Lecture 7: Advanced Policy Gradients

Zhi Wang & Chunlin Chen

Department of Control and Systems Engineering
Nanjing University

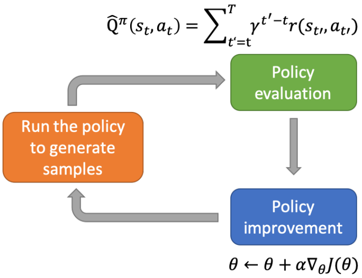
Nov. 15th, 2021

Table of Contents

Review: Vanilla policy gradient (REINFORCE)

REINFORCE algorithm: Loop:

1. sample $\{\tau^i\}$ from $\pi_\theta(a_t|s_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=t}^{T}\gamma^{t'-t}r(s_t^i, a_t^i)$
3. $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$


$\hat{Q}^\pi(s_t, a_t) = \sum_{t'=t}^{T}\gamma^{t'-t}r(s_{t'}, a_{t'})$
Policy evaluation
Run the policy to generate samples
Policy improvement
$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$

Problems of vanilla policy gradient (REINFORCE)

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

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

- 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

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

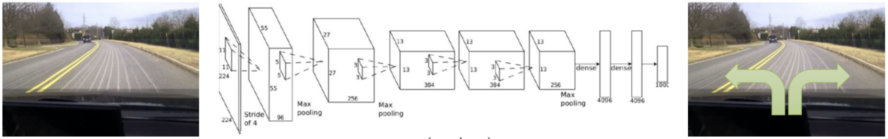
  - Euclidean distance in parameter space

- Stochastic gradient descent (SGD)

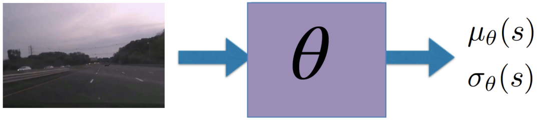
$$\theta \leftarrow \theta + d^*$$

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

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

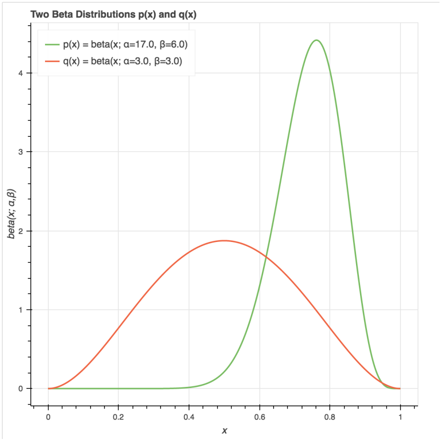
  - **KL divergence** in distribution space
  - Easier to pick the distance threshold!!!

Distance for probability distributions

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

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

  - Euclidean distance



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

Kullback-Leibler (KL) divergence

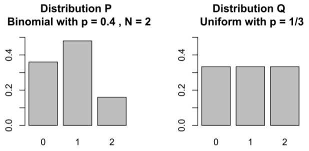
- A measure of how one probability distribution, $p(x)$, is different from a second, reference probability distribution, $q(x)$

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

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

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

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

$$\begin{aligned} D_{\mathrm{KL}}(P \parallel Q) &= \sum_{x\in\mathcal{X}} P(x)\ln\left(\frac{P(x)}{Q(x)}\right) \\ &= 0.36\ln\left(\frac{0.36}{0.333}\right) + 0.48\ln\left(\frac{0.48}{0.333}\right) + 0.16\ln\left(\frac{0.16}{0.333}\right) \\ &= 0.0852996 \\ D_{\mathrm{KL}}(Q \parallel P) &= \sum_{x\in\mathcal{X}} Q(x)\ln\left(\frac{Q(x)}{P(x)}\right) \\ &= 0.333\ln\left(\frac{0.333}{0.36}\right) + 0.333\ln\left(\frac{0.333}{0.48}\right) + 0.333\ln\left(\frac{0.333}{0.16}\right) \\ &= 0.097455 \end{aligned}$$

KL divergence: A test

- Suppose two Gaussian distributions:

$$p(x) \sim \mathcal{N}(\mu_1, \sigma_1^2), \qquad q(x) \sim \mathcal{N}(\mu_2, \sigma_2^2)$$

- What is $\mathrm{D}_{\mathrm{KL}}(p(x)||q(x))$?

$$\log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

KL divergence between two Gaussians

$$p(x) \sim \mathcal{N}(\mu_1, \sigma_1^2), \quad \mathbb{E}_{p(x)}[x] = \mu_1, \quad var_{p(x)}[x] = \mathbb{E}[(x-\mu_1)^2] = \sigma_1^2$$

$$\begin{aligned}
\mathrm{D_{KL}}(p(x)||q(x)) &= \mathbb{E}_{p(x)}[\log p(x) - \log q(x)] \\
&= \mathbb{E}_{p(x)}\left[-\log(\sqrt{2\pi\sigma_1}) - \frac{(x-\mu_1)^2}{2\sigma_1^2} + \log(\sqrt{2\pi\sigma_2}) + \frac{(x-\mu_2)^2}{2\sigma_2^2}\right] \\
&= \log\frac{\sigma_2}{\sigma_1} - \frac{\mathbb{E}_{p(x)}[(x-\mu_1)^2]}{2\sigma_1^2} + \frac{\mathbb{E}_{p(x)}[(x-\mu_1+\mu_1-\mu_2)^2]}{2\sigma_2^2} \\
&= \log\frac{\sigma_2}{\sigma_1} - \frac{\sigma_1^2}{2\sigma_1^2} + \frac{\mathbb{E}_{p(x)}[(x-\mu_1)^2 + 2(x-\mu_1)(\mu_1-\mu_2) + (\mu_1-\mu_2)^2]}{2\sigma_2^2} \\
&= \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2}
\end{aligned}$$

Table of Contents

Back to natural gradient descent

- How to solve this constrained optimization problem?

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

- What tool to use?
  - Turn the constrained optimization problem to an unconstrained one?

Lagrangian multiplier

- 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

$$
\begin{aligned}
f(x) &= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \\
&= f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2 + \dots
\end{aligned}
$$

- Examples

$$
e^x = ?
$$

$$
\frac{1}{1-x} = ?
$$

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

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

- 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 $J(\theta + d)$

- Let $\theta' = \theta + d$ is the independent variable
- That is, $x = \theta'$, $a = \theta$, $x - a = d$
- **What is the Taylor expansion of $J(\theta + d)$?**

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

Taylor expansion of $J(\theta + d)$

- Let $\theta' = \theta + d$ is the independent variable
- That is, $x = \theta'$, $a = \theta$, $x - a = d$
- **What is the Taylor expansion of** $J(\theta + d)$**?**

- First-order Taylor expansion for the loss:

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

Taylor expansion of KL

- Let $\theta' = \theta + d$ is the independent variable
- That is, $x = \theta'$, $a = \theta$, $x - a = d$
- **What is the Taylor expansion of** $\mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta+d})$**?**

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

Taylor expansion of KL

- Let $\theta' = \theta + d$ is the independent variable
- That is, $x = \theta'$, $a = \theta$, $x - a = d$
- **What is the Taylor expansion of $\mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta+d})$?**

- Second-order Taylor expansion for $\mathrm{D}_{\mathrm{KL}}(\pi_\theta || \pi_{\theta'})$:

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

Taylor expansion of KL

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

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

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

Taylor expansion of KL

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

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

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

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

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

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

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

- Set the gradient to 0:

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

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

Table of Contents

Natural gradient descent

- The natural gradient:

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

- Natural gradient ascent:

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

- How to determine the learning rate α:

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

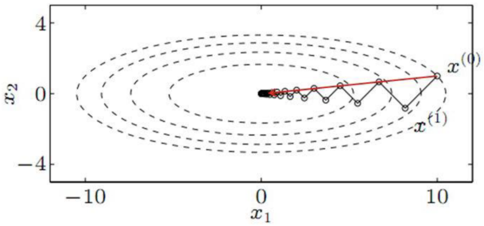
$$\frac{1}{2}(\alpha\hat{g})^T \boldsymbol{F}(\alpha\hat{g}) = \epsilon$$

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

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 $\theta_0$
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

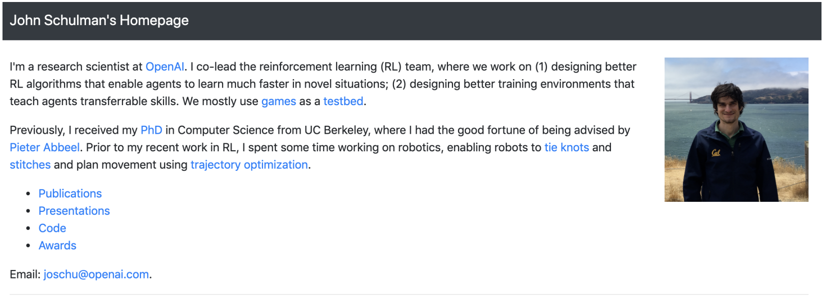
Review of natural policy gradient

- 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

Table of Contents

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

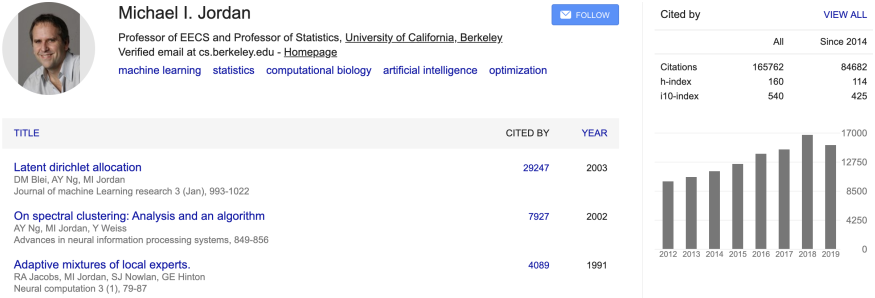
I'm a research scientist at OpenAI. I co-lead the reinforcement learning (RL) team, where we work on (1) designing better RL algorithms that enable agents to learn much faster in novel situations; (2) designing better training environments that teach agents transferrable skills. We mostly use games as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

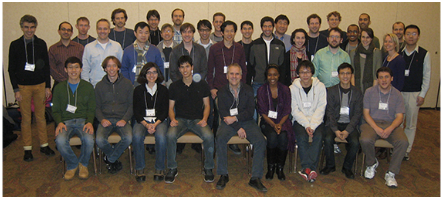
Professor of EECS and Professor of Statistics, University of California, Berkeley
Verified email at cs.berkeley.edu - Homepage

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

- The objective function:

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

$$\text{subject to} \quad \hat{\mathbb{E}}_t \left[\mathrm{D}_{\mathrm{KL}}[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)] \right] \leq \delta$$

- Also worth considering using a penalty instead of a constraint:

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

  - Again the KL penalized problem!

TRPO = NPG + Line search + Monotonic improvement theorem

Algorithm 3 Trust Region Policy Optimization

```
Input: initial policy parameters $\theta_0$
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
```

- Still very **expensive** to compute the **inverse of Hessian matrix** for a large number of parameters

Table of Contents

Proximal policy optimization (PPO): Clipped objective

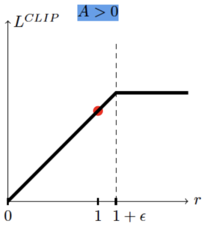
- The surrogate objective function:

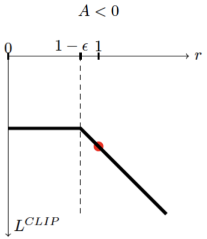
$$\mathcal{L}^{\mathsf{IS}}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t [r_t(\theta)\hat{A}_t]$$

- Form a lower bound via clipped importance ratios

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

  - 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 $\theta_0$, initial KL penalty $\beta_0$, target KL-divergence $\delta$
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

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

$$\text{subject to} \quad \hat{\mathbb{E}}_t \left[\mathrm{D}_{\mathrm{KL}}[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)] \right] \le \delta$$

Learning objectives of this lecture

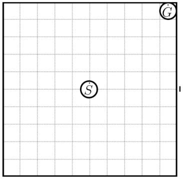
- You should be able to...
  - Know how to derive the natural policy gradient
  - Be aware of several advanced algorithms, e.g., TRPO, PPO
  - Enhance your mathematical skills

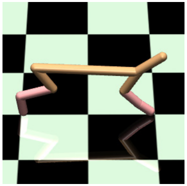
References

- Lecture 9 of CS285 at UC Berkeley, *Deep Reinforcement Learning, Decision Making, and Control*
  - `http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-9.pdf`
- Classic papers
  - 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.

Homework 3

- Study the policy gradient algorithm in detail
- Implement the series of policy gradient algorithms on problems 1 & 2
  - Problem 1: the point maze navigation, continuous state-action space ($s, a \in \mathbb{R}^2$, $s \in [-0.5, 0.5]^2, a \in [-0.1, 0.1]^2$)
  - Problem 2: the MuJoCo HalfCheetah, make the robot run forward
  - Must use vanilla policy gradient and natural policy gradient, encourage to use TRPO and PPO
- Write a report introducing the algorithms and your experimentation
  - Explanations, steps, evaluation results, visualizations...
  - Submit the code and the report to *mg20150005@smail.nju.edu.cn*





THE END