Policy Gradients

CS 294-112: Deep Reinforcement Learning
Sergey Levine
Class Notes

1. Homework 1 due today (11:59 pm)!
   • Don’t be late!

2. Remember to start forming final project groups
Today’s Lecture

1. The policy gradient algorithm
2. What does the policy gradient do?
3. Basic variance reduction: causality
4. Basic variance reduction: baselines
5. Policy gradient examples

• Goals:
  • Understand policy gradient reinforcement learning
  • Understand practical considerations for policy gradients
The goal of reinforcement learning

\[ p_\theta(s_1, a_1, \ldots, s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t) \]

\[ p_\theta(\tau) \]

\[ \theta^* = \arg \max_{\theta} E_{\tau \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]
The goal of reinforcement learning

\[ \theta^* = \arg \max_{\theta} E_{\tau \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

\[ \theta^* = \arg \max_{\theta} E_{(s,a) \sim p_\theta(s,a)} [r(s, a)] \]

infinite horizon case

\[ \theta^* = \arg \max_{\theta} \sum_{t=1}^T E_{(s_t,a_t) \sim p_\theta(s_t,a_t)} [r(s_t, a_t)] \]

finite horizon case
Evaluating the objective

\[ \theta^* = \arg \max_{\theta} E_{\tau \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

\[ J(\theta) \]

\[ J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t}) \]

sum over samples from \( \pi_\theta \)
Direct policy differentiation

\[ \theta^* = \arg\max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[ \sum_{t} r(s_t, a_t) \right] \]

\[ J(\theta) = \frac{1}{T} \sum_{t=1}^{T} r(s_t, a_t) \]

\[ J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)}[r(\tau)] = \int \pi_{\theta}(\tau)r(\tau)d\tau \]

\[ \nabla_{\theta} J(\theta) = \int \nabla_{\theta} \pi_{\theta}(\tau)r(\tau)d\tau = \int \pi_{\theta}(\tau)\nabla_{\theta} \log \pi_{\theta}(\tau)r(\tau)d\tau = E_{\tau \sim \pi_{\theta}(\tau)}[\nabla_{\theta} \log \pi_{\theta}(\tau)r(\tau)] \]

a convenient identity

\[ \pi_{\theta}(\tau)\nabla_{\theta} \log \pi_{\theta}(\tau) = \pi_{\theta}(\tau) \frac{\nabla_{\theta} \pi_{\theta}(\tau)}{\pi_{\theta}(\tau)} = \nabla_{\theta} \pi_{\theta}(\tau) \]
Direct policy differentiation

\[ \theta^* = \arg \max_{\theta} J(\theta) \]

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[r(\tau)] \quad \text{log of both sides} \]

\[ \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)] \]

\[ \nabla_\theta \left[ \log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t) \right] \]

\[ \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left( \sum_{t=1}^{T} r(s_t, a_t) \right) \right] \]
Evaluating the policy gradient

recall: \( J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t}) \)

\( \nabla_{\theta} J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \left( \sum_{t=1}^{T} \nabla_{\theta} \log \pi_\theta(a_t|s_t) \right) \left( \sum_{t=1}^{T} r(s_t, a_t) \right) \right] \)

\( \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_{\theta} \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \)

\( \theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta) \)

REINFORCE algorithm:
1. sample \( \{\tau^i\} \) from \( \pi_\theta(a_t|s_t) \) (run the policy)
2. \( \nabla_{\theta} J(\theta) \approx \sum_i \left( \sum_t \nabla_{\theta} \log \pi_\theta(a^i_t|s^i_t) \right) \left( \sum_t r(s^i_t, a^i_t) \right) \)
3. \( \theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta) \)
Evaluating the policy gradient

recall: $J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t})$

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[ \left( \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left( \sum_{t=1}^T r(s_t, a_t) \right) \right]$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left( \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right)$$

what is this?

$s_t$  $\pi_\theta(a_t|s_t)$  $a_t$
Comparison to maximum likelihood

Policy gradient: \( \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \)

Maximum likelihood: \( \nabla_{\theta} J_{ML}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \right) \)
Example: Gaussian policies

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]

example: \[ \pi_\theta(a_t|s_t) = \mathcal{N}(f_{\text{neural network}}(s_t); \Sigma) \]

\[ \log \pi_\theta(a_t|s_t) = -\frac{1}{2} \| f(s_t) - a_t \|_\Sigma^2 + \text{const} \]

\[ \nabla_\theta \log \pi_\theta(a_t|s_t) = -\frac{1}{2} \Sigma^{-1}(f(s_t) - a_t) \frac{df}{d\theta} \]

REINFORCE algorithm:

1. sample \( \{\tau^i\} \) from \( \pi_\theta(a_t|s_t) \) (run it on the robot)
2. \( \nabla_\theta J(\theta) \approx \sum_i \left( \sum_t \nabla_\theta \log \pi_\theta(a^i_t|s^i_t) \right) \left( \sum_t r(s^i_t, a^i_t) \right) \)
3. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)
What did we just do?

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\tau_i) r(\tau_i) \quad \text{maximum likelihood:} \quad \nabla_\theta J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau_i) \]

good stuff is made more likely

bad stuff is made less likely

simply formalizes the notion of “trial and error”!

REINFORCE algorithm:

1. sample \( \{\tau^i\} \) from \( \pi_\theta(a_t|s_t) \) (run it on the robot)
2. \[ \nabla_\theta J(\theta) \approx \sum_{i} \left( \sum_{t} \nabla_\theta \log \pi_\theta(a^i_t|s^i_t) \right) \left( \sum_{t} r(s^i_t, a^i_t) \right) \]
3. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)
Partial observability

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|o_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]

Markov property is not actually used!

Can use policy gradient in partially observed MDPs without modification
What is wrong with the policy gradient?

\[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau) \]

even worse: what if the two “good” samples have \( r(\tau) = 0 \)?

high variance
Review

• Evaluating the RL objective
  • Generate samples

• Evaluating the policy gradient
  • Log-gradient trick
  • Generate samples

• Understanding the policy gradient
  • Formalization of trial-and-error

• Partial observability
  • Works just fine

• What is wrong with policy gradient?

\[ \sum_{t=1}^{T} r(x_t, u_t) \]

\[ \theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta) \]
Break
Reducing variance

\[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]

*Causality:* policy at time \( t' \) cannot affect reward at time \( t \) when \( t < t' \)

\[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi_{\theta}(a_{i,t}|s_{i,t}) \left( \sum_{t'=t}^{T} r(s_{i,t''}, a_{i,t''}) \right) \]

\[ \hat{Q}_{i,t} \]

“reward to go”
Baselines

\[ \nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_{\theta} \log \pi_{\theta}(\tau)[n(\tau)] - b \]

\[ b = \frac{1}{N} \sum_{i=1}^{N} r(\tau) \]

but... are we *allowed* to do that??

\[ E[\nabla_{\theta} \log \pi_{\theta}(\tau)b] = \int \pi_{\theta}(\tau)\nabla_{\theta} \log \pi_{\theta}(\tau)b \, d\tau = \int \nabla_{\theta} \pi_{\theta}(\tau)b \, d\tau = b\nabla_{\theta} \int \pi_{\theta}(\tau) \, d\tau = b\nabla_{\theta} 1 = 0 \]

subtracting a baseline is *unbiased* in expectation!

average reward is *not* the best baseline, but it’s pretty good!

\[ \pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) = \nabla_{\theta} \pi_{\theta}(\tau) \]

a convenient identity
Analyzing variance

can we write down the variance?

\[
\text{Var}[x] = E[x^2] - E[x]^2
\]

\[
\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)]
\]

\[
\text{Var} = E_{\tau \sim \pi_\theta(\tau)}[(\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b))^2] - E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)(r(\tau) - b)]^2
\]

this bit is just \(E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)r(\tau)]\)
(baselines are unbiased in expectation)

\[
\frac{d\text{Var}}{db} = \frac{d}{db} E[g(\tau)^2(r(\tau) - b)^2] = \frac{d}{db} \left( E[g(\tau)^2r(\tau)^2] - 2E[g(\tau)^2r(\tau)b] + b^2 E[g(\tau)^2] \right)
\]

\[
= -2E[g(\tau)^2r(\tau)] + 2bE[g(\tau)^2] = 0
\]

\[
b = \frac{E[g(\tau)^2r(\tau)]}{E[g(\tau)^2]}
\]

This is just expected reward, but weighted by gradient magnitudes!
Review

- The high variance of policy gradient
- Exploiting causality
  - Future doesn’t affect the past
- Baselines
  - Unbiased!
- Analyzing variance
  - Can derive optimal baselines

\[
\hat{Q}^\pi(x_t, u_t) = \sum_{t'=t}^{T} r(x_{t'}, u_{t'})
\]

\[
\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)
\]
Policy gradient is on-policy

\[ \theta^* = \arg \max_{\theta} J(\theta) \]

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[r(\tau)] \]

\[ \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)] \]

- Neural networks change only a little bit with each gradient step
- On-policy learning can be extremely inefficient!

REINFORCE algorithm:
1. sample \( \{\tau^i\} \) from \( \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \) (run it on the robot)
2. \( \nabla_\theta J(\theta) \approx \sum_i \left( \sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \right) \left( \sum_t r(s_t^i, a_t^i) \right) \)
3. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)
Off-policy learning & importance sampling

\[ \theta^* = \arg \max_{\theta} J(\theta) \]

\[ J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[r(\tau)] \]

does not have samples from \( \pi_\theta(\tau) \)?

(we have samples from some \( \tilde{\pi}(\tau) \) instead)

\[ J(\theta) = E_{\tau \sim \tilde{\pi}(\tau)} \left[ \frac{\pi_\theta(\tau)}{\tilde{\pi}(\tau)} r(\tau) \right] \]

\[ \pi_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t) \]

\[ \frac{\pi_\theta(\tau)}{\tilde{\pi}(\tau)} = \frac{p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)}{p(s_1) \prod_{t=1}^{T} \tilde{\pi}(a_t|s_t)p(s_{t+1}|s_t, a_t)} = \frac{\prod_{t=1}^{T} \pi_\theta(a_t|s_t)}{\prod_{t=1}^{T} \tilde{\pi}(a_t|s_t)} \]

importance sampling

\[ E_{x \sim p(x)}[f(x)] = \int p(x) f(x) \, dx \]

\[ = \int \frac{q(x)}{q(x)} p(x) f(x) \, dx \]

\[ = \int q(x) \frac{p(x)}{q(x)} f(x) \, dx \]

\[ = E_{x \sim q(x)} \left[ \frac{p(x)}{q(x)} f(x) \right] \]
Deriving the policy gradient with IS

$$\theta^* = \arg \max_{\theta} J(\theta)$$

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[r(\tau)]$$

can we estimate the value of some new parameters $\theta'$?

$$J(\theta') = E_{\tau \sim \pi_\theta(\tau)} \left[ \frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)} r(\tau) \right]$$

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim \pi_\theta(\tau)} \left[ \frac{\nabla_{\theta'} \pi_{\theta'}(\tau)}{\pi_\theta(\tau)} r(\tau) \right] = E_{\tau \sim \pi_\theta(\tau)} \left[ \frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)} \nabla_{\theta'} \log \pi_{\theta'}(\tau) r(\tau) \right]$$

now estimate locally, at $\theta = \theta'$:  
$$\nabla_{\theta} J(\theta) = E_{\tau \sim \pi_\theta(\tau)}[\nabla_{\theta} \log \pi_\theta(\tau)r(\tau)]$$

a convenient identity

$$\pi_\theta(\tau) \nabla_{\theta} \log \pi_\theta(\tau) = \nabla_{\theta} \pi_\theta(\tau)$$
The off-policy policy gradient

\[ \theta^* = \arg \max_{\theta} J(\theta) \]

\[ J(\theta) = E_{T \sim \pi_\theta(\tau)}[r(\tau)] \]

\[ \nabla_{\theta'} J(\theta') = E_{T \sim \pi_\theta(\tau)} \left[ \frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)} \nabla_{\theta'} \log \pi_{\theta'}(\tau) r(\tau) \right] \quad \text{when } \theta \neq \theta' \]

\[ = E_{T \sim \pi_\theta(\tau)} \left[ \left( \prod_{t=1}^{T} \frac{\pi_{\theta'}(a_t | s_t)}{\pi_\theta(a_t | s_t)} \right) \left( \sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t | s_t) \right) \left( \sum_{t=1}^{T} r(s_t, a_t) \right) \right] \quad \text{what about causality?} \]

\[ = E_{T \sim \pi_\theta(\tau)} \left[ \sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t | s_t) \left( \prod_{t'=1}^{t} \frac{\pi_{\theta'}(a_{t'} | s_{t'})}{\pi_\theta(a_{t'} | s_{t'})} \right) \left( \sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \left( \prod_{t''=t'}^{t''} \frac{\pi_{\theta'}(a_{t''} | s_{t''})}{\pi_\theta(a_{t''} | s_{t''})} \right) \right) \right] \]

future actions don’t affect current weight

if we ignore this, we get a policy iteration algorithm

(more on this in a later lecture)
A first-order approximation for IS (preview)

\[ \nabla_{\theta'} J(\theta') = \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[ \sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t | s_t) \left( \prod_{t'=1}^{t} \frac{\pi_{\theta'}(a_{t'} | s_{t'})}{\pi_\theta(a_{t'} | s_{t'})} \right) \left( \sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \right) \right] \]

let's write the objective a bit differently...

\[ \theta^* = \arg \max_{\theta} \sum_{t=1}^{T} \mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)} [r(s_t, a_t)] \]

\[ J(\theta) = \sum_{t=1}^{T} \mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)} [r(s_t, a_t)] = \sum_{t=1}^{T} \mathbb{E}_{s_t \sim p_\theta(s_t)} \left[ \mathbb{E}_{a_t \sim \pi_\theta(a_t | s_t)} [r(s_t, a_t)] \right] \]

\[ J(\theta') = \sum_{t=1}^{T} \mathbb{E}_{s_t \sim p_\theta(s_t)} \left[ \frac{p_{\theta'}(s_t)}{p_\theta(s_t)} \mathbb{E}_{a_t \sim \pi_\theta(a_t | s_t)} \left[ \frac{\pi_{\theta'}(a_t | s_t)}{\pi_\theta(a_t | s_t)} r(s_t, a_t) \right] \right] \]

We’ll see why this is reasonable later in the course!
Policy gradient with automatic differentiation

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \hat{Q}_{i,t} \]

pretty inefficient to compute these explicitly!

How can we compute policy gradients with automatic differentiation?

We need a graph such that its gradient is the policy gradient!

maximum likelihood:  \[ \nabla_\theta J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \quad J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log \pi_\theta(a_{i,t}|s_{i,t}) \]

Just implement “pseudo-loss” as a weighted maximum likelihood:

\[ \tilde{J}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log \pi_\theta(a_{i,t}|s_{i,t}) \hat{Q}_{i,t} \]

cross entropy (discrete) or squared error (Gaussian)
Policy gradient with automatic differentiation

Pseudocode example (with discrete actions):

Maximum likelihood:

```python
# Given:
# actions - (N*T) x Da tensor of actions
# states - (N*T) x Ds tensor of states
# Build the graph:
logits = policy.predictions(states)  # This should return (N*T) x Da tensor of action logits
negative_likelihoods = tf.nn.softmax_cross_entropy_with_logits(labels=actions, logits=logits)
loss = tf.reduce_mean(negative_likelihoods)
gradients = loss.gradients(loss, variables)
```
Policy gradient with automatic differentiation

Pseudocode example (with discrete actions):

Policy gradient:

```python
# Given:
# actions - (N*T) x Da tensor of actions
# states - (N*T) x Ds tensor of states
# q_values - (N*T) x 1 tensor of estimated state-action values
# Build the graph:
logits = policy.predictions(states)  # This should return (N*T) x Da tensor of action logits
negative_likelihoods = tf.nn.softmax_cross_entropy_with_logits(labels=actions, logits=logits)
weighted_negative_likelihoods = tf.multiply(negative_likelihoods, q_values)
loss = tf.reduce_mean(weighted_negative_likelihoods)
gradients = loss.gradients(loss, variables)
```

\[
\hat{J}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log \pi_{\theta}(a_{i,t}|s_{i,t}, Q_{i,t})
\]
Policy gradient in practice

• Remember that the gradient has high variance
  • This isn’t the same as supervised learning!
  • Gradients will be really noisy!

• Consider using much larger batches

• Tweaking learning rates is very hard
  • Adaptive step size rules like ADAM can be OK-ish
  • We’ll learn about policy gradient-specific learning rate adjustment methods later!
Review

• Policy gradient is on-policy

• Can derive off-policy variant
  • Use importance sampling
  • Exponential scaling in $T$
  • Can ignore state portion (approximation)

• Can implement with automatic differentiation – need to know what to backpropagate

• Practical considerations: batch size, learning rates, optimizers
What else is wrong with the policy gradient?

\[ r(s_t, a_t) = -s_t^2 - a_t^2 \]

\[ \log \pi_\theta(a_t|s_t) = -\frac{1}{2}\sigma^2 (ks_t - a_t)^2 + \text{const} \quad \theta = (k, \sigma) \]

Essentially the same problem as this:
Covariant/natural policy gradient

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \quad \pi_\theta(a_t|s_t)$$

some parameters change probabilities a lot more than others!

$$\theta' \leftarrow \arg\max_{\theta'} (\theta' - \theta)^T \nabla_\theta J(\theta) \quad \text{s.t.} \quad \|\theta' - \theta\|^2 \leq \epsilon$$

controls how far we go can we rescale the gradient so this doesn’t happen?

$$\theta' \leftarrow \arg\max_{\theta'} (\theta' - \theta)^T \nabla_\theta J(\theta) \quad \text{s.t.} \quad D(\pi_{\theta'}, \pi_\theta) \leq \epsilon$$

parameterization-independent divergence measure usually KL-divergence: $D_{KL}(\pi_{\theta'}||\pi_\theta) = E_{\pi_\theta'}[\log \pi_\theta - \log \pi_{\theta'}]$ 

$$D_{KL}(\pi_{\theta'}||\pi_\theta) \approx (\theta' - \theta)^T \mathbf{F}(\theta' - \theta)$$

Fisher-information matrix

$$\mathbf{F} = E_{\pi_\theta} [\log \pi_\theta(a|s) \log \pi_\theta(a|s)^T]$$

can estimate with samples
Covariant/natural policy gradient

\[ D_{KL}(\pi_{\theta'} || \pi_\theta) \approx (\theta' - \theta)^T F (\theta' - \theta) \]

\[ F = E_{\pi_\theta}[\log \pi_{\theta}(a|s) \log \pi_{\theta}(a|s)^T] \]

\[ \theta' \leftarrow \arg \max_{\theta'} (\theta' - \theta)^T \nabla_{\theta} J(\theta) \text{ s.t. } \|\theta_0 - \theta\|_F^2 \leq \epsilon \]

\[ \theta \leftarrow \theta + \alpha F^{-1} \nabla_{\theta} J(\theta) \]

natural gradient: pick \( \alpha \)

trust region policy optimization: pick \( \epsilon \)

can solve for optimal \( \alpha \) while solving \( F^{-1} \nabla_{\theta} J(\theta) \)

conjugate gradient works well for this

see Schulman, L., Moritz, Jordan, Abbeel (2015) Trust region policy optimization

(figure from Peters & Schaal 2008)
Advanced policy gradient topics

• What more is there?
• Next time: introduce value functions and Q-functions
• Later in the class: natural gradient and automatic step size adjustment
Example: policy gradient with importance sampling

\[ \nabla_{\theta'} J(\theta') = E_{\tau \sim \pi_\theta(\tau)} \left[ \sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t | s_t) \left( \prod_{t'=1}^{t} \frac{\pi_{\theta'}(a_{t'} | s_{t'})}{\pi_{\theta}(a_{t'} | s_{t'})} \right) \left( \sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \right) \right] \]

- Incorporate example demonstrations using importance sampling
- Neural network policies

Levine, Koltun ‘13
Example: trust region policy optimization

- Natural gradient with automatic step adjustment
- Discrete and continuous actions
- Code available (see Duan et al. ‘16)
Policy gradients suggested readings

• Classic papers
  • Williams (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning: introduces REINFORCE algorithm
  • Baxter & Bartlett (2001). Infinite-horizon policy-gradient estimation: temporally decomposed policy gradient (not the first paper on this! see actor-critic section later)
  • Peters & Schaal (2008). Reinforcement learning of motor skills with policy gradients: very accessible overview of optimal baselines and natural gradient

• Deep reinforcement learning policy gradient papers
  • Levine & Koltun (2013). Guided policy search: deep RL with importance sampled policy gradient (unrelated to later discussion of guided policy search)
  • Schulman, Wolski, Dhariwal, Radford, Klimov (2017). Proximal policy optimization algorithms: deep RL with importance sampled policy gradient