Lecture 7: Decision-making under uncertainty: Part 1

Lecturer: *Sanjeev Arora* Scribe: *Sanjeev Arora*

This lecture is an introduction to *decision theory*, which gives tools for making *rational* choices in face of *uncertainty*. It is useful in all kinds of disciplines from electrical engineering to economics. In computer science, a compelling setting to consider is an autonomous vehicle or robot navigating in a new environment. It may have some prior notions about the environment but inevitably it encounters many different situations and must respond to them. The actions it chooses (drive over the object on the road or drive around it?) *changes* the set of future events it will see, and thus its choice of the immediate action must necessarily take into account the continuing effects of that choice *far into the future*. You can immediately see that the same issues arise in any kind of decision-making in real life: save your money in stocks or bonds; go to grad school or get a job; marry the person you are dating now, or wait a few more years?

Of course, italicized terms in the previous paragraph are all very loaded. What is a rational choice? What is "uncertainty"? In everyday life uncertainty can be interpreted in many ways: risk, ignorance, probability, etc.

Decision theory suggests some answers —perhaps simplistic, but a good start. The theory has the following three elements. The first element is its *probabilistic* interpretation of uncertainty: there is a probability distribution on future events, and furthermore, the decision maker knows this distribution. The second element is how it quantifies what the decision-make *wants*: he/she derives some *utility* from the events that happen. Utility is a number that satisfies some intuitive axioms such as monotonicity and concavity (look it up on wikipedia). The third element of the theory is its definition of a "rational choice." The decision-making is said to be *rational* if it maximises the *expected* utility.

Example 1 Say your utility involves job satisfaction quantified in some way. If you decide to go for a PhD the distribution of your utility is given by random variable X_0. If you decide to take a job instead, your return is a random variable X_1. Decision theory assumes that you (i.e.,the decision-maker) know and understand these two random variables. You choose to get a PhD if $\mathbf{E}[X_0] > \mathbf{E}[X_1]$.

Example 2 17th century mathematician Blaise Pascal's famous *wager* is an early example of an argument recognizable as modern decision theory. He tried to argue that it is the rational choice for humans to believe in God (he meant Christian god, of course). If you choose to be a disbeliever and sin all your life, you may have infinite loss if God exists (eternal damnation). If you choose to believe and live your life in virtue, and God doesn't exist it is all for naught. Therefore if you think that the probability that God exists is nonzero, you must choose to live as a believer to avoid an infinite expected loss. (Aside: how convincing is this argument to you?) □

We will not go into a precise definition of utility (wikipedia moment) but illustrate it with an example. You can think of it as a quantification of "satisfaction ". In computer science we also use *payoff*, *reward* etc.

EXAMPLE 3 (Meaning of utility) You have bought a cake. On any single day if you eat x percent of the cake your utility is $\sqrt{x}$. (This happiness is sublinear because the 5th bite of the cake brings less happiness than the first.) The cake reaches its expiration date in 5 days and if any is still left at that point you might as well finish it (since there is no payoff from throwing away cake).

What schedule of cake eating will maximise your total utility over 5 days? If x_i is the percent of the cake that you eat on day i, then you wish to maximise $\sum_i \sqrt{x_i}$ such that $\sum_i x_i = 1$. Optimizing this using the usual Lagrange multiplier method, you discover that your optimal choice is to eat 20% of the cake each day, since it yields a payoff of $5 \times \sqrt{20}$, which is a lot more than any of the alternatives. For instance, eating it all on day 1 would produce a much lower payoff $\sqrt{5 \times 20}$.

This example is related to Modigliani's *Life cycle hypothesis,* which suggests that consumers consume wealth in a way that evens out consumption over their lifetime. (For instance, it is rational to take a loan early in life to get an education or buy a house, because it lets you enjoy a certain quality of life, and pay for it later in life when your earnings are higher.)

In our class discussion some of you were unconvinced about the axiom about maximising expected utility. (And the existence of lotteries in real life suggests you are on to something.) Others objected that one doesn't truly know —at least very precisely—the distribution of outcomes, as in the PhD vs job example. Very true. (The financial crash of 2008 relates to some of this, but that's a story for another day.) It is important to understand the limitations of this powerful theory.

0.1 Decision-making as dynamic programming

Often you can think of decision-making under uncertainty as playing a game against a random opponent, and the optimum policy can be computed via dynamic programming.

EXAMPLE 4 (Cake eating revisited) Let's now complicate the cake-eating problem. In addition to the expiration date, your decision must contend with actions of your housemates, who tend to eat small amounts of cake when you are not looking. On each day with probability 1/2 they eat 10% of the cake.

Assume that each day the amount you eat as a percentage of the original is a multiple of 10. You have to compute the cake eating schedule that maximises your expected utility.

Now you can draw a tree of depth 5 that describes all possible outcomes. (For instance the first level consists of a 11-way choice between eating $0\%, 10\%, \ldots, 100\%$.) Computing your optimum cake-eating schedule is a simple dynamic programming over this tree. Each leaf has an obvious utility associated with it (derived from the cake you ate while getting to that leaf.) For each intermediate node you compute the best action using the utility calculation from the nodes below. $\square$

The above cake-eating examples can be seen as a metaphor for all kinds of decision-making in life: e.g., how should you spend/save throughout your life to maximize overall happiness[1]?

Decision choice theory says that all such decisions can be made by an appropriate dynamic programming over some tree. Say you think of time as discrete and you have a finite choice of actions at each step: say, two actions labeled 0 and 1. In response the environment responds with a coin toss. (In cake-eating if the coin comes up heads, 10% of the cake disappears.) Then you receive some payoff/utility, which is a real number, and depends upon the sequence of T moves made so far. If this goes on for T steps, we can represent this entire game as a tree of depth T.

Then the best decision at each step involves a simple dynamic programming where the operation at each action node is *max* and the operation at each probabilistic node is *average*. If the node is a leaf it just returns its value. Note that this takes time *exponential* [2] in T. Interestingly, dynamic programming was invented by R. Bellman in this decision-theory context. (If you ever wondered what the "dynamic"in dynamic programming refers to, well now you know. Check out wikipedia for the full story.) The dynamic programming is also related to the game-theoretic notion of *backwards induction*.

The cake example had a finite horizon of 5 days and often such a finite horizon is *imposed* on the problem to make it tractable.

But one can consider a process that goes on for ever and still make it tractable using *discounted* payoffs. The payoff is being accumulated at every step, but the decision-maker discounts the value of payoffs at time t as γ^t where γ is the discount factor. This notion is based upon the observation that most people, given a choice between getting 10 dollars now versus 11 a year from now, will choose the former. This means that they discount payoffs made a year from now by 10/11 at least.

Since $\gamma^t \to 0$ as t gets large, discounting ensures that payoffs obtained a large time from now are perceived as almost zero. Thus it is a "soft "way to impose a finite horizon.

Aside: Children tend to be fairly shortsighted in their decisions, and don't understand the importance of postponement of gratification. Is growing up a process of adjusting your γ to a higher value? There is evidence that people are born with different values of γ, and this is known to correlate with material success later in life. (Look up Stanford marshmallow experiment on wikipedia.)

0.2 Markov Decision Processes (MDPs)

This is the version of decision-making most popular in AI and robotics, and is used in autonomous vehicles, drones etc. (Of course, the difficult "engineering"part is figuring out the correct MDP description.) The literature on this topic is also vast.

The MDP framework is a way to succinctly represent the decision-maker's interaction with the environment. The decision-maker has a finite number of *states* and a finite number of actions it is allowed to take in each state. (For example, a state for an autonomous vehicle could be defined using a finite set of variables: its speed, what lane it is in, whether or not

[1] Several Nobel prizes were awarded for figuring out the implications of this theory for explaining economic behavior, and even phenomena like marriage/divorce.

[2] In fact in a reasonable model where each node of the tree can be computed in time polynomial in the description of the node, Papadimitriou showed that the problem of computing the optimum policy is PSPACE-complete, and hence $\exp(T)$ time is unavoidable.

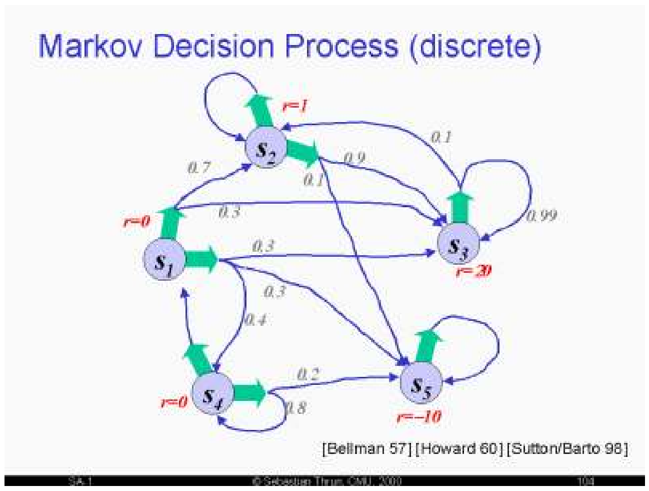
Markov Decision Process (discrete)
s_1
s_2
s_3
s_4
s_5
r=0
r=1
r=20
r=0
r=-10
0.7
0.3
0.3
0.3
0.4
0.1
0.9
0.1
0.99
0.2
0.8
[Bellman 57] [Howard 60] [Sutton/Barto 98]
SA-1
© Sebastian Thrun, CMU, 2000
104

Figure 1: An MDP (from S. Thrun's notes)

there is a vehicle in front/back/left/right, whether or not one of them is getting closer at a fast rate.) Upon taking an action the decision-maker gets a *reward* and then "nature" or "chance" transitions him probabilistically to another state. The optimal policy is defined as one that maximises the total reward (or discounted reward).

For simplicity assume the set of states is labeled by integers $1, \ldots, n$, the possible actions in each state are $0/1$. For each action b there is a probability $p(i, b, j)$ of transitioning to state j if this action is taken in that state. Such a transition brings an immediate reward of $R(i, b, j)$. Note that this process goes forever; the decision-maker keeps taking actions, which affect the sequence of states it passes through and the rewards it gets.
The name Markov: This refers to the *memoryless* aspect of the above setup: the reward and transition probabilities do not depend upon the past history.

EXAMPLE 5 If the decision-maker always takes action 0 and $s_1, s_2, \ldots,$ are the random variables denoting the states it passes through, then its total reward is

$$\sum_{t=1}^{\infty} R(s_t, 0, s_{t+1}).$$

Furthermore, the distribution of s_t is completely determined (as described above) given s_{t-1} (i.e., we don't need to know the earlier sequence of states that were visited).

This sum of rewards is typically going to be infinite, so if we use a discount factor γ then the discounted reward of the above sequence is

$$\sum_{t=1}^{\infty} \gamma^t R(s_t, 0, s_{t+1}).$$

$\square$

0.3 Optimal MDP policies via LP

A *policy* is a strategy for the decision-maker to choose its actions in the MDP. You can think of it as the *driver* of the hardware whose workings are described by the MDP. One idea — based upon the discussion above—is to let the policy be dictated by a dynamic programming that is limited to lookahead T steps ahead. But this is computationally difficult for even moderate T. Ideally we would want a simple precomputed answer.

The problem with precomputed answers is that in general the optimal action in a state at a particular time could depend upon the precise sequence of states traversed in the past. Dynamic programming allows this possibility.

We are interested in *history-independent* policies: each time the decision-maker enters the state it takes the same action. This is computationally trivial to implement in real-time. The above example contained a very simple history-independent policy: always take the action 0. In general such a policy is a mapping $\pi : \{1, \ldots, n\} \to \{0, 1\}$. So there are 2^n possible policies. Are they any good?

For each fixed policy the MDP turns into a simple (but infinite) random walk on states, where the probability of transitioning from i to j is $p(i, \pi(i), j)$. To talk sensibly about an optimum policy one has to make the total reward finite, so we assume a discount factor $\gamma < 1$. Then the expression for reward is

$$\sum_{i=1}^{\infty} \gamma^t (\text{reward at time } t).$$

Clearly this converges. Under some technical condition it can be shown that the optimum policy is history-independent[3]

To compute the rewards from the optimum policy one ignores *transient* effects as the random walk settles down, and look at the final steady state. This computation can be done via linear programming.

Let V_i be the expected reward of following the optimum policy if one starts in state i. In the first step the policy takes action $\pi(i) \in \{0, 1\}$, and transititions to another state j. Then the subpolicy that kicks in after this transition must also be optimal too, though its contribution is attenuated by γ. So V_i must satisfy

$$V_i = \sum_{j=1}^{n} p(i, \pi(i), j)(R(i, \pi(i), j) + \gamma V_j). \tag{1}$$

Thus if the allowed actions are $0, 1$ the optimum policy must satisfy:

$$V_i \geq \sum_{j=1}^{n} p(i, 0, j)(R(i, 0, j) + \gamma V_j),$$

and

$$V_i \geq \sum_{j=1}^{n} p(i, 1, j)(R(i, 1, j) + \gamma V_j).$$

[3]This condition has to do with the *Ergodicity* of the MDP. For each fixing of the policy the MDP turns into a simple random walk on the state space. One needs this to converge to a stationary distribution whereby each state i appears during the walk some p_i fraction of times.

The objective is to maximize V_0 subject to the above constraints, where V_0 denotes the reward for the starting state of the MDP. (Note that each V_i appears on both the left and right hand side of constraints, which constrains it.) So the LP is really solving for

$$V_i = \max_{b \in \{0,1\}} \sum_{j=1}^{n} p(i,b,j)(R(i,b,j) + \gamma V_j).$$

After solving the LP one has to look at which of the above two inequalities involving V_i is tight to figure out whether the optimum action $\pi(i)$ is 0 or 1.

In practice solving via LP is considered too slow (since the number of states could be 100,000 or more) and iterative methods are used instead. We'll see some iterative methods later in the course in other contexts.

Bibliography

1. C. Papadimitriou, *Games against Nature.* JCSS 31, 288-301 (1985)