Introduction to Reinforcement Learning

Jim Dai

iDDA, CUHK-Shenzhen

January 21, 2019

Sequential decision problems

- Let $N > 0$ be the time horizon of the decision problem.
- For each $k \in [0, N+1]$, $x_k \in \mathcal{X}_k$ is the state at time k.
- At time k, observing state x_k, an action $a_k \in \mathcal{A}_k$ is taken.
- Given (x_k, a_k), a new (random) state x_{k+1} is observed and a (one-step) cost $g_k(x_k, a_k, x_{k+1})$ is incurred.
- The sequence

$$(x_0, a_0, x_1, a_1, \ldots, x_N, a_N, x_{N+1})$$

is known as an episode.

Policies and total costs

- The total cost for the episode is

$$\sum_{k=0}^{N} g_k(x_k, a_k, x_{k+1}).$$

- $\pi = \{\mu_0, \mu_1, \ldots, \mu_N\}$ is known as a policy if for $k \in [0, N]$
  - $\mu_k : \mathcal{X}_k \to \mathcal{A}_k$,
  - $a_k = \mu_k(x_k)$.
- Expected total cost $J_\pi(x) = \mathbb{E}_\pi\Big[\sum_{k=0}^{N} g_k(x_k, a_k, x_{k+1})|x_0 = x\Big]$, where $\mathbb{E}_\pi$ is the expectation over randomness for transitions from x_k to x_{k+1}, $k \in [0, N]$, under policy π.

Models for dynamics

- The system state at the next decision epoch is determined by

$$\mathbb{P}\Big\{x_{k+1} = y \,\Big|\, x_k = x, a_k = a\Big\} = P_k(x, a, y)$$

  for each $x \in \mathcal{X}_k$, $a_k \in \mathcal{A}_k$, and $y \in \mathcal{X}_{k+1}$.

- Case I: transition probabilities are known. The model is known.
- Case II: transition probabilities are unknown, but episodes can be observed from data.
- Case III: transition probabilities are unknown, but given (x_k, a_k), x_{k+1} can be sampled. A simulator is available.

Objective and optimal value function

- Π is the set of feasible policies. The optimal value function is

$$J^*(x) = \inf_{\pi \in \Pi} J_\pi(x), \quad x \in \mathcal{X}_0. \tag{1}$$

  A policy π^* is an optimal policy if $J^*(x) = J_{\pi^*}(x)$.

- In general, the infimum in (1) may not be achievable. In such a case, an optimal policy does not exist.

Bellman equation and backward induction algorithm

- Bellman equation when N is finite.

$$J_{N+1}(x) = 0, x \in \mathcal{X}_{N+1}, \text{ and for } k = N, \ldots, 0,$$

$$J_k(x) = \min_{a \in \mathcal{A}_k(x)} \sum_{y \in \mathcal{X}_{k+1}} P_k(x,a,y)\Big(g_k(x,a,y) + J_{k+1}(y)\Big) \quad \text{for } x \in \mathcal{X}_k,$$

(cost-to-go function J_k)

$$J^*(x) = J_0(x), \quad x \in \mathcal{X}_0. \qquad \text{complexity: } \Pi_{k=0}^{N} |\mathcal{A}_k||\mathcal{X}_{k+1}|.$$

- Bellman equation when N is infinite, assuming time homogeneity with a discounted factor $\beta < 1$, ($P_k = P$ and $g_k = \beta^k g$)

$$J^*(x) = \min_{a \in \mathcal{A}} \sum_{y \in \mathcal{X}} P(x,a,y)\Big(g(x,a,y) + \beta J^*(y)\Big)$$

Bellman's equation: optimal value function

Theorem (Bellman optimality equation; Bertsekas, Proposition 1.2.3)

Assume that the state space $\mathcal{X}$ and action space $\mathcal{A}$ are finite.

(a) The optimal value function J^* satisfies

$$J^*(x) = \min_{a \in \mathcal{A}(x)} \mathbb{E}_{x' \sim P(\cdot|x,a)} \Big[g(x,a,x') + \beta J^*(x') \Big] \quad \textit{for all } x \in \mathcal{X}$$

(b) J^* is the unique solution of the Bellman's equation.

Notation: $\mathbb{P}\{x' = y | x, a\} = P(x,a,y)$ for $y \in \mathcal{X}$.

$$\sum_{y \in \mathcal{X}} P(x,a,y) \Big(g(x,a,y) + \beta J^*(y) \Big) = \mathbb{E}_{x' \sim P(\cdot|x,a)} \Big[g(x,a,x') + \beta J^*(x') \Big]$$

Bellman's equation: optimal policy

Theorem (Bertsekas, Proposition 1.2.5)

Assume that the state space $\mathcal{X}$ and action space $\mathcal{A}$ are finite.

Let $J^* : \mathcal{X} \to \mathbb{R}$ be the unique solution to the Bellman equation.

Define $\mu^* : \mathcal{X} \to \mathcal{A}$ via

$$\mu^*(x) = \arg\min_{a \in \mathcal{A}(x)} \mathbb{E}_{x' \sim P(\cdot|x,a)} \Big[g(x,a,x') + \beta J^*(x') \Big] \quad \textit{for all } x \in \mathcal{X}.$$

The stationary policy $\pi^* = \{\mu^*, \ldots, \mu^*, \ldots\}$ is optimal.

For a function $h : \mathcal{X} \to \mathbb{R}$, define h-greedy policy $\mu_h : \mathcal{X} \to \mathcal{A}$ via

$$\mu_h(x) = \arg\min_{a \in \mathcal{A}(x)} \mathbb{E}_{x' \sim P(\cdot|x,a)} \Big[g(x,a,x') + \beta h(x') \Big] \quad \text{for all } x \in \mathcal{X}.$$

Bellman operator

- Define Bellman operator T: for $J : \mathcal{X} \to \mathbb{R}$

$$(TJ)(x) = \min_{a \in \mathcal{A}(x)} \mathbb{E}_{x' \sim P(\cdot|x,a)} \left[g(x, a, x') + \beta J(x') \right]$$

- Fix a stationary policy μ. Define its Bellman operator T_μ: for $J : \mathcal{X} \to \mathbb{R}$

$$(T_\mu J)(x) = \mathbb{E}_{x' \sim P(\cdot|x,\mu(x))} \left[g(x, \mu(x), x') + \beta J(x') \right]$$

- For any J,

$$(T_{\mu_J} J)(x) = (TJ)(x) \quad x \in \mathcal{X}.$$

Value iteration

Theorem (Bertsekas, Proposition 1.2.1)

For any function $J : \mathcal{X} \to \mathbb{R}$, we have for all $x \in \mathcal{X}$,

$$J^*(x) = \lim_{N \to \infty} (T^N J)(x).$$

- Value iteration: $J^n = TJ^{n-1}$, starting with arbitrary $J^0 = J$.
- Convergence rate:

$$\max_{x \in \mathcal{X}} |J^N(x) - J^*(x)| = \max_{x \in \mathcal{X}} |TJ^{N-1}(x) - TJ^*(x)| \leq$$

$$\beta \max_{x \in \mathcal{X}} |J^{N-1}(x) - J^*(x)| \leq \beta^N \max_{x \in \mathcal{X}} |J(x) - J^*(x)|.$$

Value iteration: Complexity

- One iteration step, for one state $x \in \mathcal{X}$:

  Let $g(x,a) = \sum_{y \in \mathcal{X}} P(x,a,y) g(x,a,y)$ be expected cost, then

  $$J^n(x) = (TJ^{n-1})(x) = \min_{a \in A(x)} \left[g(x,a) + \beta \sum_{y \in \mathcal{X}} P(x,a,y) J^{n-1}(y) \right]$$

  The complexity is $|\mathcal{A}||\mathcal{X}|$.

- Complexity of value iteration algorithm for N steps:

  $$N|\mathcal{A}||\mathcal{X}|^2.$$

Policy evaluation

- Given a stationary policy μ, its value function J_μ satisfies Bellman equation

$$J_\mu(x) = g(x, \mu(x)) + \beta \sum_{y \in \mathcal{X}} P(x, \mu(x), y) J_\mu(y) \quad x \in \mathcal{X}. \tag{2}$$

- Thus

$$J_\mu = (I - \beta P_\mu)^{-1} g_\mu,$$

where g is an $\mathcal{X}$-vector with entries $g_\mu(x) = g(x, \mu(x))$ and P_μ is an $\mathcal{X} \times \mathcal{X}$ matrix with entries $P_\mu(x, y) = P(x, \mu(x), y)$.

- There are many algorithms solving (2).

Policy iteration

- Step 1: (Initialization) Guess an initial stationary policy μ^0.
- Step 2: (Policy evaluation: Find J_{μ^k})

  Solve $J_{\mu^k} = T_{\mu^k} J_{\mu^k}$ or the linear system of equations w.r.t. J:

  $$(I - \beta P_{\mu^k})J = g_{\mu^k}$$

- Step 3: (Policy improvement: Find J_{μ^k}-greedy policy)

  Obtain a new stationary policy μ^{k+1} that is J_{μ^k}-greedy.

  If $J_{\mu^k} = J_{\mu^{k+1}}$ stop; else return to step 2.

Policy iteration

μ^k		J_{μ^k}		μ^{k+1}
old policy	$\longrightarrow$	value function of the old policy	$\longrightarrow$	new improved policy

Theorem (Bertsekas, Proposition 2.3.1)

Fix a policy μ. Let $\hat{\mu}$ be a J_μ-greedy policy. Then we have

$$J_{\hat{\mu}}(x) \le J_\mu(x), \quad \text{for each } x \in \mathcal{X}.$$

Moreover, if μ is not optimal, strict inequality holds for at least one state.

Reinforcement learning

- Dynamic Programming:

  - model of the environment's dynamics is given (P, g are known).

- Reinforcement learning:

  - model of the environment's dynamics is not given (P, g are unknown).

Policy evaluation

- Given a stationary policy μ we want to estimate

$$J_\mu(x) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \beta^k g(x_k, \mu(x_k), x_{k+1}) | x_0 = x\Big]$$

- J_μ has to satisfy Bellman equation:

$$\begin{aligned} J_\mu(x) &= \mathbb{E}[g(x, \mu(x), x_1)] + \beta \mathbb{E}[J_\mu(x_1)] \\ &= g_\mu(x) + \beta(P_\mu J_\mu)(x) = (T_\mu J_\mu)(x) \end{aligned}$$

- Solving fixed point J_μ from $J_\mu = T_\mu J_\mu$ is equivalent solving

$$J_\mu = \arg\min_{J \in \mathbb{R}^{|\mathcal{X}|}} \sum_{x \in \mathcal{X}} |J(x) - (T_\mu J)(x)|^2 \xi(x).$$

Bellman error

- Lost function:

$$||J - T_\mu J||_\xi \equiv \sum_{x \in \mathcal{X}} \left| J(x) - \big(g_\mu(x) + \beta (P_\mu J)(x)\big) \right|^2 \xi(x). \tag{3}$$

  is known as the (weighted) Bellman error.

- Most visited states should be weighted more. So ξ is often chosen to be the stationary distribution of the DTMC with transition matrix P_μ.

Policy evaluation

- By setting the gradient of (3) (w.r.t J) to 0, we get

$$D(J - (g_\mu + \beta P_\mu J)) = 0, \tag{4}$$

where D is the diagonal matrix with ξ along the diagonal.

- Note that equation (4) is equivalent to the fixed point problem

$$J = J - \gamma D\Big[(I - \beta P_\mu)J - g_\mu\Big]$$

- $D\Big[(I - \beta P_\mu)J - g_\mu\Big] = \mathbb{E}_\xi\Big[J(x) - \beta J(x') - g(x, x')\Big]$

Tabular TD(0) learning

- Step 1 (Initialization): arbitrary initialize J^0, choose initial state x_0
- Step 2 (Simulation): simulate one step starting from x_k with decision given by μ.

  Observe next state x_{k+1} and one-step cost g_k.
- Step 3 (Update):

$$\begin{cases} J^{k+1}(x_k) = J^k(x_k) - \gamma_k \Big[J^k(x_k) - \big(g_k + \beta J^k(x_{k+1}) \big) \Big], \\ J^{k+1}(y) = J^k(y), \quad \text{for } y \neq x_k \end{cases}$$

- Step 4: $k = k+1$, move to Step 2.

Tabular TD(0) learning

Theorem (Sutton, 1988)

Assume a Markov chain associate with the policy μ is finite, irreducible and aperiodic. Given bounded costs $|g(x,a,x')| < G$ and learning rate s.t. $\sum_{k=0}^{\infty} \gamma_k = \infty$, $\sum_{k=0}^{\infty} \gamma_k^2 < \infty$

$$\lim_{k\to\infty} J^k = J_\mu \quad \textit{a.s.}$$

The ℓ-step Bellman equation:

$$J_\mu(x) = \mathbb{E}\Big[\sum_{k=0}^{\ell} \beta^k g(x_k, \mu(x), x_{k+1}) + \beta^{\ell+1} J_\mu(x_{\ell+1})\Big].$$

When $\ell \sim$ Geometric(λ), $0 \le \lambda \le 1$, the corresponding algorithm is TD(λ) learning algorithm.

Q-factor

- Optimal Q-factor is defined as

$$\begin{aligned} Q^*(x,a) &= \sum_{x' \in \mathcal{X}} P(x,a,x') \Big[g(x,a,x') + \beta J^*(x') \Big] \\ &= \mathbb{E}[g(x,a,x') + \beta J^*(x')] \\ &\approx \frac{1}{K} \sum_{i=1}^{K} \Big(g(x,a,x'_i) + \beta J^*(x'_i) \Big) \end{aligned}$$

- Optimal policy inference

$$\mu^*(x) = \arg\min_{a \in \mathcal{A}(x)} Q^*(x,a). \tag{5}$$

- When Q^* is too big for memory or (5) is too difficult, a low-dimensional representation of Q^* is needed.

Bellman equation for Q-factor

- Define

$$(TQ)(x,a) = \sum_{y \in \mathcal{X}} P(x,a,y)\left[g(x,a,y) + \beta \min_{v \in A(y)} Q(y,v)\right]$$

- Q^* is the unique fixed point to equation

$$Q = TQ.$$

- If P and g are known, value iteration (VI)

$$Q_{k+1} = TQ_k$$

converges to Q^* from any starting Q_0.

Q-learning as stochastic VI

- We can generate infinitely long sequence of triples $\{x_k, a_k, g_k\}$, s.t. each state-action pair (x, a) appears infinitely often.
- the Q-factor of (x_k, a_k) pair is updated:

$$\begin{cases} Q_{k+1}(x_k, a_k) = (1-\gamma_k)Q_k(x_k, a_k) + \gamma_k\Big(g_k + \beta \min_v Q_k(x'_{k+1}, v)\Big) \\ Q_{k+1}(x, a) = Q_k(x, a), \quad \text{if } (x, a) \neq (x_k, a_k) \end{cases}$$

- Note that $g_k + \beta \min_v Q_k(x'_{k+1}, v)$ is a single sample approximation of the expected value $(TQ)(x_k, a_k)$.

Q-learning

Theorem (Watkins and Dayan, 1992)

Given

- *A sequence where each state-action pair appears infinitely often*
- *bounded costs* $|g(x, a, y)| < G$
- *learning rate s.t.* $0 < \gamma_k < 1$, $\sum_{k=0}^{\infty} \gamma_k = \infty$, $\sum_{k=0}^{\infty} \gamma_k^2 < \infty$

Q-learning algorithm converges:

$$\lim_{k \to \infty} Q_k(x, a) = Q^* \quad \text{a.s.}$$

Policy iteration for Q-factors

- Policy evaluation: given current policy μ^k find the fixed point Q_{μ^k} of

$$Q(x,a) = \sum_{y \in \mathcal{X}} P(x,a,y)[g(x,a,y) + \beta Q(y, \mu^k(y))]$$

- Policy improvement: $\mu^{k+1}(x) = \arg\min_{a \in \mathcal{A}(x)} Q_{\mu^k}(x,a)$

optimistic PI for Q-factors: SARSA

- Step 1 (Initialization): arbitrary initialize Q^0, choose initial state x_0, initial decision a_0.
- Step 2 (Simulation): simulate one step starting from x_k with decision given by a_k. Observe next state x_{k+1} and cost g_k.
- Step 3 (Evaluation&improvement) $a_{k+1} = \begin{cases} \arg\min_{a \in \mathcal{A}} Q^k(x_{k+1}, a) & \text{w.p. } 1-\epsilon \\ \text{other action} & \text{w.p. } \epsilon \end{cases}$
- Step 4 (Update):

$$\begin{cases} Q^{k+1}(x_k, a_k) = (1-\gamma_k)Q^k(x_k, a_k) + \gamma_k \left[g_k + \beta Q^k(x_{k+1}, a_{k+1}) \right], \\ Q^{k+1}(y, v) = Q^k(y, v), \quad \text{for } (y, v) \neq (x_k, a_k) \end{cases}$$

- Step 5: $k = k+1$, move to Step 2. SARSA: state, action, reward, state,

References

- Sutton, R. S. (1988). Learning to predict by the method of temporal differences. Machine Learning, 3:9–44.
- Watkins, C. J. C. H., Dayan, P. (1992). Q-learning. Machine Learning, 8:279–292.
- Bertsekas, D. P. (2012). Dynamic Programming and Optimal Control, Volume 2: Approximate Dynamic Programming, 4th edition. Athena Scientific, Belmont.

Properties

Theorem (Monotonicity)

For any functions $J, J' : \mathcal{X} \to \mathbb{R}$*, s.t. for all* $x \in \mathcal{X}$*,*

$$J(x) \leq J'(x),$$

and any stationary policy $\mu : X \to \mathcal{A}$*, we have*

$$(TJ)(x) \leq (TJ')(x) \qquad (T_\mu J)(x) \leq (T_\mu J')(x), \text{ for each } x \in \mathcal{X}.$$

Theorem (Constant Shift)

For every k*, function* $J : \mathcal{X} \to \mathbb{R}$*, stationary policy* μ*,* $r \in \mathbb{R}$*, and* $x \in \mathcal{X}$*,*

$$(T(J + re))(x) = (TJ)(x) + \beta r,$$

$$(T_\mu(J + re))(x) = (T_\mu J)(x) + \beta r.$$

Contraction mapping

- For any $J, J' : \mathcal{X} \to \mathbb{R}$, there holds

$$\max_{x \in X} |(TJ)(x) - (T^k J')(x)| \leq \beta \max_{x \in X} |J(x) - J'(x)|.$$

- Proof. Let $c = \max_{x \in X} |J(x) - J'(x)|$, then

  $J(x) - c \leq J'(x) \leq J(x) + c$ for each $x \in \mathcal{X}$.

  By Monotonicity Lemma: $T(J - ce)(x) \leq T(J')(x) \leq T(J + ce)(x)$

  By Constant shift Lemma: $(TJ)(x) - \beta c \leq T(J')(x) \leq (TJ)(x) + \beta c$