

12. PPO 알고리즘 분석

김호철

Contents

1 TRPO(Trust Region Policy Optimization)

- 1.1 개요
- 1.2 신뢰 영역(Trust region)
- 1.3 TRPO 단계별 이론
- 1.4 Natural Policy Gradient (NPG)
- 1.5 TRPO 알고리즘
- 1.6 TRPO의 단점

2 PPO(Proximal Policy Optimization)

- 2.1 개요
- 2.2 PPO 클리핑 써로게이트 목적 함수(clipped surrogate objective function)
- 2.3 실제 구현

3 DRL 알고리즘들의 비교

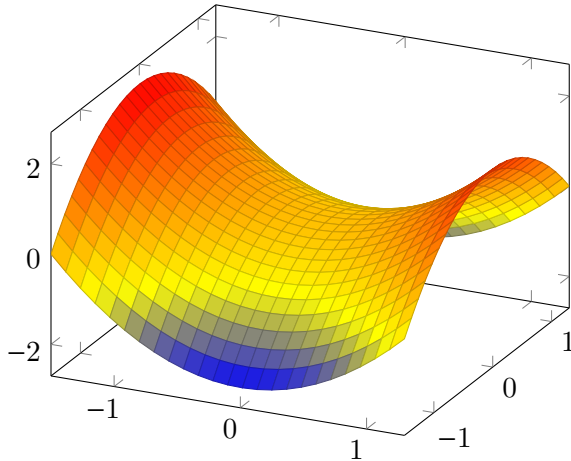
1 TRPO(Trust Region Policy Optimization)

1.1 개요

- Trust Region Policy Optimization, UC Berkeley 2015
- DDPG는 에이전트가 연속(continuous) 액션 공간에서 수행할 수 있도록 하는 획기적인 알고리즘이지만, 성능이 단조롭게(monotonically) 향상되지는 않는다.
- TRPO는 **Minorization-Maximization** 알고리즘과 **Trust region**이라는 두 가지 개념을 채택하여 DDPG에서 발생하는 문제를 성공적으로 해결하였다.
- TRPO는 정책 공간의 쿨백-라이블러(Kullback-Leibler) 발산(다이버전스) 제약 조건에 따라 신뢰 영역(Trust region)에서 정책 파라미터 θ (기대 수익 극대화)를 업데이트한다.
- 신뢰 영역(Trust region)에 대한 정책 업데이트는 기대 리턴의 단조로운 개선을 보장하므로, TRPO는 스텝 크기(학습율)에 걱정할 필요가 없다.
- 실용적인 알고리즘 TRPO를 생성하기 위해 이론적으로 정당화된 알고리즘에 대한 일련의 근사치를 만들었다.

- TRPO는 고성능을 달성했지만 구현이 매우 복잡하고 고비용 계산이 필요하므로, 딥 CNN이나 RNN이 필요한 작업에는 실용적이지 않다.

1.2 신뢰 영역(Trust region)



- DDPG를 사용할 때 중요한 문제는 적절한 범위 내에서 스텝 크기를 선택하는 방법이다.
- 스텝 크기가 너무 크면 노이즈에 압도되어 성능이 저하되는 경향이 있고, 너무 작으면 훈련 진행이 매우 느려진다.
- 기울기 상승에서는 가장 가파른 방향을 결정한 다음 앞으로 나아간다. 그러나 스텝 크기가 너무 크면(그래서 너무 앞으로 나아가면) 때때로 재앙이 된다.
- δ 에 의해 제어되는 신뢰 영역(Trust region)을 사용하여 영역 내로 이동을 제한한다.
- 이론적으로 이 신뢰 영역은 새로운 최적 정책이 이전 정책보다는 우수함을 보장하므로, 반복을 계속하면 결국 로컬 또는 글로벌 최적 지점에 도달한다.

1.3 TRPO 단계별 이론

[0-단계] 기대 리턴 최대화

- DRL의 목표는 기대 리턴 $\eta(\pi) = \mathbb{E}_{s_0, a_0, \dots} \left[\sum_t \gamma^t r(s_t) \right]$ 를 극대화하는 것이다.

[1-단계] 이전 정책에서 새 정책을 계산

- π_{old} 에 대한 어드밴티지 $A_{\pi_{old}}$ 를 사용해서, 기존 정책 π_{old} 의 $\eta(\pi_{old})$ 로 새 정책 π 의 $\eta(\pi)$ 를 계산한다.

$$\begin{aligned}
 \eta(\pi) &= \eta(\pi_{old}) + \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} A_{\pi_{old}}(s_t, a_t) \right] \\
 &= \eta(\pi_{old}) + \sum_s \rho_{\pi}(s) \sum_a \pi(a|s) A_{\pi_{old}}(s, a)
 \end{aligned} \tag{1}$$

– 상태 방문 빈도 (state visitation frequency)

$$\rho_{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t P(s_t = s | \pi) \quad (2)$$

– 증명을 위한 식 1

$$\begin{aligned} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{old}}(s_t, a_t) \right] &= \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t (r(s_t) + \gamma V_{\pi_{old}}(s_{t+1}) - V_{\pi_{old}}(s_t)) \right] \\ &= \eta(\pi) + \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t (r(s_{t+1}) + \gamma V_{\pi_{old}}(s_{t+1}) - \sum_{t=0}^{\infty} \gamma^t V_{\pi_{old}}(s_t)) \right] \\ &= \eta(\pi) - \mathbb{E}_{s_0} [V_{\pi_{old}}(s_0)] = \eta(\pi) - \eta(\pi_{old}) \end{aligned} \quad (3)$$

– 증명을 위한 식 2

$$\begin{aligned} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t A_{\pi_{old}}(s_t, a_t) \right] &= \sum_{t=0}^{\infty} \sum_s P(s_t = s | \pi) \sum_a \pi(a|s) \gamma^t A_{\pi_{old}}(s, a) \\ &= \sum_{t=0}^{\infty} \sum_s \gamma^t P(s_t = s | \pi) \sum_a \pi(a|s) A_{\pi_{old}}(s, a) \quad (4) \\ &= \sum_s \rho_{\pi}(s) \sum_a \pi(a|s) A_{\pi_{old}}(s, a) \end{aligned}$$

– 이전 어드밴티지에 대한 식

$$\begin{aligned} A_{\pi_{old}}(s_t, a_t) &= Q_{\pi_{old}}(s_t, a_t) - V_{\pi_{old}}(s_t) \\ &= r(s_t) + \gamma \mathbb{E}_{s_{t+1} \sim P(s_{t+1}|s_t, a_t)} [V_{\pi_{old}}] - V_{\pi_{old}}(s_t) \end{aligned} \quad (5)$$

- 그래서 어떤 정책 업데이트 $\pi_{old} \rightarrow \pi$ 를 하는 경우에, 모든 상태에서 $\sum_a \pi(a|s) A_{\pi_{old}}(s, a) \geq 0$ 이면, 정책 성능 η 는 증가함이 보장된다.
- 그런데, 근사를 할 때 기대 어드밴티지가 음수인 상태 s 들이 있는 경우가 있다. (그래서 다음 단계에서 신뢰 영역(Trust Region)을 사용하게 된다.)
- 더구나, π 에 대한 $\rho_{\pi}(s)$ 의 복잡한 의존성(새로운 정책에 대한 상태 방문 빈도를 구하기 어려움)은 이 방정식을 최적화하기 어렵게 만든다.

[2-단계] 로컬 근사(local approximation)

- 정책 업데이트할 때의 상태 방문 빈도의 변화는 무시하여 ρ_{π} 대신 $\rho_{\pi_{old}}$ 를 사용하는 $\eta(\pi)$ 에 대한 로컬 근사치 $L_{\pi_{old}}(\pi)$ 를 사용한다.

$$L_{\pi_{old}} = \eta(\pi_{old}) + \sum_s \rho_{\pi_{old}}(s) \sum_a \pi(a|s) A_{\pi_{old}}(s, a) \quad (6)$$

- 정책 π_{θ} 가 θ 에 대해 미분 가능한 파라미터(신경망)인 경우, 어떤 파라미터 값 θ_0 에 대해 $L_{\pi_{old}}(\pi_{\theta_0}) = \eta(\pi_{\theta_0})$ 와 $\nabla_{\theta} L_{\pi_{old}}(\pi_{\theta})|_{\theta=\theta_0} = \nabla_{\theta} \eta(\pi_{\theta})|_{\theta=\theta_0}$ 가 되어, $L_{\pi_{old}}$ 는 η 를 매치(근사)한다.

- 이러한 $L_{\pi_{old}}$ 를 개선시키는 것은, 충분히 작은 단계로 $\pi_{old} \rightarrow \pi$ (오래된 정책을 새로운 정책으로 업데이트)시에 η 도 개선(근사로 사용 가능하다)시킬 것이라는 것을 의미하지만, 얼마나 큰 단계를 밟아야 하는지에 대한 지침을 제공하지는 않는다.

[3-단계] 보수적 정책 반복 업데이트

- 이러한 이슈를 해결하기 위해 보수적 정책 반복 업데이트(conservative policy iteration update:Kakade 2002)를 사용한다.
- $\pi' = \operatorname{argmax}_{\pi} L_{\pi_{old}}(\pi)$ 인 경우, 새로운 혼합 정책(mixture policy) $\pi(a|s) = (1-\alpha)\pi_{old}(a|s) + \alpha\pi'(a|s)$ 을 사용한다.
- 이것은 다음의 하한(lower bound)을 가지게 한다.

$$\eta(\pi) \geq L_{\pi_{old}}(\pi) - \frac{2\epsilon\gamma}{(1-\gamma)^2}\alpha^2 \quad (7)$$

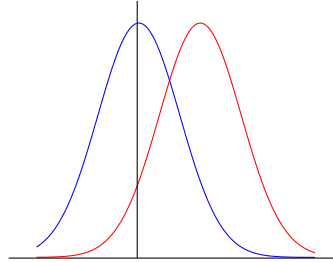
where $\epsilon = \max_s \left| \mathbb{E}_{a \sim \pi'(a|s)} [A_{\pi_{old}}(s, a)] \right|$

- 그러나 이러한 bound는 위의 혼합 정책에만 적용되며 실제로는 제한적이다.

[4-단계] 일반적 확률 정책

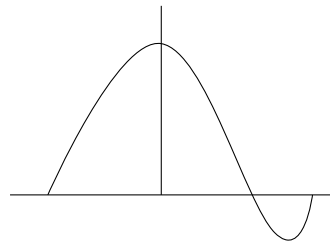
- 이전의 정책 개선 바운드를 확장하여 혼합 정책보다는 실제에서 사용가능한 일반적 확률 정책(general stochastic policies)을 사용한다.
- Total Variation distance와 KL다이버전스는 두 확률 분포간의 차이를 계산하는데 사용한다.
 - Total Variation distance는 두 확률 분포의 차에 대한 면적을 2로 나눈 것

$$D_{TV}(P, Q) = \frac{1}{2} \|P - Q\|$$



- KL다이버전스는 정보 엔트로피를 사용

$$D_{KL}(P \| Q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx$$



- 머신러닝에서는 주로 KL다이버전스를 사용한다.
- 이전 bound 식의 하이퍼-파라미터 α 대신에 Total Variation distance를 사용한다.

$$\alpha = \max_s D_{TV}(\pi_{old}(\cdot|s) \| \pi(\cdot|s)) \quad (8)$$

- ϵ 을 다음으로 변경한다.

$$\max_s |A_{\pi_{old}}(s, a)| \leftarrow \max_s |\mathbb{E}_{a \sim \pi'(a|s)} [A_{\pi_{old}}(s, a)]| \quad (9)$$

- $D_{TV}(p||q)^2 \leq D_{KL}(p||q)$ 이기 때문에, Total Variation distance를 KL다이버전스로 대체해도 된다.

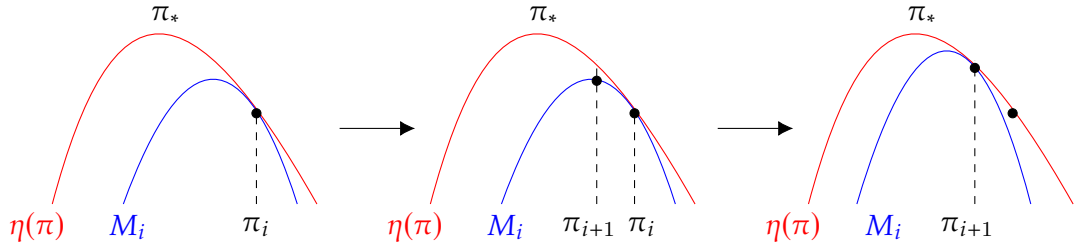
$$\eta(\pi) \geq L_{\pi_{old}}(\pi) - CD_{KL}^{max}(\pi_{old}, \pi) \quad (10)$$

$$\text{where } C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$$

[5-단계] 최소-최대 알고리즘

- 최소-최대 알고리즘(Minorization-Maximization algorithm) (Hunter 2004)

- 최소-최대(MinMax, MM) 알고리즘은 이터러티브 방법으로 다음을 반복한다.
 1. 대행 목적 함수(surrogate objective function) M_i 를 찾는다.
 - * 현재 정책 π_i 에 목적 함수 η 를 근사한다. (M_i 는 π_i 에서 같은 값을 가지는 η 를 최소화한다.)
 - * η 에 대한 하한(lower bound) 함수
 - * η 보다 최적화하기 쉬움(2차 방정식같은)
 2. M_i 에 대한 최적점 π_{i+1} 을 찾아 다음 정책으로 사용한다. (M_i 최대화)
 3. 새 정책에서 하한 M_{i+1} 를 재평가하고 다시 반복한다.



- 프로세스를 계속 진행함에 따라 정책은 계속 개선된다.
- 정책의 유한성으로 인해 정책은 결국 로컬 또는 전역 최적으로 수렴된다.

- 모델에 MM 알고리즘 적용하기

- 대행 목적 함수(surrogate objective function)

$$M_i(\pi) = L_{\pi_i}(\pi) - CD_{KL}^{max}(\pi_i, \pi) \quad (11)$$

- 여기에서,

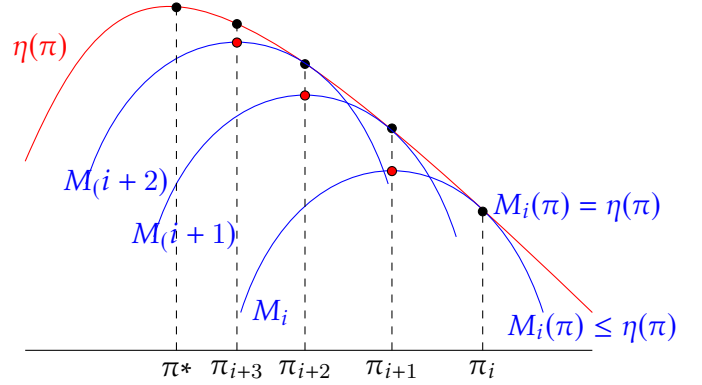
$$L_{\pi_{old}} = \eta(\pi_{old}) + \sum_s \rho_{\pi_{old}}(s) \sum_a \pi(a|s) A_{\pi_{old}}(s, a)$$

$$C = \frac{4\epsilon\gamma}{(1-\gamma)^2} \quad (12)$$

$$D_{KL}^{max}(\pi_i, \pi) = \max D_{KL}(\pi_{old}(\cdot|s) || \pi(\cdot|s))$$

- 실제 목적 함수 η 의 단조(Monotonic) 증가가 보장된다

$$\begin{aligned} \eta(\pi_i) &= M_i(\pi_i) \text{ and } \eta(\pi_i) \geq M_i(\pi_i) \\ \Rightarrow \eta(\pi_i) &= M_i(\pi_i) \leq M_i(\pi_{\pi_{i+1}}) \leq \eta(\pi_{i+1}) \\ \text{for } \pi_{i+1} &= \operatorname{argmax}_{\pi} M_i \pi \end{aligned}$$



- 이제 이러한 이론적 토대위에 실용적인 알고리즘을 도출하면(정책 $\pi_{\theta}(a|s)$ 에 대해 정책 네트워크 파라미터 θ 를 사용),
- 이전 파라미터를 θ_{old} 로, 새로운 파라미터를 θ 로, surrogate objective 함수에 적용하면,

$$\eta(\theta) \geq L_{\theta_{old}}(\theta) - CD_{KL}^{\max}(\theta_{old}, \theta) \quad (13)$$

- 그런데 surrogate objective 함수를 최대화하는 다음 함수는, 최적(optimal) 정책이 아니라, 다음(next) 정책이 된다.

$$\max_{\theta} [L_{\theta_{old}}(\theta) - CD_{KL}^{\max}(\theta_{old}, \theta)] \quad (14)$$

- 이러한 계산을 최적이 될 때까지 반복해야 함으로 TRPO는 상당히 많은 계산을 필요로 한다.
- 그리고 $C = \frac{4\epsilon\gamma}{(1-\gamma)^2}$ 에서 γ 가 1에 가깝게 커지면 실질적으로 학습 스텝 크기가 너무 작아져서 학습이 느려진다.

[6-단계] 신뢰 구간 제약(Trust region constraint)

- 다음을 KL 페널티라이즈드 목적(KL penalized objective)라 할 수 있고,

$$\max_{\theta} [L_{\theta_{old}}(\theta) - CD_{KL}^{\max}(\theta_{old}, \theta)] \quad (15)$$

- 다음은 KL 제약 목적(KL constrained objective)이 된다.

$$\max_{\theta} L_{\theta_{old}}(\theta) \text{ subject to } D_{KL}^{\max}(\theta_{old}, \theta) \leq \delta \quad (16)$$

- 이 두 오브젝티브들은 무한히 반복하면 수학적으로 일치한다(Lagrangian duality).
- 실제로, 앞선 이론에 따라 C값이 크게될 가능성이 높아서, 스텝 크기를 매우 작게 해야한다.
- 스텝 크기를 크게할 수 있는 방법중에 하나는 δ 로 KL 제약 조건을 바운딩(δ 보다 작거나 같게)하는 것이다.
- 이를 hard constraint라 하는데, 정책 공간에서 나쁜 사례를 제어하기 위해 사용되는 것으로, C보다 δ 를 조정하는 것이 더 쉽다.

- 결론적으로 $\max_{\theta} L_{\theta_{old}}$ 를 만족하는 θ 가, $D_{KL}^{\max}(\theta_{old}, \theta) \leq \delta$ 를 만족하면 정책 개선은 보장이 된다. 그래서 이 KL 제약 조건을 신뢰 구간(Trust region)이라 한다.
- 이 제약 조건은 모든 상태에 대해서 만족해야하기 때문에, KL 제약 조건의 최대값이 작은 δ 보다 작아야한다.
- 그런데 불행히도, 많은 수의 제약 조건이나 상태로 인해 이를 만족하는 것을 찾기가 쉽지 않다.

[7-단계] 휴리스틱 근사(Heuristic approximation)

- 상태들에 \max 대신 기대(expected, 샘플링) KL 발산을 하는 휴리스틱 근사를 사용한다.

$$\max_{\theta} L_{\theta_{old}}(\theta) \text{ subject to } \mathbb{E}_{s \sim \rho_{\theta_{old}}} [D_{KL}(\pi_{\theta_{old}}(\cdot|s) \parallel \pi_{\theta}(\cdot|s))] \leq \delta \quad (17)$$

[8-단계] 몬테카를로 시뮬레이션(Monte Carlo simulation)

- 샘플 기반의 에피소드 추정

$$\begin{aligned} & \max_{\theta} \sum_s \rho_{\theta_{old}}(s) \sum_a \pi_{\theta}(a|s) A_{\theta_{old}}(s, a) \\ & \text{subject to } \mathbb{E}_{s \sim \rho_{\theta_{old}}} [D_{KL}(\pi_{\theta_{old}}(\cdot|s) \parallel \pi_{\theta}(\cdot|s))] \leq \delta \end{aligned} \quad (18)$$

- 다음을 대체

- $\sum_s \rho_{\theta_{old}}(s)[\dots]$ 를 $\frac{1}{1-\gamma} \mathbb{E}_{s \sim \rho_{\theta_{old}}}$ 로 대체
- 어드밴티지 $A_{\theta_{old}}$ 를 Q-벨류 $Q_{\theta_{old}}$ 로 대체
- 임포턴스 샘플링(Importance sampling)

$$\sum_a \pi_{\theta}(a|s) A_{\theta_{old}}(s, a) = \mathbb{E}_{a \sim \pi_{\theta_{old}}} \left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)} A_{\theta_{old}}(s, a) \right] \quad (19)$$

- 일반적인 PG에서는 정책이 업데이트될 때마다 새로운 샘플을 수집하고 이전 샘플은 폐기한다. 임포턴스 샘플링을 사용하면 이전 정책의 샘플을 사용하여 정책 기울기를 계산한다.

$$\begin{aligned} \mathbb{E}_{x \sim p}[f(x)] &= \mathbb{E}_{x \sim q} \left[\frac{p(x)}{q(x)} f(x) \right] \approx \frac{1}{N} \sum_{n=1}^N w(x_n) f(x_n) \\ & \text{from } x_n \sim q(x) \text{ where } w(x) = \frac{p(x)}{q(x)} \end{aligned} \quad (20)$$

- 최종 최적화 문제

$$\begin{aligned} & \max_{\theta} \mathbb{E}_{s \sim \rho_{\theta_{old}}, a \sim \pi_{\theta_{old}}} \left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)} Q_{\theta_{old}}(s, a) \right] \\ & \text{subject to } \mathbb{E}_{s \sim \rho_{\theta_{old}}} [D_{KL}(\pi_{\theta_{old}}(\cdot|s) \parallel (\pi_{\theta}(\cdot|s)))] \leq \delta \end{aligned} \quad (21)$$

- 뒤의 식의 조건을 만족하면서 앞의 식을 최대화(maximize)하는 θ 는 최적 정책이 아니라, π_{old} 에서 π 로 한번 업데이트하는 것이다.
- 남은 것은 기대값을 샘플 평균(몬테카를로 시뮬레이션)으로 대체하고, Q-벨류를 추정하기 위해서 별도의 네트워크를 가지고 있어야 한다.

[9-단계] TRPO를 해결하는 실용적인 알고리즘

- 각 이터레이션(정책을 π_{old} 에서 π 로 한번 업데이트 하는 것)에서 다음 3단계를 반복적으로 수행하여, TRPO를 실용적으로 해결한다.
 - π_{old} 로 상태-액션 쌍을 샘플하여 트라젝토리들을 만들고, 그것을 가지고 Q-네트워크에 있는 Q-벨류를 추정한다.
 - 최종 최적화 함수의 목적 함수와 제약 조건을 샘플들에 대해 평균을 구해서 계산한다.
 - 정책 파라미터 θ 를 업데이트하기 위해 이 제한된 최적화(constrained optimization)를 근사적으로 해결한다. 그런데 기존의 Policy Gradient가 아닌 Natural Policy Gradient를 사용한다. 이 NPG의 효율을 높이기 위해 Conjugate gradient를 사용하고, 이 업데이트에 의해 개선되었는지와 제한 조건을 만족하는지를 확인하기 위해 Line search라는 방법을 사용한다.

1.4 Natural Policy Gradient (NPG)

- 8단계의 최종 최적화 문제를 간략히 하면,
 - 목적 함수(Objective function)는,

$$\max_{\theta} L_{\theta_{old}} \theta \quad (22)$$

- 신뢰 구간(Trust region)은,

$$\overline{D}_{KL}(\theta_{old} \parallel \theta) = \mathbb{E}_{s \sim \rho_{\theta_{old}}} [D_{KL}(\pi_{\theta_{old}}(\cdot | s) \parallel (\pi_{\theta}(\cdot | s)))] \leq \delta \quad (23)$$

- NPG는 $L_{\theta_{old}}$ 는 1차(선형)까지, $D_{KL}(\theta_{old} \parallel \theta)$ 는 2차(quadratic)까지 테일러 급수를 사용하여, $L_{\theta_{old}}$ 및 예상 KL 발산 $D_{KL}(\theta_{old} \parallel \theta)$ 를 근사한다.
 - 목적 함수(Objective function)는,

$$\max_{\theta} \nabla_{\theta} L_{\theta_{old}}(\theta) |_{\theta=\theta_{old}} \quad (24)$$

- 신뢰 구간(Trust region)은,

$$\frac{1}{2}(\theta - \theta_{old})^T H(\theta - \theta_{old}) \leq \delta \quad (25)$$

- 헤시안(Hessian)으로 Fisher Information Matrix $H = \nabla_{\theta}^2 \overline{D}_{KL}(\theta_{old} \parallel \theta) = \frac{\partial^2 \overline{D}_{KL}(\theta_{old} \parallel \theta)}{\partial \theta_i \partial \theta_j}$ 를 사용하고, N개의 샘플을 위해 $H \approx \left(\frac{1}{N} \sum_{n=1}^N \frac{\partial^2}{\partial \theta_i \partial \theta_j} D_{KL}(\pi_{\theta_{old}}(\cdot | s_n) \parallel \pi_{\theta}(\cdot | s_n)) \right) |_{\theta=\theta_{old}}$ 를 사용한다.
- 테일러 시리즈에서 $L_{\theta_{old}}$ 의 2차는 $\overline{D}_{KL}(\theta_{old} \parallel \theta)$ 보다 너무 작아서 무시된다.

$$L_{\theta_{old}}(\theta) \approx \cancel{L_{\theta_{old}}(\theta_{old})} + \nabla_{\theta} L_{\theta_{old}}(\theta) \Big|_{\theta=\theta_{old}} (\theta - \theta_{old}) + \dots \quad \bar{D}_{KL}(\theta_{old} \parallel \theta_{old}) = 0 \text{ and } \bar{D}_{KL}(\theta_{old} \parallel \theta) \geq 0$$

$$\bar{D}_{KL}(\theta_{old} \parallel \theta) \approx \cancel{\bar{D}_{KL}(\theta_{old} \parallel \theta_{old})} + \nabla_{\theta} \bar{D}_{KL}(\theta_{old} \parallel \theta) \Big|_{\theta=\theta_{old}} (\theta - \theta_{old}) + \frac{1}{2}(\theta - \theta_{old})^T H (\theta - \theta_{old})$$

- 기울기 상승 대신 사용되는 **Natural gradient** $H^{-1} \nabla_{\theta} L_{\theta_{old}}(\theta) \Big|_{\theta=\theta_{old}}$ 는 가장 가파른(steepest) 방향이다.

- $\nabla_{\theta} L_{\theta_{old}}(\theta) \Big|_{\theta=\theta_{old}}$ 는 기존의 폴리시 그래디언트이다.
- H 는 모델 파라미터 θ 에 대한 정책의 민감도(curvature:곡률)를 측정한다.

- 정책 업데이트는 분석적으로 얻을 수 있다.

- Nature Policy Gradient**

$$\theta = \theta_{old} + \sqrt{\frac{2\delta}{g^T H^{-1} g}} H^{-1} g \text{ where } g = \nabla_{\theta} L_{\theta_{old}}(\theta) \Big|_{\theta=\theta_{old}} \quad (26)$$

- $\theta = \theta_{old} + \beta H^{-1} g$ 가 $\delta = \frac{1}{2}(\beta H^{-1} g)^T H (\beta H^{-1} g)$ 를 만족하도록 최대 스텝 길이 β 를 찾는다.
- NPG는 2차 최적화이며 일반 1차 기울기 상승보다 빠르게 수렴하지만, 정책이 DNN처럼 많은 파라미터로 파라미터라이즈된 경우 H^{-1} 을 찾는 데 비용이 많이 든다.
- 대신 $f'(x) = Hx - g = 0$ 에서 $\min_{\theta} f(x) = \frac{1}{2}x^T Hx - gx$ 를 찾아서, $Hx = g$ 처럼 $x = H^{-1}g$ (그래서 $g^T = H^{-1}g$)를 직접 계산하다.
- Conjugate gradient** 방법은 정정 행렬(positive-definite matrix) H 를 사용하여 특정 선형 방정식 시스템 $Hx = g$ 를 일반 기울기 하강법보다 더 적은 이터레이션 횟수로 수치적으로 풀지만, 구현을 더 복잡하게 만든다.
- Line search**는 목적 함수를 개선하는지와 KL 제약 조건을 만족하는지에 대한 보장을 위해 사용한다. 최대값 β 에서 시작하여 목적 함수가 개선되고 KL 제약 조건이 충족될 때까지 반 복적으로 $\alpha(0 < \alpha < 1)$ 를 곱하여 β 를 감소 시킨다.

1.5 TRPO 알고리즘

입력 : 정책 파라미터 θ 를 초기화

$k = 0, 1, 2, \dots$ 를 반복(한번의 정책 업데이트 단위)

정책 $\pi_k = \pi(\theta_k)$ 에서 트라젝토리 \mathcal{D}_k 의 집합을 수집(샘플링)한다.

어떤 어드밴티지 추정 알고리즘을 사용해서 어드밴티지 $\hat{A}_t^{\pi_k}$ 를 추정한다.

이 샘플들을 사용해서 다음을 구한다.

어드밴티지 추정을 사용해서 정책 기울기 g_k

KL-발산 헤시안-벡터 함수 $f(v) = \hat{H}_k v$

n_{cg} 이터레이션을 하는 Conjugate Gradient를 사용해서, $x_k \approx \hat{H}_k^{-1} g_k$ 를 얻는다.

제안된 스텝을 추정 $\Delta_k \approx \sqrt{\frac{2\delta}{x_k^T \hat{H}_k x_k}} x_k$

최종 업데이트를 얻기위해 지수적 감쇠를 하는 역추적 Line Search를 수행한다.

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

1.6 TRPO의 단점

- TRPO는 Adam과 같은 1차 옵티마이저를 사용하는 정책 그라디언트에 비해 샘플 효율성이 떨어진다.
- TRPO는 2차 방정식 $\min_x \frac{1}{2} x^T H x - g x$ 를 근사 $H^{-1} g$ 로 최소화한다. 그러나 각 파라미터 업데이트에서 여전히 H를 위한 고가의 컴퓨팅이 필요하며 이는 확장성을 손상시킨다.
- TRPO는 심층 CNN 및 RNN이 필요한 작업에는 실용적이지 않다.

2 PPO(Proximal Policy Optimization)

2.1 개요

- TRPO는 높은 성능을 달성했지만, 정책이 너무 크게 변경되는 것을 방지하는, KL 제약 조건으로 인해 구현 및 계산이 복잡하다. 다음은 TRPO의 써로게이트 오브젝티브 평션이었다.

$$\begin{aligned} \max_{\theta} L^{TRPO}(\theta) &= \mathbb{E} \left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)} A_{\theta_{old}}(s, a) \right] \\ \text{subject to } \mathbb{E} [D_{KL}(\pi_{\theta_{old}}(\cdot|s) \parallel \pi_{\theta}(\cdot|s))] &\leq \delta \end{aligned} \quad (27)$$

- π_{old} 와 π 사이의 거리에 대한 신뢰 영역에 대한 제약 조건이 없으면, 최대화는 큰 정책 비율(policy ratio)을 유발하여 큰 파라미터 업데이트로 불안정성을 초래한다.
- PPO는 정책 비율(policy ratio) $r(\theta) = \frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)}$ 을 클리핑하는, **클리핑 써로게이트 목적 함수 (Clipped surrogate objective function)**를 도입하였다.

$$\max_{\theta} L^{CLIP}(\theta) = \mathbb{E} \left[\min(r(\theta) A_{\theta_{old}}(s, a), \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) A_{\theta_{old}}(s, a)) \right] \quad (28)$$

- KL 제약을 대체하는 이 클리핑은 정책 업데이트가 너무 클 때 페널티를 제공하고 무거운 계산을 줄인다(1차 확률적 경사 상승만 사용).
- $L^{CLIP}(\theta)$ 는 정책 성능의 하한을 보장하는 $L^{TRPO}(\theta)$ 의 하한이다. 이것으로 PPO가 OpenAI의 기본 DRL 알고리즘이 되었다.

2.2 PPO 클리핑 써로게이트 목적 함수(clipped surrogate objective function)

$$L^{CLIP}(\theta) = \mathbb{E}[\min(r(\theta)A_{\theta_{old}}(s, a), \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon)A_{\theta_{old}}(s, a))] \quad (29)$$

$$\text{for } r(\theta) = \frac{\pi_{\theta}(a|a)}{\pi_{\theta_{old}}(a|s)}$$

- $A_{\theta_{old}}(s, a) > 0$ 일 때, 액션 a는 상태 s의 모든 행동의 평균보다 좋고, a가 채택될 확률이 높아 지도록 $r(\theta)$ 를 증가시켜 a를 장려해야 한다.
- 그러나 $r(\theta)$ 가 너무 높으면 큰 변화를 방지하기 위해 $1 + \epsilon$ 만큼 클리핑한다. (논문에서 $\epsilon = 0.2$)
- $A_{\theta_{old}}(s, a) < 0$ 인 경우, 액션이 권장되지 않으며, $r(\theta)$ 는 $1 - \epsilon$ 만큼만 감소한다.

2.3 실제 구현

$$L^{CLIP+VF+S}(\theta) = \mathbb{E}[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_{\theta}](s)] \quad (30)$$

- $L^{VF}(\theta) = (V_{target} - V_{\theta}(s))^2$ 는 벨류(value) 추정시 에러 항
- 분산 감소 어드밴티지 함수 추정을 계산하기 위한 대부분의 기술은 학습된 상태-벨류(state-value) 함수 $V_{\theta}(s)$ 를 사용한다.
- 정책과 가치 함수 간에 파라미터를 공유하는 신경망을 사용하는 경우, 손실 함수는 정책 써로게이트와 이 에러 항을 결합해야 한다.
- $S[\pi_{\theta}](s)$ 는 충분한 탐색(exploration)을 보장하기 위한 엔트로피 보너스이다.

3 DRL 알고리즘들의 비교

알고리즘	모델	상태 공간	액션 공간	정책	오브젝티브
살사	모델-프리 RL	이산적	이산적	온-폴리시	Q-벨류
Q-러닝	모델-프리 RL	이산적	이산적	오프-폴리시	Q-벨류
DQN	모델-프리 DRL	연속적	이산적	오프-폴리시	Q-벨류
A3C	모델-프리 DRL	연속적	연속적	온-폴리시	어드밴티지
DDPG	모델-프리 DRL	연속적	연속적	오프-폴리시	Q-벨류
TRPO	모델-프리 DRL	연속적	연속적	온-폴리시	어드밴티지
PPO	모델-프리 DRL	연속적	연속적	온-폴리시	어드밴티지

- DQN은 RL의 상태 공간을 이산에서 연속 공간으로 크게 개선시켰다.
- A3C 및 DDPG는 에이전트가 정책 기울기(Policy Gradient) 알고리즘으로 지속적인 작업을 수행할 수 있게 하여, 로봇 제어와 같은 RL의 적용을 확장할 수 있는 또 다른 혁신이었다.
- TRPO는 KL 발산 제약 조건이 있는 써로게이트 목적 함수를 도입하여, 무감소 리턴을 보장함으로써 DDPG의 성능을 향상시켰다.
- PPO는 클리핑된 써로게이트 목적 함수를 사용하는데, TRPO를 수정하여 성능을 개선하고, 구현 및 계산의 복잡성을 줄였다.