Study Notes for AI (CS 440/520)
Lectures 16 & 17: Decision Networks - Markov Decision Processes

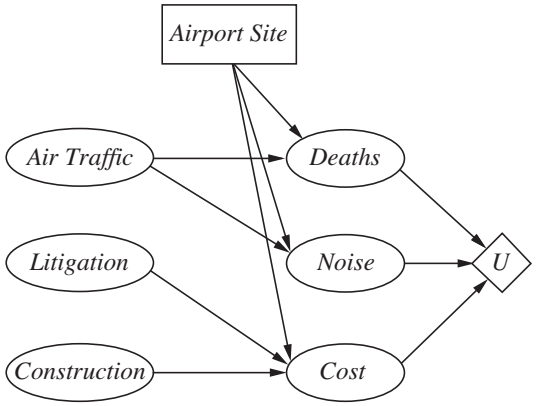
Corresponding Book Chapters: 16.5 - 16.6 - 17.1 - 17.2 - 17.3
Note: These notes provide only a short summary and some highlights of the material covered in the corresponding lecture based on notes collected from students. Make sure you check the corresponding chapters. Please report any mistakes in or any other issues with the notes to the instructor.

1 Decision Networks

Decision networks combine Bayesian networks with additional nodes for types of actions and utilities. In particular, there are:

- **Chance Nodes**(ovals): as in a typical Bayesian network they represent the possible attributes that effect the problem's state.
- **Decision Nodes**(rectangles): They correspond to the different actions that the agent can take to solve a problem.
- **Utility nodes**(diamonds): They are used to show which attributes affect the utility.

Consider the airport figure below. The airport site is a decision node and U is a utility node.



1.1 Evaluating decision networks

The airport site action can take different values (actions of the agent), which influences cost, noise, and death (safety). The actions should be selected by evaluating the network for each setting of the airport site. The algorithm for evaluating decisions networks is the following:

- Set evidence variables for the current state
- For each possible value of the decision node
  - Set decision node to that value
  - Calculate posterior probabilities for the state nodes that are parents to the utility node
  - Compute expected utility for this action
- Return action that maximizes the expected utility

2 Value of Information

A problem in real life is that often not all information is available and we must collect data to build a proper decision network for our problem. But then the following question arises: what kind of data should we collect? And most importantly: how much should we invest in acquiring data? The example below studies the value of information provided to assist an agent in taking a decision.
Consider the following example: An oil company has n indistinguishable blocks of ocean drilling rights.

- Only one of them will contain oil worth $C
- The price of each block is $\$\frac{C}{n}$

A seismologist offers to survey block 3 and will definitely indicate whether the block has oil or not. How much should the company pay the seismologist? In other words, what is the "value of information" that the seismologist offers?

There are two outcomes when using the seismologist:

1. $\exists$ oil in block 3, which has a probability of $\frac{1}{n}$
   Then the best action is to buy block 3
   Because we want profit (C) - cost ($\frac{C}{n}$), the total profit is $C - \frac{C}{n} = (n-1)\frac{C}{n}$
2. Does not $\exists$ oil in block 3, which has a probability of $\frac{n-1}{n}$
   Then the best action is to buy a block other than 3
   Because expected profit is $\frac{C}{n-1}$ and since we choose among n-1 blocks, cost is $\frac{C}{n}$ of buying a block.
   The total profit is therefore $\frac{C}{n-1} - \frac{C}{n} = \frac{C}{n(n-1)}$

The expected utility in the case that we use the seismologist is:

$$EU = \frac{1}{n} * \frac{(n-1)C}{n} + \frac{n-1}{n} * \frac{C}{n(n-1)} = \frac{C}{n}$$

If we do not use the seismologist:

$$EU = \text{ expected profit - cost } = \frac{C}{n} - \frac{C}{n} = 0$$

Consequently, if the company can pay less than $\frac{C}{n}$ to acquire the seismologist's information, then it pays off to hire him.

2.1 Generalization

The above problem, however, generalizes to many problems. Lets say that:
E: the current evidence variables
E_j: potentially new evidence (new information) we can acquire
In order to compute the value of acquiring the new information, we must compute the expected utility in the two cases:

1. If we do not acquire the evidence E_j

$$E(U)(a|E) = \max_A \sum_i U(Result_i(A)) * P(Result_i(A)|D_0(A), E)$$

where A is the action space for the agent.

2. If we do acquire the evidence E_j

$$E(U)(a_{E_j}|E, E_j) = \max_A \sum_i U(Result_i(A)) * P(Result_i(A)|(A), E, E_j)$$

In this case we have to consider the fact that E_j can take many values. So the expected utility of acquiring E_j is the weighted sum of the expected utilities we get for each possible value of E_j (e.g. whether the seismologist finds or does not find oil in block 3). The weights in the sum are the probabilities of E_j acquiring these values. Then we have:

$$E(U)_{E_j} = \sum_k P(E_j = e_{jk}|E) * E(U)(a_{E_{jk}}|E, E_j = e_{jk})$$

Overall, the value of perfect information is given by:

$$VPI_E(E_j) = (\sum_k P(E_j = e_{jk}|E) * E(U)(a_{E_{jk}}|E, E_j = e_{jk})) - EU(a|E)$$

The above equation shows that information has value:

- if it is likely to cause change of plan
- and the new plan is significantly better than the old

3 Introduction to Sequential Decision Making

In previous lectures we have been discussing the basics of utility theory, which is a method to allow intelligent agents to make a decision based on state's utility and the probability of reaching the state. Here, we will focus on sequential decision making problems when there is uncertainty in motion/action. This means that the agent does not make a single decision but has to select a sequence of actions over time to solve a problem.

3.1 Example: 4 × 3 world example

We will begin discussing the concept of sequential decision making in relationship to the 3x4 grid world shown in Figure 1 (left). In a deterministic environment (an environment where you always move where you expect to move) if we start in cell (1,1) the optimal path to reach the goal node (+1) would be {up, up, right, right, right}.

However, in a non-deterministic grid world, where there is a probability of 0.8 moving in the direction you want and 0.1 moving either sideways, the probability of ending up in the goal is lower. The probability of selecting the sequence {up, up, right, right, right} and executing this sequence is actually as low as: 0.8^5. However, we might also reach the goal through a different path when we execute the above sequence. For example, we might actually execute the sequence {right, right, up, up, right}, which has a probability $0.1^4 \cdot 0.8$ of occurring. There are 3 such sequences that can bring the agent to the goal. So overall the probability of reaching the goal with this sequence is: $0.8^5 + 3(0.1)^4 * 0.8$ or 0.32776. This is not a very good chance of ending up in the desired location.

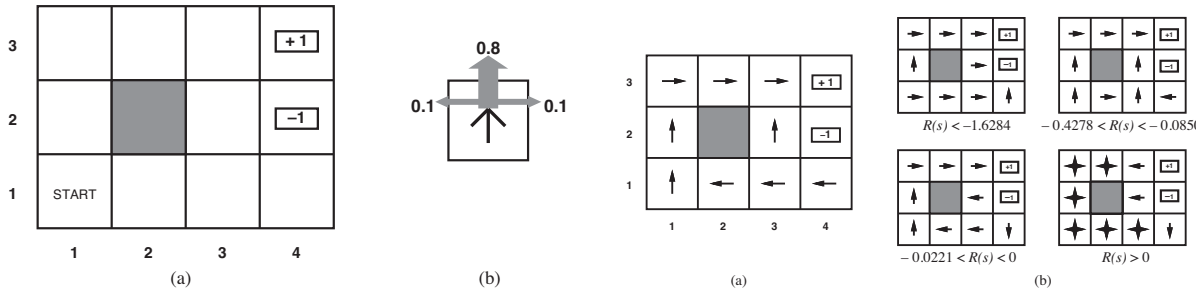


Figure 1: Optimal policies for different rewards.

3.2 Markov Decision Process (MDP)

The above example is an instance of a Markov Decision Process. In order to use an MDP, we must first define three elements:

1. **Transition model:** $T(s, \alpha, s')$
   In the past we have defined the transition model as:
   $$P(x_{t+1}|x_t)$$
   The MDP notation is:
   $$\mathrm{T}(s, \alpha, s')$$
   which is the probability of going from state s to state s' by applying the action α.
2. **Rewards:** $R(s)$
   We typically assign some positive value to the desired goal state (e.g., +1 in the example), and some negative value to a finish node that is undesirable (e.g., -1 in the example). In addition to this, we assign a small negative value to each cell that the agent visits in hopes to entice the agent to find the goal quickly (assume -0.04 for each cell in the example).
3. **Initial state:** s_0
   It is also necessary to know the initial state of our agent.

Note than an MDP does not require the definition of an observation model, since sensing is considered perfect in MDPs.

3.3 Policies

In a deterministic world, it is easy to tell an agent how to reach a goal from its current location by instructing it what steps to has to take. However, in a non-deterministic world assigning it does not make sense to define sequence of states (i.e. paths) as a solution, because while we plan to follow them due to uncertainty in action we might follow a different path. As we saw above, moving just five steps gives a probability of 0.32776 of actually ending up in the location which we wanted to get to.

Instead, we must assign a policy for the agent to use, which will be based on the rewards placed in the system. A policy tells the agent the best action to execute at each state so as to maximize expected utility. Figure 1 (right) shows various optimal policies for different reward values. This means that if we change the rewards of states, the policy can change drastically.

Objective in MDPs: Compute an optimal policy π^*, a policy that yields the highest expected utility.

An optimal policy is one that maximizes the expected utility by taking into account all possible paths resulting from this policy. Once an optimal policy is available, then an agent has just to check its current state and then apply the action according to the optimal policy. There are a few questions to consider when searching for an optimal policy:

- Is there a **horizon** or deadline by which the problem must be solved?
  - If a horizon exists then we are working with a finite horizon problem which adds additional constraints to the policy, constraints than might even cause the agent to be unable to solve the problem. For example, if the shortest possible path to a goal is five steps, as in our example, but the horizon is four steps, the problem would have no solution. Furthermore, for different horizons, we will acquire different solutions.
  - In the case that no horizon exists we say that we have an infinite horizon problem. For infinite horizon problems, we have fewer constraints which makes them easier if they are in a fully-observable environment. However the agent will continue looking for a solution even if no solution exists.
- How to compute **utilities of state sequences**?
  There are two ways to compute the reward of a path.

  1. Simply add up the rewards of the state along the path:

     $$U([s_0, s_1, ...]) = R(s_0) + R(s_1) + ...$$

     The danger with simply adding rewards, however, is that if you get an infinite path, it is difficult to distinguish one path from another as the reward for all of the paths will be infinite.

  2. Use *discounting*:
     select a value γ such that $0 < \gamma < 1$ and then define

     $$\begin{aligned} U([s_0, s_1, ...]) &= R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + ... \\ &= \sum_t^\infty \gamma^t R(S_t) \ \leq \ \sum_t^\infty \gamma^t R_{max} \\ &= \frac{R_{max}}{(1-\gamma)} \end{aligned}$$

     The above computation shows that with discounted rewards the utility is always bounded even if there is not a terminal state or the agent never reaches one.
     Consequently, when we say that we want to find an optimal policy that maximizes expected utility, we define it as follows:

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

  Overall:

- The utility of a path is the discounted reward of states along the path
- The utility of a policy is the expected utility over all paths resulting from this policy.
- The optimal policy maximizes the expected utility over all policies.

There are two algorithms that we will cover and which find an optimal policy:

- Value iteration
- Policy iteration.

3.4 Value iteration

Idea: 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 sequences that might follow it. State sequences depend upon a policy, therefore we must define a random policy in order to begin applying the algorithm.

If we define the utility of a state for a given policy to be $U^{\pi}(s)$ then:

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

where s_t is the state the agent if in after executing policy π for t steps.
Note that we want to compute $U(s) = U^{\pi^*}(s)$ so we need to set

$$\pi^*(s) = argmax_{\alpha} \sum_{s'} \mathrm{T}(s, \alpha, s') * U(s')$$

This leads us to the **Bellman equation** which states:

$$\mathrm{U}(s) = R(s) + \gamma * max_{\alpha} \sum_{s'} \mathrm{T}(s, \alpha, s') * U(s')$$

Given the above equation, how can we compute the optimal policy through value iteration?

We have n equations and n unknowns ($U(s)$ for n states), so we should be able to solve for the optimal solution using linear algebra. However, since we are taking the maximum of the options at each step these equations are not linear. So in order to solve them, we can follow an iterative approach.

Value Iteration Algorithm:

- Make an initial assignment.
- Calculate the right hand side of the Bellman equation.
- Plug the values into the Left hand side: $U_{i+1}(s) = R(s) + \gamma * max \sum_{s'} T(s, \alpha, s') * U_i(s')$
- Repeat until you reach equilibrium.

Properties:

- The algorithm always converges to an optimal solution.
- At each iteration, you always know your error from the actual solution.
- Unfortunately it might take a very long time to reach a solution.

As an example, we will employ the Bellman equation in the (4x3) world example and we compute the utility of the cell(1,1):

$$
\begin{array}{lll}
U(1,1) = -0.04 + \gamma * max\{ & 8.0 * U(1,2) + 0.1 * U(2,1) + 0.1 * U(1,1) & \text{(up)} \\
& 0.9 * U(1,1) + 0.1 * U(1,2) & \text{(left)} \\
& 0.9 * U(1,1) + 0.1 * U(2,1) & \text{(down)} \\
& 0.8 * U(2,1) + 0.1 * U(1,2) + 0.1 * U(1,1) \quad \} & \text{(right)}
\end{array}
$$

3.5 Policy Iteration

Policy iteration is an alternative to the value iteration approach. The inspiration for policy iteration comes from the fact that when the utilities at the states are slightly inaccurate, then the optimal policy tends to be the same.

The basic principle of policy iteration is the following: Given a policy π_i, calculate $U_i = U^{\pi_i}$, the utility of each state if π_i were to be executed. Then given U_i, we can calculate a new policy π_{i+1} that maximizes expected utility using one-step look-ahead.

The important advantage of policy iteration is that once you assume the first policy, computing the utilities for the cells is easier than in value iteration. Instead of the operand max in the equations, we have the operand $\sum$, which implies that the equations are now linear in nature. Thus, we end up with n linear equations and n unknowns. Solving this linear system of equations has an $O(n^3)$ complexity. Policy evaluation is often more efficient in small state spaces, however, for large state spaces $O(n^3)$ may still be prohibitive.

As an example, lets compute the utility of the (1,1) state (bottom left) in the 4x3 world given a policy that dictates that the best action at that state is to move up: $\pi(1,1) = Up$. Then:

$$U_i(1,1) = -0.04 + 0.8 \cdot U_i(1,2) + 0.1 \cdot U_i(1,1) + 0.1 \cdot U_i(2,1)$$

4 What's next?

Next we will be looking at the problems that arise when you not only have uncertainty in your motion, but also uncertainty in observatons and thus, state uncertaint as well. Such problems are called Partially Observable Markov Decision Processes (POMDPs).