Policy Gradient
Policy Gradient (PG)

- Given a class of parameterized policies $\pi_\theta$, optimize
  \[ J(\pi_\theta) := \mathbb{E}_{s \sim d_0}[V^{\pi_\theta}(s)] \]

- We will often make the dependence of $\pi_\theta$ on $\theta$ implicit, i.e.,
  when we write $\pi$ we mean $\pi_\theta$ in this part of the course

- Simple idea: can run (stochastic) gradient descent if we can obtain (an unbiased estimate of) $\nabla_\theta J(\pi_\theta)$
  - will abbreviate as $J(\pi)$

- Beautiful result: an unbiased estimate can be obtained from a single on-policy trajectory, without using knowledge of $P$ and $R$ of the MDP!

- Has a strong connection to IS

- “Vanilla” PG (e.g., REINFORCE) is considered a Monte-Carlo method—it does not leverage Bellman equation
Why PG?

- RL methods can be categorized according to what we try to approximate: model-based RL, value-based RL, policy search
- Eventually we only care about a good policy!
- value-based RL is indirect (model-based even more)
- If a value function induces a good greedy policy, but the function itself severely violates Bellman equation, you won’t be able to find such a policy via value-based methods
- In other words, policy search is agnostic against misspecification of function approximation
  - Apart from difficulties in optimization, there is nothing that prevents policy search from finding the best policy in class
- Value- (and model-) based methods have their advantages—will come back later
Example of policy parametrization

• Linear + softmax:
  • Featurize state-action: $\phi : S \times A \rightarrow \mathbb{R}^d$
  • Policy: $\pi(a|s) \propto e^{\theta^T \phi(s,a)}$

• Recall that in SARSA we’ve also seen the softmax policy
• There we include a temperature parameter, $\pi(a|s) \propto e^{\theta^T \phi(s,a)/T}$
• Why the difference?
  • In TD, we want $\theta^T \phi(s,a) \approx Q^\pi(s,a)$. We don’t have the freedom to rescale it; i.e., if $\theta^T \phi(s,a) \approx Q^\pi(s,a)$, then $(2\theta)^T \phi(s,a) \neq Q^\pi(s,a)$.
  • We need an additional knob ($T$) to control the stochasticity of $\pi$
  • In PG, $\theta^T \phi(s,a)$ does not carry any meaning—it’s totally possible that eventually we find a $\theta$ but $\theta^T \phi(s,a) \neq Q^\pi^\theta(s,a)$!
  • That’s why we can absorb the temperature parameter in $\theta$
  • Reflection of the agnosticism of PG
Derivation of PG

- Use $\tau := (s_1, a_1, r_1, \ldots, s_H, a_H, r_H)$ to denote a trajectory (episodic)
- Use $\tau \sim \pi$ as a shorthand for distribution induced by $\pi$
- Let $R(\tau) := \sum_{t=1}^{H} \gamma^{t-1} r_t$
- Ver 1: $\nabla J(\pi) = \mathbb{E}_{\tau \sim \pi}[R(\tau) \sum_{t=1}^{H} \nabla \log \pi(a_t | s_t)]$
  - Will derive using a “MC”-style proof
- Ver 2: $\nabla J(\pi) = \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^{\pi}, a \sim \pi(s)}[Q^{\pi}(s, a) \nabla \log \pi(a | s)]$
  - $d^{\pi}$ is the normalized occupancy (from $d_0$ as init distribution)
  - Possible implementation: (1) roll out $\tau \sim \pi$, (2) pick a random time step $t$ w.p. $\propto \gamma^{t-1}$, (3) $\mathbb{E}[\sum_{t=t}^{H} \gamma^{t' - 1} r_t] \nabla \log \pi(a_t | s_t)$
    - Note that $\mathbb{E}[\sum_{t=t}^{H} \gamma^{t' - 1} r_t | s_t, a_t] = Q^{\pi}(s_t, a_t)$
    - Take expectation over step (2) gives an alternative form: $\nabla J(\pi) = \mathbb{E}_{\tau \sim \pi}[\sum_{t=1}^{H} (\sum_{t=t}^{H} \gamma^{t' - 1} r_t \cdot ) \nabla \log \pi(a_t | s_t)]$
  - Will derive using a “DP”-style proof; can also be derived using the MC-style proof for ver 1
• Use $\tau := (s_1, a_1, r_1, \ldots, s_H, a_H, r_H)$ to denote a trajectory (episodic)
• Use $\tau \sim \pi$ as a shorthand for distribution induced by $\pi$
• Let $R(\tau) := \sum_{t=1}^{H} \gamma^{t-1} r_t$

\[\nabla J(\pi) = \nabla_{\theta} \mathbb{E}_{\tau \sim \pi} [ R(\tau) \sum_{t=1}^{H} \nabla \log \pi(a_t | s_t) ]\]

\[J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=1}^{H} \gamma^{t-1} r_t \right]\]

\[= \mathbb{E}_{\tau \sim \pi} \left[ R(\tau) \right].\]

\[
\frac{\text{dlog} y}{\text{dx}} = \frac{1}{y} \frac{dy}{dx}.
\]

\[\nabla \log P^{\pi_0}(\tau) \cdot R(\tau) = \mathbb{E}_{\tau \sim \pi} \left[ \nabla_{\theta} \log P^{\pi_0}(\tau) \cdot R(\tau) \right].\]

\[\nabla_{\theta} \log P^{\pi_0}(\tau).\]

\[= \nabla_{\theta} \log \left( d_0(s_1) \cdot \pi(a_1 | s_1) \cdot P(s_2 | s_1, a_1) \cdot \pi(a_2 | s_2) \right)\]

\[= \nabla_{\theta} \log d_0(s_1) + \nabla_{\theta} \log \pi(a_1 | s_1) + \nabla_{\theta} \log P(s_2 | s_1, a_1) + \nabla_{\theta} \log \pi(a_2 | s_2) + \ldots \]
$$\nabla \log \prod_{t} \phi(a_t | s_t) = \frac{e^{\Theta^T \phi(s, a)}}{\sum_{a_t} e^{\Theta^T \phi(s, a)}}.$$ 

$$= \nabla_{\Theta} \left( \log \left( e^{\Theta^T \phi(s, a)} \right) - \log \left( \sum \frac{e^{\Theta^T \phi(s, a')}}{\sum_{a'} e^{\Theta^T \phi(s, a')}} \phi(s, a') \right) \right)$$

$$= \phi(s, a) - \frac{\sum_{a'} e^{\Theta^T \phi(s, a')} \cdot \phi(s, a')}{\sum_{a'} e^{\Theta^T \phi(s, a')}}$$

$$= \phi(s, a) - E_{a_t \sim \pi} [ \phi(s, a_t) ].$$

$$\nabla J(\pi) = E_{\pi} \left[ \left( \sum_{t=1}^{H} g^{t-1} r_t \right) \left( \sum_{t=1}^{H} \nabla \log \pi(a_t | s_t) \right) \right].$$

$$= E_{\pi} \left[ \sum_{t=1}^{H} \left( \nabla \log \pi(a_t | s_t) \sum_{t'=1}^{H} g^{t'-1} r_{t'} \right) \right].$$

$$= E_{\pi} \left[ \sum_{t=1}^{H} \nabla \log \pi(a_t | s_t) \sum_{t'=t}^{H} g^{t'-1} r_{t'} \right].$$
\[
\frac{dJ(\theta)}{d\theta} = \lim_{\Delta \theta \to 0} \frac{J(\theta + \Delta \theta) - J(\theta)}{\Delta \theta}
\]

\[
E_\pi \left[ \sum_{t=1}^{H} \nabla \log \pi_t(a_t|s_t) \sum_{t'=t}^{H} y_{t'-1} r_{t'} \right]
\]

\[
= \sum_{s_t, a_t} \sum_{t=1}^{H} \nabla \log \pi_t(a_t|s_t) \sum_{t'=t}^{H} y_{t'-1} r_{t'} \left[ s_t, a_t \right]
\]

\[
= \sum_{t=1}^{H} \frac{y_{t}-1}{t} E_{s_t, a_t} \left[ \nabla \log \pi_t(a_t|s_t) - Q^\pi(s, a) \right]
\]

\[
= \left(1 - \frac{1}{t} \right) \sum_{t=1}^{H} \frac{d_t}{d_t} \left[ \nabla \log \pi_t(a_t|s_t) - Q^\pi(s, a) \right]
\]

\[
E_p[f] + E_q[f] = 2 \cdot E_{p+q} \frac{f}{2} [f].
\]
\[ Q^*(s, \alpha) \approx \Theta^T \phi(s) \]

\[ Q^*(s, \beta) \approx \Theta^T \phi(s) \]

\[ Q^*(s, \alpha) = \Theta^T \phi(s, \alpha) \]

\[ \phi(s, \alpha) = \begin{bmatrix} \theta_L^T \\ \theta_R \end{bmatrix}^T \phi(s, \alpha) \]

\[ \phi(s, \alpha) = \begin{bmatrix} \Psi(s) \\ 0 \end{bmatrix} \]

\[ \phi(s, \beta) = \begin{bmatrix} 0 \\ \Psi(s) \end{bmatrix} \]

\[ s_t = s, \ \alpha_t = \alpha \]
Pros & Cons of PG, and beyond

• Standard PG is fully on-policy, and it’s hard to reuse data
  • after each update step, the policy changes and we need to generate MC trajectories from the new policy
• in practice, it suffers from noisy gradient estimate
• Blend PG with value-based method:
  • Instead of using MC estimate \( \sum_{t=t}^{H} \gamma^{t-1}r_t \) for \( Q^\pi(s_t, a_t) \), use an approximate value-function \( \hat{Q}^\pi(s_t, a_t) \), often trained by TD
  • e.g., using expected Sarsa—can leverage previous (off-policy) data to learn \( \hat{Q}^\pi(s_t, a_t) \)
  • “Actor-critic”: the parametrized policy is called the actor, and the value-function estimate is called the critic

\[
\nabla J(\pi) = \frac{1}{1-\gamma} [\mathbb{E}_{s \sim d^\pi,a \sim \pi(s)}[Q^\pi(s, a) \nabla \log \pi(a | s)]
\]

\[
\hat{Q}^\pi(s_t, a_t) = \sum_{t=t}^{H} \gamma^{t-1}r_t
\]
Baseline in PG

- \( \nabla J(\pi) = \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^s, a \sim \pi(s)} [Q^\pi(s, a) \nabla \log \pi(a|s)] \)

- For any \( f : S \rightarrow \mathbb{R} \), \( \nabla J(\pi) = \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^s, a \sim \pi(s)} [(Q^\pi(s, a) - f(s)) \nabla \log \pi(a|s)] \)
  - for any \( s \), \( \mathbb{E}_{a \sim \pi(s)} [f(s) \nabla \log \pi(a|s)] = f(s) \cdot \mathbb{E}_{a \sim \pi(s)} [\nabla \log \pi(a|s)] = 0 \)
  - proof: \( \mathbb{E}_{a \sim \pi(s)} [\nabla \log \pi(a|s)] = \sum_a \pi(a|s) \nabla \log \pi(a|s) = \sum_a \nabla \pi(a|s) = \nabla \sum_a \pi(a|s) = \nabla 1 = 0 \)
  - One choice: \( f = V^\pi(s) \)
- \( \nabla J(\pi) = \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d^s, a \sim \pi(s)} [A^\pi(s, a) \nabla \log \pi(a|s)] \)
- recall that \( A \) is the advantage function
Comparing AC with Policy Iteration

- $\nabla J(\pi) \approx \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^n, a \sim \pi(s)}[\hat{Q}^\pi(s, a) \nabla \log \pi(a|s)]$

- A different but related procedure: freeze $\pi$, update the parameter of another policy $\pi'$ (whose parameters are $\theta'$) by
  \[ \theta' \leftarrow \theta' + \alpha \cdot \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^n, a \sim \pi(s)}[\hat{Q}^\pi(s, a) \nabla \log \pi'(a|s)] \]

- gradient = 0 at $\pi' = \pi_{Q^n} \Rightarrow$ policy iteration

- This can run into serious issues
  - Tabular PI theory assumes that we get $\hat{Q}^\pi$ that is accurate for every single state-action pair
  - Simply unrealistic if problem is complex and we can only rollout trajectories (instead of sweeping the entire state space)
  - in the middle of learning, part of the state space may be under-explored
  - at best we can hope $\hat{Q}^\pi$ to be accurate under distribution of state space we have data for
Comparing AC with Policy Iteration

- \[ \nabla J(\pi) \approx \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^n, a \sim \pi(s)}[\hat{Q}^\pi(s, a) \nabla \log \pi(a|s)] \]
- A different but related procedure: freeze \( \pi \), update the parameter of another policy \( \pi' \) (whose parameters are \( \theta' \)) by
  \[
  \theta' \leftarrow \theta' + \alpha \cdot \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^n, a \sim \pi(s)}[\hat{Q}^\pi(s, a) \nabla \log \pi'(a|s)]
  \]
- gradient = 0 at \( \pi' = \pi_{Q^n} \) => policy iteration
- This can run into serious issues
  - (cont.) if \( \pi' \) visits new states, \( \hat{Q}^\pi \) may be highly inaccurate in those states, and policy improvement no longer holds
- Perhaps better idea: move \( \pi' \) a little more but not too far from \( \pi \), so that their state occupancies are still similar.
- Theory: CPI [Kakade & Langford’02]
- Modern implementations & variants: TRPO, PPO, etc
RL Algorithms Landscape

- policy search
  - Policy Optimization
    - DFO / Evolution
    - Policy Gradients
      - Actor-Critic Methods
    - 0-th order opt.
  - Actor-Critic Methods

- value-based RL
  - Dynamic Programming
    - modified policy iteration
    - Policy Iteration
    - Value Iteration
  - Q-Learning
Practical considerations

• Recall that one way to implement PG/AC is:
  1. roll out \( \tau \sim \pi \),
  2. gradient from step \( t \): \( Q^\pi(s_t, a_t) \nabla \log \pi(a_t|s_t) \)
  3. sum up the gradients from all time steps, with weight \( \propto \gamma^{t-1} \),
• What if a trajectory length \( \gg 1/(1-\gamma) \)?
  • Most of the data points are wasted!
• Deep RL implementation in Atari games:
  • Trajectory length = \(~5\) min
  • Effective horizon = \( \approx \) secs
    \( \gamma = 0.99 \), frame rate 60Hz \( \Rightarrow \) effective horizon = \( O(1/(1-\gamma) * 1/60) \) \( = \approx \) sec
Practical considerations

• Actual implementation:
  1. roll out $\tau \sim \pi$,
  2. gradient from step $t$: $Q^\pi(s_t, a_t) \nabla \log \pi(a_t | s_t)$
  3. put equal weights on gradients from all time steps

• Pro: use all data points; Con: biased gradient.
• Is there no discounting then?
  • $Q^\pi(s_t, a_t)$ is still learned using $\gamma$ (e.g., by TD in actor-critic)
• How to understand/make sense of this?