CA684 Jia Yuan Yu

7: Markov decision problems

DCU April 27, 2014

So far, we have seen learning problems where the data is modeled as i.i.d. random variables or arbitrary deterministic sequences. However, many important problems have data that don't fall in these models. Another important model of data is the notion of Markov chains. This model is a good reflection of problems of sequential decision-making under uncertainty.

1 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

$$X_0, X_1, \dots$$

of random variables that take values in a finite set S. For example, $S = \{1, \dots, d\}$. 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 $t = 1, 2, \dots$, and every deterministic sequence $x_0, \dots, x_{t-1}$,

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

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:

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

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 1.1 (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 1.2 (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!

2 MDPs

Markov decision problems or Markov decision processes can be described by a set of objects:

- A state space S, assumed finite in this lecture;
- An action space A, assumed finite;
- A set of discrete time instants $1, 2, \ldots$;
- A reward function $r : S \times A \to \mathbb{R}$;
- A set of transition probabilities $\{P(j \mid i, a)\}$.

In this setting, we are interested in the following random processes. First, the state and action processes:

$$X_1, X_2, \ldots,$$
$$A_1, A_2, \ldots,$$

where at every time step t and for all i, j, a, if X_t takes value i and the decision-maker takes action $A_t = a$, then the next state X_{t+1} takes value j with probability $P(j \mid i, a)$. Next, the induced reward process:

$$r(X_1, A_1), r(X_2, A_2), \ldots$$

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 idea of *reinforcement learning*.

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

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

Definition 2.1 (Unichain)**.** For every policy $\sigma : S \to A$, the resulting Markov chain has a single ergodic class[2] of states.

In this course, we consider only MDPs satisfying the unichain condition.

2.1 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:

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

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

[2] Two distinct states x and y communicate if x is accessible from y (in finite time with positive probability) and y is accessible from x. A class C of states is a non-empty set of states such that each $x \in C$ communicates with every other state $y \in C$ and communicates with no $y \notin C$.

2.2 How do we find optimal policies?

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. Define:

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

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.

2.2.1 Value iteration

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

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

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.

2.2.2 What if we don't transition probabilities?

In many case, we do not know the probabilities $\{P(s' \mid s, a)\}$. Consider a sequence of iterates $\{Q_t : S \times A \to \mathbb{R}\}$ defined as follows:

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

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.

2.3 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 $h(z) = F(z) - z$.

2.4 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."

3 I want to learn more please!

Good news: There's a lot more to learn about machine learning! Don't stop at this course:

- Multi-agent systems,
- Learning in games,
- MDPs against a rational opponent: stochastic games,
- Change-point detection.