INSE6300 Jia Yuan Yu

4: Reinforcement Learning

Concordia September 26, 2017

Reinforcement learning is about making repeated decisions based on the latest observations. Whenever you take a decision, you receive some immediate receive some money or incur some cost. The goal is to make lots of money over time.

The key insight is that you have to look at the big picture: you may want to make some decisions that cost you money now, but will make you lots of money later.

Example 0.1 (Supervised Learning vs Reinforcement Learning)**.** Suppose you are the owner in a restaurant and the customer asks you to recommend a dish. You remember this customer coming many times before. If you recommend what you honestly think he or she will like best, then you're doing supervised learning. You remember that you have too many avocados that may go bad soon, should you recommend avocado toasts, guacamole, avocado maki? This is reinforcement learning. The difference is the notion of state, e.g., your inventory.

1 Stochastic inventory management

Figure 1: From `shutterstock.com`

The EOQ model has deterministic demand, the newboy model does not allow inventory carry-over from one time period to the next (perishable inventory). These problem are essentially one-decision problems. In this section, we model stochastic demand for inventory that is not perishable, where solving the problem requires a sequence of decisions. This model of stochastic inventory management is closer in spirit to the beer game.

Consider a single product (e.g., oysters), and discrete time steps (e.g., day 1, 2, etc.). Every time step (e.g., every day), the decision maker oberserves the current

inventory level, and decides how much inventory to order from the supplier. There are costs for holding inventory. The demand is random, but we know the distribution of the random variable. The goal is maximize the expected value of the profit (revenue minus costs) over a number N of days.

Assumptions:

- Delivery is instantaneous (no lead-time);
- The demand take integer values;
- The demand is i.i.d. with given distribution $p_j = \mathbb{P}(D_t = j)$ for $j = 0, 1, \ldots$;
- Inventory has a capacity M.

For time steps $t = 1, 2, \ldots$, let s_t denote the inventory level (this is the *state*), a_t the order size, and D_t the demand at time t—these are all integer-valued. The inventory level from one time step to the next follows this dynamics:

$$s_{t+1} = \max\{s_t + a_t - D_t, 0\}.$$

The reward or profit at time t is

$$r_t(s_t, a_t) = \underbrace{F(s_t + a_t)}_{\text{present value of inventory}} - \underbrace{O(a_t)}_{\text{order}} - \underbrace{h(s_t + a_t)}_{\text{holding}}, \quad \text{for } t = 1, \ldots, N-1,$$
$$r_N(s_N, a_N) = \underbrace{g(s_N, a_N)}_{\text{salvage value}}.$$

where the expected present value of inventory is

$$F(z) = \sum_{j=0}^{z-1} \underbrace{f(j)}_{\text{revenue from } j \text{ sales}} p_j + \sum_{j \geq z} \underbrace{f(z)}_{\text{revenue capped to } z \text{ sales}} p_k, \quad \text{for } z = 0, 1, \ldots$$

The order and holding cost function can be arbitrary; for instance, $O(z) = [K + c(z)]1_{[z>0]}$.

Remark 1. Backorder costs (missed sales) are implicitly accounted for in the profit.

1.1 MDP

We can describe the stochastic inventory management problem as an MDP. The inputs are:

- Holding cost function h, order cost O, sales revenue f, salvage revenue g;
- Probabilities $p_0, p_1, \ldots$;
- Time horizon: $\{1, 2, \ldots, N\}$;

- State space: $S = \{0, 1, \ldots, M\}$;
- Action space: $A = \{0, 1, \ldots, M\}$;
- Expected reward: $r_1, r_2, \ldots, r_N$;
- State transition probabilities:

$$P(s' \mid s, a) = \begin{cases} 0 & \text{if } s' \in (s+a, M], \\ p_{s+a-s'} & \text{if } s' \in (0, s+a] \text{ and } s+a \leq M, \\ \sum_{k>s+a} p_k & \text{if } s' = 0 \text{ and } s+a \leq M. \end{cases}$$

  This is the probability of having an inventory level s' at the next time step when the inventory level at the current time step is s and we order a units of inventory.

The output is a optimal sequence of policies $\sigma_1, \sigma_2, \ldots$, where $\sigma_j : S \to A$. These policies are used to pick the optimal action to take at each time step: suppose that at time $t = 1, 2, \ldots, N$, we observe the state s_t (a random variable), then the optimal action is $\sigma_t(s_t)$.

1.2 Policies

A (pure[1]) policy is a sequence of maps $\{\sigma_t\}$ from histories of observations to actions:

$$\sigma_t : (S \times A)^t \times S \to A.$$

The goal is to improve the maps $\{\sigma_t\}$ as we get more observations $X_1, \ldots, X_t$. This is the goal of *reinforcement learning*.

A pure stationary policy is a function: $\sigma : S \to A$, which corresponds to action processes of the form:

$$A_1, A_2, \ldots = \sigma(X_1), \sigma(X_2), \ldots$$

1.3 Objective or baseline

We define an utility function v that assigns to every policy σ, and initial state x, a value $v(x, \sigma)$. For every initial state x, we also define its value:

$$V(x) = \sup_{\sigma} v(x, \sigma).$$

A policy σ^* is optimal if $v(x, \sigma^*) = V(x)$ for all x.

A number of objectives make sense in MDPs. Consider a fixed initial state $X_0 = x_0$ and a stationary policy σ

- finite-horizon of length T:

$$J_N(\sigma, x_0) = \mathbb{E} \sum_{t=0}^{T-1} r(X_t, \sigma(X_t))$$

[1] A mixed policy is more general and allows mixed or random actions.

- discounted reward with discount factor $\lambda \in (0,1)$:

$$J_\lambda(\sigma, x_0) = \mathbb{E} \sum_{t=0}^{\infty} \lambda^t r(X_t, \sigma(X_t))$$

- time-averaged reward:

$$\bar{J}(\sigma, x_0) = \frac{1}{T} \mathbb{E} \sum_{t=0}^{T-1} \lambda^t r(X_t, \sigma(X_t)).$$

1.4 Solving finite-horizon MDP by backward induction

How do we compute the optimal policies $\sigma_1, \sigma_2, \ldots$? We propose a method of dynamic programming called backward induction.

To illustrate how it works, consider first the game of tic-tac-toe and solving it by backward induction. Take the point of view of one player (e.g., the X-player), consider every possible board configuration with only one last move remaining for the X-player[2], record in a look-up table the outcome. Next, consider every possible board configuration with two moves remaining, find the best next move using the last look-up table, record the outcome corresponding to the best move in a new look-up table. Repeat until we are at the first move for the X-player.

As another example, consider how to solve the *shortest path problem* by backward induction[3].

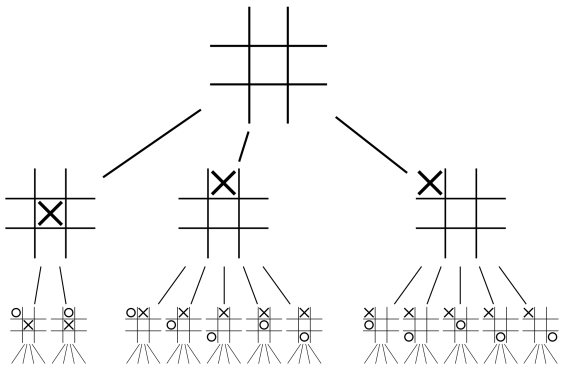

Figure 2: From `https://en.wikipedia.org/wiki/Tic-tac-toe`

The backward induction algorith for MDPs proceeds as follows.

1. Set $j = N$, and $V_N(s) = \max_{a \in A} r_N(s,a) = g(s)$ for all $s \in S$;

[2]These are all board configurations where neither player has three squares in a row, and each player has made four moves.

[3]Cf. Chapter 11 of *Applied Mathematical Programming* by Bradley, Hax, and Magnanti (Addison-Wesley, 1977), `http://web.mit.edu/15.053/www/AMP-Chapter-11.pdf`.

2. For $j = N-1, N-2, \ldots, 1$:

   (a) For $s \in S$:

      i. Compute

$$V_j(s) = \max_{a \in A} \left\{ r_j(s,a) + \sum_{s' \in S} P(s' \mid s,a) V_{j+1}(s) \right\};$$

      ii. Output $\sigma_j(s) \in \arg\max_{a \in A} \left\{ r_j(s,a) + \sum_{s' \in S} P(s' \mid s,a) V_{j+1}(s) \right\}$.

The output policies $\sigma_1, \ldots, \sigma_N$ are optimal (cf. Puterman, Section 4.3).

2 Model-free Reinforcement Learning

In the inventory management and shortest-path routing problems, we assumed that we know how our state will evolve over time. This is however not the case in many other situations. What if the demand is not independent and identically distributed, but depends on the inventory level (e.g., artificially limited supply)?

Example 2.1 (Model-based vs model-free)**.** Suppose that the demand is described by a sequence of i.i.d. random variables (each potential customer flips a coin, each inventory item has a chance to go bad). If those probabilities are known, then the transition probabilities can be easily calculated. This is model-based. If the probabilities are unknown, we don't know the transition probabilities between states, then it's model-free.

In many case, we do not know the probabilities $\{P(s' \mid s,a)\}$. Define the following sequence of vectors $\{Q_t \in \mathbb{R}^{S \times A} : t = 1, 2 \ldots\}$. Recall that you observe X_t and take action A_t at time t, and observe the next state X_{t+1}.

$$Q_{t+1}(s,a) = Q_t(s,a) + 1_{[(X_t,A_t)=(s,a)]} \alpha_t \left[r(s,a) + \lambda \max_{a' \in A} Q_t(X_{t+1}, a') - Q_t(s,a) \right], \quad (s,a) \in S \times A.$$

Observed that we have replace the expectation of next-step value by its simulated quantity $\max_{a' \in A} Q_t(X_{t+1}, a')$. This is called Q-learning, and is an example of (asynchronous) stochastic approximation.

2.1 Stochastic approximation

We now present an important result to solve an equation with an unknown function. First, we introduce the stochastic approximation algorithm in $\mathbb{R}^d$:

$$X_{n+1} = X_n + \alpha_n (h(X_n) + M_{n+1}), \quad n = 0, 1, \ldots$$

Theorem 2.1. *1. Suppose that $h : \mathbb{R}^d \to \mathbb{R}^d$ is L-Lipschitz.*

2. *Suppose that* $\sum_{n\geq 0}\alpha_n = \infty$ *and* $\sum_{n\geq 0}\alpha_n^2 < \infty$.

3. *Suppose that* $\{M_n\}$ *is a bounded martingale difference sequence:* $\mathbb{E}[M_{n+1} \mid X_0, M_1, M_2, \ldots, M_n] = 0$ *almost surely.*

4. *Suppose that* X_n *is bounded almost surely.*

Under these four assumptions, X_n *converges to a solution of the differential equation* $z'(t) = h(z(t))$.

Observe that in the case of Q-learning, we want to find a vector Q such that$Q = F(Q)$, where F is defined by:

$$Q(s,a) = r(s,a) + \lambda \max_{a'\in A} \sum_{s'\in S} P(s' \mid s,a)Q(s',a'), \quad s,a \in S \times A.$$

Consequently, in the Q_t iterations, we set $h(z) = F(z) - z$.

3 Markov chains

Markov chains are particular random processes (sequences of random variables). In this class, we consider only finite Markov chains, in the sense that we consider sequences

$$X_0, X_1, \ldots$$

of random variables that take values in a finite set S. For example, $S = \{1, \ldots, d\}$. These sequences have the following property: there exists a matrix P such that X_{t+1} is distributed according to a fixed probability distribution $P(\cdot, X_t)$. More precisely, a homogeneous Markov chain has the property: for every $x, y \in S$, every $t = 1, 2, \ldots$, and every deterministic sequence $x_0, \ldots, x_{t-1}$,

$$\begin{aligned}&\mathbb{P}(X_{t+1} = y \mid \{X_t = x\} \cap \{X_{t-1} = x_{t-1}\} \ldots \cap \{X_0 = x_0\}) \\ &= \mathbb{P}(X_{t+1} = y \mid \{X_t = x\})\end{aligned}$$

whenever the conditional probability is defined. We define the transition matrix P of this Markov chain as the matrix with elements:

$$P(y,x) = \mathbb{P}(X_{t+1} = y \mid \{X_t = x\}).$$

In other words, the matrix P describes every transition.

We denote the distribution (probability density vector) of X_t by $\mu_t \in \mathbb{R}^d$, *i.e.*,

$$\mu_t(x) = \mathbb{P}(X_t = x), \text{for all } x \in S.$$

Observe that, by conditioning, we have

$$\mu_{t+1}(y) = \sum_{x\in S} P(y,x)\mu_t(x), \text{for all } x \in S.$$

Hence, in vector form:

$$\mu_{t+1} = P\mu_t, \text{for all } t = 0, 1, \ldots$$

and

$$\mu_t = P^t\mu_0, \text{for all } t = 0, 1, \ldots$$

The most important results on Markov chains concern the convergence of the sequence $\{\mu_t\}$. To describe convergence, we first need to introduce two additional properties.

Definition 3.1 (Irreducibility)**.** A Markov chain is irreducible if for every two states $x, y \in S$, there exists an integer t such that $P^t(x, y) > 0$. In other words, it is possible to go from any state to any other state in finite time with positive probability.

Definition 3.2 (Periodicity)**.** The period of a state x is the greatest common divisor (gcd) of the values of n for which $P^n(x, x) > 0$. If the period is 1, the state is called aperiodic.

A Markov chain that is irreducible and and whose states are all aperiodic is said to be *ergodic*.

Markov chains are useful in many places: Monte Carlo simulation, random walks, finance, computer graphics, and card tricks!

3.1 Optimal policies for infinite horizon

Suppose that we know that the policy $\sigma^* = \{\sigma_t^*\}$ is optimal (in one of the above senses). Then, at time t, we can assign a value $V_t^*(s)$ to each state $s \in S$ that reflect the expected future reward that can be accumulated by starting at that state s and choosing actions according the optimal policy σ^*. Consequently, the optimal action at time t can be written as

$$A_t = \sigma_t^*(X_t) \in \arg\max_{a \in A} \left[r(X_t, a) + \sum_{s' \in S} P(s' \mid X_t, a) V_t^*(s') \right],$$

where X_t is the observed state at time t and the sum represents the expected future reward. Next, we consider variants of this approach for the finite-time-horizon setting and the infinite-horizon discounted-reward setting. In the first case, V_t^* can be computed exactly; in the second case, we estimate it by observing that it must be equal for every time step t.

For the finite-horizon problem, backward recursion is a reasonable approach. De-

fine:

$$V_T(s) = 0, \quad \text{for all } s \in S,$$
$$V_{T-1}(s) = \max_{a \in A} \left[r(s,a) + \sum_{s' \in S} P(s' \mid s,a) V_T(s') \right],$$
$$\dots$$
$$V_t(s) = \max_{a \in A} \left[r(s,a) + \sum_{s' \in S} P(s' \mid s,a) V_{t+1}(s') \right], \quad \text{for } t < T.$$

At each time step t, if we pick an action A_t that achieves the above maximization, we are optimal. This holds for every x_0. This is the idea of *dynamic programming*.

For the discounted objective, we solve for a vector V^* the following system of equations:

$$V^*(s) = \max_{a \in A} \left[r(s,a) + \lambda \sum_{s' \in S} P(s' \mid s,a) V^*(s') \right], \quad s \in S. \tag{1}$$

This is the Bellman equation. An optimal policy is to pick an action A_t that achieves the above maximization at every time step.

3.1.1 Value iteration

How do we find a vector V that solves (1)? Consider the following sequence of iterates $V_1, V_2, \dots$:

$$V_0 = 0,$$
$$V_{t+1}(s) = \max_{a \in A} \left[r(s,a) + \lambda \sum_{s' \in S} P(s' \mid s,a) V_t(s') \right], \quad s \in S.$$

This sequence converges if each iteration is a contraction, and the limit is a solution V to (1). This is called value iteration. Other well-known methods exist to compute V: policy iteration, linear programming.

3.2 What are MDPs good for?

Applications of MDPs:

- communication networks, routing,
- operations research (logistics, inventory management, queuing, scheduling, Gittins allocation indices),
- investment.

To learn more about MDPs, check out Puterman's book "Markov Decision Processes."

4 Reading material

- Chapters 1, 2, and 3 of Markov Decision Processes (Puterman).
- Chapter 11 of *Applied Mathematical Programming* by Bradley, Hax, and Magnanti (Addison-Wesley, 1977), `http://web.mit.edu/15.053/www/AMP-Chapter-11.pdf`.