Informatics 2D. Tutorial 10
Decision Networks

Week 11

Decision Network

> *James needs to buy a flat. He has three options: a, b or c. As he plans to resell the house in a few years, he is interested in an increase of the flat's value higher than inflation. Depending on the place he chooses, there is some probability of public works in the area, that will increase the value of the property. He also needs to go to work: therefore he is interested in the distance to the bus stop on the route that brings him to work.*

As described in R&N, decision networks combine belief networks with additional node types for actions and utilities. The network in Figure 1 shows the decision network for the house purchase problem. It contains three types of nodes:

Chance nodes (Ovals) that represent random variables, just as they do in belief nets. In this example, the agent may be uncertain about the increase of the flat's value and about the distance to the bus stop, as they both depend on which flat is chosen. He is also uncertain about the works in the area, as they depend both on the site of the chosen flat and on a decision by the city council.

Decision nodes (Rectangular) that represent points where the decision-maker has a choice of action. In this case the node *Flat* can take the values a, b or c.

Utility nodes (Diamonds) represent the agent's utility function. The utility node has as parents all those variables describing the outcome state that directly affect utility. The table associated with the a utility node is thus a tabulation of the agent's utility as a function of the attributes that determine it.

Actions are selected by evaluating the decision network for each possible setting of the decision node. Once the decision node is set, it behaves exactly like a chance node that has been set as an evidence variable.

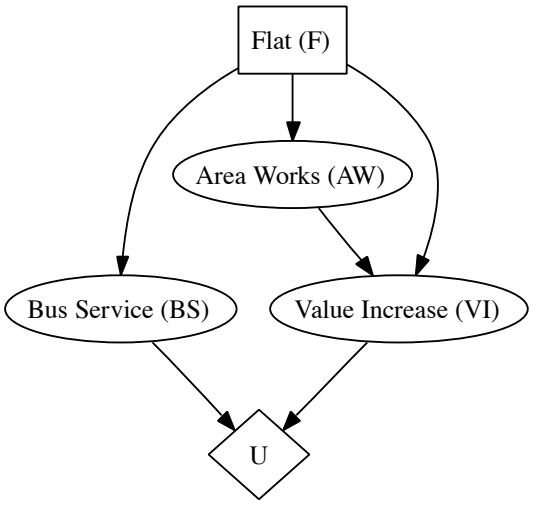

Figure 1: Decision Network

1. Given the conditional probabilities in table 1, compute the utilities for flats a, b and c and then identify the flat with the highest utility.

$F = \langle a, b, c \rangle$

F	$P(AW)$
a	0.4
b	0.2
c	0.4

F	$P(BS)$
a	0.1
b	0.3
c	0.6

F	AW	$P(VI)$
a	T	0.6
a	F	0.1
b	T	0.2
b	F	0.1
c	T	0.7
c	F	0.2

BS	VI	U
T	T	0.9
T	F	0.3
F	T	0.5
F	F	0.1

Table 1: Conditional probabilities and utility function

Complex Decisions

As in R&N, sections 17.1-17.3.

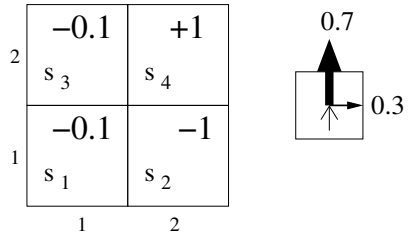

Figure 2: 2×2 environment that presents the agent with a sequential decision problem

Decision networks are used for one-shot, episodic decision problems in which the utility of each action outcome is well known. When the agent's utility depends on a sequence of decisions, another approach is needed. The problem arises when the environment is non deterministic: actions are unreliable.

In the environment in Figure 2 each action achieves the intended effect with probability 0.7, while the rest of the times the agent moves to the right. If the agent bumps into a wall, it stays in the same cell. For example, if the agent is in cell s_1 and wants to go to cell s_3, it will succeeds with a 70% of probability. With 30% of probability, it will end up in cell s_2. If the agent is in cell s_3, and wants to go back to cell s_1, it will succeed with 70% of probability, and it will remain in the same cell with 30% of probability (as he bumps against the wall).

Transition model

A specification of the outcome probabilities for each action in each possible state is called a *transition model*: $T(s, a, s')$ denotes the probability of reaching state s' if action a is done in state s. Transitions are *Markovian*: probability of reaching s' from s depends only on s and not on the history of earlier states.

Reward

In each state s the agent receives a *reward* $R(s)$, which may be positive or negative. For the environment in figure, the reward is -0.1 for cells s_1 and s_3, -1 for s_2 and $+1$ for s_4.

Utility of states

Because the decision problem is sequential, the utility function will depend on a sequence of states rather than on a single state. The utility of an environment history is the (discounted) sum of the rewards received.

The rewards can be *additive*:
$U_h([s_0, s_1, s_2, \ldots]) = R(s_o) + R(s_1) + R(s_2) + \ldots$
or *discounted*:
$U_h([s_0, s_1, s_2, \ldots]) = R(s_o) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$
where the *discount factor* γ is in [0,1]

Markov Decision Process

The specification of a sequential decision problem for a fully observable environment with a Markovian transition model and additive reward is called the Markov Decision Process. The solution of a MDP cannot be a fixed sequence of actions, as the agent might end up in a state that is other than the goal. A solution must specify what the agent should do for any state that the agent might reach. A solution of this kind is called policy.

The quality of a policy π is measured by the *expected utility* of the possible environment histories generated by the policy. An optimal policy π^* is a policy that yields the highest expected utility.

Value iteration

Value iteration is an algorithm for calculating an optimal policy. The basic idea is to calculate the utility of each state and then use the state utilities to select an optimal action in each state.

The utility of states is defined in terms of the utility of states sequences: the utility function $U(s)$ allows the agent to select actions by using the Maximum Expected Utility:

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

If the utility of a state is the expected sum of discounted rewards from that point onwards, there is a direct relationship between the utility of a state and the utility of its neighbours, the utility of a state is the immediate rewards for that state plus the expected discounted utility of the next state, assuming that the agent chooses the optimal action.

The utility, expressed by the Bellman equation, is:

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

where $T(s, a, s')$ is the transition model from s to s', γ is the discount factor for the reward and $U(s')$ is the utility of state s'.

The utilities of states, defined as the expected utility of subsequent state sequences, are solutions of the set of Bellman equations.

Bellman update

The Bellman equation is the basis of the value iteration algorithm for solving MDPs. If there are n possible states, then there are n Bellman equations, one for each state. The n equations contains n unknowns (the utilities of the states).

The equations are non linear, as they contain the *max* operator. One approach to solve non linear equations is to use an iterative method. We start with arbitrary initial values for the utilities, calculate the right-hand side of the equation, and plug it on the left-hand side, updating the utility of each state from the utilities of each neighbours. We repeat until we reach an equilibrium. The Bellman update has the form:

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

Algorithm 1 Value Iteration

```
function VALUE-ITERATION(mdp, ε) returns a utility function
 inputs: mdp, a Markov Decision Process defined by ⟨S, T, R, γ⟩
         ε the maximum error allowed in the utility
 local variables: U, U′ vector of utilities for states in S
                  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]+γmax_a Σ_{s'} T(s, a, s')U[s']
     if |U'[s] − U[s]| > δ then δ ← |U'[s] − U[s]|
until δ < ε(1 − γ)/γ
return U
```

The applet at the url `http://www.cs.ubc.ca/~poole/demos/mdp/vi.html` (under the link for value iteration) allows you to play with the discount factor on a 10x10 grid: setting the factor close to 1, you will need a greater number of iterations for the values to stabilise.

The number of iterations needed to reach an error of at most ϵ is given by:

$$N = \left\lceil \frac{\log\left(\frac{2R_{max}}{\epsilon(1-\gamma)}\right)}{\log(1/\gamma)} \right\rceil$$

2. Discuss the formula and the graph in figure 4: discuss how the value of N changes depending on the different factors.

3. Use the formula to compute the number of iterations N needed to obtain a maximum error ϵ of 0.01, given a R_{max} of 0.45 andγ of 0.1.

4. The first iteration of the algorithm value Iteration in the grid world in Figure 2 using a discount γ of 0.1 is given in Figure 3. Using the results obtained from the first iteration, compute the utility for s_1 in the second iteration.

According to the transition model $T(s, a, s')$, the intended outcome of an action occurs with probability 0.7, while with probability 0.3 the agents slips right.
U is the utility vector. Initially $U_0 = \langle 0, 0, 0, 0 \rangle$
$\delta = 0$

$$U_1(s_1) = R(s_1) + \gamma \max_a \left\{ \begin{array}{cc} 0.7U_0(s_3) + 0.3U_0(s_2), & (up) \\ 0.7U_0(s_2) + 0.3U_0(s_1), & (right) \\ 1.0U_0(s_1), & (down) \\ 0.7U_0(s_1) + 0.3U_0(s_3) & (left) \end{array} \right\}$$

$$= -0.1 + 0.1 \max_a \left\{ \begin{array}{cc} 0 & (up) \\ 0 & (right) \\ 0 & (down) \\ 0 & (left) \end{array} \right\} = -0.1 \text{ (any pick)}$$

$$U_1(s_2) = R(s_2) + \gamma \max_a \left\{ \begin{array}{cc} 0.7U_0(s_4) + 0.3U_0(s_2), & (up) \\ 1.0U_0(s_2), & (right) \\ 0.7U_0(s_3) + 0.3U_0(s_1), & (down) \\ 0.7U_0(s_1) + 0.3U_0(s_4) & (left) \end{array} \right\}$$

$$= -1 + 0.1 \max_a \left\{ \begin{array}{cc} 0 + 0, & (up) \\ 0, & (right) \\ 0 + 0, & (down) \\ 0 + 0 & (left) \end{array} \right\} = -1 \text{ (any pick)}$$

$$U_1(s_3) = R(s_3) + \gamma \max_a \left\{ \begin{array}{cc} 0.7U_0(s_3) + 0.3U_0(s_4), & (up) \\ 0.7U_0(s_4) + 0.3U_0(s_1), & (right) \\ 0.7U_0(s_1) + 0.3U_0(s_3), & (down) \\ 1.0 * U_0(s_3) & (left) \end{array} \right\}$$

$$= -0.1 + 0.1 \max_a \left\{ \begin{array}{cc} 0 + 0, & (up) \\ 0 + 0, & (right) \\ 0 + 0, & (down) \\ 0 & (left) \end{array} \right\} = -0.1 \text{ (any pick)}$$

$$U_1(s_4) = R(s_4) + \gamma \max_a \left\{ \begin{array}{cc} 1.0U_0(s_4), & (up) \\ 0.7U_0(s_4) + 0.3U_0(s_2), & (right) \\ 0.7U_0(s_2) + 0.3U_0(s_3), & (down) \\ 0.7U_0(s_3) + 0.3U_0(s_4) & (left) \end{array} \right\}$$

$$= +1 + 0.1 \max_a \left\{ \begin{array}{cc} 0 & (up) \\ 0 + 0 & (right) \\ 0 + 0 & (down) \\ 0 + 0 & (left) \end{array} \right\} = 1 \text{ (any pick)}$$

Figure 3: Step 1 of Value iteration

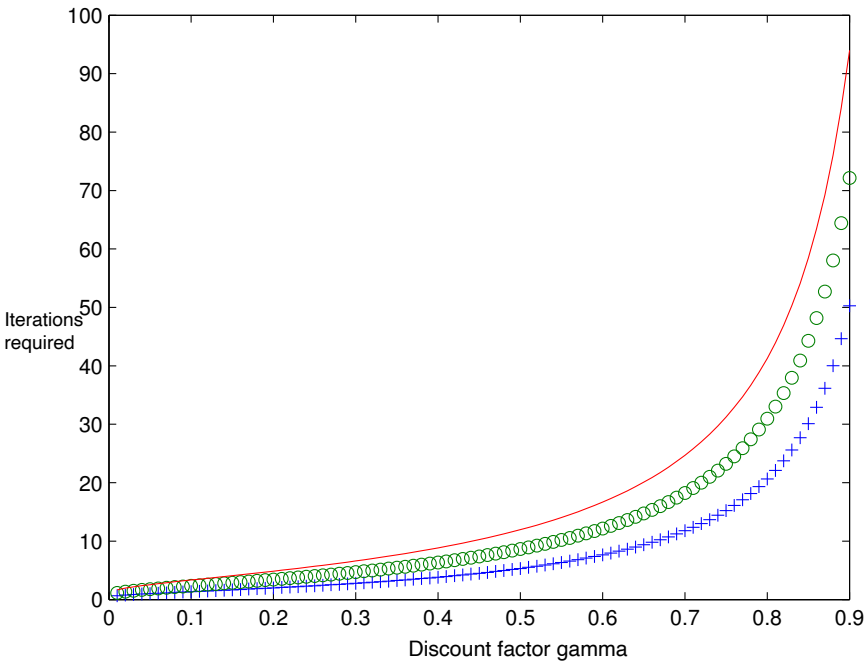

Figure 4: Number of iterations required for convergence for ϵ/R_{max}= 0.1, 0.01, 0.001 depending on γ. (based on Fig 17.5b from R&N)