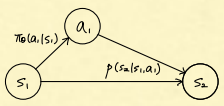
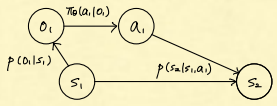
DRL Lec 1

- Motivation
  - ⇒ deep learning helps us handle unstructured environments
    (it can process complex sensory inputs)
  - ⇒ reinforcement learning provides a formalism for behavior
    (it can choose complex actions)
  - ∴ deep models are what allow reinforcement learning algorithms to solve complex problems end-to-end
- Strength of RL on real-world sequential decision making
  - ⇒ reinforcement learning can be naturally integrated with artificial neural networks to obtain high quality generalization resulting in a significant learning speedup
  - ⇒ reinforcement learning agents can save many learning trials by using an action model which can be learned online.
  - ⇒ reinforcement learning agents can take advantages of instructive training instances provided by human teachers resulting in a significant learning speedup
  - ⇒ reinforcement learning agents can significantly reduce learning time by hierarchical learning (first solve elementary learning problems and then by combining the solutions, solve a complex one)
  - ⇒ reinforcement learning agents can deal with non-Markovian environments by having a memory of their past
- What DRL can do
  - ⇒ acquire high degree of proficiency in domains governed by simple, known rules
  - ⇒ learn simple skills with raw sensory inputs, given enough experience
  - ⇒ learn from imitating enough human provided expert behavior.
- Challenges
  - ⇒ slow convergence, transfer learning, reward function, role of prediction
- ☆ Words from Alan Turing
  : Instead of trying to produce a program to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education, one would obtain an adult brain

DRL Lec 2

- Supervised learning of behaviors
  1. Notations
     - ⇒ s_t : state, o_t : observation, a_t : action
     - ⇒ $\pi_\theta(a_t|s_t)$: fully observed policy
     - ⇒ $\pi_\theta(a_t|o_t)$: partially observed policy





  2. Imitation learning (≈ behavior cloning)
     : training data $D=\{o_t,a_t\}_{t=1}^N$ is made by human experts, then the policy $\pi_\theta(a_t|o_t)$ is trained via supervised learning framework.
     - ⇒ problems : $p_{data}(o_t) \neq p_{\pi_\theta}(o_t)$ (≈ distributional drift)
       - → little mistake in the beginning results in large error at the end.
         (∵ it gets to states that have never seen by the human experts)
     - ⇒ solutions
       - → DAgger (Dataset Aggregation)
         - ⇒ train on several trajectories that involve mistakes and corrections
         - ⇒ collect training data from $p_{\pi_\theta}(o)$ instead of $p_{data}(o)$ so that $p_{data}(o) \approx p_{\pi_\theta}(o)$
         1. train $\pi_\theta(a_t|o_t)$ from human data $D=\{o_t,a_t\}_{t=1}^N$
         2. run $\pi_\theta(a_t|o_t)$ to get dataset $D_\pi=\{o_i\}_{i=1}^M$
         3. ask human to label D_π with actions $\{a_i\}_{i=1}^M$ → limitation
         4. aggregate $D \leftarrow D \cup D_\pi$
       - → others : hacks, better models, stable trajectory distribution
  3. Cause of failure
     - ⇒ Non-markovian behavior
       - → behavior depends on all the past observations
       - → may be resolved by using RNN on outputs of several past observations
         (but, this may also fail by causal confusion)
     - ⇒ Multi-modal behavior
       - → may be resolved by the follows.
         1. output mixture of gaussians → $\omega_1,\mu_1,\Sigma_1,\cdots,\omega_N,\mu_N,\Sigma_N$
         2. latent variable models → $\xi \sim N(0,I)$
         3. autoregressive discretization → discretize high dimensional action space by one dimension at a time
  4. Theoretical analysis
     - ⇒ Assumption → $c(s,a)=\begin{cases}0 & \text{if } a=\pi^*(s)\\ 1 & \text{otherwise}\end{cases}$
       1. $\pi_\theta(a\neq\pi^*(s)\,|\,s)\le\varepsilon \quad \forall s\in D_{train}$
          ⇒ $E\left[\sum_t c(s_t,a_t)\right] \le \varepsilon T+(1-\varepsilon)\cdot\varepsilon(T-1)+(1-\varepsilon)^2\varepsilon(T-2)+\cdots = O(\varepsilon T^2)$
          (once it makes mistake, it ends up facing cost every time)
       2. $\pi_\theta(a\neq\pi^*(s)\,|\,s)\le\varepsilon \quad \forall s\sim p_{train}(s)$ (Only from human)
          ⇒ $p_{\pi_\theta}(s_t)=(1-\varepsilon)^t p_{train}(s_t)+(1-(1-\varepsilon)^t)\,p_{mistake}(s_t)$
          ⇒ $\sum_{s_t}\left|p_{\pi_\theta}(s_t)-p_{train}(s_t)\right| = \sum_{s_t}(1-(1-\varepsilon)^t)\left|p_{mistake}(s_t)-p_{train}(s_t)\right| \le 2(1-(1-\varepsilon)^t) \le 2\varepsilon t$
          ⇒ $E\left[\sum_t c(s_t,a_t)\right] = \sum_t\sum_{s_t} p_{\pi_\theta}(s_t)\,c(s_t,a_t) \le \sum_t\sum_{s_t} p_{train}(s_t)\,c(s_t,a_t)+\left|p_{\pi_\theta}(s_t)-p_{train}(s_t)\right| c_{max} \le \sum_t \varepsilon+2\varepsilon t = O(\varepsilon T^2)$
       3. DAgger ($p_{\pi_\theta}(s) \to p_{train}(s)$)
          ⇒ $E\left[\sum_t c(s_t,a_t)\right] \le \varepsilon T = O(\varepsilon T)$
          (even it makes mistake, it immediately face correction)

DRL Lec 3

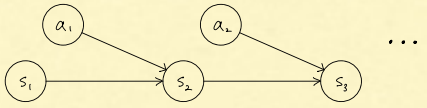
- Introduction to reinforcement learning

1. Markov Decision Process ($M = \{S, A, T, r\}$)

⇒ S : state space, A : action space, T : transition operator, r : reward

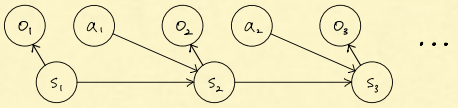
($\mu_{t,j} = p(s_t = j)$, $\xi_{t,k} = p(a_t = k)$, $T_{i,j,k} = p(s_{t+1} = i \mid s_t = j, a_t = k)$, $r(s_t, a_t)$)

($\mu_{t+1,i} = \sum_{j,k} T_{i,j,k} \mu_{t,j} \xi_{t,k}$)



2. Partially observed Markov Decision Process ($M = \{S, A, O, T, \mathcal{E}, r\}$)

⇒ O : observation space, $\mathcal{E}$: emission probability ($p(o_t \mid s_t)$)

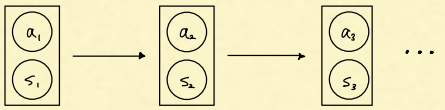


3. Reinforcement learning

⇒ $p_\theta(\tau) = p(s_1, a_1, \cdots, s_T, a_T)$

$= p(s_1) \cdot \pi_\theta(a_1 \mid s_1) \cdot \prod_{t=1}^{T-1} p(s_{t+1} \mid s_t, a_t) \pi_\theta(a_{t+1} \mid s_{t+1})$

$= p_\theta(s_1, a_1) \prod_{t=1}^{T-1} p_\theta(s_{t+1}, a_{t+1} \mid s_t, a_t)$ → markov chain on (s_t, a_t)



($p_\theta(s_t, a_t)$ converges to stationary distribution $p_\theta(s, a)$ under ergodicity)

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

$= \arg\max_\theta \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)} \left[r(s_t, a_t)\right]$ → finite horizon case

$\approx \arg\max_\theta \mathbb{E}_{(s,a) \sim p_\theta(s,a)} \left[r(s, a)\right]$ as $T \to \infty$ → infinite horizon case

4. Estimating reward

Q-function ($Q^\pi(s_t, a_t)$)

: $Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} \mathbb{E}_\pi \left[r(s_{t'}, a_{t'}) \mid s_t, a_t\right] = r(s_t, a_t) + \mathbb{E}_{s_{t+1}} \left[V^\pi(s_{t+1})\right]$

Value-function ($V^\pi(s_t)$)

: $V^\pi(s_t) = \sum_{t'=t}^{T} \mathbb{E}_\pi \left[r(s_{t'}, a_{t'}) \mid s_t\right] = \mathbb{E}_{a_t} \left[Q^\pi(s_t, a_t)\right]$

⇒ $\mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} r(s_t, a_t)\right] = \mathbb{E}_{s_1 \sim p(s_1)} \left[\mathbb{E}_{a_1 \sim \pi_\theta(a_1 \mid s_1)} \left[r(s_1, a_1) + \mathbb{E}_{s_2 \sim p(s_2 \mid s_1, a_1)} \left[r(s_2, a_2) + \cdots\right]\right]\right]$

$= \mathbb{E}_{s_1 \sim p(s_1)} \left[\mathbb{E}_{a_1 \sim \pi_\theta(a_1 \mid s_1)} Q^{\pi_\theta}(s_1, a_1)\right] = \mathbb{E}_{s_1 \sim p(s_1)} \left[V^{\pi_\theta}(s_1)\right]$

→ idea 1 : for all t, set $\pi'(a \mid s_t) = 1$ when $a = \arg\max_{a_t} Q^\pi(s_t, a_t)$

→ idea 2 : modify $\pi(a_t \mid s_t)$ by computing gradients when $Q^\pi(s_t, a_t) > V^\pi(s_t)$

- Algorithms

1. Policy gradients

① estimate $R_\tau = \mathbb{E}\left[\sum_t r(s_t, a_t)\right]$

② $\theta \leftarrow \theta + \alpha \nabla_\theta R_\tau$

2. Value-based

① fit $V^\pi(s)$ or $Q^\pi(s, a)$

② set $\pi'(a \mid s_t) = 1$ when $a = \arg\max_{a_t} Q^\pi(s_t, a_t)$

3. Actor-critic

① fit $V^\pi(s)$ or $Q^\pi(s, a)$

② $\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}\left[Q^\pi(s_t, a_t)\right]$

4. Model-based

① learn f_ϕ s.t. $s_{t+1} \approx f_\phi(s_t, a_t)$

② planning (eg Monte Carlo Tree search)
learn a value function (e.g. dynamic programming)
...

- Comparison

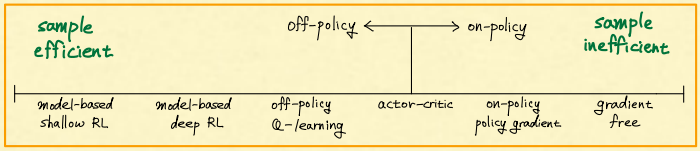
1. Sample efficiency (≠ wall clock time)

⇒ on-policy → inefficient

: each time the policy is changed, new samples need to be generated

⇒ off-policy

: each time the policy is changed, new samples need not to be generated



2. Stability and ease of use

⇒ no guarantee to converge when not using gradient methods

① policy gradient → high variance

② value-based → fitted value may not equal to expected reward

③ model-based → better model may not lead to better policy

3. Assumptions

① Full observability → value-based

② Episodic learning → policy gradient, model-based

③ Continuity or smoothness → value-based, model-based

DRL Lec 4

- Policy Gradients

1. Derivation

$\Rightarrow J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_t r(s_t,a_t)\right] = \mathbb{E}_{\tau\sim p_\theta(\tau)}[r(\tau)] = \int p_\theta(\tau)\, r(\tau)\, d\tau$

$\Rightarrow \nabla_\theta J(\theta) = \int \nabla_\theta p_\theta(\tau)\, r(\tau)\, d\tau$

$= \int p_\theta(\tau)\cdot \nabla_\theta \log p_\theta(\tau)\cdot r(\tau)\, d\tau$ → score function trick

$= \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\nabla_\theta \log p_\theta(\tau)\cdot r(\tau)\right]$

$= \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\nabla_\theta\left(\cancel{\log p(s_1)} + \sum_{t=1}^{T}\log \pi_\theta(a_t|s_t) + \sum_{t=1}^{T+1}\cancel{\log p(s_{t+1}|s_t,a_t)}\right) r(\tau)\right]$

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

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

$\Rightarrow$ REINFORCE

① sample $\{\tau^i\}$ from $\pi_\theta(a_t|s_t)$

② $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t},a_{i,t})\right)$ → episode with high reward becomes more likely

③ $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$

VS Maximum likelihood

① sample $\{\tau^i\}$ from $\pi_\theta(a_t|s_t)$

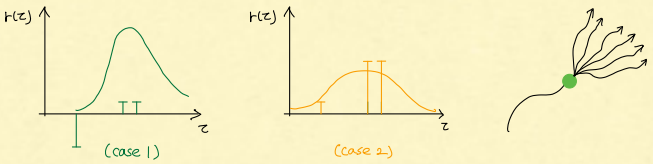
② $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)$ → every episode is equally likely

③ $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$

* Partial observability can be used without modification as Markov property is not used.

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

* Limitation : highly variable on the scale of the reward and the choice of episode



2. Variance reduction

① Causality

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

(future actions cannot affect past rewards)

② Subtracting baseline

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

→ unbiasedness

$\Rightarrow \mathbb{E}_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)\cdot b] = \nabla_\theta \int p_\theta(\tau)\, d\tau \cdot b = 0$

→ optimal baseline

$\left(\text{Var} = \mathbb{E}_\tau\left[\left(\nabla_\theta \log p_\theta(\tau)\cdot(r(\tau)-b)\right)^2\right] - \mathbb{E}_\tau\left[\nabla_\theta \log p_\theta(\tau)\cdot(r(\tau)-b)\right]^2\right.$

$\left.\frac{d\text{Var}}{db} = -2\,\mathbb{E}_\tau\left[\left(\nabla_\theta \log p_\theta(\tau)\right)^2(r(\tau)-b)\right] = 0 \quad \text{when } b = b^*\right)$

$\Rightarrow b^* = \frac{\mathbb{E}_\tau\left[\left(\nabla_\theta \log p_\theta(\tau)\right)^2 r(\tau)\right]}{\mathbb{E}_\tau\left[\left(\nabla_\theta \log p_\theta(\tau)\right)^2\right]}$

$\left(b = \frac{1}{N}\sum_{i=1}^{N} r(\tau^i)\right)$ works just fine, too)

3. Turning to off-policy

: use importance sampling to compute expected reward with updated parameter θ' from the samples with previous parameter θ

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

$\Rightarrow \nabla_{\theta'} J(\theta') = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\frac{\nabla_{\theta'} p_{\theta'}(\tau)}{p_\theta(\tau)} r(\tau)\right] = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\nabla_{\theta'} \log p_{\theta'}(\tau)\cdot\frac{p_{\theta'}(\tau)}{p_\theta(\tau)}\cdot r(\tau)\right]$

$= \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\nabla_{\theta'} \log p_{\theta'}(\tau)\cdot\prod_{t=1}^{T}\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} r(\tau)\right]$

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

(future actions cannot affect weights on past rewards)

$\approx \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T}\nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t)\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\left(\sum_{t'=t}^{T} r(s_t,a_t)\right)\right]$ (∵ first order approximation)

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

DRL Lec 5

- Actor-critic

1. Derivation

$\Rightarrow \nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\sum_{t'=t}^{T} r(s_{i,t'},a_{i,t'})\right)$ → rewards to go

(using a single trajectory for computing gradients) Variance ↑, unbiased

$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s_{i,t'},a_{i,t'})\,|\,s_{i,t},a_{i,t}\right]\right)$ → True expected rewards to go

(using multiple trajectories for computing gradients) Variance ↓, unbiased

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

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

A2C $\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(Q^{\pi_\theta}(s_{i,t},a_{i,t}) - V^{\pi_\theta}(s_{i,t})\right)$ → Advantage Variance ↓↓↓

* $V^{\pi_\theta}(s_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s_{t'},a_{t'})\,|\,s_t\right] = \mathbb{E}_{a_t}\left[Q^{\pi_\theta}(s_t,a_t)\right]$

* $Q^{\pi_\theta}(s_t,a_t) = r(s_t,a_t) + \mathbb{E}_{\pi_\theta}\left[\sum_{t'=t+1}^{T} r(s_{t'},a_{t'})\,|\,s_t,a_t\right] = r(s_t,a_t) + \mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t,a_t)}\left[V^{\pi_\theta}(s_{t+1})\right]$

$\approx r(s_t,a_t) + V^{\pi_\theta}(s_{t+1})$ → from "the" single trajectory

* $A^{\pi_\theta}(s_t,a_t) \approx r(s_t,a_t) + V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$

∴ When $V^{\pi_\theta}(s_t)$ is fitted, $Q^{\pi_\theta}(s_t,a_t)$ and $A^{\pi_\theta}(s_t,a_t)$ can be easily computed

$\Rightarrow$ Fitting value function

① Monte Carlo policy evaluation

$\Rightarrow V^{\pi_\theta}(s_t) \approx \sum_{t'=t}^{T} r(s_{t'},a_{t'})$ or $\frac{1}{N}\sum_{i=1}^{N}\sum_{t'=t}^{T} r(s_{t'},a_{t'})$

② Monte Carlo policy evaluation with function approximation (critic)

$\Rightarrow \mathcal{L}(\phi) = \frac{1}{2}\sum_{i=1}^{N}\left\|\hat{V}_\phi^{\pi_\theta}(s_{i,t}) - \sum_{t'=t}^{T} r(s_{i,t'},a_{i,t'})\right\|^2$ → Monte Carlo target

$= \frac{1}{2}\sum_{i=1}^{N}\left\|\hat{V}_\phi^{\pi_\theta}(s_{i,t}) - \left(r(s_{i,t},a_{i,t}) + \hat{V}_\phi^{\pi_\theta}(s_{i,t+1})\right)\right\|^2$ → Bootstrapped target (from "the" single trajectory)

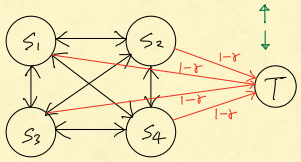
③ Policy gradient is applied with the estimated value from the critic network (actor)

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

$\Rightarrow$ Discount factor (γ)

: better to get reward sooner than later

→ allows trade-off between bias & variance



↑ ⇒ γ ↓

↓ ⇒ γ ↑

→ Policy gradients

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

$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(r(s_{i,t},a_{i,t}) + \gamma\hat{V}_\phi^{\pi_\theta}(s_{i,t+1}) - \hat{V}_\phi^{\pi_\theta}(s_{i,t})\right)$ → Actor-critic

$\Rightarrow$ batch actor-critic

① sample $\{s_t,a_t\}_{t=1}^{T}$ from $p_\theta(\tau)$

② fit $\hat{V}_\phi^{\pi_\theta}(s_t)$ to Monte Carlo target $\left(\sum_{t'=t}^{T}\gamma^{t'-t} r(s_t,a_t)\right)$

③ evaluate $\hat{A}_\phi^{\pi_\theta}(s_t,a_t) = r(s_t,a_t) + \gamma\hat{V}_\phi^{\pi_\theta}(s_{t+1}) - \hat{V}_\phi^{\pi_\theta}(s_t)$

④ $\nabla_\theta J(\theta) \approx \sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\,\hat{A}_\phi^{\pi_\theta}(s_t,a_t)$

⑤ $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$

$\Rightarrow$ online actor-critic

① take action $a\sim\pi_\theta(a|s)$ and get (s,a,r,s')

② fit $\hat{V}_\phi^{\pi_\theta}(s)$ to bootstrapped target $\left(r(s,a) + \gamma\hat{V}_\phi^{\pi_\theta}(s')\right)$

③ evaluate $\hat{A}_\phi^{\pi_\theta}(s,a) = r(s,a) + \gamma\hat{V}_\phi^{\pi_\theta}(s') - \hat{V}_\phi^{\pi_\theta}(s)$

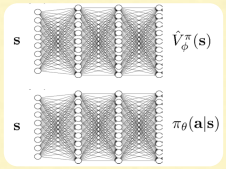
④ $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a|s)\,\hat{A}_\phi^{\pi_\theta}(s,a)$

⑤ $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$

* Limitation : highly biased when the critic is not perfect, especially at the beginning

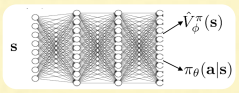
2. Architecture design

⇒ two networks



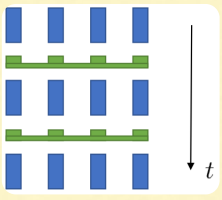
(No shared features)

⇒ shared network



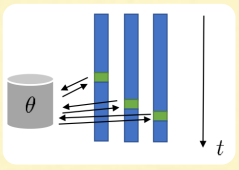
(difficulty on convergence)

⇒ synchronized parallel actor-critic



(slow synchronization)

⇒ asynchronous parallel actor-critic



(faster and easier)

3. Correcting bias

① state-dependent baseline

$$\Rightarrow \nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\sum_{t'=t}^{T}\gamma^{t'-t}r(s_{i,t'},a_{i,t'}) - \hat{V}^{\pi_\theta}_{\phi}(s_{i,t})\right) \longrightarrow \text{unbiased}$$

② action-dependent baseline

$$\Rightarrow \nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\sum_{t'=t}^{T}\gamma^{t'-t}r(s_{i,t'},a_{i,t'}) - \hat{Q}^{\pi_\theta}_{\phi}(s_{i,t},a_{i,t})\right) \longrightarrow \text{biased}$$

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

$$+ \underline{\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \mathbb{E}_{a}\left[\hat{Q}^{\pi_\theta}_{\phi}(s_{i,t},a)\right]} \longrightarrow \text{correction term : Q-Prop}$$

③ eligibility trace

$$\Rightarrow \nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\sum_{t'=t}^{t+n}\gamma^{t'-t}r(s_{i,t'},a_{i,t'}) - \hat{V}^{\pi_\theta}_{\phi}(s_{i,t}) + \gamma^{n}\hat{V}^{\pi_\theta}_{\phi}(s_{i,t+n})\right)$$

(as $n \uparrow$, bias $\downarrow$, variance $\uparrow$)

④ generalized advantage estimation

$$\Rightarrow \nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\sum_{n=1}^{\infty} \underbrace{w_n}_{w_n \propto \lambda^{n-1}}\left(\sum_{t'=t}^{t+n}\gamma^{t'-t}r(s_{i,t'},a_{i,t'}) - \hat{V}^{\pi_\theta}_{\phi}(s_{i,t}) + \gamma^{n}\hat{V}^{\pi_\theta}_{\phi}(s_{i,t+n})\right)\right)$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\sum_{t'=t}^{\infty}(\gamma\lambda)^{t'-t}\left(r(s_{i,t'},a_{i,t'}) - \hat{V}^{\pi_\theta}_{\phi}(s_{i,t}) + \gamma\hat{V}^{\pi_\theta}_{\phi}(s_{i,t+n})\right)\right)$$

DRL Lec 6

- Value-function method

1. Knowing the transition dynamics

⇒ policy iteration

① evaluate $V^{\pi}(s) \leftarrow r(s,a) + \gamma\,\mathbb{E}_{s'\sim p(s'|s,a),\,a\sim\pi(a|s)}\left[V^{\pi}(s')\right]$ until convergence

② compute $A^{\pi}(s,a) = r(s,a) + \gamma\,\mathbb{E}_{s'\sim p(s'|s,a)}\left[V^{\pi}(s')\right] - V^{\pi}(s)$

③ update $\pi(a|s) \leftarrow \pi'(a|s) = \begin{cases} 1 & \text{if } a = \underset{a}{\operatorname{argmax}}\, A^{\pi}(s,a) \\ 0 & \text{otherwise} \end{cases}$

① evaluate $Q^{\pi}(s,a) \leftarrow r(s,a) + \gamma\,\mathbb{E}_{s'\sim p(s'|s,a)}\left[Q^{\pi}(s',a')\right]$, $a'\sim\pi(a|s')$

② update $\pi(a|s) \leftarrow \pi'(a|s) = \begin{cases} 1 & \text{if } a = \underset{a}{\operatorname{argmax}}\, Q^{\pi}(s,a) \\ 0 & \text{otherwise} \end{cases}$

⇒ value iteration

: Value function is represented by a big table, one entry for each discrete s

① compute $Q(s,a) = r(s,a) + \gamma\,\mathbb{E}_{s'\sim p(s'|s,a)}\left[V(s')\right]$

② update $V(s) \leftarrow \max_{a} Q(s,a)$

⇒ fitted value iteration

: Value function is represented by a neural network with parameter ϕ

① set $y_i \leftarrow \max_{a_i}\left(r(s_i,a_i) + \gamma\,\mathbb{E}_{s'\sim p(s'|s,a)}\left[\hat{V}_{\phi}(s_i')\right]\right)$

② update $\phi \leftarrow \underset{\phi}{\operatorname{argmin}}\, \frac{1}{2}\sum_{i}\left\|\hat{V}_{\phi}(s_i) - y_i\right\|^2$

2. Not knowing the transition dynamics

⇒ fitted Q iteration

: Q function is represented by a neural network with parameter ϕ

① collect dataset $\{(s_i,a_i,s_i',r_i)\}$ using some policy ⟶ off-policy

② set $y_i \leftarrow r(s_i,a_i) + \gamma \max_{a'}\hat{Q}_{\phi}(s_i',a')$ ⟶ from "the" single trajectory

③ update $\phi \leftarrow \underset{\phi}{\operatorname{argmin}}\, \frac{1}{2}\sum_{i}\left\|\hat{Q}_{\phi}(s_i,a_i) - y_i\right\|^2$

xk

⇒ online Q iteration ($\approx$ Q-learning)

① take some action a_i and observe (s_i,a_i,s_i',r_i) ⟶ off-policy

② set $y_i \leftarrow r(s_i,a_i) + \gamma \max_{a'}\hat{Q}_{\phi}(s_i',a')$ ⟶ from "the" single trajectory

③ update $\phi \leftarrow \phi - \alpha\left.\frac{d\hat{Q}_{\phi}}{d\phi}\right|_{s=s_i,a=a_i}\left(\hat{Q}_{\phi}(s_i,a_i) - y_i\right)$

* Exploration in ①

⟶ ε-greedy : $\pi(a|s) = \begin{cases} 1-\varepsilon & \text{if } a = \underset{a}{\operatorname{argmax}}\, \hat{Q}_{\phi}(s,a) \\ \varepsilon/(|A|-1) & \text{otherwise} \end{cases}$

⟶ Boltzmann exploration : $\pi(a|s) \propto \exp\left(\hat{Q}_{\phi}(s,a)\right)$

* Limitation

⟶ batch size is 1 ⟶ hard to optimize DNN

⟶ Samples are correlated ⟶ DNN needs i.i.d. samples

⟶ No gradient through the target value y_i ⟶ y_i also depends on ϕ

⟶ Overestimation on Q-function ⟶ action and value depend on the same network

⟶ Biased target when Q-function is inaccurate ⟶ slow learning at the early stage

⟶ Taking max on continuous action space is hard ⟶ worse when Q-function is complex

3. Learning theory

① operator $\mathcal{B}$

: $\mathcal{B}V = \max_a r_a + \gamma T_a V$ (or $\mathcal{B}Q = r + \gamma T \max_a Q$)

→ value iteration : $V \leftarrow \mathcal{B}V$

→ V^* is a fixed point of $\mathcal{B}$: $V^* = \mathcal{B}V^*$

→ for any V and $\bar{V}$, we have $\|\mathcal{B}V - \mathcal{B}\bar{V}\|_\infty \le \gamma \|V - \bar{V}\|_\infty$ → contraction

→ applying $\mathcal{B}$ to any V would leads to V^* in terms of ∞-norm

($\because \|\mathcal{B}V - \mathcal{B}V^*\|_\infty = \|\mathcal{B}V - V^*\|_\infty \le \gamma \|V - V^*\|_\infty$)

② operator π

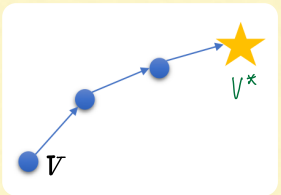
: $\pi V = \operatorname{argmin}_{V' \in \Omega} \frac{1}{2} \sum_s \|V'(s) - V(s)\|_2^2$ (or $\pi Q = \operatorname{argmin}_{Q' \in \Omega} \frac{1}{2} \sum_s \|Q'(s,a) - Q(s,a)\|_2^2$)

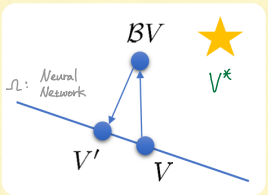
→ fitted value (or Q) iteration : $V \leftarrow \pi\mathcal{B}V$ (or $Q \leftarrow \pi\mathcal{B}Q$)

→ for any V and $\bar{V}$, we have $\|\pi V - \pi\bar{V}\|_2^2 \le \|V - \bar{V}\|_2^2$ → contraction

→ $\pi\mathcal{B}$ is not contraction of any kind

(fitted bootstrapped policy evaluations are not guaranteed for convergence)





DRL Lec 7

- Deep RL with Q-functions

1. Improving Q-learning

⇒ increase batch size by parallel learning

→ synchronized parallel Q-learning , asynchronized parallel Q-learning

⇒ decorrelate samples

→ replay buffers ($\mathcal{B}$) : dataset of transitions from some previous policies

* Q-learning with replay buffers

① collect dataset using some policy and add it to $\mathcal{B}$

② sample a batch $\{(s_i, a_i, s_i', r_i)\}_{i=1}^N$ from $\mathcal{B}$

xk ③ set $y_i \leftarrow r(s_i,a_i) + \gamma \max_{a'} \hat{Q}_\phi(s_i', a')$

④ update $\phi \leftarrow \phi - \alpha \sum_{i=1}^N \frac{d\hat{Q}_\phi}{d\phi}\Big|_{s=s_i, a=a_i} (\hat{Q}_\phi(s_i,a_i) - y_i)$

⇒ flow gradients to the target value

→ loosely update the target parameters (simply copying or polyak averaging) $\phi' \leftarrow \tau\phi' + (1-\tau)\phi$

* Q-learning with replay buffers and target network (≈ DQN if $N=k=1$)

① save target network parameters $\phi' \leftarrow \phi$

xN ② collect dataset using some policy and add it to $\mathcal{B}$

xN ③ sample a batch $\{(s_i, a_i, s_i', r_i)\}_{i=1}^N$ from $\mathcal{B}$

xk ④ set $y_i \leftarrow r(s_i,a_i) + \gamma \max_{a'} \hat{Q}_{\phi'}(s_i', a')$

⑤ update $\phi \leftarrow \phi - \alpha \sum_{i=1}^N \frac{d\hat{Q}_\phi}{d\phi}\Big|_{s=s_i, a=a_i} (\hat{Q}_\phi(s_i,a_i) - y_i)$

⇒ decorrelate the noises when computing the target value

→ use $\hat{Q}_\phi$ for choosing the action and $\hat{Q}_{\phi'}$ for evaluating the value

* Double Q-learning

① save target network parameters $\phi' \leftarrow \phi$

xN ② collect dataset using some policy and add it to $\mathcal{B}$

xN ③ sample a batch $\{(s_i, a_i, s_i', r_i)\}_{i=1}^N$ from $\mathcal{B}$

xk ④ set $y_i \leftarrow r(s_i,a_i) + \gamma \hat{Q}_{\phi'}(s_i', \operatorname{argmax}_{a'} \hat{Q}_\phi(s_i, a'))$

⑤ update $\phi \leftarrow \phi - \alpha \sum_{i=1}^N \frac{d\hat{Q}_\phi}{d\phi}\Big|_{s=s_i, a=a_i} (\hat{Q}_\phi(s_i,a_i) - y_i)$

⇒ utilize multi-step returns

→ need transition history and only actually correct when learning on-policy

* Q-learning with N-step returns

① save target network parameters $\phi' \leftarrow \phi$

xN ② collect dataset using some policy and add it to $\mathcal{B}$

xN ③ sample a batch $\{(s_i, a_i, s_i', r_i)\}_{i=1}^N$ from $\mathcal{B}$

xk ④ set $y_i \leftarrow \sum_{t'=t}^{t+N-1} \gamma^{t'-t} r(s_{i,t'}, a_{i,t'}) + \gamma^N \max_{a'} \hat{Q}_{\phi'}(s_{i,t+N}, a')$

⑤ update $\phi \leftarrow \phi - \alpha \sum_{i=1}^N \frac{d\hat{Q}_\phi}{d\phi}\Big|_{s=s_i, a=a_i} (\hat{Q}_\phi(s_i,a_i) - y_i)$

⇒ work on continuous action space

→ optimization (sample actions from uniform distribution / cross-entropy method (CEM, CMA-ES))

→ use function class for Q-function that is easy to optimize

(Normalized Advantage functions (NAF): $\hat{Q}_\phi(s,a) = -\frac{1}{2}(a - \mu(s))^T P_\phi(s)(a - \mu(s)) + \hat{V}_\phi(s)$)

→ learn an approximate maximizer $\mu_\theta(s)$ s.t. $\mu_\theta(s) \approx \operatorname{argmax}_a \hat{Q}_\phi(s,a)$

* DDPG

① save target network parameters $\phi' \leftarrow \phi$, $\theta' \leftarrow \theta$

xN ② collect dataset using some policy and add it to $\mathcal{B}$

xN ③ sample a batch $\{(s_i, a_i, s_i', r_i)\}_{i=1}^N$ from $\mathcal{B}$

xk ④ set $y_i \leftarrow r(s_i,a_i) + \gamma \hat{Q}_{\phi'}(s_i', \mu_{\theta'}(s_i'))$

⑤ update
$\phi \leftarrow \phi - \alpha \sum_{i=1}^N \frac{d\hat{Q}_\phi}{d\phi}\Big|_{s=s_i, a=a_i} (\hat{Q}_\phi(s_i,a_i) - y_i)$
$\theta \leftarrow \theta + \beta \sum_{i=1}^N \frac{d\mu_\theta}{d\theta}\Big|_{s=s_i} \cdot \frac{d\hat{Q}_\phi}{da}\Big|_{s=s_i, a=a_i}$

DRL Lec 8

- Advanced policy gradients

1. Policy gradients ≈ Policy iteration

$$\Rightarrow J(\theta') - J(\theta) = \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] - \mathbb{E}_{s_0\sim p(s_0)}\left[V^{\pi_\theta}(s_0)\right]$$

$$= \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] - \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[V^{\pi_\theta}(s_0)\right]$$

$$= \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] - \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t V^{\pi_\theta}(s_t) - \sum_{t=1}^{\infty}\gamma^t V^{\pi_\theta}(s_t)\right]$$

$$= \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] + \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t\left(\gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)\right)\right]$$

$$= \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t\left(r(s_t,a_t) + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)\right)\right]$$

$$= \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi_\theta}(s_t,a_t)\right]$$

$\therefore$ Improvement on policy gradients ≈ Improvement on policy iteration

2. Bounding the distribution change

$$\Rightarrow \mathbb{E}_{\tau\sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi_\theta}(s_t,a_t)\right] = \sum_t \mathbb{E}_{s_t\sim p_{\theta'}(s_t)}\left[\mathbb{E}_{a_t\sim\pi(a_t|s_t)}\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\gamma^t A^{\pi_\theta}(s_t,a_t)\right]\right]$$

$$\geq \sum_t \mathbb{E}_{s_t\sim p_\theta(s_t)}\left[\mathbb{E}_{a_t\sim\pi(a_t|s_t)}\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\gamma^t A^{\pi_\theta}(s_t,a_t)\right]\right] - \sum_{s_t}\left|p_\theta(s_t) - p_{\theta'}(s_t)\right| \max_{s_t}\mathbb{E}_{a_t\sim\pi(a_t|s_t)}\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\gamma^t A^{\pi_\theta}(s_t,a_t)\right]$$

$$\geq \bar{A}(\theta') - 2\varepsilon t C$$

where
- $\bar{A}(\theta') = \sum_t \mathbb{E}_{s_t\sim p_\theta(s_t)}\left[\mathbb{E}_{a_t\sim\pi(a_t|s_t)}\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\gamma^t A^{\pi_\theta}(s_t,a_t)\right]\right]$
- $C = \max_{s_t}\mathbb{E}_{a_t\sim\pi(a_t|s_t)}\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\gamma^t A^{\pi_\theta}(s_t,a_t)\right]$
- $\left|\pi_\theta(a_t|s_t) - \pi_{\theta'}(a_t|s_t)\right| \leq \varepsilon$

$\therefore \theta' \leftarrow \arg\max_{\theta'} \bar{A}(\theta')$ s.t. $\left|\pi_\theta(a_t|s_t) - \pi_{\theta'}(a_t|s_t)\right| \leq \varepsilon$

3. Using more convenient and practical bound

$$\Rightarrow \left|\pi_\theta(a_t|s_t) - \pi_{\theta'}(a_t|s_t)\right| \leq \sqrt{\tfrac{1}{2}KL\left(\pi_{\theta'}(a_t|s_t)\,\|\,\pi_\theta(a_t|s_t)\right)}$$

$\therefore \theta' \leftarrow \arg\max_{\theta'} \bar{A}(\theta')$ s.t. $KL\left(\pi_{\theta'}(a_t|s_t)\,\|\,\pi_\theta(a_t|s_t)\right) \leq \varepsilon$

$\therefore \theta' \leftarrow \arg\max_{\theta'} \nabla_\theta \bar{A}(\theta)^T(\theta'-\theta)$ s.t. $KL\left(\pi_{\theta'}(a_t|s_t)\,\|\,\pi_\theta(a_t|s_t)\right) \leq \varepsilon$ (linearization)

where $\nabla_{\theta'}\bar{A}(\theta') = \sum_t \mathbb{E}_{s_t\sim p_\theta(s_t)}\left[\mathbb{E}_{a_t\sim\pi(a_t|s_t)}\left[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\gamma^t \nabla_{\theta'}\log\pi_{\theta'}(a_t|s_t)\, A^{\pi_\theta}(s_t,a_t)\right]\right]$

so that $\nabla_\theta\bar{A}(\theta) = \sum_t \mathbb{E}_{s_t\sim p_\theta(s_t)}\left[\mathbb{E}_{a_t\sim\pi(a_t|s_t)}\left[\gamma^t\nabla_\theta\log\pi_\theta(a_t|s_t)\, A^{\pi_\theta}(s_t,a_t)\right]\right]$ (policy gradients)

4. Natural gradient

$$\Rightarrow KL\left(\pi_{\theta'}(a_t|s_t)\,\|\,\pi_\theta(a_t|s_t)\right) \approx \frac{1}{2}(\theta'-\theta)^T F(\theta'-\theta) \quad \text{where } F = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta\log\pi_\theta(a|s)\,\nabla_\theta\log\pi_\theta(a|s)^T\right]$$

$$\therefore \theta^* = \theta + \sqrt{\frac{2\varepsilon}{\nabla_\theta\bar{A}(\theta)^T F\,\nabla_\theta\bar{A}(\theta)}}F^{-1}\nabla_\theta\bar{A}(\theta)$$ (Trust region policy optimization)