Reasoning under uncertainty

Huy L. Nguyễn

In this note, we will study tools for *rational decision making* under uncertainty. Unlike the traditional settings where we are given all of the input at once, there are cases where the algorithm needs to interact with the environment and make decisions based on partial information. We will explore one particular formalism to define uncertainty and rationality.

In our simple setup, uncertainty is modeled using probability: there is a distribution on future events that is known to the decision maker. To define rational choices, we also need to assign to each outcome an utility, which is a number. The decision maker is rational if it maximizes the *expected* utility.

Example 1. A famous example is Pascal's wager. He argues that a rational human should live as though God exists. This is because there are 2 possibilities. If god exists and you live a sinful life, you will suffer infinite loss (eternal damnation, etc). If god does not exist and you live like a believer, it is all for naught. Thus, if our prior is that the probability that god exists is nonzero, you must choose to believe to avoid an infinite expected loss.

We consider another example to illustrate the meaning of utility.

Example 2. You bought a cake and have to eat it within 5 days before it is spoiled. Suppose that eating x fraction of the cake brings $x^{1/3}$ amount of satisfaction. You would like to divide the cake to maximize your total amount of satisfaction.

Example 3. You are eating cake again but since immediate satisfaction is better than future satisfaction, the amount of satisfaction on day i is discounted by a factor γ^i. You again would like to maximize your total amount of satisfaction.

Notice that our model does not capture many real life instances such as lottery where the expected utility is worse than the ticket price. Another problem is that in many cases, the utility is not known to the decision maker. These are just some of the limitation of the model.

1 Markov decision process

The Markov decision process (MDP) framework is a way to model the task of the decision maker in its interaction with the environment. There are *states* that model all necessary knowledge from the past that might affect the future outcome. In each state, there are a number of actions that can be performed by the decision maker. Given the current state and the action, there is a probability distribution that will move the decision maker to a new state. Additionally, the decision maker will also get a reward that is a function of the previous state, the new state, and the action. The name Markov comes from the memoryless property of the model: conditioned the current state, the future outcome is independent of the history.

Formally, let's label the time by integers $1, 2, 3, \ldots$ Suppose that there are N states and A actions. Let s_i be the state at time i. For each state s and action a, there is a probability distribution $p(s'|s, a)$ that determines the next state s'. The transition from state s to state s' using action a also results in the reward $R(a, s, s')$. We also include a discount factor γ^i to reward from day i. Thus, the total

reward is

$$\sum_{i=1}^{\infty} \gamma^i R(a_i, s_i, s_{i+1})$$

Note that for $\gamma < 1$ this sum is bounded.

2 Optimal MDP policies with finite time horizon

First we consider the setting where we are only concerned with the utility up to a finite time T. Let $V_{x,t}$ be the maximum expected utility if we start from state x at time t and finish at time T. For the base case, we have $V_{x,T+1} = 0 \ \forall x$.

For the recursive case, the decision maker should pick the action with the maximum expected utility:

$$V_{x,t} = \max_a \sum_{y=1}^{N} p(s_{t+1} = y | s_t = x, a_t = a)(R(a, x, y) + \gamma V_{y,t+1})$$

In order to compute the best action and the maximum expected utility, we can use dynamic programming. That is, we apply the above recursive relation and store all the intermediate answers so that we do not need to recompute any of them. The running time is $O(N^2AT)$.

3 Optimal MDP policies using LP

In this section, we are concerned with computing the best possible policies if we have an infinite time horizon. There are several ways to define our objective. One way is to use the discount factor $\gamma < 1$ and thus, the sum of the reward is finite. Another way is to use no discount factor i.e. $\gamma = 1$ but we are concerned with the limit of the average reward

$$\lim_{T \to \infty} \frac{1}{T} \sum_{i=1}^{T} R(a_i, s_i, s_{i+1})$$

A special class of policies we are interested in is history-independent policies: we choose the same action every time we enter the same state. Note that the number of history independent policies is A^N, which is very large. Under certain technical conditions, it turns out that there are optimal policies that are history independent.

Thus, our task is reduced to computing the optimal policy that is history independent. Let $\pi : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, A\}$ be the optimal policy. Let V_x be the expected utility of the optimal policy if we start from state x. We have

$$V_x = \sum_{y=1}^{N} p(s' = y | s = x, a = \pi(x))(R(\pi(x), x, y) + \gamma V_y)$$

Since this is the optimal policy, the action $\pi(x)$ must give utility at least as good as any other action. Thus, for any action z, we have

$$V_x \geq \sum_{y=1}^{N} p(s' = y | s = x, a = z)(R(z, x, y) + \gamma V_y) \ \forall z \in \{1, \ldots, A\}$$

We can thus formulate an LP to compute the values V_x. The above inequalities are the constraint. Since we are looking for the optimal policy, the objective is to minimize $\sum_{x=1}^{N} V_x$.

In practice, solving the LP is typically too slow and the preferred methods are iterative algorithms. We will see some examples of iterative algorithms later in the course.