16.4 Multiattribute Utility Functions

- Most often the utility is determined by the values \( x = (x_1, \ldots, x_n) \) of multiple variables (attributes) \( X = X_1, \ldots, X_n \).
- For simplicity, we will assume that each attribute is defined in such a way that, all other things being equal, higher values of the attribute correspond to higher utilities.
- If for a pair of attribute vectors \( x \) and \( y \) it holds that \( x_i \geq y_i \ \forall \ i \), then \( x \) strictly dominates \( y \).
- Suppose that airport site \( S_1 \) costs less, generates less noise pollution, and is safer than site \( S_2 \), one would not hesitate to reject the latter.
- In the general case, where the action outcomes are uncertain, strict dominance occurs less often than in the deterministic case.
- Stochastic dominance is more useful generalization.

Suppose we believe that the cost of siting an airport is uniformly distributed between
- \( S_1 \): 2.8 and 4.8 billion euros
- \( S_2 \): 3.0 and 5.2 billion euros

Then by examining the cumulative distributions, we see that \( S_1 \) stochastically dominates \( S_2 \) (because costs are negative).
Cumulative distribution integrates the original distribution

If two actions $A_1$ and $A_2$ lead to probability distributions $p_1(x)$ and $p_2(x)$ on attribute $X$ then $A_1$ stochastically dominates $A_2$ on $X$ if

$$
\forall x: \int_{-\infty}^{x} p_1(x') \, dx' \leq \int_{-\infty}^{x} p_2(x') \, dx'
$$

If

- $A_1$ stochastically dominates $A_2$,
- then for any monotonically nondecreasing utility function $U(x)$,

the expected utility of $A_1$ is at least as high as that of $A_2$

Hence, if an action is stochastically dominated by another action on all attributes, then it can be discarded

---

16.6 The Value of Information

GP is hoping to buy one of $n$ indistinguishable blocks of ocean drilling rights at the Artic Sea

- Exactly one of the blocks contains oil worth $C$ euros

- The price for each block is $C/n$ euros

A seismologist offers GP the results of a survey of block #3, which indicates definitively whether the block contains oil

- With probability $1/n$, the survey will indicate oil in block #3, in which case GP will buy the block for $C/n$ euros and make a profit of $(n - 1)C/n$ euros

- With probability $(n - 1)/n$, the survey will show that the block contains no oil, in which case GP will buy a different block
– Now the probability of finding oil in one of the other blocks changes to \( \frac{1}{n-1} \), so GP makes an expected profit of
\[
\frac{C}{n-1} + \frac{n}{n} = \frac{C}{n(n-1)} \text{ euros}
\]

- Now we can calculate the expected profit, given the survey information:
\[
\frac{1}{n} \cdot \left( \frac{(n-1)C}{n} \right) + \frac{n-1}{n} \cdot \left( \frac{C}{n(n-1)} \right) = \frac{C}{n}
\]

- Therefore, GP should be willing to pay the seismologist up to the price of the block itself
- With the information, one’s course of action can be changed to suit the actual situation
- Without the information, one has to do what’s best on average over the possible situations

17 MAKING COMPLEX DECISIONS

- The agent’s utility now depends on a sequence of decisions
- In the following 4 \times 3 grid environment the agent makes a decision to move (U, R, D, L) at each time step
- When the agent reaches one of the goal states, it terminates
- The environment is fully observable — the agent always knows where it is
If the environment were deterministic, a solution would be easy: the agent will always reach +1 with moves [U, U, R, R, R].

Because actions are unreliable, a sequence of moves will not always lead to the desired outcome.

Let each action achieve the intended effect with probability 0.8 but with probability 0.1 the action moves the agent to either of the right angles to the intended direction.

If the agent bumps into a wall, it stays in the same square.

Now the sequence [U, U, R, R, R] leads to the goal state with probability

\[ 0.8^5 = 0.32768 \]

In addition, the agent has a small chance of reaching the goal by accident going the other way around the obstacle with a probability \( 0.1^4 \times 0.8 \), for a grand total of 0.32776.

A transition model specifies outcome probabilities for each action in each possible state.

Let \( P(s' | s, a) \) denote the probability of reaching state \( s' \) if action \( a \) is done in state \( s \).

The transitions are Markovian in the sense that the probability of reaching \( s' \) depends only on \( s \) and not the earlier states.

We still need to specify the utility function for the agent.

The decision problem is sequential, so the utility function depends on a sequence of states — an environment history — rather than on a single state.

For now, we will simply stipulate that is each state \( s \), the agent receives a reward \( R(s) \), which may be positive or negative.
• For our particular example, the reward is \(-0.04\) in all states except in the terminal states.
• The utility of an environment history is just (for now) the sum of rewards received.
• If the agent reaches the state +1, e.g., after ten steps, its total utility will be 0.6.
• The small negative reward gives the agent an incentive to reach \([4,3]\) quickly.

• A sequential decision problem for a fully observable environment with
  – A Markovian transition model and
  – Additive rewards
is called a **Markov decision process** (MDP).

An MDP is defined by the following four components:

– Initial state \(s_0\),
– A set \(\text{Actions}(s)\) of actions in each state,
– Transition model \(P(s' | s, a)\), and
– Reward function \(R(s)\).

As a solution to an MDP we cannot take a fixed action sequence, because the agent might end up in a state other than the goal.

• A solution must be a **policy**, which specifies what the agent should do for any state that the agent might reach.
• The action recommended by policy \(\pi\) for state \(s\) is \(\pi(s)\).
• If the agent has a complete policy, then no matter what the outcome of any action, the agent will always know what to do next.
Each time a given policy is executed starting from the initial state, the stochastic nature of the environment will lead to a different environment history.

The quality of a policy is therefore measured by the expected utility of the possible environment histories generated by the policy.

An optimal policy $\pi^*$ yields the highest expected utility.

- A policy represents the agent function explicitly and is therefore a description of a simple reflex agent.

$-0.0221 < R(s) < 0$: 

$-0.4278 < R(s) < -0.0850$: 

17.1.1 Utilities over time

- In case of an **infinite horizon** the agent’s action time has no upper bound.
- With a finite time horizon, the optimal action in a given state could change over time — the optimal policy for a finite horizon is **nonstationary**.
- With no fixed time limit, on the other hand, there is no reason to behave differently in the same state at different times, and the optimal policy is **stationary**.
- The **discounted** utility of a state sequence $s_0, s_1, s_2, ...$ is
  \[ R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots, \]
  where $0 \leq \gamma \leq 1$ is the discount factor.
• When $\gamma = 1$, discounted rewards are exactly equivalent to additive rewards
• The latter rewards are a special case of the former ones
• When $\gamma$ is close to 0, rewards in the future are viewed as insignificant

• If an infinite horizon environment does not contain a terminal state or if the agent never reaches one, then all environment histories will be infinitely long
• Then, utilities with additive rewards will generally be infinite
• With discounted rewards ($\gamma < 1$), the utility of even an infinite sequence is finite

Let $R_{\text{max}}$ be an upper bound for rewards. Using the standard formula for the sum of an infinite geometric series yields:

$$\sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \frac{\gamma R_{\text{max}}}{1 - \gamma}$$

• Proper policy guarantees that the agent reaches a terminal state when the environment contains such
• With proper policies infinite state sequences do not pose a problem, and we can use $\gamma = 1$ (i.e., additive rewards)

• An optimal policy using discounted rewards is

$$\pi^* = \arg \max_{\pi} \mathbb{E} \left( \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi \right)$$

where the expectation is taken over all possible state sequences that could occur, given that the policy is executed
### 17.2 Value Iteration

- For calculating an optimal policy we
  - calculate the utility of each state and
  - then use the state utilities to select an optimal action in each state
- The utility of a state is the expected utility of the state sequence that might follow it
- Obviously, the state sequences depend on the policy $\pi$ that is executed
- Let $s_t$ be the state the agent is in after executing $\pi$ for $t$ steps
- Note that $s_t$ is a random variable
- Then, executing $\pi$ starting in $s (= s_0)$ we have
  $$U^\pi(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \right]$$

- The true utility of a state $U(s)$ is just $U^\pi(s)$
- $R(s)$ is the short-term reward for being in $s$, whereas $U(s)$ is the long-term total reward from $s$ onwards
- In our example grid the utilities are higher for states closer to the +1 exit, because fewer steps are required to reach the exit

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.812</td>
<td>0.868</td>
<td>0.912</td>
<td>+1</td>
<td></td>
</tr>
<tr>
<td>0.762</td>
<td>0.660</td>
<td></td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>0.705</td>
<td>0.655</td>
<td>0.611</td>
<td>0.388</td>
<td></td>
</tr>
</tbody>
</table>
The Bellman equation for utilities

- The agent may select actions using the MEU principle
  \[ \pi^*(s) = \arg \max_a \sum_{s'} P(s' \mid s, a) U(s') \] (*)
- The utility of state \( s \) is the expected sum of discounted rewards from this point onwards, hence, we can calculate it:
  - Immediate reward in state \( s \), \( R(s) \)
  - The expected discounted utility of the next state, assuming that the agent chooses the optimal action
  \[ U(s) = R(s) + \gamma \max_a \sum_{s'} P(s' \mid s, a) U(s') \]
- This is called the Bellman equation
- If there are \( n \) possible states, then there are \( n \) Bellman equations, one for each state

\[ U(1,1) = -0.04 + \gamma \max\{ 0.6096 + 0.0655 + 0.0705 = 0.7456, \ (U) \\
0.6345 + 0.0762 = 0.7107, \ (L) \\
0.6345 + 0.0655 = 0.7000, \ (D) \\
0.5240 + 0.0762 + 0.0705 = 0.6707 \} \ (R) \]

- Using the values from the previous picture, this becomes:

\[ U(1,1) = -0.04 + \gamma \max\{ 0.6096 + 0.0655 + 0.0705 = 0.7456, \ (U) \\
0.6345 + 0.0762 = 0.7107, \ (L) \\
0.6345 + 0.0655 = 0.7000, \ (D) \\
0.5240 + 0.0762 + 0.0705 = 0.6707 \} \ (R) \]

- Therefore, \( Up \) is the best action to choose
Simultaneously solving the Bellman equations does not work using the efficient techniques for systems of linear equations, because max is a nonlinear operation.

In the iterative approach we start with arbitrary initial values for the utilities, calculate the right-hand side of the equation and plug it into the left-hand side:

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} P(s' | s, a) U_i(s'),$$

where index $i$ refers to the utility value of iteration $i$.

If we apply this Bellman update infinitely often, we are guaranteed to reach an equilibrium, in which case the final utility values must be solutions to the Bellman equations.

They are also the unique solutions, and the corresponding policy is optimal:

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

### 17.3 Policy Iteration

- Beginning from some initial policy $\pi_0$, alternate:
  - **Policy evaluation**: given a policy $\pi_i$, calculate $U_i = U^{\pi_i}$, the utility of each state if $\pi_i$ were to be executed.
  - **Policy improvement**: Calculate the new MEU policy $\pi_{i+1}$, using one-step look-ahead based on $U_i$ (Equation (*)).

- The algorithm terminates when the policy improvement step yields no change in utilities.
- At this point, we know that the utility function $U_i$ is a fixed point of the Bellman update and a solution to the Bellman equations, so $\pi_i$ must be an optimal policy.
- Because there are only finitely many policies for a finite state space, and each iteration can be shown to yield a better policy, policy iteration must terminate.
• Because at the $i$th iteration the policy $\pi_i$ specifies the action $\pi_i(s)$ in state $s$, there is no need to maximize over actions in policy iteration.

• We have a simplified version of the Bellman equation:

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

• For example:

$$U_i(1,1) = -0.04 + 0.8 U_i(1,2) + 0.1 U_i(1,1) + 0.1 U_i(2,1)$$
$$U_i(1,2) = -0.04 + 0.8 U_i(1,3) + 0.2 U_i(1,2)$$

etc.

• Now the nonlinear $\max$ has been removed, and we have linear equations.

• A system of linear equations with $n$ equations with $n$ unknowns can be solved exactly in time $O(n^3)$ by standard linear algebra methods.

• Instead of using a cubic amount of time to reach the exact solution (for large state spaces), we can instead perform some number simplified value iteration steps to give a reasonably good approximation of the utilities:

$$U_{i+1} = R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) U_i(s')$$

• This algorithm is called modified policy iteration.

• In asynchronous policy iteration we pick any subset of the states on each iteration and apply either policy improvement or simplified value iteration to that subset.

• Given certain conditions on the initial policy and initial utility function, asynchronous policy iteration is guaranteed to converge to an optimal policy.

• We can design, e.g., algorithms that concentrate on updating the values of states that are likely to be reached by a good policy.
18 LEARNING FROM EXAMPLES

- An intelligent agent may have to learn, for instance, the following components:
  - A direct mapping from conditions on the current state to actions
  - A means to infer relevant properties of the world from the percept sequence
  - Information about the way the world evolves and about the results of possible actions the agent can take
  - Utility information indicating the desirability of world states
  - Action-value information indicating the desirability of actions
  - Goals that describe classes of states whose achievement maximizes the agent’s utility

- The type of feedback available for learning determines the nature of the learning problem that the agent faces
  - **Supervised learning** involves learning a function from examples of its inputs and outputs
  - **Unsupervised learning** involves learning patterns in the input when no specific output values are supplied
  - In **reinforcement learning** the agent must learn from reinforcement (reward, punishment, less exact feedback than in supervised learning)
  - In **semi-supervised learning** we are given a few labeled examples and must make what we can of a large collection of unlabeled examples

- The representation of the learned information plays an important role in determining how the learning algorithm must work
18.2 Supervised Learning

- In deterministic supervised learning the aim is to recover the unknown function $f$ given examples $(x, f(x))$, where $x$ is the input (vector).
- In pure inductive inference (or induction) the result is a hypothesis $h$, which is a function that approximates $f$.
- A good hypothesis will generalize well — will predict unseen instances correctly.
- The hypothesis is chosen from a hypothesis space $\mathcal{H}$.
- For example, when both $x$ and $f(x)$ are real numbers, then $\mathcal{H}$ can be, e.g., the set of polynomials of degree at most $k$:
  
  $$3x^2 + 2, \quad x^{17} - 4x^3, \ldots$$

- A *consistent hypothesis* agrees with all the data.
- How do we choose from among multiple consistent hypotheses?
- *Occam’s (Ockham’s) razor*: prefer the simplest hypothesis consistent with the data.
- How to define simplicity?