Lecture 7: Policy Gradient I

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2023

Additional reading: Sutton and Barto 2018 Chp. 13

1With many slides from or derived from David Silver and John Schulman and Pieter Abbeel
Consider doing experience replay over a finite, but extremely large, set of \((s, a, r, s')\) tuples. Q-learning is initialized to 0 everywhere and all rewards are positive. Select all that are true.

1. Assume all tuples were gathered from a fixed, deterministic policy \(\pi\). Then in the tabular setting, if each tuple is sampled at random and used to do a Q-learning update, and this is repeated an infinite number of times, then there exists a learning rate schedule so that the resulting estimate will converge to the true \(Q^\pi\).

2. In situation (1) (the first option above) the resulting Q estimate will be identical to if one computed an estimated dynamics model and reward model using maximum likelihood evaluation from the tuples, and performed policy evaluation using the estimated dynamics and reward models.

3. If one uses DQN to populate the experience replay set of tuples, then doing experience replay with DQN is always guaranteed to converge to the optimal Q function.

4. Not sure
Consider doing experience replay over a finite, but extremely large, set of \((s,a,r,s')\) tuples. Q-learning is initialized to 0 everywhere and all rewards are positive. Select all that are true

1. Assume all tuples were gathered from a fixed, deterministic policy \(\pi\). Then in the tabular setting, if each tuple is sampled at random and used to do a Q-learning update, and this is repeated an infinite number of times, then there exists a learning rate schedule so that the resulting estimate will converge to the true \(Q^\pi\).

2. In situation (1) (the first option above) the resulting Q estimate will be identical to if one computed an estimated dynamics model and reward model using maximum likelihood evaluation from the tuples, and performed policy evaluation using the estimated dynamics and reward models.

3. If one uses DQN to populate the experience replay set of tuples, then doing experience replay with DQN is always guaranteed to converge to the optimal Q function.

4. Not sure

Answer: 1 is true and 2 is true. 3 is false.
RL Algorithms Involve

- Optimization
- Delayed consequences
- Exploration
- Generalization
- And statistical and computational efficiency matters
Can use structure and additional knowledge to help constrain and speed reinforcement learning.
Last time: Deep RL

This time: Policy Search

Next time: Policy Search Cont.
Policy-Based Reinforcement Learning

- In the last lecture we approximated the value or action-value function using parameters $w$, 
  \[ V_w(s) \approx V^\pi(s) \]
  \[ Q_w(s, a) \approx Q^\pi(s, a) \]

- A policy was generated directly from the value function
  - e.g. using $\epsilon$-greedy

- In this lecture we will directly parametrize the policy, and will typically use $\theta$ to show parameterization:
  \[ \pi_\theta(s, a) = \mathbb{P}[a|s; \theta] \]

- Goal is to find a policy $\pi$ with the highest value function $V^\pi$

- We will focus again on model-free reinforcement learning
Value-Based and Policy-Based RL

- **Value Based**
  - learned Value Function
  - Implicit policy (e.g. $\epsilon$-greedy)

- **Policy Based**
  - No Value Function
  - Learned Policy

- **Actor-Critic**
  - Learned Value Function
  - Learned Policy

![Diagram showing Value Function and Policy](image)
Types of Policies to Search Over

- So far have focused on deterministic policies or $\epsilon$-greedy policies
- Now we are thinking about direct policy search in RL, will focus heavily on stochastic policies
Example: Rock-Paper-Scissors

- Two-player game of rock-paper-scissors
  - Scissors beats paper
  - Rock beats scissors
  - Paper beats rock
- Let state be history of prior actions (rock, paper and scissors) and if won or lost
- Is deterministic policy optimal? Why or why not?
Example: Rock-Paper-Scissors, Vote

- Two-player game of rock-paper-scissors
  - Scissors beats paper
  - Rock beats scissors
  - Paper beats rock
- Let state be history of prior actions (rock, paper and scissors) and if won or lost
  Deterministic policy is easily exploited by an adversary. System is not Markov. A uniform random policy is optimal (Nash equilibrium).
Example: Aliased Gridword (1)

- The agent cannot differentiate the grey states
- Consider features of the following form (for all N, E, S, W)
  \[ \phi(s, a) = 1 \text{(wall to N, } a = \text{move E)} \]
- Compare value-based RL, using an approximate value function
  \[ Q_\theta(s, a) = f(\phi(s, a); \theta) \]
- To policy-based RL, using a parametrized policy
  \[ \pi_\theta(s, a) = g(\phi(s, a); \theta) \]
Under aliasing, an optimal deterministic policy will either
- move W in both grey states (shown by red arrows)
- move E in both grey states

Either way, it can get stuck and never reach the money

Value-based RL learns a near-deterministic policy
- e.g. greedy or $\epsilon$-greedy

So it will traverse the corridor for a long time
An optimal stochastic policy will randomly move E or W in grey states

\[ \pi_\theta(\text{wall to N and S, move E}) = 0.5 \]

\[ \pi_\theta(\text{wall to N and S, move W}) = 0.5 \]

- It will reach the goal state in a few steps with high probability
- Policy-based RL can learn the optimal stochastic policy
Policy Objective Functions

- Goal: given a policy $\pi_\theta(s, a)$ with parameters $\theta$, find best $\theta$
- But how do we measure the quality for a policy $\pi_\theta$?
- In episodic environments can use policy value at start state $V(s_0, \theta)$
- For simplicity, today will mostly discuss the episodic case, but can easily extend to the continuing / infinite horizon case
Policy based reinforcement learning is an optimization problem

Find policy parameters $\theta$ that maximize $V(s_0, \theta)$
Policy optimization

- Policy based reinforcement learning is an **optimization** problem
- Find policy parameters $\theta$ that maximize $V(s_0, \theta)$
- Can use gradient free optimization
  - Hill climbing
  - Simplex / amoeba / Nelder Mead
  - Genetic algorithms
  - Cross-Entropy method (CEM)
  - Covariance Matrix Adaptation (CMA)
Human-in-the-Loop Exoskeleton Optimization (Zhang et al. Science 2017)

- Optimization was done using CMA-ES, variation of covariance matrix evaluation

**Figure:** Zhang et al. Science 2017
Gradient Free Policy Optimization

- Can often work embarrassingly well: "discovered that evolution strategies (ES), an optimization technique that’s been known for decades, rivals the performance of standard reinforcement learning (RL) techniques on modern RL benchmarks (e.g. Atari/MuJoCo)" (https://blog.openai.com/evolution-strategies/)
Gradient Free Policy Optimization

- Often a great simple baseline to try

**Benefits**
- Can work with any policy parameterizations, including non-differentiable
- Frequently very easy to parallelize

**Limitations**
- Often less sample efficient because it ignores temporal structure
Policy based reinforcement learning is an optimization problem
Find policy parameters $\theta$ that maximize $V(s_0, \theta)$
Can use gradient free optimization:
Greater efficiency often possible using gradient
  - Gradient descent
  - Conjugate gradient
  - Quasi-newton
We focus on gradient descent, many extensions possible
And on methods that exploit sequential structure
Define \( V(\theta) = V(s_0, \theta) \) to make explicit the dependence of the value on the policy parameters [but don’t confuse with value function approximation, where parameterized value function]

Assume episodic MDPs (easy to extend to related objectives, like average reward)
Define $V^{\pi_\theta} = V(s_0, \theta)$ to make explicit the dependence of the value on the policy parameters.

Assume episodic MDPs.

Policy gradient algorithms search for a \textit{local} maximum in $V(s_0, \theta)$ by ascending the gradient of the policy, w.r.t parameters $\theta$

$$\Delta \theta = \alpha \nabla_{\theta} V(s_0, \theta)$$

Where $\nabla_{\theta} V(s_0, \theta)$ is the \textit{policy gradient}

$$\nabla_{\theta} V(s_0, \theta) = \left( \begin{array}{c} \frac{\partial V(s_0, \theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial V(s_0, \theta)}{\partial \theta_n} \end{array} \right)$$

and $\alpha$ is a step-size parameter.
Example: Training AIBO to Walk by Finite Difference Policy Gradient

- Goal: learn a fast AIBO walk (useful for Robocup)
- Adapt these parameters by finite difference policy gradient
- Evaluate performance of policy by field traversal time

---

Advantages:
- Better convergence properties
- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies

Disadvantages:
- Typically converge to a local rather than global optimum
- Evaluating a policy is typically inefficient and high variance

Shortly will see some ideas to help with this last limitation
Score functions and Policy Gradient

- Differentiable Policies
- Temporal Structure
- Baseline
- Alternatives to MC Returns
Computing the gradient analytically

- We now compute the policy gradient \textit{analytically}.
- Assume policy \( \pi_\theta \) is \textit{differentiable whenever it is non-zero}.
- Assume we can calculate gradient \( \nabla_\theta \pi_\theta(s, a) \) analytically.
- What kinds of policy classes can we do this for?
Many choices of differentiable policy classes including:
- Softmax
- Gaussian
- Neural networks
Value of a Parameterized Policy

- Now assume policy $\pi_\theta$ is differentiable whenever it is non-zero and we know the gradient $\nabla_\theta \pi_\theta(s, a)$
- Recall policy value is $V(s_0, \theta) = \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{T} R(s_t, a_t); \pi_\theta, s_0 \right]$ where the expectation is taken over the states & actions visited by $\pi_\theta$
- We can re-express this in multiple ways
  - $V(s_0, \theta) = \sum_a \pi_\theta(a|s_0) Q(s_0, a, \theta)$
  - $V(s_0, \theta) = \sum_{\tau} P(\tau, \theta) R(\tau)$
Value of a Parameterized Policy

- Now assume policy $\pi_\theta$ is differentiable whenever it is non-zero and we know the gradient $\nabla_\theta \pi_\theta(s, a)$
- Recall policy value is $V(s_0, \theta) = \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{T} R(s_t, a_t); \pi_\theta, s_0 \right]$ where the expectation is taken over the states & actions visited by $\pi_\theta$
- We can re-express this in multiple ways
  - $V(s_0, \theta) = \sum_a \pi_\theta(a|s_0) Q(s_0, a, \theta)$
  - $V(s_0, \theta) = \sum_\tau P(\tau; \theta) R(\tau)$
    - where $\tau = (s_0, a_0, r_0, ..., s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ is a state-action trajectory,
    - $P(\tau; \theta)$ is used to denote the probability over trajectories when executing policy $\pi(\theta)$ starting in state $s_0$, and
    - $R(\tau) = \sum_{t=0}^{T} R(s_t, a_t)$ the sum of rewards for a trajectory $\tau$
- To start will focus on this latter definition. See Chp 13.1-13.3 of SB for a nice discussion starting with the other definition
Likelihood Ratio Policies

- Denote a state-action trajectory as 
  \[ \tau = (s_0, a_0, r_0, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T) \]
- Use \( R(\tau) = \sum_{t=0}^{T} R(s_t, a_t) \) to be the sum of rewards for a trajectory \( \tau \)
- Policy value is 
  \[
  V(\theta) = \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{T} R(s_t, a_t); \pi_\theta \right] = \sum_{\tau} P(\tau; \theta) R(\tau)
  \]
  where \( P(\tau; \theta) \) is used to denote the probability over trajectories when executing policy \( \pi(\theta) \)
- In this new notation, our goal is to find the policy parameters \( \theta \):
  \[
  \arg \max_{\theta} V(\theta) = \arg \max_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)
  \]
• Goal is to find the policy parameters $\theta$:

$$\arg \max_{\theta} V(\theta) = \arg \max_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

• Take the gradient with respect to $\theta$:

$$\nabla_{\theta} V(\theta) = \nabla_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

$$= \sum_{\tau} \nabla_{\theta} P(\tau; \theta) R(\tau)$$

$$= \sum_{\tau} R(\tau) \frac{\nabla_{\theta} P(\tau; \theta)}{P(\tau; \theta)}$$

$$= \sum_{\tau} R(\tau) P(\tau; \theta) \nabla_{\theta} \log P(\tau; \theta)$$
Likelihood Ratio Policy Gradient

- Goal is to find the policy parameters $\theta$:
  \[
  \arg \max_\theta V(\theta) = \arg \max_\theta \sum_\tau P(\tau; \theta)R(\tau)
  \]

- Take the gradient with respect to $\theta$:
  \[
  \nabla_\theta V(\theta) = \nabla_\theta \sum_\tau P(\tau; \theta)R(\tau)
  \]
  \[
  = \sum_\tau \nabla_\theta P(\tau; \theta)R(\tau)
  \]
  \[
  = \sum_\tau \frac{P(\tau; \theta)}{P(\tau; \theta)} \nabla_\theta P(\tau; \theta)R(\tau)
  \]
  \[
  = \sum_\tau P(\tau; \theta)R(\tau) \frac{\nabla_\theta P(\tau; \theta)}{P(\tau; \theta)}
  \]
  \[
  = \sum_\tau P(\tau; \theta)R(\tau) \nabla_\theta \log P(\tau; \theta)
  \]
Goal is to find the policy parameters $\theta$:

$$\arg \max_\theta V(\theta) = \arg \max_\theta \sum_\tau P(\tau; \theta)R(\tau)$$

Take the gradient with respect to $\theta$:

$$\nabla_\theta V(\theta) = \sum_\tau P(\tau; \theta)R(\tau)\nabla_\theta \log P(\tau; \theta)$$

Approximate with empirical estimate for $m$ sample trajectories under policy $\pi_\theta$:

$$\nabla_\theta V(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^m R(\tau^{(i)})\nabla_\theta \log P(\tau^{(i)}; \theta)$$
Decomposing the Trajectories Into States and Actions

- Approximate with empirical estimate for $m$ sample paths under policy $\pi_\theta$:

$$\nabla_\theta V(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} R(\tau^{(i)}) \nabla_\theta \log P(\tau^{(i)})$$

$$\nabla_\theta \log P(\tau^{(i)}; \theta) = \nabla_\Theta \log \left[ \mu(s_0) \prod_{t=0}^{T} \pi_\Theta(a_t | s_t) \right] p(s_{t+1} | s_t, a_t)$$

$$= \nabla_\Theta \log \mu(s_0) + \sum_{t=0}^{T} \log \pi_\Theta(a_t | s_t) + \log p(s_{t+1} | s_t, a_t)$$

$$\nabla_\Theta \log \mu(s_0)$$

$$= \sum_{t=0}^{T-1} \nabla_\Theta \log \pi_\Theta(a_t | s_t)$$
Decomposing the Trajectories Into States and Actions

- Approximate with empirical estimate for \( m \) sample paths under policy \( \pi_\theta \):

\[
\nabla_\theta V(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} R(\tau^{(i)}) \nabla_\theta \log P(\tau^{(i)})
\]

\[
\nabla_\theta \log P(\tau^{(i)}; \theta) = \nabla_\theta \log \left[ \underbrace{\mu(s_0)}_{\text{Initial state distrib.}} \prod_{t=0}^{T-1} \underbrace{\pi_\theta(a_t | s_t)}_{\text{policy}} \underbrace{P(s_{t+1} | s_t, a_t)}_{\text{dynamics model}} \right]
\]

\[
= \nabla_\theta \left[ \log \mu(s_0) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t | s_t) + \log P(s_{t+1} | s_t, a_t) \right]
\]

\[
= \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t | s_t)
\]

no dynamics model required!
Decomposing the Trajectories Into States and Actions

Approximate with empirical estimate for $m$ sample paths under policy $\pi_\theta$:

$$\nabla_\theta V(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} R(\tau^{(i)}) \nabla_\theta \log P(\tau^{(i)})$$

$$\nabla_\theta \log P(\tau^{(i)}; \theta) = \nabla_\theta \log \left[ \mu(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t) \right]$$

$$= \nabla_\theta \left[ \log \mu(s_0) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t) + \log P(s_{t+1}|s_t, a_t) \right]$$

$$= \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Initial state distrib.  \hspace{1cm} \text{policy}  \hspace{1cm} \text{dynamics model}
A score function is the derivative of the log of a parameterized probability / likelihood.

Example: let $\pi(s; \theta)$ be the probability of state $s$ under parameter $\theta$.

Then the score function would be

$$\nabla_\theta \log \pi(s; \theta)$$ (1)

For many policy classes, it is not hard to compute the score function.
Softmax Policy

- Weight actions using linear combination of features $\phi(s, a)^T \theta$
- Probability of action is proportional to exponentiated weight

$$\pi_\theta(s, a) = \frac{e^{\phi(s, a)^T \theta}}{\sum_a e^{\phi(s, a)^T \theta}}$$

Made error in live Lecture, purple derivation is correct
Softmax Policy

- Weight actions using linear combination of features $\phi(s, a)^T \theta$
- Probability of action is proportional to exponentiated weight
  \[ \pi_\theta(s, a) = e^{\phi(s, a)^T \theta} / \left( \sum_a e^{\phi(s, a)^T \theta} \right) \]
- The score function is
  \[ \nabla_\theta \log \pi_\theta(s, a) = \phi(s, a) - \mathbb{E}_{\pi_\theta}[\phi(s, \cdot)] \]
In continuous action spaces, a Gaussian policy is natural
Mean is a linear combination of state features $\mu(s) = \phi(s)^T \theta$
Variance may be fixed $\sigma^2$, or can also parametrised
Policy is Gaussian $a \sim \mathcal{N}(\mu(s), \sigma^2)$
The score function is
\[
\nabla_\theta \log \pi_\theta(s, a) = \frac{(a - \mu(s))\phi(s)}{\sigma^2}
\]
Putting this together

Goal is to find the policy parameters $\theta$:

$$\arg \max_{\theta} V(\theta) = \arg \max_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

Approximate with empirical estimate for $m$ sample paths under policy $\pi_\theta$ using score function:

$$\nabla_{\theta} V(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} R(\tau^{(i)}) \nabla_{\theta} \log P(\tau^{(i)}; \theta)$$

$$= \frac{1}{m} \sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

Do not need to know dynamics model
Consider generic form of $R(\tau^{(i)}) \nabla_\theta \log P(\tau^{(i)}; \theta)$:

$$\hat{g}_i = f(x_i) \nabla_\theta \log p(x_i|\theta)$$

$f(x)$ measures how good the sample $x$ is.

Moving in the direction $\hat{g}_i$ pushes up the logprob of the sample, in proportion to how good it is.

Valid even if $f(x)$ is discontinuous, and unknown, or sample space (containing $x$) is a discrete set.
\[ \hat{g}_i = f(x_i) \nabla_{\theta} \log p(x_i | \theta) \]
\[ \hat{g}_i = f(x_i) \nabla \theta \log p(x_i | \theta) \]
The policy gradient theorem generalizes the likelihood ratio approach.

**Theorem**

For any differentiable policy \( \pi_\theta(s, a) \), for any of the policy objective function \( J = J_1 \), (episodic reward), \( J_{avR} \) (average reward per time step), or \( \frac{1}{1-\gamma} J_{avV} \) (average value), the policy gradient is

\[
\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_\theta} [\nabla_{\theta} \log \pi_\theta(s, a) Q^{\pi_\theta}(s, a)]
\]

Chapter 13.2 in SB has a nice derivation of the policy gradient theorem for episodic tasks and discrete states.
Table of Contents

- Differentiable Policies

4 Policy Gradient Algorithms and Reducing Variance
  - Temporal Structure
  - Baseline
  - Alternatives to MC Returns
\[ \nabla_{\theta} V(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_{t}^{(i)}|s_{t}^{(i)}) \]

- Unbiased but very noisy
- Fixes that can make it practical
  - Temporal structure
  - Baseline
Policy Gradient: Use Temporal Structure

- Previously:

\[
\nabla_\theta \mathbb{E}_\tau [R] = \mathbb{E}_\tau \left[ \left( \sum_{t=0}^{T-1} r_t \right) \left( \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta (a_t | s_t) \right) \right]
\]

- We can repeat the same argument to derive the gradient estimator for a single reward term \( r_{t'} \).

\[
\nabla_\theta \mathbb{E}[r_{t'}] = \mathbb{E} \left[ r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta (a_t | s_t) \right]
\]

- To see this, recall \( V(s_0, \theta) = \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{T} R(s_t, a_t); \pi_\theta, s_0 \right] \) where the expectation is taken over the states & actions visited by \( \pi_\theta \).
Previously:

\[
\nabla_\theta \mathbb{E}_\tau [R] = \mathbb{E}_\tau \left[ \left( \sum_{t=0}^{T-1} r_t \right) \left( \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta (a_t | s_t) \right) \right]
\]

We can repeat the same argument to derive the gradient estimator for a single reward term \( r_{t'} \).

\[
\nabla_\theta \mathbb{E}[r_{t'}] = \mathbb{E} \left[ r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta (a_t | s_t) \right]
\]

Summing this formula over \( t \), we obtain

\[
V(\theta) = \nabla_\theta \mathbb{E}[R] = \mathbb{E} \left[ \sum_{t'=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta (a_t | s_t) \right]
\]
Policy Gradient: Use Temporal Structure

- Previously:

$$\nabla_\theta \mathbb{E}_\tau [R] = \mathbb{E}_\tau \left[ \left( \sum_{t=0}^{T-1} r_t \right) \left( \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t | s_t) \right) \right]$$

- We can repeat the same argument to derive the gradient estimator for a single reward term $r_{t'}$.

$$\nabla_\theta \mathbb{E}[r_{t'}] = \mathbb{E} \left[ r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t | s_t) \right]$$

- Summing this formula over $t$, we obtain

$$V(\theta) = \nabla_\theta \mathbb{E}[R] = \mathbb{E} \left[ \sum_{t'=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t | s_t) \right]$$

$$= \mathbb{E} \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \sum_{t'=t}^{T-1} r_{t'} \right]$$
Recall for a particular trajectory $\tau^{(i)}$, $\sum_{t'=t}^{T-1} r^{(i)}_{t'}$ is the return $G^{(i)}_t$

\[
\nabla_\theta \mathbb{E}[R] \approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) G^{(i)}_t
\]
Monte-Carlo Policy Gradient (REINFORCE)

- Leverages likelihood ratio / score function and temporal structure

\[ \Delta \theta_t = \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t \]

**REINFORCE:**

Initialize policy parameters \( \theta \) arbitrarily

for each episode \( \{s_1, a_1, r_2, \cdots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta \) do

for \( t = 1 \) to \( T - 1 \) do

\[ \theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t \]

endfor

endfor

return \( \theta \)
\[ \nabla_\theta V(\theta) \approx (1/m) \sum_{i=1}^{m} R(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \]

- Unbiased but very noisy
- Fixes that can make it practical
  - Temporal structure
  - Baseline
  - Alternatives to using Monte Carlo returns \( R(\tau^{(i)}) \) as targets
Desired Properties of a Policy Gradient RL Algorithm

- Goal: Converge as quickly as possible to a local optima
  - Incurring reward / cost as execute policy, so want to minimize number of iterations / time steps until reach a good policy
Table of Contents

- Differentiable Policies
- Temporal Structure

Policy Gradient Algorithms and Reducing Variance
- Baseline
- Alternatives to MC Returns
Policy Gradient: Introduce Baseline

- Reduce variance by introducing a baseline $b(s) = G_t - b(s_t)$

$$\nabla_\theta \mathbb{E}_\tau [R] = \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi (a_t | s_t; \theta) \left( \sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right) \right]$$

- For any choice of $b$, gradient estimator is unbiased.
- Near optimal choice is the expected return,

$$b(s_t) \approx \mathbb{E}[r_t + r_{t+1} + \cdots + r_{T-1}]$$

- Interpretation: increase logprob of action $a_t$ proportionally to how much returns $\sum_{t'=t}^{T-1} r_{t'}$ are better than expected
Baseline $b(s)$ Does Not Introduce Bias–Derivation

$$
= 0
$$

$$
E_T [\nabla_\theta \log \pi(a_t | s_t; \theta) b(s_t)] \\
= E_{s_0:t, a_0:(t-1)} [E_{s(t+1):T, a_{t:(T-1)}} [\nabla_\theta \log \pi(a_t | s_t; \theta) b(s_t)]]
$$

$$
= E_{s_0:t, a_0:(t-1)} [b(s_t) E_{s(t+1):T} \sum_a \pi(a_{t+1} | s_{t+1}; \theta) \nabla_\theta \log \pi(a_{t+1} | s_{t+1}; \theta) \pi(a_t | s_t; \theta)]
$$

$$
= E_{s_0:t, a_0:(t-1)} [b(s_t) \sum_a \pi(a_t | s_t; \theta) \nabla_\theta \pi(a_t | s_t; \theta)]
$$

$$
= 0
$$
Baseline $b(s)$ Does Not Introduce Bias–Derivation

$$E_{\tau}[\nabla_\theta \log \pi(a_t|s_t; \theta)b(s_t)]$$

$$= E_{s_0:t, a_0:(t-1)}[E_{s(t+1):T, a_t:(T-1)}[\nabla_\theta \log \pi(a_t|s_t; \theta)b(s_t)]]$$ (break up expectation)

$$= E_{s_0:t, a_0:(t-1)}[b(s_t)E_{s(t+1):T, a_t:(T-1)}[\nabla_\theta \log \pi(a_t|s_t; \theta)]]$$ (pull baseline term out)

$$= E_{s_0:t, a_0:(t-1)}[b(s_t)E_{a_t}[\nabla_\theta \log \pi(a_t|s_t; \theta)]]$$ (remove irrelevant variables)

$$= E_{s_0:t, a_0:(t-1)}[b(s_t)\sum_a \pi_\theta(a_t|s_t) \frac{\nabla_\theta \pi(a_t|s_t; \theta)}{\pi_\theta(a_t|s_t)}]$$ (likelihood ratio)

$$= E_{s_0:t, a_0:(t-1)}[b(s_t)\sum_a \nabla_\theta \pi(a_t|s_t; \theta)]$$

$$= E_{s_0:t, a_0:(t-1)}[b(s_t)\nabla_\theta \sum_a \pi(a_t|s_t; \theta)]$$

$$= E_{s_0:t, a_0:(t-1)}[b(s_t)\nabla_\theta 1]$$

$$= E_{s_0:t, a_0:(t-1)}[b(s_t) \cdot 0] = 0$$
“Vanilla” Policy Gradient Algorithm

Initialize policy parameter $\theta$, baseline $b$

for iteration=1, 2, ⋯ do
  Collect a set of trajectories by executing the current policy
  At each timestep $t$ in each trajectory $\tau^i$, compute
  Return $G^i_t = \sum_{t'=t}^{T-1} r^i_{t'}$, and
  Advantage estimate $\hat{A}^i_t = G^i_t - b(s_t)$.
  Re-fit the baseline, by minimizing $\sum_i \sum_t ||b(s_t) - G^i_t||^2$,
  Update the policy, using a policy gradient estimate $\hat{g}$,
  Which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta) \hat{A}_t$.
  (Plug $\hat{g}$ into SGD or ADAM)
endfor
Other Choices for Baseline?

Initialize policy parameter $\theta$, baseline $b$

for iteration = 1, 2, ... do

Collect a set of trajectories by executing the current policy
At each timestep $t$ in each trajectory $\tau^i$, compute

Return $G^i_t = \sum_{t'=t}^{T-1} r^i_{t'}$, and $\checkmark$

Advantage estimate $\hat{A}^i_t = G^i_t - b(s_t)$.

Re-fit the baseline, by minimizing $\sum_i \sum_t \| b(s_t) - G^i_t \|^2$.

Update the policy, using a policy gradient estimate $\hat{g}$,

Which is a sum of terms $\nabla_\theta \log \pi (a_t | s_t, \theta) \hat{A}_t$.

(Plug $\hat{g}$ into SGD or ADAM)

endfor
Choosing the Baseline: Value Functions

- Recall Q-function / state-action-value function:
  \[ Q^\pi(s, a) = \mathbb{E}_\pi [r_0 + \gamma r_1 + \gamma^2 r_2 \cdots | s_0 = s, a_0 = a] \]

- State-value function can serve as a great baseline
  \[ V^\pi(s) = \mathbb{E}_\pi [r_0 + \gamma r_1 + \gamma^2 r_2 \cdots | s_0 = s] = \mathbb{E}_{a \sim \pi}[Q^\pi(s, a)] \]
Policy Gradient Algorithms and Reducing Variance

- Alternatives to MC Returns
Policy gradient:

\[ \nabla_\theta \mathbb{E}[R] \approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t)(G_t^{(i)} - b(s_t)) \]

- Fixes that improve simplest estimator
  - Temporal structure (shown in above equation)
  - Baseline (shown in above equation)
- **Alternatives to using Monte Carlo returns** $G_t^i$ as estimate of expected discounted sum of returns for the policy parameterized by $\theta$?
Choosing the Target

- $G_t^i$ is an estimation of the value function at $s_t$ from a single roll out
- Unbiased but high variance
- Reduce variance by introducing bias using bootstrapping and function approximation
  - Just like in we saw for TD vs MC, and value function approximation
Actor-critic Methods

- Estimate of $V/Q$ is done by a critic
- **Actor-critic** methods maintain an explicit representation of policy and the value function, and update both
- A3C (Mnih et al. ICML 2016) is a very popular actor-critic method
Policy Gradient Formulas with Value Functions

- Recall:

\[
\nabla \theta \mathbb{E}_\tau [R] = \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla \theta \log \pi (a_t | s_t; \theta) \left( \sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right) \right]
\]

\[
\nabla \theta \mathbb{E}_\tau [R] \approx \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla \theta \log \pi (a_t | s_t; \theta) \left( Q(s_t, a_t; w) - b(s_t) \right) \right]
\]

- Letting the baseline be an estimate of the value \( V \), we can represent the gradient in terms of the state-action advantage function

\[
\nabla \theta \mathbb{E}_\tau [R] \approx \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla \theta \log \pi (a_t | s_t; \theta) \hat{A}^\pi (s_t, a_t) \right]
\]

- where the advantage function \( A^\pi (s, a) = Q^\pi (s, a) - V^\pi (s) \)
Choosing the Target: N-step estimators

\[ \nabla_\theta V(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{T-1} R_t^i \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \]

- Note that critic can select any blend between TD and MC estimators for the target to substitute for the true state-action value function.
Choosing the Target: N-step estimators

\[ \nabla_\theta V(\theta) \approx (1/m) \sum_{i=1}^{m} \sum_{t=0}^{T-1} R_t^i \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \]

- Note that critic can select any blend between TD and MC estimators for the target to substitute for the true state-action value function.

\[
\begin{align*}
\hat{R}_t^{(1)} &= r_t + \gamma V(s_{t+1}) \\
\hat{R}_t^{(2)} &= r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) \\
\hat{R}_t^{(\text{inf})} &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots
\end{align*}
\]

- If subtract baselines from the above, get advantage estimators

\[
\begin{align*}
\hat{A}_t^{(1)} &= r_t + \gamma V(s_{t+1}) - V(s_t) \\
\hat{A}_t^{(\text{inf})} &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + \cdots - V(s_t)
\end{align*}
\]
Check Your Understanding: Blended Advantage Estimators

\[ \nabla_{\theta} V(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{T-1} R^i_t \nabla_{\theta} \log \pi_{\theta}(a_{t}^{(i)}|s_{t}^{(i)}) \]

- If subtract baselines from the above, get advantage estimators

\[ \hat{A}^{(1)}_t = r_t + \gamma V(s_{t+1}) - V(s_t) \]
\[ \hat{A}^{(\infty)}_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + \cdots - V(s_t) \]

- Select all that are true
- \( \hat{A}^{(1)}_t \) has low variance & low bias.
- \( \hat{A}^{(1)}_t \) has high variance & low bias.
- \( \hat{A}^{(\infty)}_t \) low variance and high bias.
- \( \hat{A}^{(\infty)}_t \) high variance and low bias.
- Not sure
Check Your Understanding: Blended Advantage Estimators

Answers

\( \nabla_\theta V(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{T-1} R_t^i \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \)

- If subtract baselines from the above, get advantage estimators

\[
\hat{A}_t^{(1)} = r_t + \gamma V(s_{t+1}) - V(s_t)
\]
\[
\hat{A}_t^{(\text{inf})} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+1} + \cdots - V(s_t)
\]

Solution: \( \hat{A}_t^{(1)} \) has low variance & high bias. \( \hat{A}_t^{(\infty)} \) high variance but low bias.
"Vanilla" Policy Gradient Algorithm

Initialize policy parameter $\theta$, baseline $b$

for iteration=1, 2, · · · do

Collect a set of trajectories by executing the current policy

At each timestep $t$ in each trajectory $\tau^i$, compute

$\text{Advantage estimate } \hat{A}^n_{it}$

Update the policy, using a policy gradient estimate $\hat{g}$,

Which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta) \hat{A}^n_{it}$.

(Plug $\hat{g}$ into SGD or ADAM)

endfor

- Note, can choose which blended estimator $\hat{A}^n$ to use
Last time: Deep Model-free Value Based RL

This time: Policy Search

Next time: Policy Search Cont.