

Reinforcement Learning : Wrap Up

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : easwar.subramanian@tcs.com / cs5500.2020@iith.ac.in

Novemer 29, 2021

- 1 On Invariance of Natural Gradients : A Brief Look
- 2 Gradient Descent and Parameterization
- 3 Landscape, Summary and References
- 4 Other Topics
- 5 Practical Tips, – Based on John Schulman's talk on Nuts and Bolts of Deep RL

On Invariance of Natural Gradients : A Brief Look

- ▶ Natural policy gradient algorithm gives an update-rule in which updates are pre-multiplied by H^{-1}
- ▶ The Hessian of the KL-divergence is the Fischer Information Matrix given by

$$F = \mathbb{E}_{\pi_{\theta}} \left[\nabla_{\theta} \log \pi_{\theta}(\cdot|s) \nabla_{\theta} \log \pi_{\theta}(\cdot|s)^T \right]$$

which now can be **computed from samples**

- ▶ The NPG direction $H^{-1}g$ is **co-variant**; i.e. it points in same direction irrespective of the parametrization that is used to compute it

Gradient Descent and Parameterization

Consider the following model that captures the joint distribution between two random variables x and y

$$p_{\theta}(x, y) = p(x)\mathcal{N}(y \mid \theta x + b, \sigma^2), \theta \in \mathbb{R},$$

where θ is a parameter and b and σ are constants.

Let $\theta = 2\mu$. We can rewrite the above model in terms of new parameter μ as

$$p_{\mu}(x, y) = p(x)\mathcal{N}(y \mid 2\mu x + b, \sigma^2), \mu \in \mathbb{R}.$$

Note that even if $p_{\theta}(x, y)$ and $p_{\mu}(x, y)$ have different analytical forms, they represent the same (family) distribution. Specifically,

$$p_{\theta=a}(x, y) \equiv p_{\mu=a/2}(x, y), \text{ for any } a \in \mathbb{R}$$

Construct a negative log-likelihood as loss function for the distribution $p(x, y)$ as

$$\begin{aligned}\mathcal{L}(p(x, y)) &= -[\log p(x, y)] \\ &= \left[\frac{1}{2\sigma^2} (\theta x + b - y)^2 + \log \sqrt{2\pi}\sigma \right] \quad (\text{for } p_\theta(x, y)) \\ &= \left[\frac{1}{2\sigma^2} (2\mu x + b - y)^2 + \log \sqrt{2\pi}\sigma \right] \quad (\text{for } p_\mu(x, y))\end{aligned}$$

- For $p_{\theta}(x, y)$ the update rule is given by $\theta_{t+1} = \theta_t - \alpha \nabla_{\theta_t} \mathcal{L}(p_{\theta_t}(x, y))$ where α is the step size with

$$\nabla_{\theta_t} \mathcal{L}(p_{\theta_t}(x, y)) = \left[\frac{x}{\sigma^2} (\theta_t x + b - y) \right]$$

- For $p_{\mu}(x, y)$ the update rule is given by $\mu_{t+1} = \mu_t - \alpha \nabla_{\mu_t} \mathcal{L}(p_{\mu_t}(x, y))$ where α is the step size with

$$\nabla_{\mu_t} \mathcal{L}(p_{\mu_t}(x, y)) = \left[\frac{2x}{\sigma^2} (2\mu_t x + b - y) \right]$$

- ▶ Let $\theta_t = 2\mu_t = a$ for some $a \in \mathbb{R}$
- ▶ That is, at the t -th step of the gradient descent, the parametric probabilistic models $p_{\theta=a}(x, y)$ and $p_{\mu=a/2}(x, y)$ represent the same distribution
- ▶ Because

$$\nabla_{\theta_t} \mathcal{L}(p_{\theta_t}(x, y))|_{\theta_t=a} = \frac{1}{2} \nabla_{\mu_t} \mathcal{L}(p_{\mu_t}(x, y))|_{\mu_t=a/2}$$

we will have,

$$\begin{aligned}\theta_{t+1} &= \theta_t - \alpha \nabla_{\theta_t} \mathcal{L}(p_{\theta_t}(x, y)) \\ &= 2\mu_t - \frac{\alpha}{2} \nabla_{\mu_t} \mathcal{L}(p_{\mu_t}(x, y)) \\ &\neq 2\mu_t - 2\alpha \nabla_{\mu_t} \mathcal{L}(p_{\mu_t}(x, y)) \\ &= 2\mu_{t+1}\end{aligned}$$

- ▶ Hence the $t + 1$ -th optimization step will result in different probabilistic models $p_{\theta_{t+1}}(x, y) \neq p_{\mu_{t+1}}(x, y)$

- ▶ One step of gradient descent will result in different models depending on which parameterization is used
- ▶ Hence, requires carefully choosing the parameterization to avoid hindering optimization
- ▶ Not all optimization procedures are parameterization dependent. For example, the Newton-Raphson method is invariant to affine transformations of model parameters
- ▶ Natural gradient methods are invariant to arbitrary differentiable transformations of model parameters when the learning rate is small enough

Sanity Check : Natural Gradients

Recall that the natural gradient of a loss function is given by,

$$\tilde{\nabla}_{\theta} \mathcal{L}(p_{\theta}(x, y)) := \mathbf{F}_{\theta}^{-1} \nabla_{\theta} \mathcal{L}(p_{\theta}(x, y)),$$

where

$$[\mathbf{F}_{\theta}]_{i,j} := \mathbb{E}_{p_{\theta}(x,y)} \left[\left(\frac{\partial}{\partial \theta_i} \log p_{\theta}(x, y) \right) \left(\frac{\partial}{\partial \theta_j} \log p_{\theta}(x, y) \right) \right],$$

For different parameterizations of Gaussian conditional distributions, the derivatives of the log densities are respectively

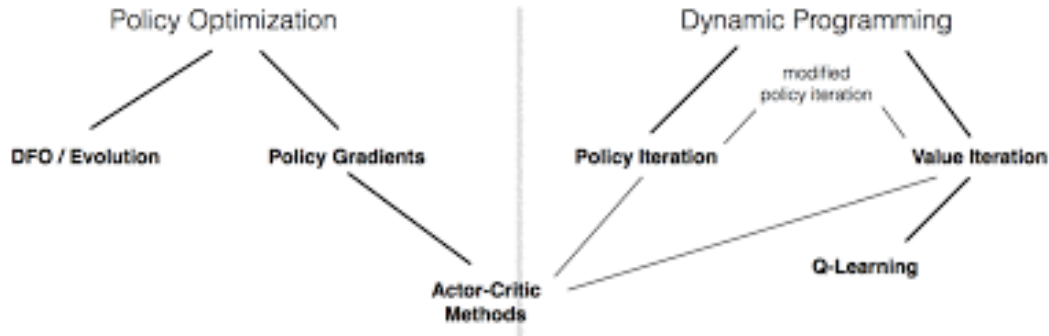
$$\begin{aligned} \frac{\partial}{\partial \theta} \log p_{\theta}(x, y) &= \frac{(y - \theta x - b)x}{\sigma^2} \\ \frac{\partial}{\partial \mu} \log p_{\mu}(x, y) &= \frac{2(y - 2\mu x - b)x}{\sigma^2} \end{aligned}$$

With $\theta_t = 2\mu_t$, we can conclude that $\mathbf{F}_{\mu=\mu_t} = 4\mathbf{F}_{\theta=\theta_t}$. Using

$$\tilde{\nabla}_{\mu} \mathcal{L}(p_{\mu}(x, y))|_{\mu=\mu_t} = \frac{1}{2} \tilde{\nabla}_{\theta} \mathcal{L}(p_{\theta}(x, y))|_{\theta=\theta_t}$$

we can conclude that, $\theta_{t+1} = 2\mu_{t+1}$

Landscape, Summary and References



Markov Property, transition probabilities, Markov reward process, Markov decision process

Three Key entities of RL

- ▶ Value Function - V
- ▶ Action Value Function - Q
- ▶ Policy - π

Optimal policies, notion of greedy policy, Bellman equations (evaluation and optimality)

Lecture Numbers : 1 to 5

Reference : David Silver's Lecture on RL

Key Algorithms

- ▶ Value Iteration
- ▶ Policy Iteration

Drawbacks

- ▶ Requires full prior knowledge of the dynamics of the environment
- ▶ Can be implemented only on small, discrete state spaces

Lecture Numbers : 6 and 7

Reference : David Silver's Lecture on RL

Proofs on convergence available at : <https://runzhe-yang.science/2017-10-04-contraction/>

Notion of bootstrap, lookahead and backup

Evaluation Algorithms

- ▶ Monte-Carlo methods (First Visit and Every Visit MC)
- ▶ Temporal difference methods
- ▶ TD- λ methods

Control Algorithms

- ▶ SARSA
- ▶ Watkin's Q-learning algorithm

Drawback : Not extendible to high dimensional state and action spaces

Lecture Numbers : 8 to 13

**References : David Silver's Lecture on RL and Relevant Chapters on Sutton
and Barto Book**

Use of neural nets as function approximators, convergence of NN based algorithms

Algorithms

- ▶ Monte Carlo based value function estimation
- ▶ Fitted V iteration and Q iteration
- ▶ Deep Q networks

Lecture Numbers : 14 to 16

References : Deep RL course in Berkeley (2017,2018), Deep RL Bootcamp, Minh (2015), Riedmiller(2006)

Notion of Policy gradients, derivation of policy gradient expression, temporal structure, baseline and discounting for variance reduction, advantage function, deterministic policy gradient

Key Algorithms

- ▶ Actor-critic algorithms A2C and A3C
- ▶ DDPG

Lecture Numbers : 17 to 20

References : Deep RL course in Berkeley (2017,2018), Deep RL Bootcamp, Minh (2016), Lillicrap(2016)

Different approach to policy gradients by looking at distance in policy space;
Surrogate loss function; Constrained policy optimization

Key Algorithms

- ▶ Natural Policy Gradient
- ▶ TRPO
- ▶ PPO

Lecture Numbers : 24 to 25

References : Deep RL course in Berkeley (2017,2018), Deep RL course in Berkeley, Joshua Aicham lecture by the same topic, NPG(Kakade, 2001), TRPO (Schulman2015) PPO (Schulman, 2017)









Bandit Concepts :

Naive Exploration, Optimistic Initialization, Optimism in the face of Uncertainty

Key Algorithms

- ▶ UCB and Thompson Sampling
- ▶ Monte Carlo Tree Search (Tree search methods)

Lecture Numbers : 21 to 23 References : David Silver's Lecture on RL and Relevant Chapters on Sutton and Barto Book

-  Reinforcement Learning : Sutton and Barto
-  Dynamic Programming and Optimal Control (I and II) by Bertsekas
-  Reinforcement Learning and Optimal Control, Bertsekas and Tsitsiklis
-  David Silver's course on Reinforcement Learning
-  Stanford course on Deep RL
-  Deep RL BootCamp (Pieter Abeel)
-  John Schulman's lectures in Policy Gradient Methods
-  ... and many others

Other Topics

- ▶ **Central Question :** How can we make decisions better if we know system dynamics ? (possibly when state and action space is high dimensional or continuous)
 - ★ Games, navigating car etc, simulated environments
- ▶ If system dynamics in not known, can we identify them ?
 - ★ System identification - fit unknown parameters to a known model
 - ★ Learning - fit a general purpose model to observed transitions

Forward Reinforcement Learning

Given states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, reward function $\mathcal{R}(s, a)$ and possibly transition probabilities $P(s'|s, a)$ and

- Learn policy $\pi^*(a|s)$

Inverse Reinforcement Learning

Given states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, a policy $\pi(a|s)$ and possibly transition probabilities $P(s'|s, a)$ and

- Learn reward function $\mathcal{R}(s, a)$

and then use it learn $\pi^*(a|s)$

- ▶ Forward transfer : train on one task, transfer to a new task
- ▶ Multi-task transfer : train on many task, transfer to a new task
- ▶ Multi-task meta learning : learn to learn from many tasks

Question

How can we better utilize our computational resources to accelerate RL progress ?

Examples

- ▶ DQN and its variants (Large scale RL) (2013)
- ▶ GORILLA (2015)
- ▶ A3C (2016)
- ▶ IMPALA (2018)

- ▶ Topics like Hierarchical RL, Feudal RL etc
- ▶ Imitation Learning
- ▶ Partially Observable MDPs
- ▶ Multi-agent RL

- ▶ Repository of multitude of environments
- ▶ Baseline implementation of several popular algorithms

Practical Tips, – Based on John Schulman's talk on Nuts and Bolts of Deep RL

- ▶ Test on small use cases and then run on medium-sized problems
- ▶ Interpret and visualize learning process: state visitation, value function, etc
- ▶ Construct toy problems where your idea will be strongest and weakest, where you have a sense of what it should do

- ▶ Progressively increase the state and action space formulation
- ▶ Reward shaping is crucial to test the working of algorithm

- ▶ Explore sensitivity to each parameter
- ▶ Health indicators
 - ★ Quality of value function
 - ★ Entropy of the policy
 - ★ KL diagnostics
- ▶ Run a battery of benchmarks

- ▶ Compare against baselines
 - ★ Cross entropy method
 - ★ Well tuned policy gradient method
 - ★ Well tuned Q-learning or SARSA based method
- ▶ Use multiple random seeds
- ▶ Don't be deterred by published works
 - ★ TRPO on Atari: 100K timesteps per batch for $KL=0:01$
 - ★ DQN on Atari: update freq=10K, replay buffer size=1M

- ▶ DQN converges slowly - for ATARI it is often necessary to wait for 10-40 million frames (couple of hours to a day of training on GPU) to see results significantly better than random policy. **Be Patient**
- ▶ Optimize memory usage carefully: you'll need it for replay buffer
- ▶ Learning rate and exploration schedules are vital
- ▶ Do use Double DQN with prioritized experience replay – significant improvement

- ▶ Policy Initialization : More important than in supervised learning: determines initial state visitation
- ▶ KL spike \rightarrow drastic loss of performance
- ▶ Not recommended to use DDPG when you have discrete action set

- ▶ Automate your experiments;
- ▶ Techniques from supervised learning don't necessarily work in RL: batch norm, dropout, big networks
- ▶ Read older textbooks and theses, not just conference papers

- ▶ Games
- ▶ Robotics
- ▶ Wealth Management
- ▶ Supply Chain Management
- ▶ Control Systems Applications

- ▶ Risk Sensitive RL
- ▶ Optimizing Expected Reward with Constraints
- ▶ Multi Agent Systems
- ▶ Transfer and Meta Learning in RL
- ▶ Improving Data efficiency in RL

- ▶ Things that we can all do (Walking) (Evolution, may be)
- ▶ Things that we learn (driving a bicycle, car etc)
- ▶ We learn a huge variety of things (music, sport, arts etc)
- ▶ We can learn 'difficult' tasks as well

We are still far from building a 'reasonable' intelligent system

- ▶ We are taking baby steps towards the goal of building intelligent systems
- ▶ **Reinforcement Learning (RL) is one of the important paradigm towards that goal**

Thank You and Good Luck