Stat 260/CS 294-102. Learning in Sequential Decision Problems.

Peter Bartlett

1. Recall: MDPs, value iteration, policy iteration.
2. Linear programming formulation.
3. Q functions.
4. Approximate methods for MDPS.

Recall: Markov Decision Processes

Definition: A *Markov Decision Process* (MDP) consists of

1. A state space $\mathcal{X}$,
2. An action space $\mathcal{A}$,
3. A set of Markov chains, $\mathcal{M} = (\mathcal{X}, P_a)$, one for each $a \in \mathcal{A}$,
4. A reward distribution $R : \mathcal{X} \times \mathcal{A} \to \Delta(\mathbb{R})$.

A policy is a sequence of functions $\pi_t : \mathcal{X} \to \Delta(\mathcal{A})$, one for each time t. (A stationary policy is constant with t.)

Recall: Dynamic programming operator

Definition: Define the operators $T, T_\pi : \mathbb{R}^{\mathcal{X}} \to \mathbb{R}^{\mathcal{X}}$ by

$$(TJ)(x) = \max_{a\in\mathcal{A}} \mathbb{E}\left[r_0 + \alpha J(x_1)\middle|\, x_0 = x, a_0 = a\right],$$
$$(T_\pi J)(x) = \mathbb{E}\left[r_0 + J(x_1)\middle|\, x_0 = x, a_0 = \pi(x_0)\right].$$

For a value function estimate $\hat{J} \in \mathbb{R}^{\mathcal{X}}$, define the greedy operator $G : \mathbb{R}^{\mathcal{X}} \to \mathcal{A}^{\mathcal{X}}$:

$$(G\hat{J})(x) := \arg\max_{a\in\mathcal{A}} \mathbb{E}\left[r_0 + \alpha \hat{J}(x_1)\middle|\, x_0 = x, a_0 = a\right].$$

Recall: Value iteration and (generalized) policy iteration

Value iteration:

$$\hat{J}_{k+1} := T\hat{J}_k, \qquad \pi_{k+1} := G\hat{J}_{k+1}.$$

Policy iteration:

$$\pi_{k+1} := GJ^{\pi_k}.$$

Generalized policy iteration:

$$J_{k+1} := T^l_{\pi_k} J_k, \qquad \pi_{k+1} := GJ_{k+1}.$$

Linear program

Bellman equations:

$$J = TJ.$$

Linear programming formulation:

Fix a probability distribution p with support $\mathcal{X}$.

$$\begin{array}{ll} \min\limits_{J} & p^T J \\ \text{s.t.} & J \geq TJ. \end{array}$$

Linear program

Proof. Uses monotonicity: $J \geq J'$ implies $TJ \geq TJ'$. So $J \geq TJ$ implies $J \geq T^k J \to J^*$. Minimizing $\mu^T J$ sets $J = J^*$. $\square$

Dual linear program

$$\begin{aligned}
\max_{\mu} \quad & \sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} \mu(x,a)\mathbb{E}\left[r_0 | x_0 = x, a_0 = a\right] \\
\text{s.t.} \quad & \forall x' \in \mathcal{X},\ \sum_{a\in\mathcal{A}} \mu(x',a) = p(x) \\
& \qquad + \alpha \sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} \mu(x,a) P[x_1 = x' | x_0 = x, a_0 = a].
\end{aligned}$$

View μ as discounted expected number of state-action visits, starting from the distribution p. So criterion is expected discounted reward.

Primal-dual are related via optimal policy: $\pi^*(x) = \arg\max_{a\in\mathcal{A}} \mu(x,a)$.

Q values

Analogous to J^*, but $\mathbb{E}$ and $\max$ are reversed:

$$Q^*(x,a) := \mathbb{E}\left[r_0 + \alpha \max_{a_1\in\mathcal{A}} Q^*(x_1,a_1) \middle| x_0 = x, a_0 = a \right],$$

$$\pi^*(x) := \arg\max_{a\in\mathcal{A}} Q^*(x,a).$$

Value iteration:

$$\hat{Q}_{k+1}(x,a) := \mathbb{E}\left[r_0 + \alpha \max_{a_1\in\mathcal{A}} \hat{Q}_k(x_1,a_1) \middle| x_0 = x, a_0 = a \right],$$

$$\pi_{k+1}(x) := \arg\max_{a\in\mathcal{A}} \hat{Q}_{k+1}(x,a).$$

Approximate dynamic programming

The grand challenge: large-scale MDPs.

In general, cannot hope to find optimal policy if state space is large. Instead, aim to compete with a policy in a restricted class.

e.g., parameterized approximations of value: $\hat{J}_\theta : \mathcal{X} \to \mathbb{R}$. Hope to compete with the best greedy policy corresponding to one of these approximations.

Approximate dynamic programming

Define features, $\Phi \in \mathbb{R}^{\mathcal{X} \times d}$. Value approximation might be linear in these features, $\hat{J}_\theta = \Phi\theta$.

The choice of features is important.

(Alternatively it might be non-linear, for example, deep neural networks. But it is difficult to prove anything; when failure occurs, it's difficult to know whether it's attributable to the choice of the class of approximating functions or to the parameter estimation heuristics.)

1. Approximate policy iteration
2. Approximate value iteration
3. Approximate linear program

Approximate policy iteration

- Find θ so $\hat{J}_\theta$ approximates J_π for current policy π.
  - e.g., linear regression, with covariate-response from simulation. (Exploration is an issue: under-represented states.)
  - e.g., $TD(\lambda)$: stochastic iterative approach to solving $\Phi\theta \approx T_\pi(\Phi\theta)$.
- Use $\hat{J}_\theta$ to determine action in each state (i.e., update policy π).

Approximate value iteration

- Initial θ_0; update θ_{t+1} so $\hat{J}_{\theta_{t+1}}$ approximates $T\hat{J}_{\theta_t}$. (e.g., fitting on a small subset of states, chosen from simulation)
- Works well if $\|\hat{J}_{\theta_{t+1}} - T\hat{J}_{\theta_t}\|_\infty$ is small. But, e.g., minimizing squared error (even on complete state space) can lead to divergence. Not a contraction; mismatch of norms.
- Q learning:
  - Maintain approximation $\hat{Q}_\theta : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ to Q. e.g., $\hat{Q}_\theta(x,a) = \Phi(x,a)^T\theta$.
  - Update using a small subset of states. (e.g., from simulation)
  - Use $\hat{Q}_\theta$ for greedy policy. (Don't need to know transition probabilities.)

Approximate linear program

Fix a probability distribution p on $\mathcal{X}$.

$$\begin{aligned} \min_{J} \quad & p^T \Phi\theta \\ \text{s.t.} \quad & \Phi\theta \geq T\Phi\theta. \end{aligned}$$

This has fewer variables, but still too many constraints. Can use, e.g., constraint sampling.