MAKING COMPLEX DECISIONS

Outline

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

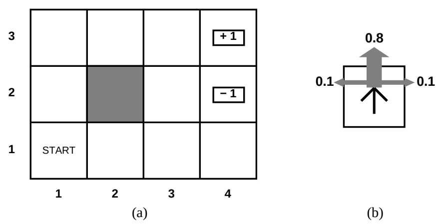
Based on the textbook by Stuart Russell & Peter Norvig:

Artificial Intelligene, A 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:



- 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.

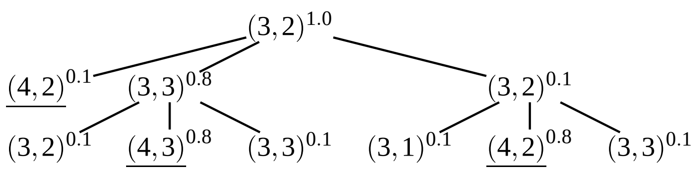
Example. (Continued) Each one of the four actions *North*, *South*, *East*, and *West* moves the agent

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 folowing probabilities:

$$\begin{aligned}
P_{(3,1)} &= 0.1\times 0.1 = & 0.01\\
P_{(3,2)} &= 0.8\times 0.1 = & 0.08\\
P_{(3,3)} &= 0.8\times 0.1 + 0.1\times 0.1 = & 0.09\\
P_{(4,2)} &= 0.1 + 0.1\times 0.8 = & 0.18\\
P_{(4,3)} &= 0.8\times 0.8 = & 0.64\\
\hline
& & 1.00
\end{aligned}$$

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 $R(s) = -\frac{1}{25}$ is for all states s except terminal states whichave rewards $+1$ and -1, respectively.

If the agent reaches the $+1$ state after 10 steps, its total utility is 0.6.

The reward of $-\frac{1}{25}$ gives the agent an incentive to reach $(4,3)$ soon.

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 folowing 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** π, 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**.

Optimal Policies

- We write $\pi(s)$ for the action recommended by π in a state s.
- The quality of a policy π is measured by the *expected utility* of the possible environment histories generated by that policy.
- An **optimal policy** π^* is a policy that yields the highest expected utility, as determined by the MEU principle.
- Given an optimal policy π^*, 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.

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

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

	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

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

- There are three ways to deal with infinite state sequences:
  1. With discounted rewards bounded by R_{max}, the utility of an infinite sequence becomes finite:
  $$U_h([s_0, s_1, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{1-\gamma}.$$
  2. Given a **proper policy**, which is guaranteed to reach a terminal state, the discount factor $\gamma = 1$ can be used.
  3. Yet another possibility is to compare infinite sequences in terms of the **average reward** obtained per time step.
- An optimal policy π^* is obtained as
  $$\arg\max_{\pi} \sum_{[s_0, s_1, \ldots]} P([s_0, s_1, \ldots] \mid \pi) U_h([s_0, s_1, \ldots])$$
  where $P([s_0, s_1, \ldots] \mid \pi)$ is determined by the transition model.

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

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

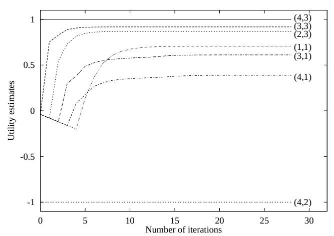
  for each $i \geq 0$ and for each state s.
- The following *termination condition* is used by the algorithm:

$$\max_s |U_{i+1}(s) - U_i(s)| < \frac{\varepsilon(1-\gamma)}{\gamma}.$$

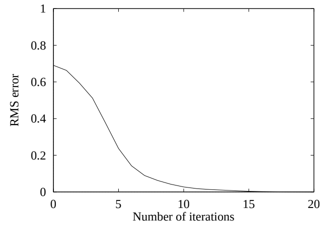
Convergence of Value Iteration

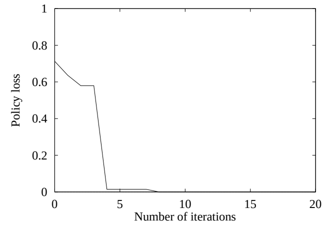
- Value iteration eventually converges to a unique set of solutions of the Bellman equations.
- The Bellman update is a **contraction** by a factor of γ on utility vectors: $\max_s |U_{i+1}(s) - U(s)| \leq \gamma \max_s |U_i(s) - U(s)|$ for all $i \geq 0$.

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



- Given stabilized utility values $U_{i+1}(s) = U_i(s)$, the corresponding *optimal policy* π^* can be determined.
- Unfortunately, it is difficult to estimate how long the value iteration algorithm should be run to get an optimal policy.
- Alternatively, policies can be evaluated using **policy loss**, i.e., the difference of expected utility with respect to the optimal policy.





☞ An optimal policy is reached long before utilities converge.

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 π_0, calculate utilities using π_0 as policy and update π_0 (repeatedly).

1. **Policy evaluation**: the utilities of states are determined using π_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 π_{i+1} is calculated (until $\pi_{i+1} = \pi_i$) using the utility values based on π_i.

Example. The utilities of states $(3,2)$ and $(3,3)$ are solved as follows:

$$\begin{cases} u_{(3,2)} = -0.04 + 0.8u_{(3,3)} + 0.1u_{(3,2)} - 0.1 \\ u_{(3,3)} = -0.04 + 0.8 + 0.1u_{(3,3)} + 0.1u_{(3,2)} \end{cases}$$

$$\Longrightarrow \begin{cases} -0.8u_{(3,3)} = -0.9u_{(3,2)} - 0.14 \\ 8.1u_{(3,3)} = 0.9u_{(3,2)} + 6.84 \end{cases}$$

$$\Longrightarrow u_{(3,3)} = \frac{6.7}{7.3} \approx 0.918 \text{ and } u_{(3,2)} = \frac{0.8u_{(3,3)} - 0.14}{0.9} \approx 0.660.$$

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

	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

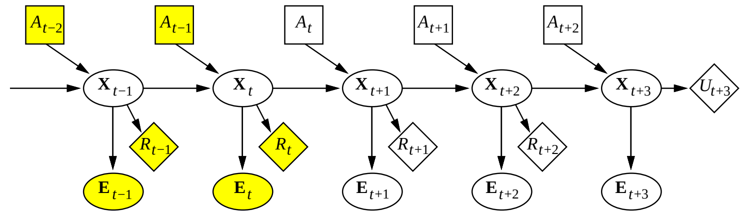
4. DECISION-THEORETIC AGENT DESIGN

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.

Dynamic Decision Networks

The generic structure of a dynamic decision network is as folows:



- The transition model $T(s_t, a, s')$ is the same as $\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{X}_t, A_t)$ where A_t denotes the action at time t.
- The observation model $O(s,o)$, which defines the probability of perceiving the observation o in state s, is the same as $\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$.

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.

QUESTIONS

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.