Lecture 4: Deep Reinforcement Learning

Policy Gradients

Chunlin Chen & Zhi Wang

Department of Control and Systems Engineering
Nanjing University

Nov. 26th, 2019

Table of Contents

What we'll cover

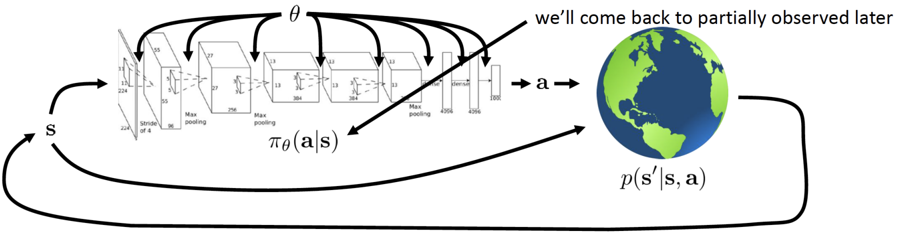
— Contents

- The policy gradient algorithm
- What does the policy gradient do?
- Basic variance reduction: causality
- Basic variance reduction: baselines
- Policy gradient examples

— Goals

- Understand policy gradient RL
- Understand practical considerations for policy gradients

Recall the goal of RL

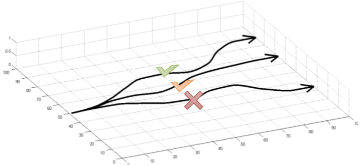


$$p_{\boldsymbol{\theta}}(\tau) = p_{\boldsymbol{\theta}}(s_0, a_0, ..., s_T, a_T) = p(s_0)\Pi_{t=0}^{T}\pi_{\boldsymbol{\theta}}(a_t|s_t)p(s_{t+1}|s_t, a_t)$$

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathbb{E}_{\tau\sim p_{\boldsymbol{\theta}}(\tau)}\left[\sum_t r(s_t, a_t)\right]$$

Evaluating the objective

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \underbrace{\mathbb{E}_{\tau\sim p_{\boldsymbol{\theta}}(\tau)}\left[\sum_t r(s_t, a_t)\right]}_{J(\boldsymbol{\theta})}$$



$$J(\boldsymbol{\theta}) = \mathbb{E}_{\tau\sim p_{\boldsymbol{\theta}}(\tau)}\left[\sum_t r(s_t, a_t)\right] \approx \frac{1}{N} \underbrace{\sum_i}_{\uparrow} \sum_t r(s_{i,t}, a_{i,t})$$

sum over samples from $\pi_{\boldsymbol{\theta}}$

Direct policy differentiation

- Objective function / cost function

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\underbrace{r(\tau)}_{\sum_{t=0}^{T} r(s_t, a_t)}] = \int \pi_\theta(\tau) r(\tau)\, \mathrm{d}\tau$$

- The gradient – differentiate the objective function

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau)\, \mathrm{d}\tau = \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) r(\tau)\, \mathrm{d}\tau$$
$$= \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]$$

- A convenient identity

$$\pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau) \frac{\nabla_\theta \log \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau)$$

Direct policy differentiation

$$\theta^\star = \arg\max_\theta J(\theta)$$

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

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

$$\underbrace{\pi_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$$

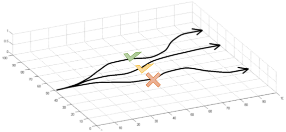
log of both sides

$$\log \pi_\theta(\tau) = \log p(\mathbf{s}_1) + \sum_{t=1}^{T}\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$$

$$\nabla_\theta\left[\cancel{\log p(\mathbf{s}_1)} + \sum_{t=1}^{T}\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \cancel{\log p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}\right]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_t,\mathbf{a}_t)\right)\right]$$

Evaluating the policy gradient

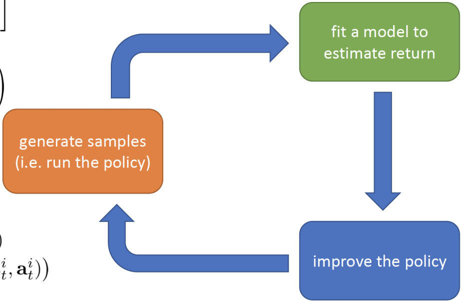


recall: $J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right] \approx \frac{1}{N}\sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right)\right]$$

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

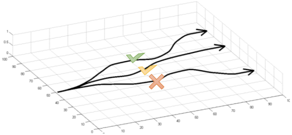
$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$



REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Evaluating the policy gradient

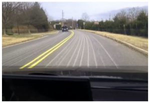


recall: $J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right] \approx \frac{1}{N}\sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$

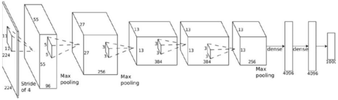
$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right)\right]$$

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

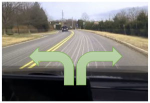
what is this?



$\mathbf{s}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

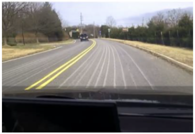


$\mathbf{a}_t$

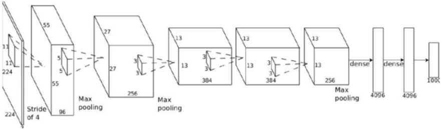
Comparison to maximum likelihood

policy gradient: $$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$

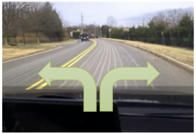
maximum likelihood: $$\nabla_\theta J_{\mathrm{ML}}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)$$



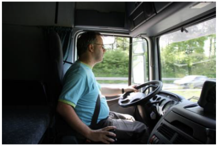
$\mathbf{s}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$



$\mathbf{a}_t$



$\mathbf{s}_t$

$\mathbf{a}_t$

training data

supervised learning

$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

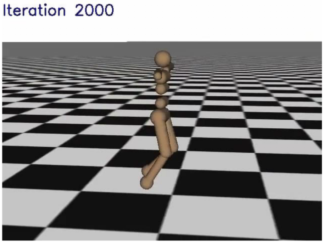
Example: Gaussian policies

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

example: $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) = \mathcal{N}(f_{\text{neural network}}(\mathbf{s}_t);\Sigma)$

$$\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) = -\frac{1}{2}\|f(\mathbf{s}_t) - \mathbf{a}_t\|_\Sigma^2 + \text{const}$$

$$\nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) = -\frac{1}{2}\Sigma^{-1}(f(\mathbf{s}_t) - \mathbf{a}_t)\frac{df}{d\theta}$$



REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i,\mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$

What did we just do?

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

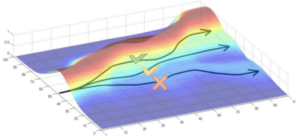
$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\underbrace{\nabla_\theta \log \pi_\theta(\tau_i)}_{\sum_{t=1}^{T}\nabla_\theta \log_\theta \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} r(\tau_i)$$

maximum likelihood: $\nabla_\theta J_{\mathrm{ML}}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\nabla_\theta \log \pi_\theta(\tau_i)$

good stuff is made more likely

bad stuff is made less likely

simply formalizes the notion of “trial and error”!

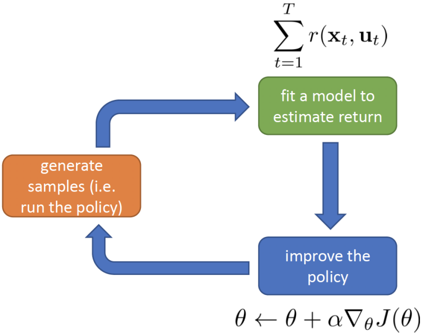


REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i,\mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

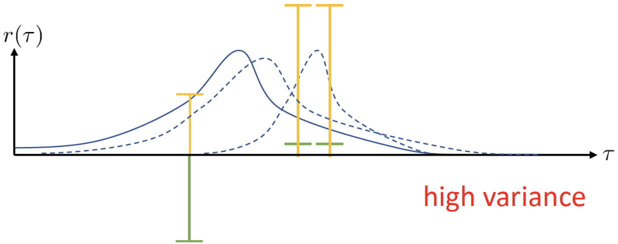
Review of the policy gradient

- Evaluating the RL objective
  - Generate samples
- Evaluating the policy gradient
  - Log gradient trick
  - Generate samples
- Understand policy gradient
  - Formalization of trial-and-error



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 $r(\tau) = 0$?

Reducing variance - Causality

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

$$\Downarrow$$

- **Causality**: policy at time t' cannot affect reward at time t when $t < t'$

$$\Downarrow$$

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

"reward to go"

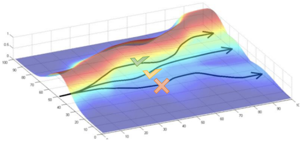
Reducing variance - Baselines

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

a convenient identity

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \nabla_\theta \pi_\theta(\tau)$$

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



- But... are we allowed to do that?

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

- Subtracting a baseline is unbiased in expectation!
- Average reward is not the best baseline, but it's pretty good!

Analyzing the variance

can we write down the variance?

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

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

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

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

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

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

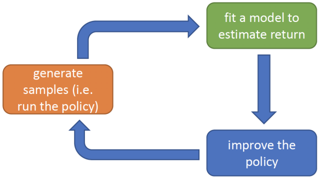
$$b = \frac{E[g(\tau)^2 r(\tau)]}{E[g(\tau)^2]} \longleftarrow$$

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

$$\hat{Q}^{\pi}(\mathbf{x}_t, \mathbf{u}_t) = \sum_{t'=t}^{T} r(\mathbf{x}_{t'}, \mathbf{u}_{t'})$$



$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Policy gradient is on-policy

$$\theta^\star = \arg\max_\theta J(\theta)$$

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

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

this is trouble...

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

can't just skip this!

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Off-policy learning & importance sampling

$$\theta^\star = \arg\max_\theta J(\theta)$$

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

what if we don't have samples from $\pi_\theta(\tau)$?

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

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

$$\pi_\theta(\tau) = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$$

$$\frac{\pi_\theta(\tau)}{\bar{\pi}(\tau)} = \frac{\cancel{p(\mathbf{s}_1)}\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}}{\cancel{p(\mathbf{s}_1)}\prod_{t=1}^{T}\bar{\pi}(\mathbf{a}_t|\mathbf{s}_t)\cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}} = \frac{\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\prod_{t=1}^{T}\bar{\pi}(\mathbf{a}_t|\mathbf{s}_t)}$$

importance sampling

$$\begin{aligned}
E_{x\sim p(x)}[f(x)] &= \int p(x)f(x)dx \\
&= \int \frac{q(x)}{q(x)}p(x)f(x)dx \\
&= \int q(x)\frac{p(x)}{q(x)}f(x)dx \\
&= E_{x\sim q(x)}\left[\frac{p(x)}{q(x)}f(x)\right]
\end{aligned}$$

Policy gradient with importance sampling

$$\theta^\star = \arg\max_\theta J(\theta)$$

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

can we estimate the value of some *new* parameters θ'?

the only bit that depends on θ'

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

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

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

$$\frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)} = \frac{\prod_{t=1}^{T} \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}$$

$$\theta^\star = \arg\max_\theta J(\theta)$$

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

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

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

$$= E_{\tau\sim\pi_\theta(\tau)}\left[\sum_{t=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)\underline{\left(\prod_{t'=1}^{t}\frac{\pi_{\theta'}(\mathbf{a}_{t'}|\mathbf{s}_{t'})}{\pi_\theta(\mathbf{a}_{t'}|\mathbf{s}_{t'})}\right)}\left(\sum_{t'=t}^{T} r(\mathbf{s}_{t'},\mathbf{a}_{t'})\right)\right]$$

future actions don't affect current weight

A first-order approximation for importance sampling

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim \pi_\theta(\tau)} \left[\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \underline{\left(\prod_{t'=1}^{t} \frac{\pi_{\theta'}(\mathbf{a}_{t'}|\mathbf{s}_{t'})}{\pi_\theta(\mathbf{a}_{t'}|\mathbf{s}_{t'})} \right)} \left(\sum_{t'=t}^{T} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \right) \right]$$

exponential in T...

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

on-policy policy gradient: $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \hat{Q}_{i,t}$

$(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \sim \pi_\theta(\mathbf{s}_t, \mathbf{a}_t)$

off-policy policy gradient: $\nabla_{\theta'} J(\theta') \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{\pi_{\theta'}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})}{\pi_\theta(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \hat{Q}_{i,t}$

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \cancel{\frac{\pi_{\theta'}(\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{s}_{i,t})}} \frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \hat{Q}_{i,t}$$

ignore this part

Research outputs using off-policy policy gradient

IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS 1

Incremental Reinforcement Learning in Continuous Spaces via Policy Relaxation and Importance Weighting

Zhi Wang, *Student Member, IEEE*, Han-Xiong Li, *Fellow, IEEE*, and Chunlin Chen, *Member, IEEE*

Algorithm 1 Policy Relaxation With Importance Sampling

Input: Number of burn-in episodes k;
learning rate α; batch size m
Output: Optimal policy parameters $\boldsymbol{\theta}^*$

```
Initialize the number of learning episodes: η ← 0
while not converged do
    if η ≤ k then
        π_r(a|s) = Uniform(A(s)), ∀s
        Sample m episodes from π_r: τ^i ~ π_r
        ∇_θ J(θ) = Σ_{i=1}^m [π_θ(τ^i)/π_r(τ^i)] ∇_θ log π_θ(τ^i) r(τ^i)
    else
        Sample m episodes from π_θ: τ^i ~ π_θ
        ∇_θ J(θ) = Σ_{i=1}^m ∇_θ log π_θ(τ^i) r(τ^i)
    end
    θ ← θ + α ∇_θ J(θ)
    η ← η + m
end
```

B. Policy Relaxation

In the new environment M_t, the agent tends to visit a small part of the whole state-action space when executing the previously learned policy, thus probably leading to a local optimum due to insufficient exploration. Hence, we propose a policy relaxation mechanism to encourage a proper exploration. Specifically, in the k burn-in learning episodes, the agent is forced to execute a relaxed policy where actions are randomly selected from the available set. For better readability, let $\boldsymbol{\theta}$ denote the current parameters in M_t, and $\pi_{\boldsymbol{\theta}}$ be the policy derived from $\boldsymbol{\theta}$. Regarding the number of learning episodes η, the agent's behavior policy π_r is relaxed as

$$\pi_r(a|s) = \begin{cases} \text{Uniform}(A(s)), & \eta \le k \\ \pi_{\boldsymbol{\theta}}(a|s), & \eta > k \end{cases} \tag{9}$$

Policy gradient with automatic differentiation

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \underline{\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}\hat{Q}_{i,t}$$

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: $\nabla_\theta J_{\mathrm{ML}}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})$ $\quad J_{\mathrm{ML}}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})$

Just implement "pseudo-loss" as a weighted maximum likelihood:

$$\tilde{J}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\hat{Q}_{i,t}$$

cross entropy (discrete) or squared error (Gaussian)

Policy gradient with automatic differentiation

Pseudocode example (with discrete actions):

Maximum likelihood:

```
# 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:

```
# 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)
```

$$\tilde{J}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\hat{Q}_{i,t}$$

q_values

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
  - Using adaptive step size rules like ADAM
  - More policy gradient specific learning rate adjustment methods...

Review

- Policy gradient is on policy
- Can derive off policy variant
  - Use importance sampling
  - Exponential scaling in T
  - Can ignore state portion (first-order approximation)
- Can implement with automatic differentiation – need to know what to backpropagate
- Practical considerations: batch size, learning rates, optimizers

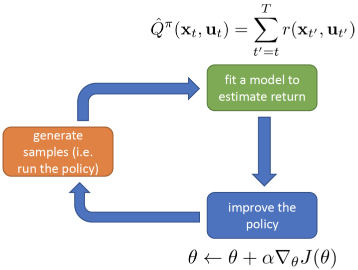


Table of Contents

Problems of vanilla policy gradient (REINFORCE)

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

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

- Hard to select the step size α
  - Too big step: Bad policy $\rightarrow$ data collected under bad policy $\rightarrow$ we cannot recover (in Supervised Learning, data does not depend on neural network weights)
  - Too small step: Not efficient use of experience (in Supervised Learning, data can be trivially re-used)

Problems of vanilla policy gradient (REINFORCE)

- Small changes in the policy parameters can unexpectedly lead to big **changes** in the policy

Gradient descent in parameter space

- The step size in gradient descent results from solving the following optimization problem, e.g., using line search

$$d^* = \arg\max_{||d|| \leq \epsilon} J(\theta + d)$$

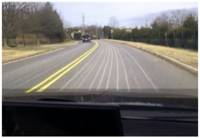
  - Euclidean distance in parameter space

- Stochastic gradient descent (SGD)

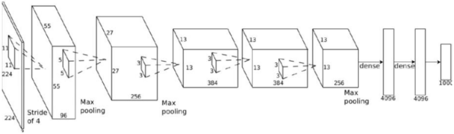
$$\theta \leftarrow \theta + d^*$$

Hard to pick the threshold ϵ

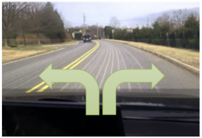
- It is hard to predict the result on the parameterized distribution
  - Especially for nonlinear function approximators, e.g., neural networks



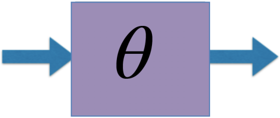
$\mathbf{s}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$



$\mathbf{a}_t$



$\mu_\theta(s)$

$\sigma_\theta(s)$

Gradient descent in distribution space

- Gradient descent in parameter space

$$d^* = \arg\max_{||d|| \leq \epsilon} J(\theta + d)$$

- **Natural gradient descent**: the step size in parameter space is determined by considering the KL divergence in the distributions before and after the update

$$d^* = \arg\max_{d} J(\theta + d), \quad s.t. \, \mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta+d}) \leq \epsilon$$

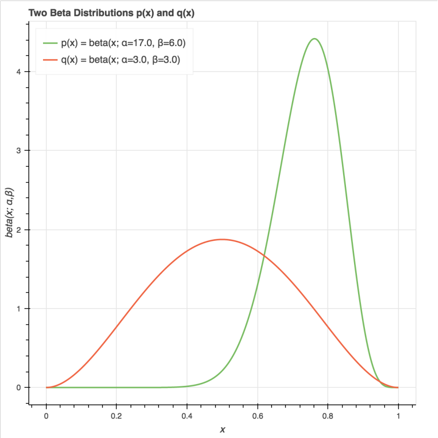
  - **KL divergence** in distribution space
  - Easier to pick the distance threshold!!!

Distance for probability distributions

- How to calculate the distance between two points in a 2D coordinate?

$$distance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

  - Euclidean distance



- How to calculate the distance between two **probability distributions**, $p(x)$ and $q(x)$?

Kullback-Leibler (KL) divergence

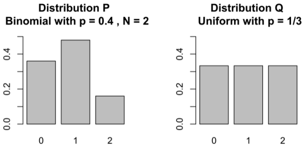
- A measure of how one probability distribution, $p(x)$, is different from a second, reference probability distribution, $q(x)$

$$\mathrm{D}_{\mathrm{KL}}(p(x)||q(x)) = \sum_i p(x_i) \log \frac{p(x_i)}{q(x_i)}$$

$$\mathrm{D}_{\mathrm{KL}}(p(x)||q(x)) = \int_x p(x) \log \frac{p(x)}{q(x)} \,\mathrm{d}x$$

  - A KL divergence of 0 indicates that the two distributions are identical

Kullback-Leibler (KL) divergence: An example



x	0	1	2
Distribution *P*(x)	0.36	0.48	0.16
Distribution *Q*(x)	0.333	0.333	0.333

$$\begin{aligned} D_{\mathrm{KL}}(P \parallel Q) &= \sum_{x \in \mathcal{X}} P(x) \ln\left(\frac{P(x)}{Q(x)}\right) \\ &= 0.36 \ln\left(\frac{0.36}{0.333}\right) + 0.48 \ln\left(\frac{0.48}{0.333}\right) + 0.16 \ln\left(\frac{0.16}{0.333}\right) \\ &= 0.0852996 \end{aligned}$$

$$\begin{aligned} D_{\mathrm{KL}}(Q \parallel P) &= \sum_{x \in \mathcal{X}} Q(x) \ln\left(\frac{Q(x)}{P(x)}\right) \\ &= 0.333 \ln\left(\frac{0.333}{0.36}\right) + 0.333 \ln\left(\frac{0.333}{0.48}\right) + 0.333 \ln\left(\frac{0.333}{0.16}\right) \\ &= 0.097455 \end{aligned}$$

- How to solve this constrained optimization problem?

$$d^* = \arg\max_d J(\theta + d), \quad s.t.\, \mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta+d}) \leq \epsilon$$

- Use the **Lagrangian multiplier** λ, turn to the **unconstrained penalized objective**

$$d^* = \arg\max_d J(\theta + d) - \lambda(\mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta+d}) - \epsilon)$$

Taylor expansion for the unconstrained penalized objective

$$d^* = \arg\max_d J(\theta + d) - \lambda(\mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta+d}) - \epsilon)$$

- First-order Taylor expansion for the loss

$$J(\theta + d) \approx J(\theta) + \nabla_{\theta'} J(\theta')|_{\theta'=\theta} \cdot d$$

- Second-order Taylor expansion for the KL

$$\mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta+d}) \approx \frac{1}{2} d^T \cdot \nabla^2_{\theta'} \mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta'})|_{\theta'=\theta} \cdot d$$

Taylor series/expansion

- A representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$
$$= f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2 + ...$$

- Examples

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...$$
$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + ...$$

Taylor expansion of KL

$$\mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta'}) \approx \mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_\theta) + d^T \nabla_{\theta'} \mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta'})|_{\theta'=\theta} + \frac{1}{2} d^T \nabla^2_{\theta'} \mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta'})|_{\theta'=\theta} d$$

$$\mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta'}) = \int \pi_\theta(x) \log \frac{\pi_\theta(x)}{\pi_{\theta'}(x)} \, \mathrm{d}x = \underbrace{\int \pi_\theta(x) \log \pi_\theta(x) \, \mathrm{d}x}_{\text{independent of } \theta'} - \int \pi_\theta(x) \log \pi_{\theta'}(x) \, \mathrm{d}x$$

$$\begin{aligned}
\nabla_{\theta'} \mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta'})|_{\theta'=\theta} &= -\nabla_{\theta'} \int \pi_\theta(x) \log \pi_{\theta'}(x) \, \mathrm{d}x|_{\theta'=\theta} \\
&= -\int \pi_\theta(x) \nabla_{\theta'} \log \pi_{\theta'}(x) \, \mathrm{d}x|_{\theta'=\theta} \\
&= -\int \frac{\pi_\theta(x)}{\pi_{\theta'}(x)} \nabla_{\theta'} \pi_{\theta'}(x) \, \mathrm{d}x|_{\theta'=\theta} \\
&= -\nabla_{\theta'} \int \pi_{\theta'}(x) \, \mathrm{d}x|_{\theta'=\theta} \\
&= 0
\end{aligned}$$

Taylor expansion of KL

$$\mathrm{D_{KL}}(\pi_\theta || \pi_{\theta'}) \approx \mathrm{D_{KL}}(\pi_\theta || \pi_\theta) + d^T \nabla_{\theta'} \mathrm{D_{KL}}(\pi_\theta || \pi_{\theta'})|_{\theta'=\theta} + \frac{1}{2} d^T \nabla^2_{\theta'} \mathrm{D_{KL}}(\pi_\theta || \pi_{\theta'})|_{\theta'=\theta} d$$

$$\begin{aligned}
\nabla^2_{\theta'} \mathrm{D_{KL}}(\pi_\theta || \pi_{\theta'})|_{\theta'=\theta} &= -\int \pi_\theta(x) \nabla^2_{\theta'} \log \pi_{\theta'}(x)\, \mathrm{d}x|_{\theta'=\theta} \\
&= -\int \pi_\theta(x) \frac{\pi_{\theta'}(x) \nabla^2_{\theta'} \pi_{\theta'}(x) - \nabla_{\theta'} \pi_{\theta'}(x) \nabla_{\theta'} \pi_{\theta'}(x)^T}{\pi_{\theta'}(x)^2}\, \mathrm{d}x|_{\theta'=\theta} \\
&= \underbrace{-\nabla^2_{\theta'} \int \pi_{\theta'}(x) dx|_{\theta'=\theta}}_{0} + \int \pi_\theta(x) \nabla_{\theta'} \log \pi_{\theta'}(x) \nabla_{\theta'} \log \pi_{\theta'}(x)^T \mathrm{d}x|_{\theta'=\theta} \\
&= \mathbb{E}_{x \sim \pi_\theta} [\nabla_{\theta'} \log \pi_{\theta'}(x) \nabla_{\theta'} \log \pi_{\theta'}(x)^T |_{\theta'=\theta}]
\end{aligned}$$

Hessian of KL = Fisher information matrix (FIM)

- **Hessian**: A square matrix of second-order partial derivatives of a scalar-valued function, which describes the local curvature of a function of many variables

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- **Fisher information**: a way of measuring the amount of information that an observable random variable X carries about an unknown parameter θ upon which the probability of X depends

$$\boldsymbol{F}(\theta) = \mathbb{E}_{x \sim \pi_\theta}[\nabla_\theta \log \pi_\theta(x) \nabla_\theta \log \pi_\theta(x)^T]$$

Hessian of KL = Fisher information matrix (FIM)

- The FIM is exactly the **Hessian matrix** of KL divergence

$$\underbrace{\nabla^2_{\theta'} \mathrm{D_{KL}}(\pi_\theta || \pi_{\theta'})|_{\theta'=\theta}}_{\text{Hessian of KL}} = \underbrace{\mathbb{E}_{x\sim\pi_\theta}[\nabla_{\theta'} \log \pi_{\theta'}(x) \nabla_{\theta'} \log \pi_{\theta'}(x)^T |_{\theta'=\theta}]}_{\text{FIM}}$$

$$\begin{aligned} \mathrm{D_{KL}}(\pi_\theta || \pi_{\theta'}) \approx \underbrace{\mathrm{D_{KL}}(\pi_\theta || \pi_\theta)}_{0} + d^T \underbrace{\nabla_{\theta'} \mathrm{D_{KL}}(\pi_\theta || \pi_{\theta'})|_{\theta'=\theta}}_{0} + \frac{1}{2} d^T \underbrace{\nabla^2_{\theta'} \mathrm{D_{KL}}(\pi_\theta || \pi_{\theta'})|_{\theta'=\theta}}_{\boldsymbol{F}(\theta)} d \\ = \frac{1}{2} d^T \boldsymbol{F}(\theta) d \\ = \frac{1}{2} (\theta' - \theta)^T \boldsymbol{F}(\theta)(\theta' - \theta) \end{aligned}$$

Back to Taylor expansion of KL

$$\mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_\theta + d) \approx \frac{1}{2} d^T \boldsymbol{F}(\theta) d$$

- KL divergence is roughly analogous to a distance measure between distributions
- Fisher information serves as a local distance metric between distributions: how much you change the distribution if you move the parameters a little bit in a given direction

Back to solving the KL constrained problem

$$
\begin{aligned}
d^* &= \arg\max_d J(\theta + d) - \lambda(\mathrm{D_{KL}}(\pi_\theta || \pi_{\theta+d}) - \epsilon) \\
&\approx \arg\max_d J(\theta) + \nabla_{\theta'} J(\theta')|_{\theta'=\theta} \cdot d - \lambda(\frac{1}{2} d^T \nabla^2_{\theta'} \mathrm{D_{KL}}(\pi_\theta || \pi_{\theta'})|_{\theta'=\theta} d - \epsilon) \\
&= \arg\max_d \nabla_{\theta'} J(\theta')|_{\theta'=\theta} \cdot d - \frac{1}{2} \lambda d^T \boldsymbol{F}(\theta) d
\end{aligned}
$$

- Set the gradient to 0:

$$
\begin{aligned}
0 &= \frac{\partial}{\partial d} \left(\nabla_{\theta'} J(\theta')|_{\theta'=\theta} \cdot d - \frac{1}{2} \lambda d^T \boldsymbol{F}(\theta) d \right) \\
&= \nabla_{\theta'} J(\theta')|_{\theta'=\theta} - \lambda \boldsymbol{F}(\theta) d
\end{aligned}
$$

$$
\boxed{d^* = \frac{1}{\lambda} \boldsymbol{F}^{-1}(\theta) \nabla_{\theta'} J(\theta')|_{\theta'=\theta} = \frac{1}{\lambda} \boldsymbol{F}^{-1}(\theta) \nabla_\theta J(\theta)}
$$

Natural gradient descent

- The natural gradient:

$$\widetilde{\nabla}_\theta J(\theta) = \boldsymbol{F}^{-1}(\theta)\nabla_\theta J(\theta)$$

- Natural gradient descent:

$$\theta' = \theta + \alpha \cdot \boldsymbol{F}^{-1}(\theta)\hat{g}$$

- How to determine the learning rate α:

$$\mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_\theta + d) \approx \frac{1}{2}(\theta' - \theta)^T \boldsymbol{F}(\theta)(\theta' - \theta) \leq \epsilon$$

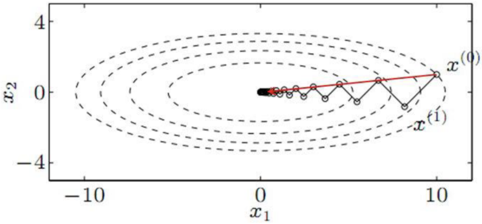
$$\frac{1}{2}(\alpha\hat{g})^T \boldsymbol{F}(\alpha\hat{g}) = \epsilon$$

$$\boxed{\alpha = \sqrt{\frac{2\epsilon}{\hat{g}^T \boldsymbol{F}\hat{g}}}}$$

Geometric interpretation of natural policy gradient

- Find **the steepest direction** for parameter updating

Essentially the same problem as this:



Natural gradient descent → Natural policy gradient (NPG)

Algorithm 1 Natural Policy Gradient

Input: initial policy parameters θ_0
for $k = 0, 1, 2, \ldots$ **do**
Collect set of trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
Form sample estimates for

- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian / Fisher Information Matrix $\hat{H}_k$

Compute Natural Policy Gradient update:

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\epsilon}{\hat{g}_k^T \hat{H}_k \hat{g}_k}} \hat{H}_k^{-1} \hat{g}_k$$

end for

- Originated from **natural gradient descent** in supervised learning
- Very **expensive** to compute the **inverse of Hessian matrix** for a large number of parameters

Review of natural policy gradient

- The gradient
  - Constrain parameter update in parameter space (using Euclidean distance)
- The natural gradient
  - Constrain parameter update in distribution space (using KL divergence)
  - The meaning of "natural": the distance metric is **invariant** to function parameterization
- Fisher information matrix (FIM)
  - Second-order information: a local distance metric between distributions
  - The FIM is exactly the Hessian matrix of KL divergence
  - Expensive to compute for a large number of parameters

Trust region policy optimization (TRPO)

- John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel, **Trust Region Policy Optimization**, ICML, 2015.
- The family of **statistical learning**
  - John Schulman $\rightarrow$ Pieter Abbeel $\rightarrow$ Andrew Ng $\rightarrow$ Michael Jordan

John Schulman's Homepage

I'm a research scientist at OpenAI. I co-lead the reinforcement learning (RL) team, where we work on (1) designing better RL algorithms that enable agents to learn much faster in novel situations; (2) designing better training environments that teach agents transferrable skills. We mostly use games as a testbed.

Previously, I received my PhD in Computer Science from UC Berkeley, where I had the good fortune of being advised by Pieter Abbeel. Prior to my recent work in RL, I spent some time working on robotics, enabling robots to tie knots and stitches and plan movement using trajectory optimization.

- Publications
- Presentations
- Code
- Awards

Email: joschu@openai.com.

Trust region policy optimization (TRPO)

Michael I. Jordan

FOLLOW

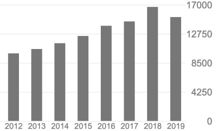
Professor of EECS and Professor of Statistics, University of California, Berkeley
Verified email at cs.berkeley.edu - Homepage

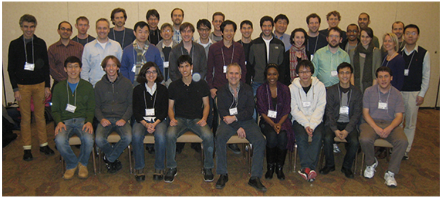
machine learning statistics computational biology artificial intelligence optimization

TITLE	CITED BY	YEAR
Latent dirichlet allocation DM Blei, AY Ng, MI Jordan Journal of machine Learning research 3 (Jan), 993-1022	29247	2003
On spectral clustering: Analysis and an algorithm AY Ng, MI Jordan, Y Weiss Advances in neural information processing systems, 849-856	7927	2002
Adaptive mixtures of local experts. RA Jacobs, MI Jordan, SJ Nowlan, GE Hinton Neural computation 3 (1), 79-87	4089	1991

Cited by VIEW ALL

	All	Since 2014
Citations	165762	84682
h-index	160	114
i10-index	540	425





TRPO - The KL constrained problem

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right]$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta.$$

- Also worth considering using a penalty instead of a constraint

$$\underset{\theta}{\text{maximize}} \qquad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] - \beta \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]]$$

Again the KL penalized problem!

TRPO = NPG + Line search + Monotonic improvement theorem

Algorithm 3 Trust Region Policy Optimization

Input: initial policy parameters θ_0

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Form sample estimates for

- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian-vector product function $f(v) = \hat{H}_k v$

Use CG with n_{cg} iterations to obtain $x_k \approx \hat{H}_k^{-1} \hat{g}_k$

Estimate proposed step $\Delta_k \approx \sqrt{\frac{2\delta}{x_k^T \hat{H}_k x_k}} x_k$

Perform backtracking line search with exponential decay to obtain final update

$$\theta_{k+1} = \theta_k + \alpha^j \Delta_k$$

end for

Line search with monotonic policy improvement

Algorithm 2 Line Search for TRPO

Compute proposed policy step $\Delta_k = \sqrt{\frac{2\delta}{\hat{g}_k^T \hat{H}_k^{-1} \hat{g}_k}} \hat{H}_k^{-1} \hat{g}_k$

for $j = 0, 1, 2, ..., L$ **do**

  Compute proposed update $\theta = \theta_k + \alpha^j \Delta_k$

  if $\mathcal{L}_{\theta_k}(\theta) \geq 0$ and $\bar{D}_{KL}(\theta || \theta_k) \leq \delta$ **then**

    accept the update and set $\theta_{k+1} = \theta_k + \alpha^j \Delta_k$

    break

  end if

end for

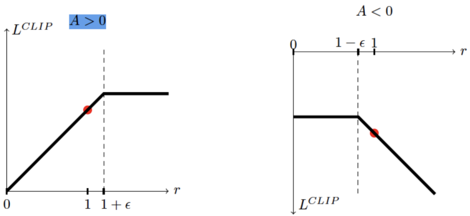
Proximal policy optimization (PPO): Clipped objective

- Recall the surrogate objective

$$L^{IS}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right].$$

- Form a lower bound via clipped importance ratios

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$



- Prevent large changes of policies, constrain the policy update
- Achieve similar performance to TRPO without second-order information (no Fisher matrix!)

Proximal policy optimization (PPO): Adaptive KL penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \boxed{\beta_k \bar{D}_{KL}(\theta || \theta_k)}$$

  by taking K steps of minibatch SGD (via Adam)
  if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \geq 1.5\delta$ **then**
    $\beta_{k+1} = 2\beta_k$
  else if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \leq \delta/1.5$ **then**
    $\beta_{k+1} = \beta_k/2$
  end if
end for

Don't use second order approximation for Kl which is expensive, use standard gradient descent

- Penalty coefficient β changes between iterations to approximately enforce KL-divergence constraint
- Achieve similar performance to TRPO without second-order information (no Fisher matrix!)

Review

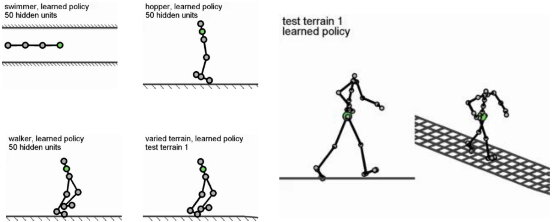
- TRPO: again the KL penalty problem
  - Natural policy gradient + Monotonic policy improvement + Line search
  - Still need to compute the natural gradient with Hessian matrix
- PPO
  - Achieve TRPO-like performance without second-order computation
  - Clipped objective, adaptive KL penalty

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta. \end{aligned}$$

Example: policy gradient with importance sampling

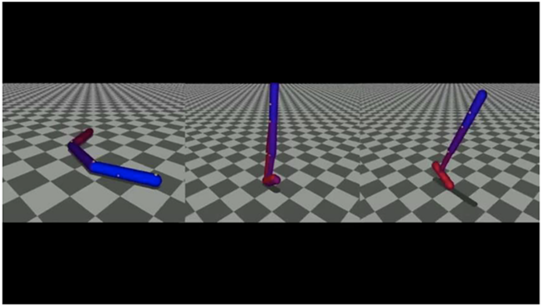
$$\nabla_{\theta'} J(\theta') = E_{\tau \sim \pi_\theta(\tau)} \left[\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \left(\prod_{t'=1}^{t} \frac{\pi_{\theta'}(\mathbf{a}_{t'}|\mathbf{s}_{t'})}{\pi_\theta(\mathbf{a}_{t'}|\mathbf{s}_{t'})} \right) \left(\sum_{t'=t}^{T} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \right) \right]$$

- Incorporate example demonstrations using importance sampling
- Neural network policies



Example: trust region policy optimization

- Natural gradient with automatic step adjustment
- Discrete and continuous actions
- Code available (see Duan et al. '16)



Learning objectives of this lecture

- You should be able to...
  - Understand and be able to use the vanilla policy gradient method
  - Be able to use the baseline to reduce the variance of policy gradient
  - Know the importance sampling technique for off-policy policy gradient
  - Know the implementation tricks in practice
  - Be aware of several advanced algorithms, natural policy gradient, TRPO, PPO
  - Enhance your mathematical skills

References

- Lecture 5 & 9 of CS285 at UC Berkeley, *Deep Reinforcement Learning, Decision Making, and Control*
  - `http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-5.pdf`
  - `http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-9.pdf`
- Classic papers
  - Williams (1992). **Simple statistical gradient following algorithms for connectionist reinforcement learning**: introduces REINFORCE algorithm.
  - Peters & Schaal (2008). **Reinforcement learning of motor skills with policy gradients**: very accessible overview of optimal baselines and natural gradient.
- DRL policy gradient papers
  - Schulman, L., Moritz, Jordan, Abbeel (2015). **Trust region policy optimization**: deep RL with natural policy gradient and adaptive step size.
  - Schulman, Wolski , Dhariwal , Radford, Klimov (2017). **Proximal policy optimization algorithms**: deep RL with importance sampled policy gradient.
  - Y. Duan, et al., **Benchmarking Deep Reinforcement Learning for Continuous Control**, *ICML*, 2016.
  - Z. Wang, et al., **Incremental Reinforcement Learning in Continuous Spaces via Policy Relaxation and Importance Weighting**, *TNNLS*, 2019.

THE END