Foundations of Artificial Intelligence

13. Acting under Uncertainty

Maximizing Expected Utility

Wolfram Burgard, Bernhard Nebel and Martin Riedmiller

Albert-Ludwigs-Universität Freiburg

Contents

1. Introduction to Utility Theory
2. Choosing Individual Actions
3. Sequential Decision Problems
4. Markov Decision Processes
5. Value Iteration

The Basis of Utility Theory

The utility function rates states and thus formalizes the desirability of a state by the agent.

$U(S)$ denotes the utility of state S for the agent.

A nondeterministic action A can lead to the outcome states $Result_i(A)$. How high is the probability that the outcome state $Result_i(A)$ is reached, if A is executed in the current state with evidence E?

$$\rightarrow P(Result_i(A) \mid Do(A), E)$$

Expected Utility:

$$\boxed{EU(A \mid E) = \sum_i P(Result_i(A) \mid Do(A), E) \times U(Result_i(A))}$$

The principle of maximum expected utility (MEU) says that a rational agent should choose an action that maximizes $EU(A \mid E)$.

Problems with the MEU Principle

$$P(Result_i(A) \mid Do(A), E)$$

requires a complete causal model of the world.

$\rightarrow$ Constant updating of belief networks

$\rightarrow$ NP-complete for Bayesian networks

$$U(Result_i(A))$$

requires search or planning, because an agent needs to know the possible future states in order to assess the worth of the current state ("effect of the state on the future").

The Axioms of Utility Theory (1)

Justification of the MEU principle, i.e., maximization of the average utility.

Scenario = Lottery L

- Possible outcomes = possible prizes
- The outcome is determined by chance
- $L = [p_1, C_1; p_2, C_2; \ldots; p_n, C_n]$

Example:
Lottery L with two outcomes, C_1 and C_2:

$$L = [p, C_1; 1 - p, C_2]$$

Preference between lotteries:

$L_1 \succ L_2$	The agent prefers L_1 over L_2
$L_1 \sim L_2$	The agent is indifferent between L_1 and L_2
$L_1 \succsim L_2$	The agent prefers L_1 or is indifferent between L_1 and L_2

The Axioms of Utility Theory (2)

Given lotteries A, B, C

- Orderability
  $(A \succ B) \vee (B \succ A) \vee (A \sim B)$
  An agent should know what it wants: it must either prefer one of the 2 lotteries or be indifferent to both.

- Transitivity
  $(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$
  Violating transitivity causes irrational behavior: $A \succ B \succ C \succ A$. The agent has A and would pay to exchange it for C. C would do the same for A.
  $\rightarrow$ The agent loses money this way.

The Axioms of Utility Theory (3)

- Continuity
  $A \succ B \succ C \Rightarrow \exists p [p, A; 1-p, C] \sim B$
  If some lottery B is between A and C in preference, then there is some probability p for which the agent is indifferent between getting B for sure and the lottery that yields A with probability p and C with probability $1-p$.

- Substitutability
  $A \sim B \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]$
  If an agent is indifferent between two lotteries A and B, then the agent is indifferent beteween two more complex lotteries that are the same except that B is substituted for A in one of them.

The Axioms of Utility Theory (4)

- Monotonicity
  $A \succ B \Rightarrow (p > q \Leftrightarrow [p, A; 1-p, B] \succ [q, A; 1-q, B])$
  If an agent prefers the outcome A, then it must also prefer the lottery that has a higher probability for A.

- Decomposability
  $[p, A; 1-p, [q, B; 1-q, C]] \sim [p, A; (1-p)q, B; (1-p)(1-q), C]$
  Compound lotteries can be reduced to simpler ones using the laws of probability. This has been called the "no fun in gambling"-rule: two consecutive gambles can be reduced to a single equivalent lottery.

Utility Functions and Axioms

The axioms only make statements about preferences.

The existence of a utility function follows from the axioms!

- Utility Principle If an agent's preferences obey the axioms, then there exists a function $U : S \mapsto R$ with
  $U(A) > U(B) \Leftrightarrow A \succ B$
  $U(A) = U(B) \Leftrightarrow A \sim B$
- Expected Utility of a Lottery:
  $U([p_1, S_1; \ldots; p_n, S_n]) = \sum_i p_i U(S_i)$
  $\rightarrow$ Since the outcome of a nondeterministic action is a lottery, an agent can act rationally only by following the Maximum Expected Utility (MEU) principle.

How do we design utility functions that cause the agent to act as desired?

Assessing Utilities

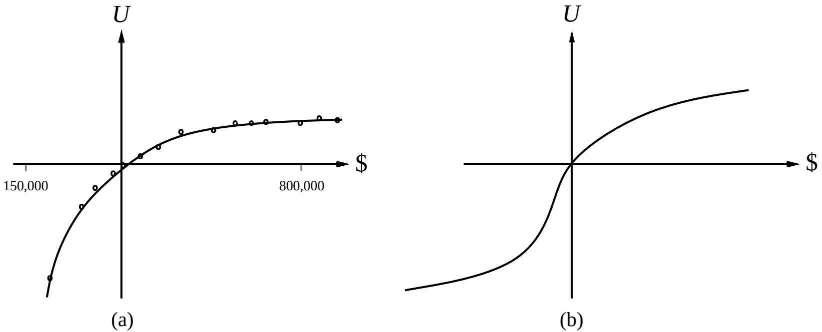
The scale of a utility function can be chosen arbitrarily. We therefore can define a 'normalized' utility:

- 'Best possible prize' $U(S) = u_{max} = 1$
- 'Worst catastrophe' $U(S) = u_{min} = 0$

Given a utility scale between u_{min} and u_{max} we can asses the utility of any particular outcome S by asking the agent to choose between S and a standard lottery $[p, u_{max}; 1-p, u_{min}]$. We adjust p until they are equally preferred.
Then, p is the utility of S. This is done for each outcome S to determine $U(S)$.

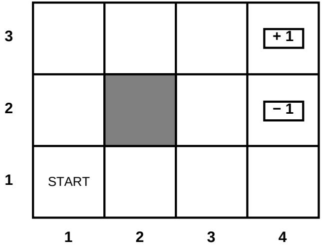
Possible Utility Functions

From economic models: The value of money



left: utility from empirical data; right: typical utility function over the full range.

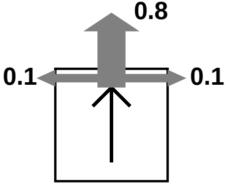
Sequential Decision Problems (1)



- Beginning in the start state the agent must choose an action at each time step.
- The interaction with the environment terminates if the agent reaches one of the goal states (4,3) (reward of $+1$) or (4,2) (reward -1). Each other location has a reward of -.04.
- In each location the available actions are Up, $Down$, $Left$, $Right$.

Sequential Decision Problems (2)

- Deterministic version: All actions always lead to the next square in the selected direction, except that moving into a wall results in no change in position.
- Stochastic version: Each action achieves the intended effect with probability 0.8, but the rest of the time, the agent moves at right angles to the intended direction.



Markov Decision Problem (MDP)

Given a set of states in an accessible, stochastic environment, an MDP is defined by

- Initial state S_0
- Transition Model $T(s, a, s')$
- Reward function $R(s)$

Transition model: $T(s, a, s')$ is the probability that state s' is reached, if action a is executed in state s.

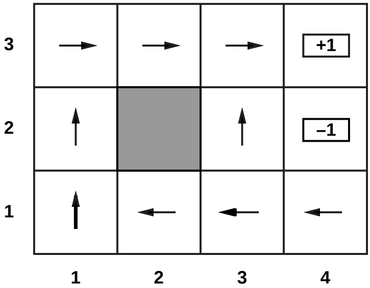
Policy: Complete mapping π that specifies for each state s which action $\pi(s)$ to take.

Wanted: The **optimal policy** π^* is the policy that maximizes the expected utility.

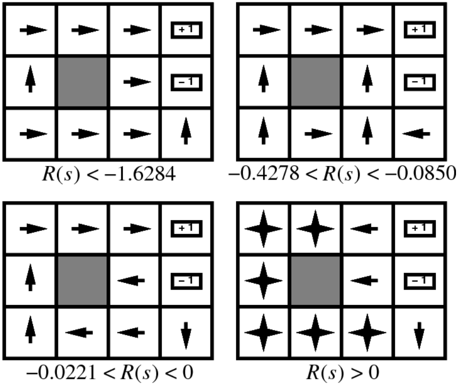
Optimal Policies (1)

- Given the optimal policy, the agent uses its current percept that tells it its current state.
- It then executes the action $\pi^*(s)$.
- We obtain a simple reflex agent that is computed from the information used for a utility-based agent.

Optimal policy for stochastic MDP with $R(s) = -0.04$:



Optimal Policies (2)



Optimal policy changes with choice of transition costs $R(s)$.
How to compute optimal policies?

Finite and Infinite Horizon Problems

- Performance of the agent is measured by the sum of rewards for the states visited.
- To determine an optimal policy we will first calculate the utility of each state and then use the state utilities to select the optimal action for each state.
- The result depends on whether we have a finite or infinite horizon problem.
- Utility function for state sequences: $U_h([s_0, s_1, \ldots, s_n])$
- Finite horizon: $U_h([s_0, s_1, \ldots, s_{N+k}]) = U_h([s_0, s_1, \ldots, s_N])$ for all $k > 0$.
- For finite horizon problems the optimal policy depends on the current state and the remaining steps to go. It therefore depends on time and is called nonstationary.
- In infinite horizon problems the optimal policy only depends on the current state and therefore is stationary.

Assigning Utilities to State Sequences

- For stationary systems there are two coherent ways to assign utilities to state sequences.
- Additive rewards:
  $U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \cdots$
- Discounted rewards:
  $U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$
- The term $\gamma \in [0, 1[$ is called the discount factor.
- With discounted rewards the utility of an infinite state sequence is always finite. The discount factor expresses that future rewards have less value than current rewards.

Utilities of States

- The utility of a state depends on the utility of the state sequences that follow it.
- Let $U^{\pi}(s)$ be the utility of a state under policy π.
- Let s_t be the state of the agent after executing π for t steps. Thus, the utility of s under π is

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

- The true utility $U(s)$ of a state is $U^{\pi^*}(s)$.
- $R(s)$ is the short-term reward for being in s and
  $U(s)$ is the long-term total reward from s onwards.

Example

The utilities of the states in our 4×3 world with $\gamma = 1$ and $R(s) = -0.04$ for non-terminal 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

Choosing Actions using the Maximum Expected Utility Principle

The agent simply chooses the action that maximizes the expected utility of the subsequent state:

$$\pi(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s, a, s') U(s')$$

The utility of a state is the immediate reward for that state plus the expected discounted utility of the next state, assuming that the agent chooses the optimal action:

$$U(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s, a, s') U(s')$$

Bellman-Equation

- The equation

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s')U(s')$$

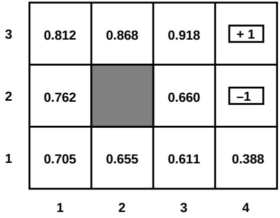
is also called the Bellman-Equation.

Bellman-Equation: Example

- In our 4×3 world the equation for the state (1,1) is

$$
\begin{aligned}
U(1,1) = -0.04 + \gamma \max\{ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && (Up)\\
& 0.9U(1,1) + 0.1U(1,2), && (Left)\\
& 0.9U(1,1) + 0.1U(2,1), && (Down)\\
& 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\} && (Right)\\
= -0.04 + \gamma \max\{ & 0.8 \cdot 0.762 + 0.1 \cdot 0.655 + 0.1 \cdot 0.705, && (Up)\\
& 0.9 \cdot 0.705 + 0.1 \cdot 0.762, && (Left)\\
& 0.9 \cdot 0.705 + 0.1 \cdot 0.655, && (Down)\\
& 0.8 \cdot 0.655 + 0.1 \cdot 0.762 + 0.1 \cdot 0.705\} && (Right)\\
= -0.04 + 1.0\,(& 0.6096 + 0.0655 + 0.0705), \quad (Up) = -0.04 + 0.7456 = 0.7056
\end{aligned}
$$

$\rightarrow$ Up is the optimal action in (1,1).



Value Iteration (1)

An algorithm to calculate an optimal strategy.

Basic Idea: Calculate the utility of each state. Then use the state utilities to select an optimal action for each state.

A sequence of actions generates a branch in the tree of possible states (histories). A utility function on histories U_h is separable iff there exists a function f such that

$$U_h([s_0, s_1, \ldots, s_n]) = f(s_0, U_h([s_1, \ldots, s_n]))$$

The simplest form is an additive reward function R:

$$U_h([s_0, s_1, \ldots, s_n]) = R(s_0) + U_h([s_1, \ldots, s_n]))$$

In the example, $R((4,3)) = +1$, $R((4,2)) = -1$, $R(other) = -1/25$.

Value Iteration (2)

If the utilities of the terminal states are known, then in certain cases we can reduce an n-step decision problem to the calculation of the utilities of the terminal states of the $(n-1)$-step decision problem.

$\rightarrow$ Iterative and efficient process

Problem: Typical problems contain cycles, which means the length of the histories is potentially infinite.

Solution: Use

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

where $U_t(s)$ is the utility of state s after t iterations.

Remark: As $t \rightarrow \infty$, the utilities of the individual states converge to stable values.

Value Iteration (3)

- The Bellman equation is the basis of value iteration.
- Because of the max-operator the n equations for the n states are nonlinear.
- We can apply an iterative approach in which we replace the equality by an assignment:

$$U'(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U(s')$$

The Value Iteration Algorithm

```
function VALUE-ITERATION(mdp, ε) returns a utility function
  inputs: mdp, an MDP with states S, actions A(s), transition model P(s' | s, a),
              rewards R(s), discount γ
          ε, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
                   δ, the maximum change in the utility of any state in an iteration

  repeat
      U ← U'; δ ← 0
      for each state s in S do
          U'[s] ← R(s) + γ max_{a ∈ A(s)} Σ_{s'} P(s' | s, a) U[s']
          if |U'[s] − U[s]| > δ then δ ← |U'[s] − U[s]|
  until δ < ε(1 − γ)/γ
  return U
```

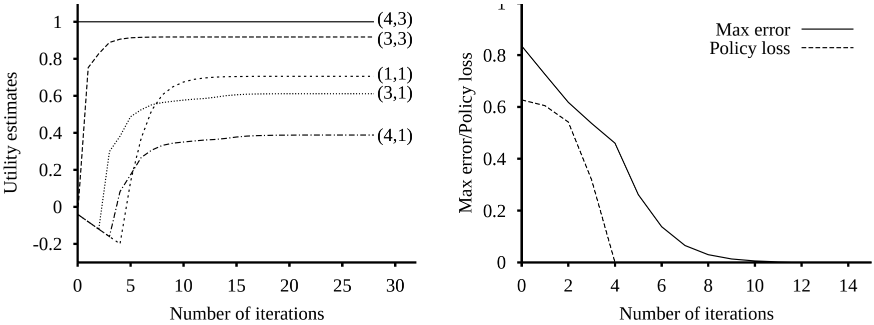
Convergence of Value Iteration

- Since the algorithm is iterative we need a criterion to stop the process if we are close enough to the correct utility.
- In principle we want to limit the policy loss $\|U^{\pi_t} - U\|$ that is the most the agent can lose by executing π_t.
- It can be shown that value iteration converges and that

  if $\quad \|U_{t+1} - U_t\| < \epsilon(1-\gamma)/\gamma \quad$ then $\quad \|U_{t+1} - U\| < \epsilon$

  if $\quad \|U_t - U\| < \epsilon \quad$ then $\quad \|U^{\pi_t} - U\| < 2\epsilon\gamma/(1-\gamma)$
- The value iteration algorithm yields the optimal policy π^*.

Application Example



In practice the policy often becomes optimal before the utility has converged.

Policy Iteration

- Value iteration computes the optimal policy even at a stage when the utility function estimate has not yet converged.
- If one action is better than all others, then the exact values of the states involved need not to be known.
- Policy iteration alternates the following two steps beginning with an initial policy π_0:
  - Policy evaluation: given a policy π_t, calculate $U_t = U^{\pi_t}$, the utility of each state if π_t were executed.
  - Policy improvement: calculate a new maximum expected utility policy π_{t+1} according to

$$\pi_{t+1}(s) = \operatorname*{argmax}_a \sum_{s'} T(s, a, s') U_t(s')$$

The Policy Iteration Algorithm

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      if $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s'] > \sum_{s'} P(s' \mid s, \pi[s])\ U[s']$ **then do**
        $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
        $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

Summary

- Rational agents can be developed on the basis of a probability theory and a utility theory.
- Agents that make decisions according to the axioms of utility theory possess a utility function.
- Sequential problems in uncertain environments (MDPs) can be solved by calculating a policy.
- Value iteration is a process for calculating optimal policies.