PG Methods, Baselines, & fitted Value function methods

Lucas Janson and Sham Kakade

CS/Stat 184: Introduction to Reinforcement Learning

Fall 2022

Today

- Recap HW3 posted today

- Today:
  1. Variance Reduction w/ Baselines
  2. Advantages and a better baseline
  3. An example: PG Example with (softmax) linear policies
  4. Fitted Value Functions:
     1. Direct approach
     2. An iterative approach

Recap

The Learning Setting:

We don't know the MDP, but we can obtain trajectories.

The Finite Horizon, Learning Setting. We can obtain trajectories as follows:

- We start at $s_0 \sim \mu_0$.
- We act for H steps and observe the trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}\}$

The Infinite Horizon, Discounted Learning Setting. We can obtain trajectories as follows:

- We start at $s_0 \sim \mu_0$.
- We can obtain a "long trajectories" $\tau = \{s_0, a_0, s_1, a_1, \ldots\}$
  - Suppose we can terminate the trajectory at will.
    (and sufficient long trajectories will well approximate the discounted value function)

Note that with a simulator, we can sample trajectories as specified in the above.

Recap: Policy Parameterization

Recall that we consider parameterized policy $\pi_\theta(\cdot \mid s) \in \Delta(A), \forall s$

1. Softmax linear Policy

Feature vector $\phi(s, a) \in \mathbb{R}^d$, and parameter $\theta \in \mathbb{R}^d$

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s, a))}{\sum_{a'} \exp(\theta^\top \phi(s, a'))}$$

2. Neural Policy:

Neural network $f_\theta : S \times A \mapsto \mathbb{R}$

$$\pi_\theta(a \mid s) = \frac{\exp(f_\theta(s, a))}{\sum_{a'} \exp(f_\theta(s, a'))}$$

Recap: the REINFORCE Algorithm

(finite horizon case)

$$\tau = \{s_0, a_0, s_1, a_1, \dots, s_{H-1}, a_{H-1}\}$$

$$J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \underbrace{\left[\sum_{h=0}^{H-1} r(s_h, a_h) \right]}_{R(\tau)}$$

$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\pi_\theta(a_1 \mid s_1)\dots$$

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \right) R(\tau) \right]$$

PG with REINFORCE:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :
   1. Obtain a trajectory $\tau \sim \rho_{\theta_t}$

      Set $\widetilde{\nabla}_\theta J(\theta_t) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta_t}(a_h | s_h) R(\tau)$
   2. Update: $\theta_{t+1} = \theta_t + \eta_t \widetilde{\nabla}_\theta J(\theta_t)$

A improved PG formulation, for sampling (finite horizon setting)

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h | s_h) \right) R(\tau) \right]$$

REINFORCE

$$= \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \left(\nabla_\theta \ln \pi_\theta(a_h | s_h) \sum_{t=h}^{H-1} r_t \right) \right]$$

✓ nice exp for sampling.

$$= \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h | s_h) Q_h^{\pi_\theta}(s_h, a_h) \right]$$

"usual"

Intuition: Change action distribution at h only affects rewards later on…)

HW: You will show these simplified version are also valid PG expressions

Proof sketch

$$\mathbb{E}_{x \sim P_\theta}\left[\nabla \log P_\theta(x)\right] = 0$$

Let $f(s_0, a_0, \ldots s_{h-1}, a_{h-1}, s_h)$ be an arbitrary function.

$$\mathbb{E}_{a_h \sim \pi_\theta(\cdot|s_h)}\left[\nabla_\theta \ln \pi_\theta(a_h \,|\, s_h) f(s_0, a_0, \ldots s_{h-1}, a_{h-1}, s_h) \,\middle|\, s_0, a_0, \ldots s_{h-1}, a_{h-1}, s_h\right]$$

$$= f(s_0, a_0, \ldots s_{h-1}, a_{h-1}, s_h)\mathbb{E}_{a_h \sim \pi_\theta(\cdot|s_h)}\left[\nabla_\theta \ln \pi_\theta(a_h \,|\, s_h) \,\middle|\, s_0, a_0, \ldots s_{h-1}, a_{h-1}, s_h\right] = ??$$

An improved PG procedure:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :
    1. Obtain a trajectory $\tau \sim \rho_{\theta_t}$

       Set $\widetilde{\nabla}_\theta J(\theta_t) = \sum_{h=0}^{H-1} \left(\nabla \ln \pi_{\theta_t}(a_h \mid s_h) \sum_{t=h}^{H-1} r_t \right)$

    2. Update: $\theta_{t+1} = \theta_t + \eta_t \widetilde{\nabla}_\theta J(\theta_t)$

Today:

Policy Gradients: Baselines & Fitted Value Function Methods

Outline:

1. Variance Reduction w/ Baselines
2. Advantages and a better baseline
3. An example: PG Example with (softmax) linear policies
4. Fitted Value Functions:
    1. Direct approach
    2. An iterative approach

With a “baseline” function:

With a "baseline" function:

For any function $b_h(s)$, we have:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \left(\sum_{t=h}^{H-1} r_t - b_h(s_h) \right) \right]$$

With a "baseline" function:

For any function $b_h(s)$, we have:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h | s_h) \left(\sum_{t=h}^{H-1} r_t - b_h(s_h) \right) \right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h | s_h) \left(Q_h^{\pi_\theta}(s_h, a_h) - b_h(s_h) \right) \right]$$

With a “baseline” function:

For any function $b_h(s)$, we have:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h | s_h) \left(\sum_{t=h}^{H-1} r_t - b_h(s_h) \right) \right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h | s_h) \left(Q_h^{\pi_\theta}(s_h, a_h) - b_h(s_h) \right) \right]$$

This is (basically) the method of control variates.

(M=1) PG with a Naive (constant) Baseline:

(M=1) PG with a Naive (constant) Baseline:

- On a trajectory τ, define:

$$R_h(\tau) = \sum_{t=h}^{H-1} r_t.$$

(M=1) PG with a Naive (constant) Baseline:

- On a trajectory τ, define:
  $$R_h(\tau) = \sum_{t=h}^{H-1} r_t.$$
- Let try to use a constant (time-dependent) baseline:
  $$b_h = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} E\left[R_h(\tau)\right]$$
  (which also depends on θ)

(M=1) PG with a Naive (constant) Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$

- On a trajectory τ, define:
$$R_h(\tau) = \sum_{t=h}^{H-1} r_t.$$
- Let try to use a constant (time-dependent) baseline:
$$b_h = \mathbb{E}_{\tau \sim \rho_{\theta}(\tau)} E\left[R_h(\tau)\right]$$
(which also depends on θ)

(M=1) PG with a Naive (constant) Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :

- On a trajectory τ, define:
  $$R_h(\tau) = \sum_{t=h}^{H-1} r_t.$$
- Let try to use a constant (time-dependent) baseline:
  $$b_h = \mathbb{E}_{\tau \sim \rho_{\theta}(\tau)} E\left[R_h(\tau)\right]$$
  (which also depends on θ)

(M=1) PG with a Naive (constant) Baseline:

- On a trajectory τ, define:

$$R_h(\tau) = \sum_{t=h}^{H-1} r_t.$$

- Let try to use a constant (time-dependent) baseline:

$$b_h = \mathbb{E}_{\tau \sim \rho_{\theta}(\tau)} E\left[R_h(\tau)\right]$$

(which also depends on θ)

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :
    1. Using N trajectories sampled under π_{θ_t}, set

$$\widetilde{b}_h = \frac{1}{N} \sum_{i=1}^{N} R_h(\tau_i)$$

(M=1) PG with a Naive (constant) Baseline:

- On a trajectory τ, define:

$$R_h(\tau) = \sum_{t=h}^{H-1} r_t.$$

- Let try to use a constant (time-dependent) baseline:

$$b_h = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} E\left[R_h(\tau)\right]$$

(which also depends on θ)

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :
   1. Using N trajectories sampled under π_{θ_t}, set

$$\widetilde{b}_h = \frac{1}{N} \sum_{i=1}^{N} R_h(\tau_i)$$

don't reuse data

   2. Obtain a trajectory $\tau \sim \rho_{\theta_t}$

Set $\widetilde{\nabla}_\theta J(\theta_t) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta_t}(a_h | s_h)\left(R_h(\tau) - \widetilde{b}_h\right)$

(M=1) PG with a Naive (constant) Baseline:

- On a trajectory τ, define:
$$R_h(\tau) = \sum_{t=h}^{H-1} r_t.$$
- Let try to use a constant (time-dependent) baseline:
$$b_h = \mathbb{E}_{\tau\sim\rho_\theta(\tau)} E\left[R_h(\tau)\right]$$
(which also depends on θ)

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :
    1. Using N trajectories sampled under π_{θ_t}, set
    $$\widetilde{b}_h = \frac{1}{N}\sum_{i=1}^{N} R_h(\tau_i)$$
    2. Obtain a trajectory $\tau \sim \rho_{\theta_t}$
    Set $\widetilde{\nabla}_\theta J(\theta_t) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta_t}(a_h | s_h)\left(R_h(\tau) - \widetilde{b}_h\right)$
    3. Update: $\theta_{t+1} = \theta_t + \eta_t \widetilde{\nabla}_\theta J(\theta_t)$

Outline:

1. Variance Reduction w/ Baselines
2. Advantages and a better baseline
3. An example: PG Example with (softmax) linear policies
4. Fitted Value Functions:
   1. Direct approach
   2. An iterative approach

The Advantage Function (finite horizon)

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r(s_\tau, a_\tau) \,\middle|\, \overset{\pi}{s_h} = s\right] \qquad Q_h^\pi(s,a) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r(s_\tau, a_\tau) \,\middle|\, (s_h, a_h) \overset{\pi}{=} (s,a)\right]$$

The Advantage Function (finite horizon)

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r(s_\tau, a_\tau) \middle| s_h = s\right] \qquad Q_h^\pi(s,a) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r(s_\tau, a_\tau) \middle| (s_h, a_h) = (s,a)\right]$$

- The Advantage function is defined as:

$$A_h^\pi(s,a) = Q_h^\pi(s,a) - V_h^\pi(s)$$

The Advantage Function (finite horizon)

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r(s_\tau, a_\tau) \,\middle|\, s_h = s\right] \qquad Q_h^\pi(s,a) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r(s_\tau, a_\tau) \,\middle|\, (s_h, a_h) = (s,a)\right]$$

$\cancel{E[\quad]}$

- The Advantage function is defined as:

$$A_h^\pi(s,a) = Q_h^\pi(s,a) - V_h^\pi(s)$$

$$= E_{a\sim\pi}\left[Q^\pi(s,a)\right] - V^\pi(s)$$

$$= V^\pi(s) - V^\pi(s) = 0$$

- We have that:

$$E_{a\sim\pi(\cdot|s)}\left[A_h^\pi(s,a) \,\middle|\, s,h\right] = \sum_a \pi(a\,|\,s) A_h^\pi(s,a) = \;?? \quad 0$$

The Advantage Function (finite horizon)

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r(s_\tau, a_\tau) \middle| s_h = s\right] \qquad Q_h^\pi(s,a) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r(s_\tau, a_\tau) \middle| (s_h, a_h) = (s,a)\right]$$

$$V^\star(s) = \max_a Q^\star(s,a)$$

- The Advantage function is defined as:

$$A_h^\pi(s,a) = Q_h^\pi(s,a) - V_h^\pi(s)$$

- We have that:

$$E_{a\sim\pi(\cdot|s)}\left[A_h(s,a) \middle| s, h\right] = \sum_a \pi(a\,|\,s)A_h(s,a) = \;\; ??$$

$\forall s, a, h$

$A_h^\star(s,a) \le 0$

- What do we know about $A_h^{\pi^\star}(s,a)$?

The Advantage Function (finite horizon)

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r(s_\tau, a_\tau) \middle| s_h = s\right] \qquad Q_h^\pi(s,a) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r(s_\tau, a_\tau) \middle| (s_h, a_h) = (s,a)\right]$$

- The Advantage function is defined as:

$$A_h^\pi(s,a) = Q_h^\pi(s,a) - V_h^\pi(s)$$

- We have that:

$$E_{a\sim\pi(\cdot|s)}\left[A_h(s,a) \middle| s, h\right] = \sum_a \pi(a\,|\,s)A_h(s,a) = \ ??$$

- What do we know about $A_h^{\pi^\star}(s,a)$?
- For the discounted case, $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$

The Advantage-based PG:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \Big(Q_h^{\pi_\theta}(s_h, a_h) - b_h(s_h) \Big) \right]$$

The Advantage-based PG:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \left(Q_h^{\pi_\theta}(s_h, a_h) - b_h(s_h) \right) \right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) A_h^{\pi_\theta}(s_h, a_h) \right]$$

$b_h(s)$

$= V_h^{\pi_\theta}(s)$

- The second step follows by choosing $b_h(s) = V_h^\pi(s)$.
- In practice, the most common approach is to use $b_h(s)$ to approximate $V_h^\pi(s)$.

Outline:

1. Variance Reduction w/ Baselines
2. Advantages and a better baseline
3. An example: PG Example with (softmax) linear policies
4. Fitted Value Functions:
    1. Direct approach
    2. An iterative approach

Policy Parameterizations

Recall that we consider parameterized policy $\pi_\theta(\cdot \mid s) \in \Delta(A), \forall s$

1. Softmax linear Policy

Feature vector $\phi(s, a) \in \mathbb{R}^d$, and parameter $\theta \in \mathbb{R}^d$

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s, a))}{\sum_{a'} \exp(\theta^\top \phi(s, a'))}$$

2. Neural Policy:

Neural network $f_\theta : S \times A \mapsto \mathbb{R}$

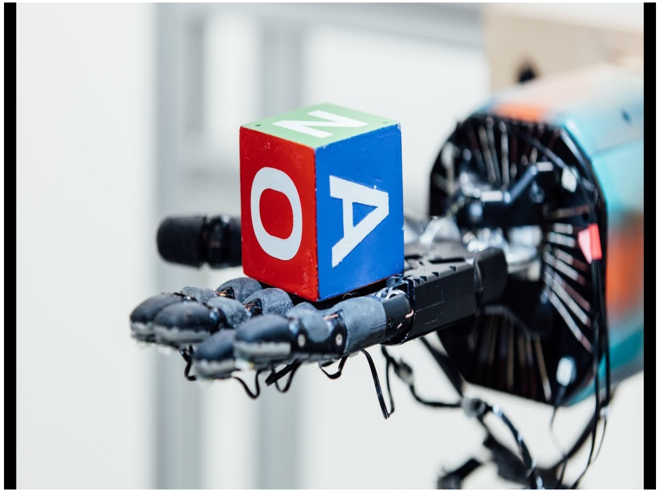
$$\pi_\theta(a \mid s) = \frac{\exp(f_\theta(s, a))}{\sum_{a'} \exp(f_\theta(s, a'))}$$

What is a “state” and a “feature vector”?

ϕ, h: hidden

[AlphaZero, Silver

[OpenAI Five,



[OpenAI,

robotics

$\phi(s,a) \in \mathbb{R}^d \quad \phi = (\vec{s}, \vec{a}, \vec{a}\otimes\vec{s}, \vec{s}\otimes\vec{s}, \vec{a}\otimes\vec{a}, \ldots)$

A state:

- Tabular case: an index in $[|S|] = \{1,\ldots|S|\}$
- Real world: a list/array of the relevant info about the world that makes the process Markovian. (we sometimes append history info into the current state)
- Let’s assume the current time h is contained in the state. (e.g. you can always add the time into the “list” that specifies the state)

Softmax Policy Properties

Recall that we consider parameterized policy $\pi_\theta(\cdot \mid s) \in \Delta(A), \forall s$

1. Softmax linear Policy

Feature vector $\phi(s, a) \in \mathbb{R}^d$, and parameter $\theta \in \mathbb{R}^d$

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s, a))}{\sum_{a'} \exp(\theta^\top \phi(s, a'))}$$

Softmax Policy Properties

Recall that we consider parameterized policy $\pi_\theta(\cdot \mid s) \in \Delta(A), \forall s$

1. Softmax linear Policy

Feature vector $\phi(s, a) \in \mathbb{R}^d$, and parameter $\theta \in \mathbb{R}^d$

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s, a))}{\sum_{a'} \exp(\theta^\top \phi(s, a'))}$$

Two properties (see HW):

Softmax Policy Properties

Recall that we consider parameterized policy $\pi_\theta(\cdot \mid s) \in \Delta(A), \forall s$

1. Softmax linear Policy

Feature vector $\phi(s, a) \in \mathbb{R}^d$, and parameter $\theta \in \mathbb{R}^d$

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s, a))}{\sum_{a'} \exp(\theta^\top \phi(s, a'))}$$

Two properties (see HW):

- More probable actions have features which align with θ. Precisely,
  $\pi_\theta(a \mid s) \geq \pi_\theta(a' \mid s)$ if and only if $\theta^\top \phi(s, a) \geq \theta^\top \phi(s, a')$

Softmax Policy Properties

Recall that we consider parameterized policy $\pi_\theta(\cdot \mid s) \in \Delta(A), \forall s$

1. Softmax linear Policy

Feature vector $\phi(s,a) \in \mathbb{R}^d$, and parameter $\theta \in \mathbb{R}^d$

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

Neural case

Two properties (see HW):

- More probable actions have features which align with θ. Precisely,

  $\pi_\theta(a \mid s) \geq \pi_\theta(a' \mid s)$ if and only if $\theta^\top \phi(s,a) \geq \theta^\top \phi(s,a')$

  please check

- The gradient is:

  $\nabla_\theta \log(\pi_\theta(a \mid s)) = \phi(s,a) - \mathbb{E}_{a' \sim \pi_\theta(\cdot \mid s)}[\phi(s,a')]$

$$\nabla_\theta \log \pi_\theta(a|s) = \nabla f_\theta(s,a) - \mathbb{E}_{a' \sim \pi(\cdot|s)}[\nabla f_\theta(s,a')]$$

PG for the (softmax) linear policies

PG for the (softmax) linear policies

- We have:

$$\nabla J(\theta) = \mathbb{E}_{\tau\sim\rho_\theta(\tau)}\left[\sum_{h=0}^{H-1} \overset{\substack{Q^{\pi}_{h} \\ \updownarrow}}{A_h^{\pi_\theta}}(s_h,a_h)\left(\overbrace{\phi(s_h,a_h) - \mathbb{E}_{a'\sim\pi_\theta(\cdot|s_h)}[\phi(s_h,a')]}^{\nabla_\theta \log \pi_\theta(s,a)}\right)\right]$$

(also true Q_h instead of A_h)

PG for the (softmax) linear policies

- We have:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} A_h^{\pi_\theta}(s_h, a_h) \Big(\phi(s_h, a_h) - \mathbb{E}_{a' \sim \pi_\theta(\cdot|s_h)}[\phi(s_h, a')] \Big) \right]$$

(also true Q_h instead of A_h)

- We can simplify this to:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} A_h^{\pi_\theta}(s_h, a_h) \phi(s_h, a_h) \right]$$

PG for the (softmax) linear policies

- We have:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} A_h^{\pi_\theta}(s_h, a_h) \Big(\phi(s_h, a_h) - \mathbb{E}_{a' \sim \pi_\theta(\cdot|s_h)}[\phi(s_h, a')] \Big) \right]$$

(also true Q_h instead of A_h)

- We can simplify this to:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} A_h^{\pi_\theta}(s_h, a_h) \phi(s_h, a_h) \right]$$

- Why?

$$\sum_h \mathbb{E}_\tau \left[A^{\pi_\theta}(s_h, a_h) \mathbb{E}_{a' \sim \pi} [\phi(s_h, a')] \right]$$

$$\mathbb{E}_\tau \left[A^\theta(s_h, a_h) \mathbb{E}_{a' \sim \pi_\theta(\cdot|s_h)} [\phi(s_h, a')] \right]$$

constant

$$\mathbb{E}_\tau \left[\mathbb{E}_{a_h \sim \pi(\cdot|s_h)} [A^\theta(s_h, a_h)] \cdot \mathbb{E}_{a' \sim \pi_\theta(\cdot|s_h)} [\phi(s_h, a')] \right] = 0$$

PG for the (softmax) linear policies

- We have:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} A_h^{\pi_\theta}(s_h, a_h) \Big(\phi(s_h, a_h) - \mathbb{E}_{a' \sim \pi_\theta(\cdot|s_h)}[\phi(s_h, a')] \Big) \right]$$

(also true Q_h instead of A_h)

- We can simplify this to:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta(\tau)} \left[\sum_{h=0}^{H-1} A_h^{\pi_\theta}(s_h, a_h) \phi(s_h, a_h) \right]$$

- Why?

Neural case $\quad \nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta} \left[\sum_h A_h^\theta(s_h, a_h) \nabla f_\theta(s_h, a_h) \right]$

Outline:

1. Variance Reduction w/ Baselines
2. Advantages and a better baseline
3. An example: PG Example with (softmax) linear policies
4. Fitted Value Functions:
    1. Direct approach
    2. An iterative approach

(M=1) PG with a Learned Baseline:

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :
    1. Using N trajectories sampled under π_{θ_t}, try to learn a $\widetilde{b}_h$
       $\widetilde{b}(s) \approx V_h^{\pi_{\theta_t}}(s)$

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :
    1. Using N trajectories sampled under π_{θ_t}, try to learn a $\widetilde{b}_h$
       $\widetilde{b}(s) \approx V_h^{\pi_{\theta_t}}(s)$
    2. Obtain a trajectory $\tau \sim \rho_{\theta_t}$

       Set $\widetilde{\nabla}_\theta J(\theta_t) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta_t}(a_h \mid s_h) \left(R_h(\tau) - \widetilde{b}(s_h) \right)$

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :
    1. Using N trajectories sampled under π_{θ_t}, try to learn a $\widetilde{b}_h$

       $\widetilde{b}(s) \approx V_h^{\pi_{\theta_t}}(s)$
    2. Obtain a trajectory $\tau \sim \rho_{\theta_t}$

       Set $\widetilde{\nabla}_\theta J(\theta_t) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta_t}(a_h | s_h)\Big(R_h(\tau) - \widetilde{b}(s_h)\Big)$
    3. Update: $\theta_{t+1} = \theta_t + \eta_t \widetilde{\nabla}_\theta J(\theta_t)$

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :
    1. Using N trajectories sampled under π_{θ_t}, try to learn a $\widetilde{b}_h$
       $\widetilde{b}(s) \approx V_h^{\pi_{\theta_t}}(s)$
    2. Obtain a trajectory $\tau \sim \rho_{\theta_t}$

       Set $\widetilde{\nabla}_\theta J(\theta_t) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta_t}(a_h | s_h) \left(R_h(\tau) - \widetilde{b}(s_h) \right)$

    3. Update: $\theta_{t+1} = \theta_t + \eta_t \widetilde{\nabla}_\theta J(\theta_t)$

Note that regardless of our choice of $\widetilde{b}_h(s)$, we still get unbiased gradient estimates.

(M=1) PG with a Learned Baseline:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :

Now let's look at our baseline fitting step.

   1. Using N trajectories sampled under π_{θ_t}, try to learn a $\widetilde{b}_h$
      $\widetilde{b}(s) \approx V_h^{\pi_{\theta_t}}(s)$
   2. Obtain a trajectory $\tau \sim \rho_{\theta_t}$
      Set $\widetilde{\nabla}_\theta J(\theta_t) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta_t}(a_h | s_h) \left(R_h(\tau) - \widetilde{b}(s_h) \right)$
   3. Update: $\theta_{t+1} = \theta_t + \eta_t \widetilde{\nabla}_\theta J(\theta_t)$

Note that regardless of our choice of $\widetilde{b}_h(s)$, we still get unbiased gradient estimates.

Baseline/Value Function Parameterizations

Now let us consider parameterized classes of functions $\mathcal{F}$, where for each $f \in \mathcal{F}$, $f : S \to R$

1. Linear Functions

Feature vector $\psi(s) \in \mathbb{R}^k$, and parameter $w \in \mathbb{R}^k$

$$f_w(s) = w^\top \psi(s)$$

2. Neural Policy:

Neural network $f_w : S \mapsto \mathbb{R}$

find w s.t.

$$f_w(s) \approx V_h^{\pi}(s)$$

“Review”

"Review"

- For a random variable $y \in R$, what is:

$$\arg\min_{c} E_{y \sim D}[(c - y)^2] = ??$$

$c = E[y]$

“Review”

- For a random variable $y \in R$, what is:

$$\arg\min_c E_{y \sim D}[(c - y)^2] = ??$$

$y \in R.$

- Now let us look at the “function” case where we have a distribution over (x, y) pairs

$$f^\star = \arg\min_{f \in \mathcal{F}} E_{(x,y) \sim D}[(f(x) - y)^2]$$

(where $\mathcal{F}$ is the class of all possible functions)

What is $f^\star(x) = ??$ $E[y/x]$

"Review"

- For a random variable $y \in R$, what is:

$$\arg\min_{c} E_{y\sim D}[(c-y)^2] = ??$$

- Now let us look at the "function" case where we have a distribution over (x, y) pairs

$$f^\star = \arg\min_{f\in\mathscr{F}} E_{(x,y)\sim D}[(f(x)-y)^2]$$

$$f_w(s) \approx V_h^\pi(s)$$

(where $\mathscr{F}$ is the class of all possible functions)

What is $f^\star(x) = ??$

$x \longleftrightarrow$ state

$y \longleftrightarrow R_h(\tau)$

$$E[R_h(\tau) \mid \pi, s_h] = V_h^\pi(s_h)$$

Let’s look at our fitting step

Let’s look at our fitting step

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$

Let's look at our fitting step

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \dots$
2. For t = 0, … :

Let's look at our fitting step

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :
   1. Sample N trajectories under π_{θ_t} to make a dataset,

$$\widetilde{w} = \arg\min_{w} \sum_{\tau \in \text{Data}} \sum_{(s_h, a_h) \in \tau} \left(f_w(s_h) - R_h(\tau) \right)^2$$

Let's look at our fitting step

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \dots$
2. For t = 0, … :
    1. Sample N trajectories under π_{θ_t} to make a dataset,

       $$\widetilde{w} = \arg\min_w \sum_{\tau \in \text{Data}} \sum_{(s_h, a_h) \in \tau} \left(f_w(s_h) - R_h(\tau) \right)^2$$

    2. Obtain a trajectory $\tau \sim \rho_{\theta_t}$

       Set $\widetilde{\nabla}_\theta J(\theta_t) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta_t}(a_h | s_h) \left(R_h(\tau) - \widetilde{b}_h \right)$

Let's look at our fitting step

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :
    1. Sample N trajectories under π_{θ_t} to make a dataset,
    $$\widetilde{w} = \arg\min_w \sum_{\tau \in \text{Data}} \sum_{(s_h, a_h) \in \tau} \left(f_w(s_h) - R_h(\tau) \right)^2$$
    2. Obtain a trajectory $\tau \sim \rho_{\theta_t}$
    Set $\widetilde{\nabla}_\theta J(\theta_t) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta_t}(a_h | s_h) \left(R_h(\tau) - \widetilde{b}_h \right)$
    3. Update: $\theta_{t+1} = \theta_t + \eta_t \widetilde{\nabla}_\theta J(\theta_t)$

Outline:

1. Variance Reduction w/ Baselines
2. Advantages and a better baseline
3. An example: PG Example with (softmax) linear policies
4. Fitted Value Functions:
    1. Direct approach
    2. An iterative approach

Is there an iterative version of Policy Evaluation?

(that is faster, but approximate?)

Algorithm (Iterative PE):

1. Initialization: $V^0 : \|V^0\|_\infty \in \left[0, \frac{1}{1-\gamma}\right]$
2. Iterate until convergence: $V^{t+1} \leftarrow R + \gamma P V^t$

Is there an iterative version of Policy Evaluation?

(that is faster, but approximate?)

Algorithm (Iterative PE):

1. Initialization: $V^0 : \|V^0\|_\infty \in \left[0, \frac{1}{1-\gamma}\right]$
2. Iterate until convergence: $V^{t+1} \leftarrow R + \gamma P V^t$

This is a "fixed point" algorithm trying to enforce Bellman consistency:

$$\forall s, V^\pi(s) = r(s, \pi(s)) + \gamma \mathbb{E}_{s \sim P(s,\pi(s))} V^\pi(s')$$

Let's look at our fitting step

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For t = 0, … :
    1. Using N trajectories sampled under π_{θ_t}, try to learn a $\widetilde{b}_h$
       $\widetilde{b}_h(s) \approx V_h^{\pi_{\theta_t}}(s)$
    2. Obtain a trajectory $\tau \sim \rho_{\theta_t}$

       Set $\widetilde{\nabla}_\theta J(\theta_t) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta_t}(a_h | s_h) \left(R_h(\tau) - \widetilde{b}_h \right)$
    3. Update: $\theta_{t+1} = \theta_t + \eta_t \widetilde{\nabla}_\theta J(\theta_t)$

Let's look at just our fitting step in the inner loop (where we want to fit the value of π_{θ_t})

Temporal Difference Learning (TD) is a an online method to do the above.

Let's look at just our fitting step in the inner loop (where we want to fit the value of π_{θ_t})

1. Sample N trajectories under π_{θ_t} to make a dataset.

Temporal Difference Learning (TD) is a an online method to do the above.

Let's look at just our fitting step in the inner loop (where we want to fit the value of π_{θ_t})

1. Sample N trajectories under π_{θ_t} to make a dataset.
2. Initialize w_0

Temporal Difference Learning (TD) is a an online method to do the above.

Let's look at just our fitting step in the inner loop (where we want to fit the value of π_{θ_t})

1. Sample N trajectories under π_{θ_t} to make a dataset.
2. Initialize w_0
3. For k = $0, \ldots, K$:

Temporal Difference Learning (TD) is a an online method to do the above.

Let's look at just our fitting step in the inner loop (where we want to fit the value of π_{θ_t})

1. Sample N trajectories under π_{θ_t} to make a dataset.
2. Initialize w_0
3. For k = $0,\ldots,K$:
    1. Update:

$$w_{k+1} = \arg\min_{w} \sum_{\tau \in \text{Data}} \sum_{(s_h,a_h)\in\tau} \Big(f_w(s_h) - \big(r_h + f_{w_k}(s_{h+1}) \big) \Big)^2$$

Temporal Difference Learning (TD) is a an online method to do the above.

Summary so far:

1. Variance Reduction w/ Baselines & Advantages.
2. An example: PG Example with (softmax) linear policies
3. Fitted Value Functions:
    1. Direct approach
    2. An iterative approach & TD

Next up: Why not just directly use a “fitted” approach for Value Iteration or Policy Iteration?

1-minute feedback form: https://bit.ly/3RHtIxy