_j = e_{jk}|E) EU(\alpha_{e_{jk}}|E, E_j = e_{jk})\right) - EU(\alpha|E)$$

Value of discovering E_j given current info E

(VPI = value of perfect information)

Information-gathering agent

```
function INFORMATION-GATHERING-AGENT(percept) returns eine Aktion
   static: D, ein Entscheidungsnetzwerk

   integriere percept in D
   j ← der Wert, der WPI(E_j) − Kosten(E_j) maximiert
   if WPI(E_j) > Kosten(E_j)
      then return REQUEST(E_j)
   else return die beste Aktion aus D
```

Agent chooses between sensing action (*REQUEST*, which will yield evidence in next percept) or „real action"

Extension: consider all possible sensing action sequences and all possible outcomes of those requests. Because values of requests depend on previous requests, need to build conditional plans

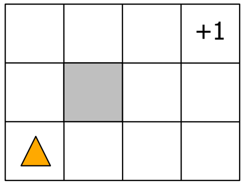
Now, making complex decisions

Sequential decision problems

- include utilities, uncertainty, and sensing
- utility depends on a sequence of actions
- generalizes search and planning problems to that of finding a suitable action „policy"

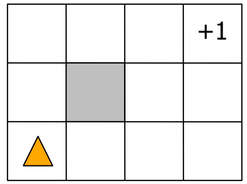
Consider for example robot navigation...

Simple Robot Navigation Problem



In each state, the possible actions are *U*, *D*, *R*, and *L*
Goal or terminal state marked +1

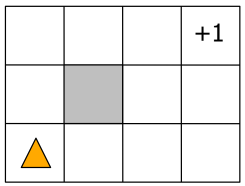
Simple Robot Navigation Problem

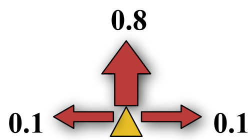


In each state, the possible actions are *U*, *D*, *R*, and *L*
Goal state marked +1

Environment fully observable and deterministic:
solution = sequence of actions (U, U, R, R, R)

Probabilistic Transition Model



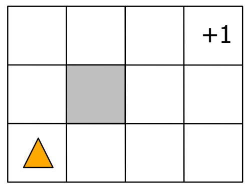


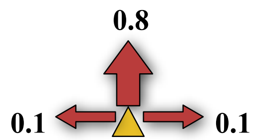
Non-deterministic actions due to stochastic motion:

Desired movement achieved with prob. 0.8, rest of the time agent moves left/right relative to the desired direction with prob. 0.1

The agent stays at same square if it bumps into a wall.

Probabilistic Transition Model

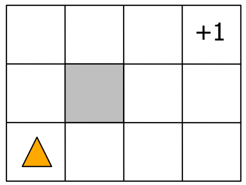


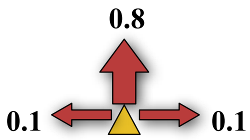


Non-deterministic actions due to stochastic motion:

sequences (U,U,R,R,R) reaches goal with prob $0.8^5 = 0.32768$
or accidentally with prob $0.1^4 x 0.8 = 0.00008$
➔ grand total prob of success = 0.32776

Probabilistic Transition Model

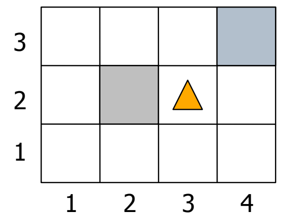




Transition model
T(s,a,s') = prob. of getting from s to s' by action a

Markovian: transition properties depend only on the current state, not on previous history (how that state was reached)

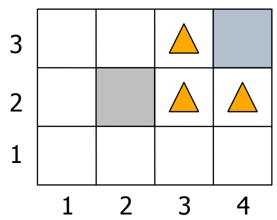
Sequence of Actions

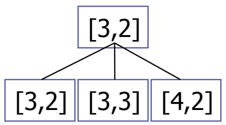


Position: [3,2]

Planned sequence of actions: (U, R)

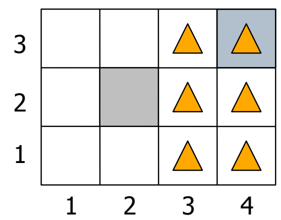
Sequence of Actions

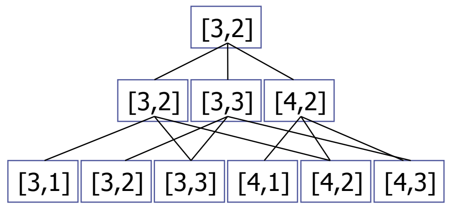




Planned sequence of actions: (U, R)
U is executed

Histories

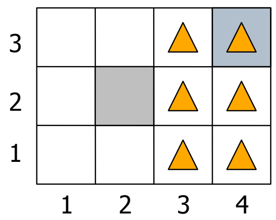




Planned sequence of actions: (U, R)
U, R is executed

There are 9 possible sequences of states – called histories – and 6 possible final states for the agent

Probability of Reaching the Goal



Only 2 possible routes:

P([4,3] | (U,R).[3,2]) =
P([4,3] | R.[3,3])
x **P**([3,3] | U.[3,2])
+ **P**([4,3] | R.[4,2])
x **P**([4,2] | U.[3,2])

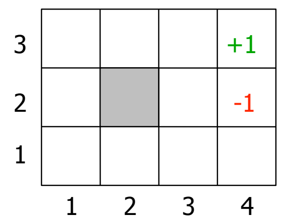
P([3,3] | U.[3,2]) = 0.8
P([4,2] | U.[3,2]) = 0.1

P([4,3] | R.[3,3]) = 0.8
P([4,3] | R.[4,2]) = 0.1

P([4,3] | (U,R).[3,2]) = 0.65

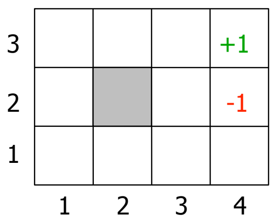
Note importance of Markov property in this derivation!

Utility Function



Agent receives reward *R(s)*
[4,3] or [4,2] are terminal states
[4,3] provides power supply, reward +1,
[4,2] is a sand area, reward -1
All other square have reward of -0.04

Utility of a History

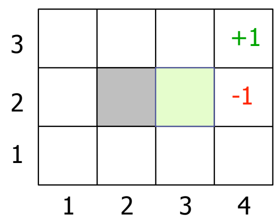


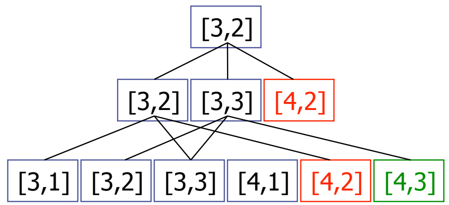
Utility is determined by the agent's history in the world.

Defined by the utility of the terminal state (+1 or –1)
plus the sum of rewards of the squares visited (additive reward)

→ agent wants to leave the environment as quickly as possible

Utility of a history

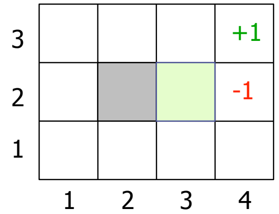


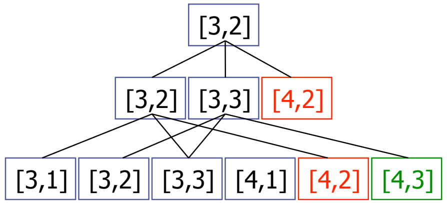


Consider the action sequence (U,R) from [3,2]
A run produces one among 7 possible histories (sequences), each with some probability P_h and an additive utility of histories:

$$\mathbf{U}_h = R(s_0)+R(s_1)+\ldots+R(s_n) = \sum R(s_i)$$

Utility of an Action Sequence





Utility of a history: $\mathbf{U}_h = R(s_0)+R(s_1)+\ldots+R(s_n) = \Sigma_i R(s_i)$

Utility of an action sequence is the expected utility of its histories:

$$\mathbf{U} = \Sigma_h \mathbf{U}_h \, \mathbf{P}(h)$$

Optimal Action Sequence

Consider the action sequence (U,R) from [3,2]
A run produces one among 7 possible histories.

The utility of the sequence is the expected utility of the histories. The optimal sequence is the one with maximal utility.

But is the optimal action sequence what we want to compute?

Only if the sequence is to be executed blindly.

A solution must specify what to do for any state the agent may reach!

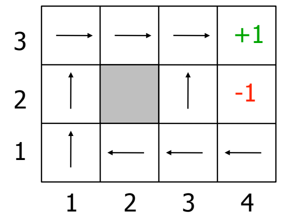
Reactive agent algorithm

Accessible or observable state

Repeat:

1. s ← sensed state
2. If s is terminal state then exit
3. a ← choose action (given s)
4. Perform a

Policy (Reactive/Closed-Loop Strategy)



A **policy** **Π** is a mapping from *states* to *actions*, i.e. recommends an action given a state *s*

A complete policy is defined for any possible world state.

Reactive agent algorithm

Repeat:

1. s ← sensed state
2. If s is terminal state then exit
3. a ← Π(s)
4. Perform a

R&N: „policy is a description of a simple reflex agent, computed from information used for a utility-based agent"

Optimal Policy

3	→	→	→	+1
2	↑		↑	-1
1	↑	←	←	←
	1	2	3	4

Note that [3,2] is a "dangerous" state that the optimal policy tries to avoid

The optimal policy Π* is the one that *always* yields a history (ending at a terminal state) with maximal expected utility

An optimal policy balances risks and rewards, depending on the value *R(s)* for non-terminal states

Key question: How to find an optimal policy?

Markov Decision Problem

Specification for sequential decision problems in *fully observable* environments:

1. Initial state s_0
2. Transition model T(s,a,s') = **P**(s'|a,s)
3. Reward function: R(s) or R(s,a,s'), additive

Question: How to compute the optimal policy Π* ?

This problem is called a
Markov Decision Problem (MDP)

Example

History h = $(s_0, s_1, \ldots, s_n)$

Utility of h: $\mathbf{U}(s_0, s_1, \ldots, s_n) = \Sigma\ \mathbf{R}(s_i)$

Robot navigation example:

- $\mathbf{R}(s_n) = +1$ if $s_n = [4,3]$
- $\mathbf{R}(s_n) = -1$ if $s_n = [4,2]$
- $\mathbf{R}(s_i) = -0.04$ if $i = 0, \ldots, n-1$

3	→	→	→	+1
2	↑		↑	-1
1	↑	←	←	←
	1	2	3	4

Given R(s), how to find an optimal policy??

Calculating the optimal policy

Idea:

Calculate the utility of all *states* first and use this information to select the optimal action when in a state

Utility of a state: $U(s_i) = R(s_i) + \max_a \sum_{s'} T(s_i, a, s')\, U(s')$ (Bellman eq.)

= *immediate reward for the current state plus the expected utility of next state assuming the agent chooses the optimal action*

R(s) = reward for *being in* s

U(s) = reward from s *onwards*

	1	2	3	4
3	0.812 →	0.868 →	0.918 →	+1
2	0.762 ↑		0.660 ↑	-1
1	0.705 ↑	0.655 ←	0.611 ←	0.388 ←

Calculating the optimal policy

With given utilities for each state, the agent can act using the MEU principle to follow the optimal policy:

→ Optimal policy: $\Pi^*(s_i) = \text{argmax}_a \sum_{s'} T(s_i,a,s')\, U(s')$

- one-step look-ahead using U(s)

Two algorithms to compute the optimal policy:

1. Value interation
2. Policy iteration

Value Iteration

For *n* states, there are *n* equations $U(s_i)$ with *n* unknowns, but with the non-linear *max*-operator → iterative approach:

- initialize the utilities, calc r.h.s., and update l.h.s. and all other utilities, until equilibrium reached.

Algorithm:

Initialize the utility of each non-terminal state s

For t = 0, 1, 2, … do:

$U_{t+1}(s) \leftarrow R(s) + \max_a \sum_{s'} T(s,a,s')\, U_t(s')$ (Bellman update)

Converges to the unique solution to Bellman eq., the correct utilities for the optimal policy

Value Iteration

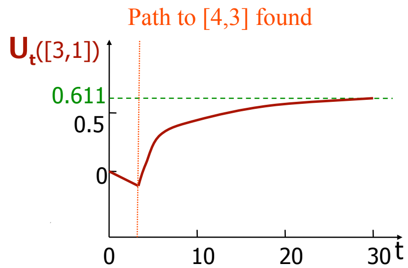
Note the importance of terminal states and connectivity of the state-transition graph

Initialize the utility of each non-terminal state s to 0

For t = 0, 1, 2, … do:

$U_{t+1}(s) \leftarrow R(s) + \max_a \sum_{s'} T(s,a,s')\, U_t(s')$

	1	2	3	4
3	0.812 →	0.868 →	0.918 →	+1
2	0.762 ↑		0.660 ↑	-1
1	0.705 ↑	0.655 ←	0.611 ←	0.388 ←



Policy Iteration

Idea:

1. policy evaluation: calculate utility of each state under Π
2. policy improvement: set new Π' using one-step look-ahead

Algorithm:

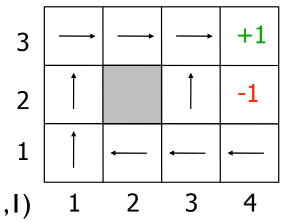
Pick a policy Π at random

Repeat

- 1. Compute the utility of each state for Π
  $U_{t+1}(s) \leftarrow R(s) + \Sigma_{s'} T(s,\Pi(s), s')U_t(s')$ Simpler, linear equations!
- 2. Compute a new policy Π' given these utilities
  $\Pi'(s) = \arg\max_a \Sigma_{s'} T(s,a,s')\ U(s')$
- If $\Pi' = \Pi$ then return Π Fixpoint of Bellman eq., optimal policy

Policy evaluation

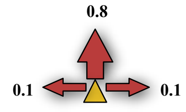
Example at time t:



→ $\Pi_t(1,1)$=Up, $\Pi_t(1,2)$=Up, ...

→ Simplified Bellman eq.:

- $U_t(1,1) = 0.8U_t(1,2) + 0.1U_t(2,1) + 0.1U_t(1,1)$
- $U_t(1,2) = 0.8U_t(1,3) + 0.1U_t(1,2)$
- .....



n linear equations with *n* unknowns, solvable in $O(n^3)$

Most efficient for small state spaces. Modified policy iteration (with simplified update) for large state spaces.

Infinite horizon - discouting

In many problems, e.g. robot navigation, histories are potentially unbounded and the same state can be reached many times.

Can use discounting to make infinite horizon problem mathematically tractable: discount factor $0 \leq \gamma \leq 1$

Utility of an infinite history: $U_h = \Sigma_i \gamma^i R(s_i)$ *(finite)*

Utility of states: $U(s) = R(s) + \gamma \max_a \Sigma_{s'} T(s, a, s')\ U(s')$

Iteration algorithms (update rules) work analog

Partially Observable Markov Decision Problem (POMDP)

MDPs assume fully observable environments and Markovian transition models.

POMDPs account in addition for partially observable environments: Which state is the agent in? Utility of s? Optimal action?

POMDPs are given by

1. Initial state s_0
2. Transition model $T(s,a,s') = \mathbf{P}(s'|a,s)$
3. Reward function $R(s)$ or $R(s,a,s')$, additive
4. Observation model: $O(s,o) = \mathbf{P}(o|s)$
   sensing operation in state s, returns multiple observations o, with a probability distribution

POMDPs

Following MEU assuming "state utilities" computed as above is not good enough, and actually is not rational

Belief state b(s) = prob. distribution over all possible states
- *Example*: in 4x3-world = point in 11-dim continuous space

The agent's policy is defined over its belief state: $\Pi^*(b)$
(actions *only* depend on beliefs, not the state the agent is in!)

POMDP decision cycle:
1. Given current belief state b, execute $a=\Pi^*(b)$
2. Get new observations o
3. Update belief states:
   $b'(s') = \alpha\, O(s',o) \sum_s T(s,a,s') b(s) =: FORWARD(b,a,o)$

POMDPs

Solving an POMDP on physical states can be reduced to solving an MDP on the corresponding belief states

- Define transition model over belief states (instead of world states) $\tau(b,a,b')$ and a reward function for belief states $\rho(b)= \Sigma_s b(s)R(s)$
- → observable MDP on (continuous, high-dim) space of belief states, whose optimal policy is also an optimal policy for the original POMDP
- need algorithmic versions of value- or policy iteration for continuous-state MDPs - possible but quickly intractable

Summary - decision-making

Simple decisions: single actions
- Preferences, utilities & MEU principle
- Bayesian Decision Networks & Value of Information

Complex decisions: sequence of actions
- Policies in probabilistic domains
- Markov Decision Problems (MDPs)
  - Value iteration
  - Policy iteration
- Partially Observable MDPs (POMDPs)

Overall summary

How to make systems behave smartly when things are (more or less) unknown?

Exact approaches:
- Search & Constraint Satisfaction
- Game Playing
- Planning

Probabilistic approaches:
- Degrees of belief & maximized expected utility
- Bayesian Networks: Modeling & inferencing
- Bayesian Decision Networks
- Markov Decision Problems