IE 3186: Approximate Dynamic Programming **Fall 2018**

Lecture 12: Policy Gradient Algorithm

Lecturer: Daniel Jiang *Scribes: Jing Yang*

References:

R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour. *Policy gradient methods for reinforcement learning with function approximation*. In Advances in Neural Information Processing Systems (pp. 1057-1063), 2000.

S. Kunnumkal, H. Topaloglu. *Using stochastic approximation methods to compute optimal base-stock levels in inventory control problems*. Operations Research, 56(3), 646-664, 2008.

W. Sun, G. J. Gordon, B. Boots, J. A. Bagnell. *Dual policy iteration*. arXiv preprint arXiv:1805.10755, 2018.

12.1 Motivation

- Instead of seeking a greedy π with respect to Q, we could directly optimize parameters of a parameterized policy $\pi_\theta(s,a)$, where

$$\pi_\theta(s,a) = \text{ prob of taking action } a \text{ in a state } s \text{ given parameter } \theta.$$

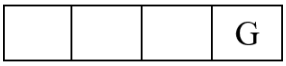
- Main idea: use an SGD type algorithm to update θ:

$$\theta \leftarrow \theta + \alpha \nabla J(\theta),$$

  where $J(\theta)$ is the expected cumulative reward of π_θ (thus a function of θ).

- Why consider $\pi_\theta(s,a)$ instead of $Q^*(s,a)$?
  - Sometimes π^* is easier to approximate than Q^*.
  - Prior knowledge can be used by π^*; perhaps you have intuition on how to act but understanding the value function is less intuitive.
  - Can more easily consider a stochastic policy.

Example 12.1 (Benefits of stochastic policies)**.** Consider a robot whose goal is to hit goal in the grid below. Two actions $\{a_1, a_2\}$ are available, but the robot doesn't know the directions to which a_1 and a_2 correspond (e.g., a_1 could be left in one state and right in another). Reward is -1 per step until G is hit.



Here is an example of how a deterministic policy learned via Q-learning could fail.

- Step 1: Design a basis function for approximation $Q_w(s,a)$ (parameter w). An extreme case is when it is independent of state:

$$X(s,a) = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \text{if } a = a_1, \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{if } a = a_2. \end{cases} \implies Q_w(s,a) = X(s,a)^T \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{cases} w_1 & \text{if } a = a_1, \\ w_2 & \text{if } a = a_2. \end{cases}$$

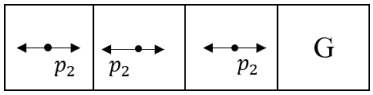
- Step 2: Apply Q-learning and get w^*.
  - Case 1: $w_1^* < w_2^*$, then choose a_2 almost always (ϵ-greedy), that is, choose a_2 with probability $1 - \epsilon/2$.
  - Case 2: $w_1^* \geq w_2^*$, then choose a_1 almost always.

  Note that under the best possible deterministic policy (go right always), reward is -3 and under best policy that Q_w can approximate, reward is -40 if $\epsilon = 0.1$. Therefore, deterministic policy can be bad.

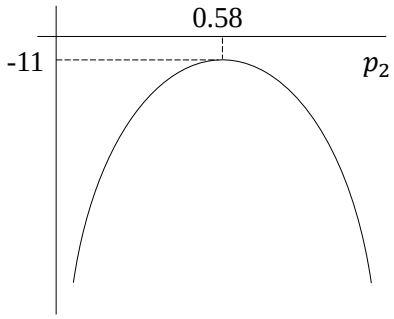
If we use a stochastic policy, but crucially, also state independent, we can be more flexible:

$$\pi_\theta(s) = \begin{cases} a_1 & \text{with probability } 1 - p_2, \\ a_2 & \text{with probability } p_2. \end{cases}$$

That is, the policy is as follows:



Then we have optimal $p_2 = 0.58$ as the following graph shows:



Another drawback of value based methods: small change in Q can cause big change in policy. Another advantage of PG: do not need to optimize over action space.

12.2 Policy Gradient

Define:

$$\begin{aligned} J(\theta) &= \text{discounted reward from running } \pi_\theta \\ &= \mathbf{E}^{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^2 r_{t+1}\right], \end{aligned}$$

where r is the reward, or $-g$ in our original notation from Bertsekas. Our update takes the form

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta).$$

Here are two ways we could calculate/approximate the gradient $\nabla_\theta J(\theta)$.

- Approach 1: Finite Difference. For each dimension $k = 1, 2, \ldots, n$,
  1. Run policy π_θ to estimate $J(\theta)$.
  2. Perturb θ in the kth dimension and run new policy to get $J(\theta + \epsilon e_k)$.
  3. Estimate the kth partial derivative by:

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon e_k) - J(\theta)}{\epsilon}.$$

- Approach 2: Analytical Computation. Let's consider a simple setting, a one-step MDP with initial state distribution $s_0 \sim d(s)$. Termination in one step with reward R_s^a. Then,

$$J(\theta) = \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s, a) R_s^a,$$

where $d(s)$ is the probability of s being the initial state. Therefore,

$$\begin{aligned} \nabla J(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \nabla \pi_\theta(s, a) R_s^a \\ &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi_\theta(s, a) \frac{\nabla_\theta(s, a)}{\pi_\theta(s, a)} R_s^a \\ &= \sum_{s \in S} d(s) \sum_{a \in A} \pi_\theta(s, a) \nabla_\theta \log \pi_\theta(s, a) R_s^a \\ &= \mathbf{E}^{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(S_t, A_t) R_{S_t}^{A^t} \right], \end{aligned}$$

where (S_t, A_t) are actual state and action encountered by simulating a policy.

Theorem 12.2 (Policy Gradient Theorem, Sutton et al 2000)**.** *For any differentiable policy $\pi_\theta(s, a)$:*

$$\nabla_\theta J(\theta) = \mathbf{E}^{\pi_\theta}[\nabla_\theta \log \pi_\theta(S_t, A_t)\, Q^{\pi_\theta}(S_t, A_t)].$$

(Compared to the one-step setting, the reward is replaced by Q^{π_θ}).

The REINFORCE algorithm is the simplest policy gradient algorithm; it uses a Monte Carlo sample of Q^{π_θ} (do one simulation).

12.3 Structured Direct Policy Search

There are many examples in operations research where some advance knowledge of the structure of the optimal policy is given. For example, consider multistage inventory management with backlogged demands:

- ordering product after observing current stock,
- D_{t+1} arrives and is subtracted from total inventory,
- holding cost h, backlogged cost b.

If we denote current stock as x_t and new stock after ordering as y_t, then we have

$$\begin{aligned} V_t^*(x_t) &= \min_{y_t \geq x_t} c(y_t - x_t) + \mathbf{E}\big[\, h(y_t - D_{t+1})^+ + b(D_{t+1} - y_t)^+ + V_{t+1}^*(y_t - D_{t+1})\big] \\ &= \min_{y_t \geq x_t} f_t(y_t) - cx_t. \end{aligned}$$

Cost in last period is $V_{T+1}^*(\cdot) = 0$.

Proposition 12.3. *$f_t(y_t)$ is convex for each t (proof using induction) with finite minimizer.*

Theorem 12.4. *Optimal policy is of "basestock form." That is, if x_t is less than a so-called basestock threshold r_t^*, then $y_t = r_t^*$; otherwise, $y_t = x_t$.*

Proof. This is clear by convexity of f and $V_t^*(x_t) = -cx_t + \min_{y_t \geq x_t} f(y_t)$. □

Policy search over $\tilde{\Pi} = \{\text{all basestock policies}\} = \{(r_1, r_2, \ldots, t_T) : r_k \in \mathbb{R}\}$. Thus we are optimizing the following system

$$\min_{r_1, r_2, \ldots, r_T} \mathbf{E}[\,\text{total cost}(r_1, r_2, \ldots, r_T)\,].$$

However, this optimization problem is not necessarily convex, as shown empirically in the paper (Kunnumkal, Topaloglu, 2008).

Proposed Solution: a mix of policy search, Bellman Equation, and SGD (Kunnumkal, Topaloglu, 2008).

Goal: we seek $\{r_t^*\}$ where $r_t^* = \operatorname{argmin} f_t(r_t)$.

$$f_t(r_t) = cr_t + \mathbf{E}[h(r_t - D_{t+1})^+ + b(D_{t+1} - r_t)^+ + V_{t+1}^*(r_t - D_{t+1})],$$

then the stochastic gradient is

$$\nabla f_t(r_t, D_{t+1}) = c + h\mathbf{1}_{\{D_{t+1} < r_t\}} - b\mathbf{1}_{\{D_{t+1} \geq r_t\}} + \nabla V_{t+1}^*(r_t - D_{t+1}).$$

So one option is

$$r_t^{n+1} = r_t^n - \alpha^n \nabla f_t(r_t, D_{t+1}^n), \text{ for each } t.$$

Need to estimate ∇V_{t+1}^* simultaneously. First:

$$V_{t+1}^*(x_t) = \begin{cases} f_t(x_t) - cx_t & \text{if } x_t \geq r_t^*, \\ \\ f_t(r_t^*) - cx_t & \text{if } x_t < r_t^*. \end{cases}$$

Since

$$\nabla f_t(x_t) = c + h\mathbf{P}(r_t \geq D_{t+1}) - b\mathbf{P}(r_t \geq D_{t+1}) + \mathbf{E}\nabla V_{t+1}^*(r_t - D_{t+1}),$$

we can have

$$\nabla V_{t+1}^*(x_t) = \begin{cases} c + h\mathbf{P}(r_t \geq D_{t+1}) + b\mathbf{P}(r_t \geq D_{t+1}) - c + \mathbf{E}\nabla V_{t+1}^*(x_t - D_{t+1}) & \text{if } x_t \geq r_t^*, \\ \\ c + h\mathbf{P}(r_t \geq D_{t+1}) + b\mathbf{P}(r_t \geq D_{t+1}) - c + \mathbf{E}\nabla V_{t+1}^*(r_t^* - D_{t+1}) & \text{if } x_t < r_t^*. \end{cases}$$

A stochastic version of $\nabla V_t^*(x_t, D_{t+1})$ is:

$$\nabla V_{t+1}^*(x_t, D_{t+1}) = \begin{cases} h\mathbf{1}_{\{D_{t+1} < x_t\}} - b\mathbf{1}_{\{D_{t+1} \geq r_t^*\}} + \nabla V_{t+1}^*(x_t - D_{t+1}) & \text{if } x_t \geq r_t^*, \\ \\ h\mathbf{1}_{\{D_{t+1} < r_t^*\}} - b\mathbf{1}_{\{D_{t+1} \geq r_t^*\}} + \nabla V_{t+1}^*(r_t^* - D_{t+1}) & \text{if } x_t < r_t^*. \end{cases}$$

Use ξ_t^n to approximate $\{\nabla V_t^*(x_t, D_{t+1}^n)\}$ on a given $(D_1^n, D_2^n, \ldots, D_T^n)$ and $(r_1^n, r_2^n, \ldots, r_{T-1}^n)$.

$$\begin{aligned} &\xi_t^n(x_t, D_{t+1}^n, \ldots, D_T^n) \\ &\quad = \begin{cases} h\mathbf{1}_{\{D_{t+1} < x_t\}} - b\mathbf{1}_{\{D_{t+1} \geq r_t^*\}} + \xi_{t+1}^n(x_t - D_{t+1}^n, D_{t+2}^n \ldots, D_T^n) & \text{if } x_t \geq r_t^*, \\ \\ h\mathbf{1}_{\{D_{t+1} < r_t^*\}} - b\mathbf{1}_{\{D_{t+1} \geq r_t^*\}} + \xi_{t+1}^n(r_t^* - D_{t+1}^n, D_{t+2}^n \ldots, D_T^n) & \text{if } x_t < r_t^*. \end{cases} \end{aligned}$$

Thus, the approximation to $\nabla f_t(x_t, D_{t+1})$ is

$$s_t^n(x_t, D_{t+1}^n, \ldots, D_T^n) = c + h\mathbf{1}_{\{D_{t+1} < r_t^*\}} - b\mathbf{1}_{\{D_{t+1} \geq r_t^*\}} + \xi_{t+1}^n(x_t - D_{t+1}^n, D_{t+2}^n, \ldots, D_T^n).$$

So, the approximate SGD update is

$$r_t^{n+1} = r_t^n - \alpha s_t^n(r_t^n, D_{t+1}^{n+1}, \ldots, D_T^{n+1}).$$

Sketch of Convergence:

$$e_t^n(x_t) = \nabla V_T^*(x_t) - \mathbf{E}[\xi_t^n(x_t, D_{t+1}^n, \ldots, D_T^n)].$$

At time T, $e_T^n(\cdot) = 0$, so $s_{t-1}^n(x_t, D_{t+1}^{n+1}, \ldots, D_t^{n+1})$ is not biased. Thus, r_{T-1}^{n+1} is updated by an SGD that converges, so $r_{T-1}^n \to r_{T-1}^*$. This, in turn, means that the bias vanishes, $e_{T-1}^n(\cdot) \to 0$, and so $r_{T-2}^n \to r_{T-2}^*$.

12.4 Paper Discussion: Dual Policy Iteration

This paper presents and analyzes Dual Policy Iteration (DPI), which is a framework that alternatively computes a non-reactive policy ia more advance and systematic search, and updates a reactive policy via imitating the non-reactive one. Based on this framework, the paper provides a simple instance of DPI for RL with unknown dynamics. The instance integrates local model fit, local model-based search, and

reactive policy improvement via imitating the local optimal policy resulting from model-based search. For the theoretical results, the authors show that integrating model-based search and imitation into policy improvement could result in larger policy improvement at each step. In computational experiment, this paper demonstrates the improved sample efficiency compared to strong baselines.

12.4.1 Preliminaries

In DPI algorithm, two policies are under consideration at any time during training:

- Reactive policy: usually learned by some form of function approximation, used for generating samples and deployed at test time.
- Intermediate policy: can only be constructed or accessed during training, used as an expert policy to guide the improvement of the reactive policy.

The RL setting considered in this paper: $(S, A, P, c, \rho_0, \gamma)$, where S, A denote the state and action space, respectively. P is the transition dynamics, c is the cost function and ρ_0 denotes the initial distribution of states, γ is the discount factor.

Side notes: For two distributions $\mathbb{P}_1$ and $\mathbb{P}_2$,

- $D_{TV}(\mathbb{P}_1, \mathbb{P}_2) = \frac{\|\mathbb{P}_1 - \mathbb{P}_2\|_1}{2}$ is the total variation distance
- $D_{KL}(\mathbb{P}_1, \mathbb{P}_2) = \int \mathbb{P}_1(x) \log(\mathbb{P}_1(x)/\mathbb{P}_2(x))dx$ is the KL divergence.

$$V^\pi(s) = \mathbf{E}\Big[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) \,|\, s_0 = s\Big];$$
$$Q^\pi(s, a) = c(s, a) + \gamma \mathbf{E}_{s' \sim P_{s,a}}[V^\pi(s')];$$
$$J(\pi) = \mathbf{E}_{s \sim \rho_0}[V^\pi(s)].$$

Define advantage function as $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. Goal is to learn a single stationary policy π^* that minimizes $J(\pi)$.

12.4.2 Main results

Lemma 12.5. *For any two policies π and π', we have*

$$J(\pi) - J(\pi') = \frac{1}{1-\gamma} \mathbf{E}_{(s,a) \sim d_\pi \pi}[A^{\pi'}(s, a)],$$

where $d_\pi \pi$ is the joint distribution such that $d_\pi \pi(s, a) = d_\pi(s)\pi(a|s)$.

Dual Policy Iteration
Consider the min-max optimization framework:

$$\min_{\pi\in\Pi}\max_{\eta\in\Pi}\mathbf{E}_{s\sim d_\pi}[\mathbf{E}_{a\sim\pi(\cdot|s)}[A^\eta(s,a)]],$$

where $d_\pi(s)=(1-\gamma)\sum_{t=0}^{\infty}\gamma^t d_\pi^t(s)$ is the state visitation distribution and $d_\pi^t(s)$ is the distribution of state at time step t, induced by running the policy π.

Then Nash equilibrium is $(\pi,\eta)=(\pi^*,\pi^*)$. DPI alternatively fixes one policy and updates the second policy.

- Updating η :
  Given π_n, the objective function for η becomes

  $$\max_{\eta}\mathbf{E}_{s\sim d_{\pi_n}}[\mathbf{E}_{a\sim\pi_n(\cdot|s)}[A^\eta(s,a)]].$$

  According to the above Lemma, updating η is equivalent to finding the optimal policy

  $$\pi^*=\text{argmax}_\eta(J(\pi_n)-J(\eta))=\text{argmin}_\eta J(\eta).$$

  However, this is as hard as the original problem.

  Therefore, the paper suggests to update η locally by constraining it to a trust region around π_n:

  $$\text{argmin}_\eta J(\eta),\ \text{s.t. }\mathbf{E}_{s\sim d_{\pi_n}}D_{TV}[(\eta(\cdot|s),\pi(\cdot|s))]\le\alpha.$$

  However, to solve this problem, we need to learn $P_{s,a}$.

  To learn $P_{s,a}$, the paper seeks a model $\hat{P}$ such that $\mathbf{E}_{s\sim d_{\pi_n}\pi_n}D_{TV}(\hat{P}_{s,a},P_{s,a})$ is small. Moreover, by Pinsker's inequality

  $$D_{KL}(\hat{P}_{s,a},P_{s,a})\ge D_{TV}(\hat{P}_{s,a},P_{s,a})^2,$$

  the following optimization problem can be considered instead.

  $$\text{argmin}_{\hat{P}\in\mathbf{P}}\mathbf{E}_{s\sim d_{\pi_n},a\sim\pi_n(s)}D_{KL}(P_{s,a},\hat{P}_{s,a})=\text{argmin}_{\hat{P}\in\mathbf{P}}\mathbf{E}_{s\sim d_{\pi_n},a\sim\pi_n(s),s'\sim P_{s,a}}[-\log\hat{P}_{s,a}(s')],$$

  and this can be solved by using existing stochastic MBOC solvers.

- Updating π:
  Given η_n, π_{n+1} is computed by performing the following constrained optimization procedure:

  $$\text{argmin}_\pi\mathbf{E}_{s\sim d_{\pi_n}}[\eth\sim\pi(\cdot|\sim)[A_n^\eta(s,a)]],\ \text{s.t. }\mathbf{E}_{s\sim d_{\pi_n}}[D_{TV}(\pi(\cdot|s)),\pi_n(\cdot|s)]\le\beta,$$

  and this can be solved by imitating η_n.

Policy Improvement
Let $\Delta_n(a) \geq 0$ be the performance gain from η_n^* over π_n, i.e.

$$J(\pi_n) - J(\eta_n^*) \geq \Delta_n(\alpha).$$

Theorem 12.6. *Assume $\hat{P}$ satisfies $\mathbf{E}_{(s,a)\sim d_{\pi_n}\pi_n} D_{TV}(\hat{P}_{s,a}, P_{s,a}) \leq \delta$, we have the*

$$J(\eta_n) \leq J(\pi_n) - \Delta_n(\alpha) + O\left(\frac{\gamma\delta}{1-\gamma} + \frac{\gamma\alpha}{(1-\gamma)^2}\right).$$

Define $\mathbb{A}_n(\pi_{n+1})$ as the disadvantage of π_{n+1} over η_n under d_{π_n}.

$$\mathbb{A}_n(\pi_{n+1}) = \mathbf{E}_{s\sim d_{\pi_n}}[\mathbf{E}_{a\sim\pi_{n+1}(\cdot|s)}[A^{\eta_\pi}(s,a)]].$$

Theorem 12.7. *The improvement of π_{n+1} over π_n is*

$$J(\pi_{n+1}) - J_{\pi_n} \leq \frac{\beta\epsilon}{(1-\gamma)^2} - \frac{|\mathbb{A}_n(\pi_{n+1})|}{1-\gamma} - \Delta_n(\alpha).$$