

Informatics 2D – Reasoning and Agents

Semester 2, 2011-12

Alex Lascarides
alex@inf.ed.ac.uk

 School of
informatics



Lecture 30 – Markov Decision Processes
30th March 2012
adapted from slides by Michael Rovatsos

Where are we?

Last time ...

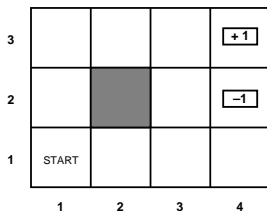
- ▶ Talked about decision making under uncertainty
- ▶ Looked at utility theory
- ▶ Discussed axioms of utility theory
- ▶ Described different utility functions
- ▶ Introduced decision networks

Today ...

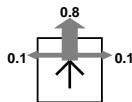
- ▶ **Markov Decision Processes**

Sequential decision problems

- ▶ So far we have only looked at one-shot decisions, but decision process are often sequential
- ▶ Example scenario: a 4x3-grid in which agent moves around (fully observable) and obtains utility of +1 or -1 in terminal states



(a)



(b)

- ▶ Actions are somewhat unreliable (in deterministic world, solution would be trivial)

Markov decision processes

- ▶ To describe such worlds, we can use a **(transition) model** $T(s, a, s')$ denoting the probability that action a in s will lead to state s'
- ▶ Model is Markovian: probability of reaching s' depends only on s and not on history of earlier states
- ▶ Think of T as big three-dimensional table (actually a DBN)
- ▶ Utility function now depends on **environment history**
 - ▶ agent receives a reward $R(s)$ in each state s (e.g. -0.04 apart from terminal states in our example)
 - ▶ (for now) utility of environment history is the sum of state rewards
- ▶ In a sense, stochastic generalisation of search algorithms!

Markov decision processes

- ▶ Definition of a **Markov Decision Process (MDP)**:

Initial state: S_0

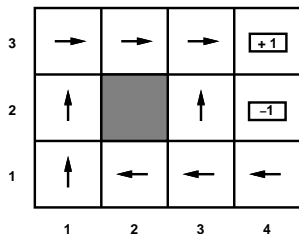
Transition model: $T(s, a, s')$

Utility function: $R(s)$

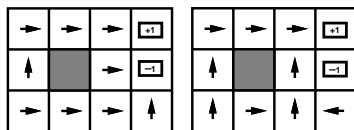
- ▶ Solution should describe what agent does in every state
- ▶ This is called **policy**, written as π
- ▶ $\pi(s)$ for an individual state describes which action should be taken in s
- ▶ **Optimal policy** is one that yields the highest expected utility (denoted by π^*)

Example

- ▶ Optimal policies in the 4x3-grid environment
 - With cost of -0.04 per intermediate state π^* is conservative for (3,1)
 - Different cost induces direct run to terminal state/shortcut at (3,1)/no risk/avoid both exits

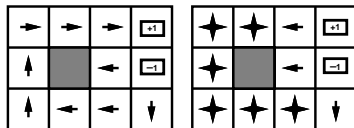


(a)



$$R(s) < -1.6284$$

$$-0.4278 < R(s) < 0.085$$



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

$$R(s) > 0$$

(b)

Optimality in sequential decision problems

- ▶ MDPs very popular in various disciplines, different algorithms for finding optimal policies
- ▶ Before we present some of them, let us look at utility functions more closely
- ▶ We have used sum of rewards as utility of environment history until now, but what are the alternatives?
- ▶ First question: **finite horizon** or **infinite horizon**
- ▶ Finite means there is a fixed time N after which nothing matters:

$$\forall k \ U_h([s_0, s_1, \dots, s_{N+k}]) = U_h([s_0, s_1, \dots, s_N])$$

Optimality in sequential decision problems

- ▶ This leads to **non-stationary** optimal policies (N matters)
- ▶ With infinite horizon, we get **stationary** optimal policies (time at state doesn't matter)
- ▶ We are mainly going to use infinite horizon utility functions
- ▶ NOTE: sequences to terminal states can be finite even under infinite horizon utility calculation
- ▶ Second issue: how to calculate utility of sequences
- ▶ **Stationarity** here is reasonable assumption:

$$s_0 = s'_0 \wedge [s_0, s_1, s_2 \dots] \succ [s'_0, s'_1, s'_2, \dots] \Rightarrow [s_1, s_2 \dots] \succ [s'_1, s'_2, \dots]$$

Optimality in sequential decision problems

- ▶ Stationarity may look harmless, but there are only two ways to assign utilities to sequences under stationarity assumptions
- ▶ **Additive rewards:**

$$U_h([s_0, s_1, s_2 \dots]) = R(s_0) + R(S_1) + R(S_2) + \dots$$

- ▶ **Discounted rewards** (for **discount factor** $0 \leq \gamma \leq 1$)

$$U_h([s_0, s_1, s_2 \dots]) = R(s_0) + \gamma R(S_1) + \gamma^2 R(S_2) + \dots$$

- ▶ Discount factor makes more distant future rewards less significant
- ▶ We will mostly use discounted rewards in what follows

Optimality in sequential decision problems

- ▶ Choosing infinite horizon rewards creates a problem
- ▶ Some sequences will be infinite with infinite (additive) reward, how do we compare them?
- ▶ Solution 1: with discounted rewards the utility is bounded if single-state rewards are

$$U_h([s_0, s_1, s_2 \dots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = R_{max}/(1 - \gamma)$$

- ▶ Solution 2: under **proper policies**, i.e. if agent will eventually visit terminal state, additive rewards are finite
- ▶ Solution 3: compare **average reward** per time step

Value iteration

- ▶ **Value iteration** is an algorithm for calculating optimal policy in MDPs

Calculate the utility of each state and then select optimal action based on these utilities

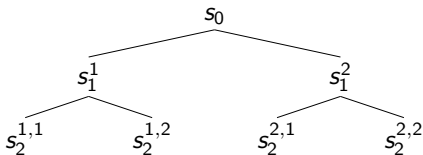
- ▶ Since discounted rewards seemed to create no problems, we will use

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

as a criterion for optimal policy

Explaining $\pi^* = \arg \max_{\pi} E [\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi]$

- ▶ Each policy π yields a tree, with root node s_0 , and daughters to a node s are the possible successor states given the action $\pi(s)$.
 - ▶ $T(s, a, s')$ gives the probability of traversing an arc from s to daughter s' .



- ▶ E is computed by:
 - (a) For each path p in the tree, getting the product of the (joint) probability of the path in this tree with its discounted reward, and then
 - (b) Summing over all the products from (a)
- ▶ So this is just a generalisation of single shot decision theory.

Utilities of states: $U(s) \neq R(s)$!

- ▶ $R(s)$ is reward for being in s *now*.
- ▶ By making $U(s)$ the utility of the states that might follow it, $U(s)$ captures *long-term* advantages from being in s

*$U(s)$ reflects what you can do from s ;
 $R(s)$ does not.*

- ▶ States that follow depend on π . So utility of s given π is:

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

- ▶ With this, “true” utility $U(s)$ is $U^{\pi^*}(s)$ (expected sum of discounted rewards if executing optimal policy)

Utilities in our example

- ▶ $U(s)$ computed for our example from algorithms to come.
- ▶ $\gamma = 1$, $R(s) = -0.04$ for nonterminals.

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

Utilities of states

- ▶ Given $U(s)$, we can easily determine optimal policy:

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

- ▶ Direct relationship between utility of a state and that of its neighbours:

Utility of a state is immediate reward plus expected utility of subsequent states if agent chooses optimal action

- ▶ This can be written as the famous **Bellman equations**:

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

The value iteration algorithm

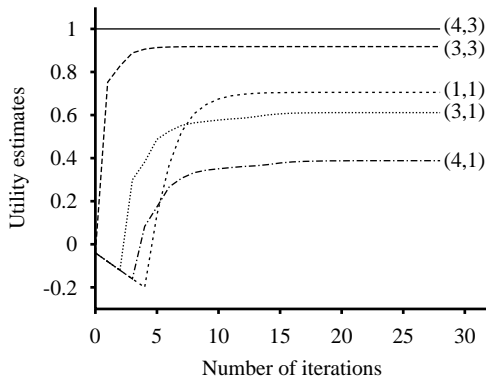
- ▶ For n states we have n Bellman equations with n unknowns (utilities of states)
- ▶ Value iteration is an iterative approach to solving the n equations.
- ▶ Start with arbitrary values and update them as follows:

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

- ▶ The algorithm converges to right and unique solution
- ▶ Like propagating values through network or utilities

The value iteration algorithm

- ▶ Value iteration in our example: evolution of utility values of states

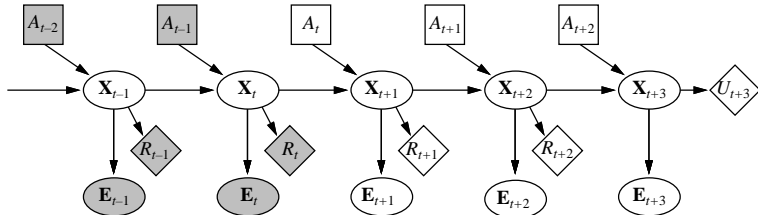


Decision-theoretic agents

- ▶ We now have (tediously) gathered all the ingredients to build decision-theoretic agents
- ▶ Transition and observation models will be described by a DBN
- ▶ They will be augmented by decision and utility nodes to obtain a **dynamic DN**
- ▶ Decisions will be made by projecting forward possible action sequences and choosing the best one
- ▶ Practical design for a **utility-based agent**

Decision-theoretic agents

- ▶ Dynamic decision networks look something like this
- ▶ General form of everything we have talked about in uncertainty part



Summary

- ▶ Sequential decision making
- ▶ Defined MDPs to model stochastic multi-step decision making processes
- ▶ Value iteration and policy iteration algorithms
- ▶ Design of decision-theoretic utility-based agents based on DDNs
- ▶ Completes our account of reasoning under uncertainty