**MAKING COMPLEX DECISIONS**

Outline
- Sequential Decision Problems
- Value Iteration
- Policy Iteration
- Decision-Theoretic Agents

Based on the textbook by Stuart Russell & Peter Norvig:

*Artificial Intelligence, Modern Approach (2nd Edition)*
Chapter 17: excluding Sections 17.4, 17.6, and 17.7

---

**1. SEQUENTIAL DECISION PROBLEMS**

**Example.** An agent is situated in a fully observable environment:

![State Grid](attachment:state_grid.png)

- The agent may perform actions *North*, *South*, *East*, and *West* in order to move between squares (or states) (1,1), ..., (4,3).
- Moving towards a wall results in no change in position.
- The operation of the agent stops and it receives a *reward/punishment* if it reaches a square marked with +1/−1.

---

**Transition Model**

- In a *deterministic setting* the outcomes of actions are known, and the agent may *plan* a sequence of actions which moves it to (4,3).
- This becomes impossible if actions are *nondeterministic/unreliable,*
- A *transition model* assigns a probability $T(s,a,s')$ to the event that the agent reaches state $s'$ when it performs action $a$ in state $s$. Transitions are *Markovian* in the sense of Chapter 15.

**Example.** (Continued) Each one of the four actions *North*, *South*, *East*, and *West* moves the agent

1. to the intended direction $d$ with a probability of 0.8, and
2. at right angles to the direction $d$ with probabilities 0.1 and 0.1.

---

**Example.** If an action sequence $S = \{North, East\}$ is performed in state (3,2) the agent reaches states with following probabilities:

\[
\begin{align*}
P_{(3,1)} &= 0.1 \times 0.1 = 0.01 \\
P_{(3,2)} &= 0.8 \times 0.1 = 0.08 \\
P_{(3,3)} &= 0.8 \times 0.1 \times 0.1 = 0.09 \\
P_{(4,2)} &= 0.1 \times 0.1 \times 0.8 = 0.08 \\
P_{(4,3)} &= 0.8 \times 0.8 = 0.64
\end{align*}
\]

These are easily inspected from a (partial) *reachability graph*:
Assigning Utility to Sequences of States

- The utility function $U$ is based on a sequence of states — an environment history — rather than a single state.
- For now, we stipulate that in each state $s$, the agent receives a reward $R(s)$, which may be positive or negative.
- An additive utility function is assumed: the utility of an environment history is just the sum of rewards received.

**Example.** In our example, the reward $R(s) = -\frac{1}{75}$ is for all states $s$ except terminal states which have rewards $+1$ and $-1$, respectively.
If the agent reaches the $+1$ state after 10 steps, its total utility is 0.6.
The reward of $-\frac{1}{75}$ gives the agent an incentive to reach $(4,3)$ soon.

Optimal Policies

- We write $\pi(s)$ for the action recommended by $\pi$ in a state $s$.
- The quality of a policy $\pi$ is measured by the expected utility of the possible environment histories generated by that policy.
- An optimal policy $\pi^*$ is a policy that yields the highest expected utility, as determined by the MEU principle.
- Given an optimal policy $\pi^*$, the agent determines the current state $s$ using its percept and chooses $\pi^*(s)$ as the next action.
- An optimal policy can be viewed as a description of a simple reflex agent extracted from the specification of a utility-based agent.

Markov Decision Processes

- The specification of a 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 three components:
  1. Initial state: $s_0$
  2. Transition model: $T(s,a,s')$ for all states $s$, $s'$, and actions $a$.
  3. Reward function: $R(s)$ for all states $s$.
- A solution is a policy $\pi$, i.e., a mapping from states to actions.
- In the sequel, we will study two basic techniques for computing policies, namely value iteration and policy iteration.

**Example.** An optimal policy for the square world appears on the left.

The expected utilities for individual states are given on the right.
- The policy is very conservative (tries to avoid punishment).
- If the cost of moves is increased, then the optimal policy becomes different for the state $(3,1)$: West is replaced by North.
- If the cost of moves is decreased to $\frac{1}{100}$, then West is chosen instead of North in state $(3,2)$.
Optimality in Sequential Decision Problems

We are interested in the possible choices for the utility function $U_h$ on environment histories $[s_0,s_1,\ldots,s_n]$.

The first question is to answer whether there is a finite horizon, i.e., $U_h([s_0,s_1,\ldots,s_{N+k}]) = U_h([s_0,s_1,\ldots,s_N])$ for some fixed time $N$ and every $k > 0$.

If not, then we have an infinite horizon.

The optimal policy for a finite horizon is nonstationary, i.e., optimal actions in particular states may change over time.

With no fixed time limit, the optimal action depends only on the current state, and the optimal policy becomes stationary.

Calculating the Utility of State Sequences

A preference independence assumption: the agent's preferences are stationary: if state sequences $[s_0,s_1,\ldots]$ and $[r_0,r_1,\ldots]$ begin with equally preferred $s_0$ and $r_0$, then these sequences should be preference ordered like $[s_1,s_2,\ldots]$ and $[r_1,r_2,\ldots]$.

Given stationarity, there are basically two ways to assign utilities:

Additive rewards: $U_h([s_0,s_1,\ldots]) = R(s_0) + R(s_1) + R(s_2) + \ldots$

Discounted rewards, which generalize additive rewards:

$U_h([s_0,s_1,\ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$

where $0 \leq \gamma \leq 1$ is a discount factor.

In discounting, future rewards $R(s_i) \leq R_{\text{max}}$ where $i > 0$ are considered less valuable than the current reward $R(s_0)$.

2. Value Iteration

In value iteration, the basic idea is to compute the utility $U(s)$ for each state $s$ and to use these utilities for selecting optimal actions.

It is difficult to determine $U(s)$ because of uncertain actions.

Given a transition model, the agent is supposed to choose the action that maximizes the expected utility of the subsequent state:

$\pi^*(s) = \arg \max_{\pi} \sum_{s'} T(s,a,s') U(s')$.

The utility of a state $s$ is the immediate reward for that state plus the discounted MEU of the next states [Bellman, 1957]:

$U(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s,a,s') U(s')$. 
The Value Iteration Algorithm

- Given \( n \) states, the Bellman equation leads to a set of \( n \) non-linear equations for utilities that can be approximated by iteration.
- We write \( U_i(s) \) for the utility of state \( s \) at the \( i \)th iteration.
- The initial value \( U_i(s) = 0 \) for each state \( s \).
- One iteration step, called a Bellman update, is defined by
  \[
  U_{i+1}(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s,a,s')U_i(s')
  \]
  for each \( i \geq 0 \) and for each state \( s \).
- The following termination condition is used by the algorithm:
  \[
  \max_s |U_{i+1}(s) - U_i(s)| < \frac{\epsilon(1 - \gamma)}{\gamma}.
  \]

Convergence of Value Iteration

- Value iteration eventually converges to a unique set of solutions of the Bellman equations.
- The Bellman update is a contraction by a factor of \( \gamma \) on utility vectors: \( \max_s |U_{i+1}(s) - U_i(s)| \leq \gamma \max_s |U_i(s) - U_i(s)| \) for all \( i \geq 0 \).

**Example.** For the square world, value iteration converges as follows:

3. **Policy Iteration**

- The optimal policy is often not very sensitive to the utility values.
- The basic idea in policy iteration is to choose an initial policy \( \pi_0 \), calculate utilities using \( \pi_0 \) as policy and update \( \pi_0 \) (repeatedly).

1. **Policy evaluation:** the utilities of states are determined using \( \pi_i \) and the simplified Bellman update for \( j \geq 0 \):
   \[
   U_{j+1}(s) = R(s) + \gamma \sum_{s'} T(s,\pi_i(s),s')U_j(s').
   \]
   Another possibility is to solve utilities directly from the simplified Bellman equation by setting \( U_{j+1}(s) = U_j(s) \).

2. **Policy improvement:** a new MEU policy \( \pi_{i+1} \) is calculated (until \( \pi_{i+1} = \pi_i \)) using the utility values based on \( \pi_i \).
Example. The utilities of states (3, 2) and (3, 3) are solved as follows:

$$
\begin{align*}
  u(3, 2) &= -0.04 + 0.8u(3, 3) + 0.1u(3, 2) = 0.1 \\
  u(3, 3) &= -0.04 + 0.8 + 0.1u(3, 3) + 0.1u(3, 2) \\
  \Rightarrow & \quad -0.8u(3, 3) = -0.9u(3, 2) - 0.14 \\
  \Rightarrow & \quad 8.1u(3, 3) = 0.9u(3, 2) + 6.84 \\
  \Rightarrow & \quad u(3, 3) = \frac{6.7}{3} \approx 0.918 \text{ and } u(3, 2) = \frac{0.8u(3, 3) - 0.14}{0.9} \approx 0.660.
\end{align*}
$$

4. DECISION-THEORETIC AGENT DESIGN

A comprehensive approach to agent design for partially observable, stochastic environments is based on the following elements:

- The transition and observation models are represented as a dynamic Bayesian network (DBN).
- This model is extended with decision and utility nodes, as in decision networks, to form a dynamic decision network (DDN).
- A filtering algorithm is used to incorporate each new percept and action, and to update the agent’s estimate on the current state.
- Decisions are made by projecting forward possible action sequences and choosing the best one.

SUMMARY

- A optimal policy associates an optimal decision with every state that the agent might reach.
- Value iteration and policy iteration are two methods for calculating optimal policies.
- Unbounded action sequences can be dealt with discounting.
- Dynamic Bayesian networks can handle sensing and updating over time, and provide a direct implementation of the update cycle.
- Dynamic decision networks can solve sequential decision problems arising for agents in complex, uncertain domains.
1. Recall the belief network that you designed for representing the ball tracking mechanism of a soccer playing agent.
   ➤ Is it possible to identify a state evolution model and a sensor model from your network?
   ➤ If not, reconstruct the network by keeping these in mind.

2. Continue the analysis of soccer playing agents.
   ➤ Can you identify other problems in this domain that can be considered as real sequential decision problems?
   ➤ Try to formalize such a problem as a dynamic decision network.