Course Notes On Dynamic Optimization (Fall 2023)
Lecture 11: Policy gradient methods

Instructor: Daniel Russo

Email: djr2174@gsb.columbia.edu

Graduate Instructor: David Cheikhi
Email: d.cheikhi@columbia.edu

These notes are partly based of scribed notes from a previous edition of the class. I have done some follow up light editing, but there may be typos or errors.

Disambiguation: I use the term policy gradient methods very broadly, encompassing any (stochastic) first-order method applied to minimize total expect costs. Others view policy gradient methods as, essentially, referring to variants of the so-called REINFORCE algorithm, which uses a specific way to estiamte gradients.

1 Problem setup

Throughout these notes we continue to study indefinite horizon problems under the assumption that all policies eventually reach the terminal state. We continue to assume that all policies are assured to reach the terminal state and $\sup_{\pi,x} \mathbb{E}^{\pi}[\tau \mid x_0 = x] \leqslant \infty$.

For simplicity, we assume the set of feasible controls $U(x)$ is the same at all states x and denote this by U. To facilitate taking gradients, we assume $U \subset \mathbb{R}^k$ is a convex set. This is satisfies automatically in some problems, like linear quadratic control. In others, we need to allow for randomized actions.

Example 1 (Stochastic actions)**.** *Suppose there are m deterministic actions. For $i \in [m]$, denote by $g(x,i)$ the immediate expected cost when action i is selected in state x and by $p(x'|x,i)$ the probability of transitioning to state x'. For a probability vector $u \in \{p \in \mathbb{R}^m : p \geqslant 0, \sum_{i=1}^m p_i = 1\}$, define*

$$g(x,u) = \sum_{i=1}^{m} u_i g(x,i) \quad \textit{and} \quad p(x'|x,u) = \sum_{i=1}^{m} u_i p(x'|x,i).$$

This converts a problem with a finite action space to one with a continuous action space.

1.1 Parameterized polices.

Consider a parameterized class of policies $\Pi_\Theta = \{\pi_\theta : \theta \in \Theta\}$ where $\Theta \subset \mathbb{R}^n$ is a convex set. For each $\theta \in \Theta$, $\pi_\theta : \mathcal{X} \to U$ is a policy.

Example 2. *In linear quadratic control, the state x is an m dimensional vector. It is known that a linear policy of the form*

$$\pi_\theta(x) = \theta x \quad \theta \in \mathbb{R}^{m\times m} \quad x \in \mathbb{R}^m,$$

is optimal for some θ. This is natural policy class to search over.

Example 3 (Directly-parameterized state-aggregated policies)**.** *Let the control space be probability vectors over k base actions, as in Example 1. Let $\phi : \mathcal{X} \to \mathcal{X}$ denote a state-aggregation rule, which associates each state x with one of n representer states $\phi(x) \in \{\bar{x}_1, \ldots, \bar{x}_n\}$. A state aggregated policy is a rule which assigns a the same action distribution to any two states with a common representer, i.e. $\phi(x) = \phi(x') \implies \pi_\theta(x) = \pi_{\theta'}(x)$.*

Let $\theta = (\theta_{j,i})_{j\in[n]\,,\,i\in[k]} \in \mathbb{R}_+^{n\times k}$ encode the action probabilities as

$$\phi(x) = \bar{x}_j \implies \pi_\theta(x) = (\theta_{j,1}, \ldots, \theta_{j,k})$$

where $\theta_{j,1} + \ldots + \theta_{j,k} = 1$. The set of feasible θ, denoted Θ, is the set of $n \times m$ matrices whose entries are non-negative and whose rows add to 1.

Example 4 (Softmax-parameterized state-aggregated policies)**.** *Repeat the following example, but now for $\phi(x) = \bar{x}_j$, let the policy be parameterized as*

$$\pi_\theta(x) = \text{softmax}(\theta_j) = \frac{1}{Z}\left(e^{\theta_{j,1}}, \ldots\, e^{\theta_{j,k}}\right) \quad Z = \sum_{i=1}^{k} e^{\theta_{j,i}}.$$

In words, the action probabilities at state x in cluster j is the softmax function applies to state the j^{th} row of θ.

Example 5 (Neural network)**.** *$f_\theta : \mathcal{X} \to \mathbb{R}^M$ is some neural network and $\pi_\theta(x) = \text{softmax}(f_\theta(x))$ tacks on a softmax operation to ensure the output is probability vector (over the k base actions.)*

1.2 Global loss function.

Define the scalar loss function

$$\ell(\pi) = \mathbb{E}_{x\sim w}\left[J_\pi(x)\right].$$

and overload notation to write $\ell(\theta) = \ell(\pi_\theta)$.

If $\mathcal{X}$ is countable, and $w(x) > 0$ for each $x \in \mathcal{X}$, then $\pi^* \in \arg\min_\pi \ell(\pi)$ if and only if $J_{\pi^*} = J^*$. Why? First suppose $J_{\pi^*} = J^*$. Then $\ell(\pi^*) = \sum_x J_{\pi^*}(x)w(x) = \sum_x J^*(x)w(x) \leqslant \sum_x J_\pi(x)w(x) \leqslant J_\pi(x)w(x)$. On the other hand, equality holds throughout if and only if $J_\pi(x) = J^*(x)$ for each x.

While this is true, if $w(x)$ is very small,

1.3 Policy gradient.

The most basic policy gradient methods is a stochastic gradient iteration of the form

$$\theta_{k+1} = \theta_k - \alpha_k\left(\nabla\ell(\theta_k) + \text{noise}\right)$$

where α is a stepsize.

This is just one example, and there are lots of stochastic first order methods. They vary in terms of stepsizes, how they deal with constraints, how they gradients are estimated, etc.

2 Policy gradient with a differentiable simulation

For simplicity, I will focus on a discounted problem, where $\tau \sim \text{Geom}(1-\alpha)$ is independent of all else. For a given τ driving noises $w = (w_0, \ldots, w_{\tau-1})$, and initial state x_0 write

$$\ell(\theta; [x_0, \tau, w]) = \sum_{k=0}^{\tau-1} g(x_k, \pi_\theta(x_k)) \quad \text{where} \quad x_{k+1} = f(x_k, u_k, w_k).$$

When states are continuous and f, g and $\pi_\theta(x)$ are appropriately smooth $\ell(\theta; [x_0, \tau, w])$ is differentiable and $\nabla \ell(\theta) = \mathbb{E}[\nabla \ell(\theta; [x_0, \tau, w])]$, where the expectation is over the 'scenario' $[x_0, \tau, w]$.

Algorithm 1 Policy gradient with a differentiable simulator

Require: initial parameter θ_0, sepsize γ
for Episode $n = 0, 1, 2, \ldots$ **do**
  Sample episode length $\tau \sim \text{Geom}(1-\alpha)$
  Sample driving noises $w = (w_0, \ldots, w_{\tau-1})$
  Sample initial stae $x_0 \sim w$
  Reset loss: $\ell = 0$
  for $k = 0, \ldots, \tau - 1$ **do** ▷ Forward pass
    Set $u_k = \pi_\theta(x_k)$
    $\ell \leftarrow \ell + g(x_k, u_k)$
    Set $x_{x+1} = f(x_k, u_k, w_k)$
  end for
  Calculate $\nabla_\theta \ell$ by backpropogation ▷ Backward pass
  Update $\theta \leftarrow \theta - \gamma \nabla_\theta \ell$ ▷ Gradient step
end for
return μ_n

3 Issues with policy gradient convergence

- **OUR FOCUS!**: The function $\pi \mapsto \ell(\pi)$ is nonconvex, with potentially bad local minima in Π_Θ.
- The policy parameterization $\theta \mapsto \pi_\theta$ means that $\theta \mapsto \ell(\pi_\theta)$ can look very flat. For instance, modify the example above so that π_θ moves right with probability θ^{100}. The gradient of $\pi_\theta(x)$ with respect to θ can be exceptionally small, leading to exceptionally slow convergence for naive gradient algorithms. In degenerate cases, where the policy becomes 'infinitely' insensitive to parameter changes, the method not to converge at all.
- How you generate stochastic gradients impacts their variance (and sometimes their bias), impacging convergence rates.

4 Policy gradient: failure of global convergence due to nonconvexity

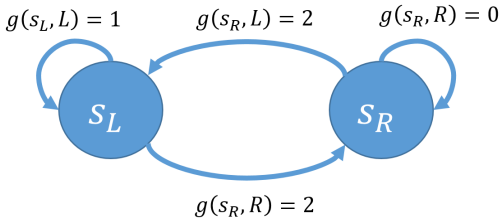
Consider a discounted formulation of the problem in Figure 1a. For large discount factor, the optimal policy plays the action R in either state. This is contained in a simple policy class $\{\pi_\theta : \theta \in$

$[0,1]\}$ where

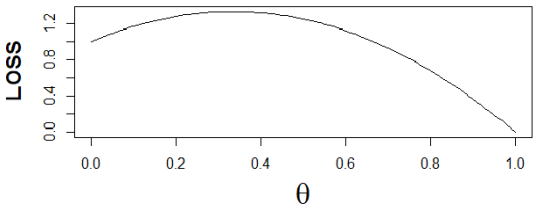
- In either state, the probability of moving right under π_θ is θ.

You can think of this as a state-aggregated policy class. But aggregation does not result inhibit optimality, since the optimal policy is contained in this restricted class.

Figure 1b shows that despite the optimal policy being contained in the class, local policy search may get stuck in a bad local minimum at $\theta = 0$.



(a) A bad example for policy gradient



(b) Normalized loss $(1-\alpha)\ell(\theta)$.

5 Policy gradient is the same a local optimization of a single-period objective

> **The policy gradient theorem:** To determine the benefit of employing a slightly different policy over an infinite horizon, it is enough to consider the benefit of employing that policy for a single period, before reverting to the status quo.

5.1 Recap of policy iteration

Beginning with some status-quo policy π, the policy iteration update looks to solve for $\pi^+ \in \arg\min_{\pi'} T_{\pi'} J_\pi$. This is meant in an element, by element way, as in $\pi^+(x) \in g(x, \pi(x)) + \sum_{x' \in \mathcal{X}} J_{\pi^+}(x')$. We could analogously write

$$\pi^+ \in \arg\min_{\pi'} L(\pi' \mid \pi, d) := \langle T_{\pi'} J_\pi \,,\, d \rangle$$

where $d(x) > 0$ is a strictly positive weighting function.

5.2 Policy gradient formula

The main result relates local optimization on the infinite horizon cost $\ell(\cdot)$ to local optimization which to the *one-step lookahead* cost considered in policy iteration.

$$\arg\min_{\theta^+ : \|\theta - \theta^+\| \leqslant \epsilon} \ell(\theta^+) \approx \arg\min_{\theta^+ : \|\theta - \theta^+\| \leqslant \epsilon} \langle T_{\pi_{\theta^+}} J_{\pi_\theta} \,,\, d_{\pi_\theta} \rangle$$

This follows from the following derivation of the policy gradient formula.

Proposition 1. *Set* $\pi^\gamma = \pi + \gamma(\pi^+ - \pi)$.

$$\ell(\pi^\gamma) = \ell(\pi) + \left\langle \underbrace{T_{\pi^\gamma} J_\pi - J_\pi}_{\text{one step imprevement}} \, , d_\pi \right\rangle + O(\gamma^2)$$

Proof. Apply the variation Bellman equation below with $\pi^+ = \pi^\gamma$ and $J = J_\pi$. THis gives

$$J_{\pi^\gamma} - J_\pi = (I - P_{\pi^\gamma})^{-1}(T_{\pi^\gamma} J_\pi - J_\pi).$$

Multiplying the left hand side by the row vector w gives

$$\begin{aligned}\ell(\pi^\gamma) - \ell(\pi) &= \langle T_{\pi^\gamma} J_\pi - J_\pi \, , \, d_\pi \rangle + \langle T_{\pi^\gamma} J_\pi - J_\pi \, , \, d_{\pi^\gamma} - d_\pi \rangle \\ &= \langle T_{\pi^\gamma} J_\pi - J_\pi \, , \, d_\pi \rangle + \underbrace{\langle T_{\pi^\gamma} J_\pi - T_\pi J_\pi \, , \, d_{\pi^\gamma} - d_\pi \rangle}_{O(\gamma^2)}.\end{aligned}$$

□

Lemma 1 (Variational form of Bellman's equation)**.** *For any* $J \in \mathbb{R}^{\mathcal{X}}$ *and stationary policy* $\tilde{\pi}$*,*

$$J_{\pi^+} - J = (I - P_{\pi^+})^{-1}(T_{\pi^+} J - J).$$

5.3 Frank-wolfe policy gradient is soft policy iteration

Consider a problem in which U denotes probability distributions over k base actions as in Example 1. Suppose there are n states indexed as $\mathcal{X} = \{1, \ldots, n\}$. Let the policy class consist of all stochastic policies as: $\Pi = \{\pi \in \mathbb{R}_+^{n \times k} : \sum_{i=1}^k \pi_{j,i} = 1 \, \forall \, j \in [n]\}$. If n is small enough, we can drop θ from the notation and just apply an optimization algorithm over these policies.

There are many first-order optimization algorithm. But the so-called *conditional gradient* or *Frank-Wolfe* algorithm has the most transparent connection to policy iteration. Given a policy π, and stepsize $\gamma \in (0, 1)$, it produces a new policy π^+ as

$$\pi^+ = (1 - \gamma)\pi + \gamma y^+$$

where

$$y^+ \arg\min_{y \in \Pi} \quad \underbrace{\ell(\pi) + \langle \nabla \ell(\pi) \, , \, y \rangle}_{\text{Linearized approximation to } \ell(\cdot)} \quad .$$

It optimized a linearized approximation to $\ell(\cdot)$ globally, and then takes just a small step in the direction of that optimizer.

> **Equivalence to soft policy iteration:** You will show on your homework that y^+ is the policy iteration update to π, so π^+ is an average of π and its policy iteration update.

5.4 Implications of the connection between policy gradient and policy iteration

We have seen both positive and negative evidence regarding policy gradient convergence. On the positive side, we saw an example in which policy gradient gets stuck in bad local minimum even though the policy class contains an optimal policy.

On the positive side, we sad that conditional-gradient (a.k.a Frank-wolfe) policy gradient is equivalent to soft-policy iteration when applied to the set of directly parameterized stochastic policies. Therefore, it converges to the global optimal policy, just as policy iteration does. It converges at a linear (even a superlinear) rate, just as policy iteration would. This isn't the behavior you would have expected from from gradient descent on a non-convex loss.

A number of policy classes lie in between these. They are (nearly) closed under policy iteration, in the sense that they (nearly) contain the policy iteration update of any policy in the class. Linear policies in linear quadratic control are a good example of a closed policy class. A paper by Jalaj Bhandari and myself titled *Global optimality guarantees for policy gradient methods* formalizes that the policy gradient objective has no bad local minima when the policy class is closed (among other conditions).