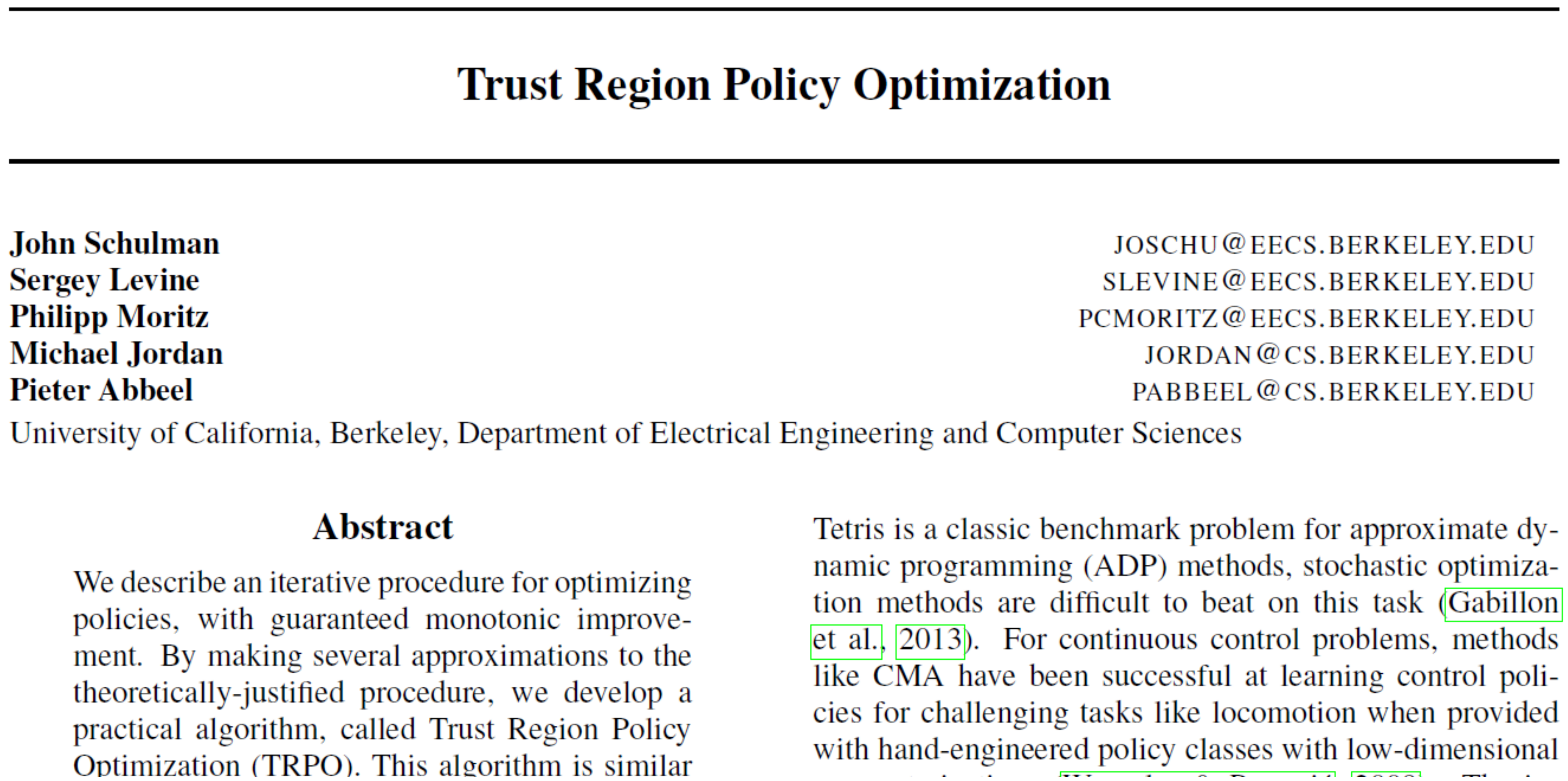
TRPO Paper Review

[쉽게 읽는 강화학습 논문 5화]

팡요랩 – 노승은, 전민영

2019.03.10



Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar

Tetris is a classic benchmark problem for approximate dynamic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional

ICML 2015, 1015회 인용

굉장히 이론적인 접근을 담고 있는 논문...

다음과 같은 분들에게 추천합니다

- RL에 어느정도 자신이 있는 분
  - RL 에서 쓰이는 notation 에 어느정도 익숙한 분
  - 팡요랩 1~10강 내용이 이제는 어렵지 않게 느껴지는 분
- TRPO를 수식까지 포함하여 깊이 있게 이해하고 싶은 분
  - TRPO는 PPO의 전신
- Notation 트레이닝을 하드하게 해보고 싶은 분

차례

1. 준비 운동

- MDP : $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$

  $\mathcal{S}$ is a finite set of states

  $\mathcal{A}$ is a finite set of actions

  $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the transition probability

  $r : \mathcal{S} \rightarrow \mathbb{R}$ is the reward function

  $\rho_0 : \mathcal{S} \rightarrow \mathbb{R}$ is the distribution of the initial state s_0

  $\gamma \in (0, 1)$ is the discount factor

- Policy

  $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$

1. 준비 운동

- Expected Cumulative Discounted reward

$$\eta(\pi) = \mathbb{E}_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right]$$

where $s_0 \sim \rho_0(s_0),\ a_t \sim \pi(a_t|s_t),\ s_{t+1} \sim P(s_{t+1}|s_t, a_t)$.

- Action value, Value, Advantage functions

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \ldots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right] \quad V_\pi(s_t) = \mathbb{E}_{a_t, s_{t+1}, \ldots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right]$$

$$A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s) \quad \text{where } a_t \sim \pi(a_t|s_t), s_{t+1} \sim P(s_{t+1}|s_t, a_t) \text{ for } t \geq 0$$

2. 모든 것의 출발점 : Kakade & Langford

- Kakade & Langford (2002)

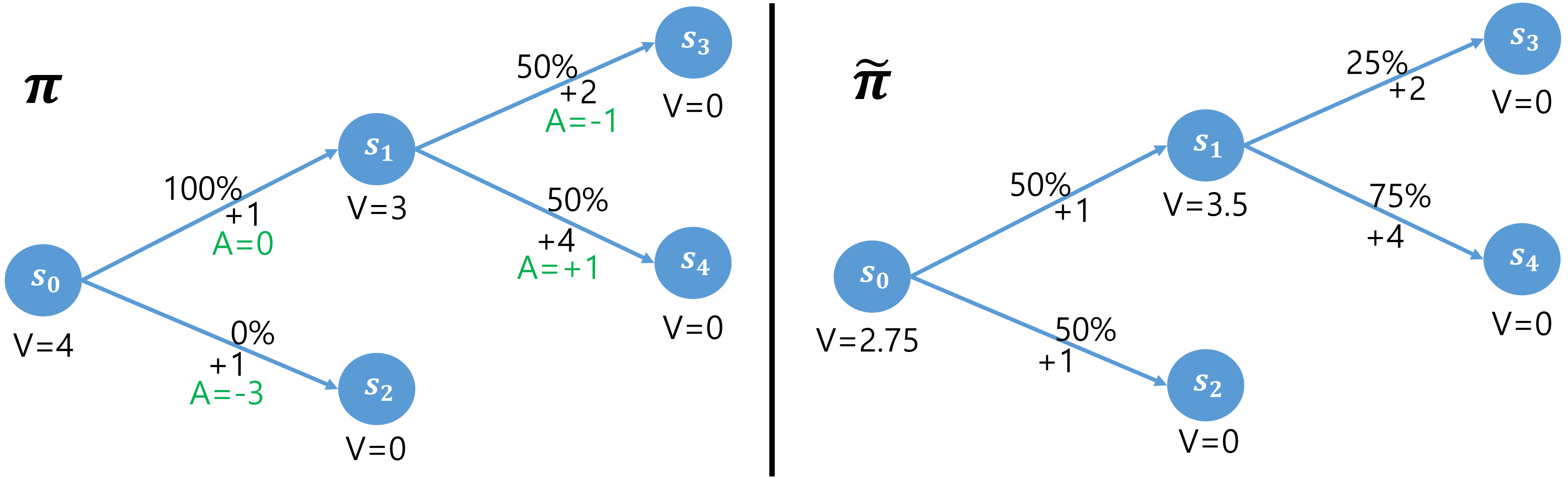
$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \cdots \sim \tilde{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t) \right]$$

where $\mathbb{E}_{s_0, a_0, \cdots \sim \tilde{\pi}} [\ldots]$ indicates that actions are sampled $a_t \sim \tilde{\pi}(\cdot|s_t)$

- Discounted (unnormalized) visitation frequency

$$\rho_\pi(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s) + \ldots$$

where $s_0 \sim \rho_0$ and the actions are chosen according to π



$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \cdots \sim \tilde{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t) \right]$$

좌변 $= 2.75$

우변 $= 4 + 0.5 * -3 + 0.5 \; * (0 + 0.25 * -1 + 0.75 * 1)$

$= 4 - 1.5 \qquad + 0.5 \; * 0.5$

$= 2.5 \qquad + 0.25$

$= 2.75$

2. 모든 것의 출발점 : Kakade & Langford

- 증명

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

2. 모든 것의 출발점 : Kakade & Langford

- Sum over timesteps 의 관점에서 => Sum over states 의 관점으로!

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \cdots \sim \tilde{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t) \right]$$

$$\Downarrow$$

$$\begin{aligned}
\eta(\tilde{\pi}) &= \eta(\pi) + \sum_{t=0}^{\infty} \sum_{s} P(s_t = s|\tilde{\pi}) \sum_{a} \tilde{\pi}(a|s) \gamma^t A_\pi(s, a) \\
&= \eta(\pi) + \sum_{s} \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\tilde{\pi}) \sum_{a} \tilde{\pi}(a|s) A_\pi(s, a) \\
&= \eta(\pi) + \sum_{s} \rho_{\tilde{\pi}}(s) \sum_{a} \tilde{\pi}(a|s) A_\pi(s, a).
\end{aligned}$$

2. 모든 것의 출발점 : Kakade & Langford

- 식의 의미

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

1. 만일 모든 state s 에 대해 $\sum_a \tilde{\pi}(a|s)A_\pi(s,a) \geq 0$. 라면, Policy의 성능인 η 가 증가하는게 **보장**된다는 뜻임!

2. 예시
   실버 강의에서 배웠던 policy iteration : $\tilde{\pi}(s) = \arg\max_a A_\pi(s,a)$
   여기서 단 한 개의 state, action 쌍에 대해 어드밴티지가 양수이면 policy는 개선됨.

3. 그러나 !
   실제에서는 A_π 를 측정할때 뉴럴넷의 approximation error가 있어서 $\sum_a \tilde{\pi}(a|s)A_\pi(s,a) < 0$
   요런 state, action 쌍이 섞여 있을 것임.

2. 모든 것의 출발점 : Kakade & Langford

- $\rho_{\tilde{\pi}}(s)$ 는 까다로워!

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_{s} \rho_{\tilde{\pi}}(s) \sum_{a} \tilde{\pi}(a|s) A_{\pi}(s,a)$$

$$\Big\downarrow \text{ 대체!}$$

$$L_{\pi}(\tilde{\pi}) = \eta(\pi) + \sum_{s} \rho_{\pi}(s) \sum_{a} \tilde{\pi}(a|s) A_{\pi}(s,a)$$

- 그래도 됨 ㅎㅎ

왜냐하면 π를 θ로 parametrize 하고 $\pi_\theta(a|s)$ 가 θ에 대해 미분가능하다고 하면 다음이 성립하기 때문.

$$L_{\pi_{\theta_0}}(\pi_{\theta_0}) = \eta(\pi_{\theta_0}),$$
$$\nabla_\theta L_{\pi_{\theta_0}}(\pi_\theta)\big|_{\theta=\theta_0} = \nabla_\theta \eta(\pi_\theta)\big|_{\theta=\theta_0}$$

2. 모든 것의 출발점 : Kakade & Langford

$$L_{\pi_{\theta_0}}(\pi_{\theta_0}) = \eta(\pi_{\theta_0}),$$
$$\nabla_\theta L_{\pi_{\theta_0}}(\pi_\theta)\big|_{\theta=\theta_0} = \nabla_\theta \eta(\pi_\theta)\big|_{\theta=\theta_0}$$

- 이 식의 의미는 L_π 와 η 가 first – order 로 근사하면 같다는 뜻이다.
- 충분히 작은 step 만큼 policy를 update 하면 : $\pi_{\theta_0} \rightarrow \tilde{\pi}$

  $L_{\pi_{\theta_{\text{old}}}}$ 를 증가시키는게 곧 η 를 증가시키는 것과 같다는 의미!
- 하지만 이 식은 그 step이 얼만큼 작아야 하는지에 대해서는 알려주지는 않는다.
- 그래서 Kakade & Langford 는 어떤 policy update 방법론을 제시한다. 그 이름은

Conservative policy iteration

2. 모든 것의 출발점 : Kakade & Langford

- Conservative policy iteration

이전 policy와 새로운 policy를 일정 비율로 섞어 쓰는 방법론.

$$\pi_{\text{new}}(a|s) = (1-\alpha)\underline{\pi_{\text{old}}(a|s)} + \alpha\underline{\pi'(a|s)}$$

π_{old} : current policy $\qquad\qquad \pi' = \arg\max_{\pi'} L_{\pi_{\text{old}}}(\pi')$

- 이 때의 lower bound를 Kakade와 Langford 가 유도했다.

$$\eta(\pi_{\text{new}}) \geq L_{\pi_{\text{old}}}(\pi_{\text{new}}) - \frac{2\epsilon\gamma}{(1-\gamma)^2}\alpha^2$$

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

하지만 이 식은 mixture policy 에서만 쓸 수 있음… 실질적으로 별 도움이 못된다.

좀더 general 하게 통용되는 방법론이 필요!

3. General한 improvement guarantee

- Total variation divergence 개념을 도입

$$D_{TV}(p \parallel q) = \tfrac{1}{2}\sum_i |p_i - q_i|$$

예컨대 p = (1/3, 1/3, 1/3) , q = (1, 0, 0) 이라면 $D_{TV}(p||q) = \frac{1}{2}(\frac{2}{3}+\frac{1}{3}+\frac{1}{3})$ = 2/3

$$D_{\mathrm{TV}}^{\max}(\pi, \tilde{\pi}) = \max_s D_{TV}(\pi(\cdot|s) \parallel \tilde{\pi}(\cdot|s))$$

- **Theorem 1.** *Let* $\alpha = D_{\mathrm{TV}}^{\max}(\pi_{\mathrm{old}}, \pi_{\mathrm{new}})$. *Then the following bound holds:*

$$\eta(\pi_{\mathrm{new}}) \geq L_{\pi_{\mathrm{old}}}(\pi_{\mathrm{new}}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2 \quad \textit{where } \epsilon = \max_{s,a}|A_\pi(s,a)|$$

3. General한 improvement guarantee

- 즉, 다음과 같이 바뀐거임

$$\eta(\pi_{\text{new}}) \geq L_{\pi_{\text{old}}}(\pi_{\text{new}}) - \frac{2\epsilon\gamma}{(1-\gamma)^2}\alpha^2 \quad \text{where } \epsilon = \max_s \left| \mathbb{E}_{a\sim\pi'(a|s)}\left[A_\pi(s,a)\right] \right|$$

$$\Downarrow$$

$$\eta(\pi_{\text{new}}) \geq L_{\pi_{\text{old}}}(\pi_{\text{new}}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{\text{TV}}^{\max}(\pi_{\text{old}}, \pi_{\text{new}})^2 \quad \textit{where } \epsilon = \max_{s,a} |A_\pi(s,a)|$$

- KL Divergence를 도입하여 한 번 더 바꿈.

$$D_{TV}(p \parallel q)^2 \leq D_{\text{KL}}(p \parallel q)$$ -------------- (Pollard, 2000)

$$D_{\text{KL}}^{\max}(\pi, \tilde{\pi}) = \max_s D_{\text{KL}}(\pi(\cdot|s) \parallel \tilde{\pi}(\cdot|s))$$

3. General한 improvement guarantee

$$\eta(\pi_{\text{new}}) \geq L_{\pi_{\text{old}}}(\pi_{\text{new}}) - \frac{2\epsilon\gamma}{(1-\gamma)^2}\alpha^2 \quad \text{where } \epsilon = \max_s \left| \mathbb{E}_{a\sim\pi'(a|s)}\left[A_\pi(s,a)\right] \right|$$

$\Downarrow$ By theorem 1

$$\eta(\pi_{\text{new}}) \geq L_{\pi_{\text{old}}}(\pi_{\text{new}}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{\text{TV}}^{\max}(\pi_{\text{old}}, \pi_{\text{new}})^2 \quad \textit{where } \epsilon = \max_{s,a} |A_\pi(s,a)|$$

$\Downarrow$ By $D_{TV}(p \parallel q)^2 \leq D_{\text{KL}}(p \parallel q)$

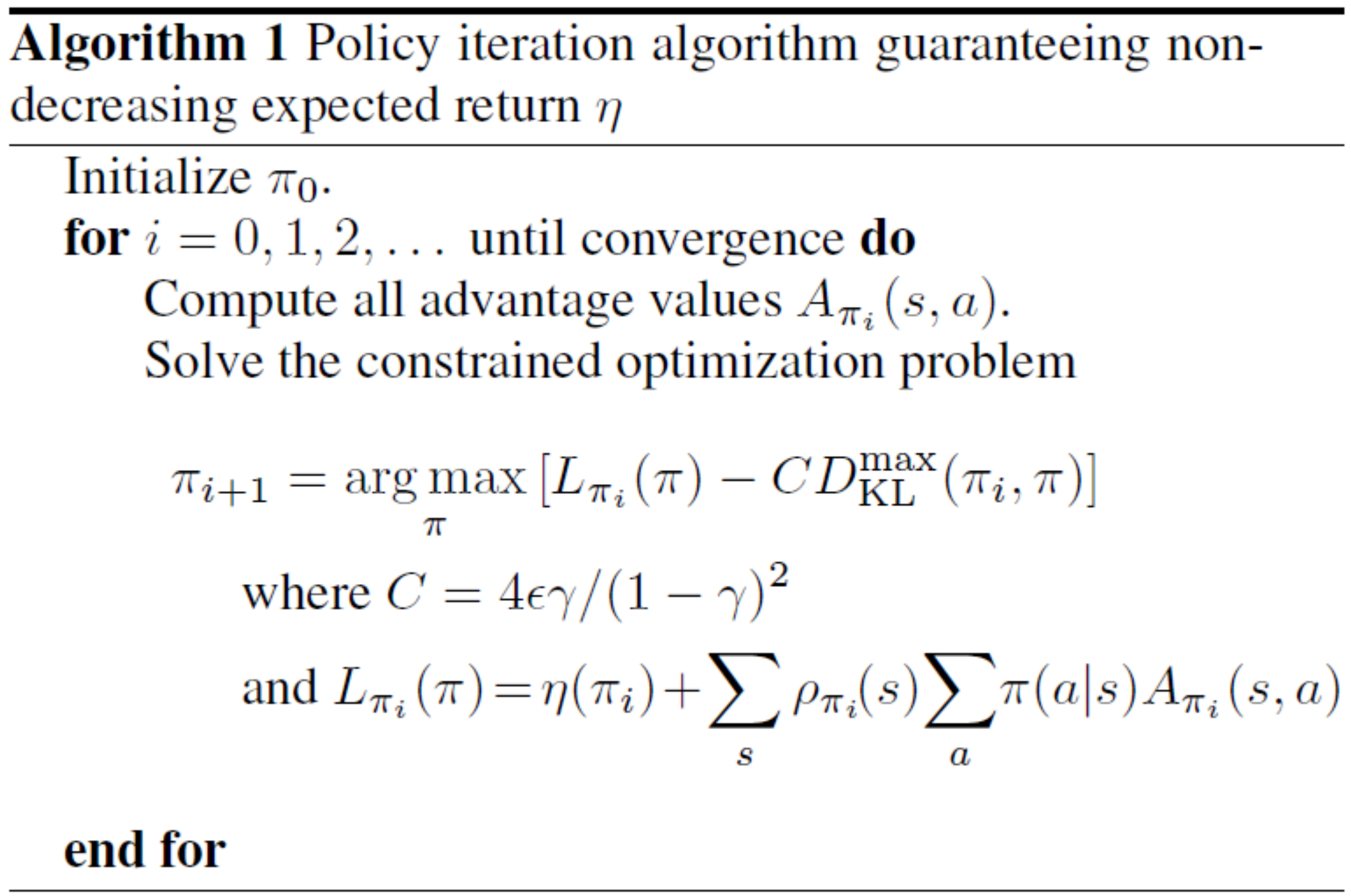
$$\eta(\tilde{\pi}) \geq L_\pi(\tilde{\pi}) - \frac{4\epsilon\gamma}{(1-\gamma)^2} D_{\text{KL}}^{\max}(\pi, \tilde{\pi})$$

$\Downarrow$ By 치환

$$\eta(\tilde{\pi}) \geq L_\pi(\tilde{\pi}) - C D_{\text{KL}}^{\max}(\pi, \tilde{\pi}), \quad \text{where } C = \frac{4\epsilon\gamma}{(1-\gamma)^2}.$$

3. General한 improvement guarantee

- 그렇게 하여 만들어진 알고리즘 !



Algorithm 1 Policy iteration algorithm guaranteeing non-decreasing expected return η

```
Initialize $\pi_0$.
for $i = 0, 1, 2, \dots$ until convergence do
    Compute all advantage values $A_{\pi_i}(s, a)$.
    Solve the constrained optimization problem

    $\pi_{i+1} = \arg\max_{\pi} [L_{\pi_i}(\pi) - C D_{\mathrm{KL}}^{\max}(\pi_i, \pi)]$
        where $C = 4\epsilon\gamma/(1-\gamma)^2$
        and $L_{\pi_i}(\pi) = \eta(\pi_i) + \sum_s \rho_{\pi_i}(s) \sum_a \pi(a|s) A_{\pi_i}(s, a)$

end for
```

3. General한 improvement guarantee

- 알고리즘은 성능 개선을 보장할까..?

1. 즉, $\pi_{i+1} = \arg\max \left[L_{\pi_i}(\pi) - C D_{\mathrm{KL}}^{\max}(\pi_i, \pi)\right]$ 일때 $\eta(\pi_0) \le \eta(\pi_1) \le \eta(\pi_2) \le \ldots.$ 가 성립 할까?
2. $M_i(\pi) = L_{\pi_i}(\pi) - C D_{\mathrm{KL}}^{\max}(\pi_i, \pi)$ 라고 하자. 이때
   1. $\eta(\pi_{i+1}) \ge M_i(\pi_{i+1})$ --------------------- by $\eta(\tilde{\pi}) \ge L_{\pi}(\tilde{\pi}) - C D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi})$
   2. $\eta(\pi_i) = M_i(\pi_i)$
   3. 그러므로 $\eta(\pi_{i+1}) - \eta(\pi_i) \ge M_i(\pi_{i+1}) - M(\pi_i)$
3. 그러므로! 결국 M_i를 극대화 하는 것은 $\eta(\pi)$가 non-decreasing 함을 보장해준다.
4. 요기서 M_i를 $\eta(\pi)$ 의 surrogate function 이라고도 한다.

4. 보다 practical 한 알고리즘

- 지금까지 π 얘기를 할 때 parametrize 하지 않고 얘기했었음.
- 이제 π를 θ를 이용하여 parametrize 하겠음.
- 그리고 notation도 다 알맞게 바꾸겠음!

$$\eta(\theta) := \eta(\pi_\theta)$$

$$L_\theta(\tilde{\theta}) := L_{\pi_\theta}(\pi_{\tilde{\theta}})$$

$$D_{\mathrm{KL}}(\theta \,\|\, \tilde{\theta}) := D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\tilde{\theta}})$$

- 그리고 지금부터 이전 policy의 parameter를 θ_{old} 로 쓰겠음.
- 최종 목적 함수는 다음과 같이 바뀜

$$\underset{\theta}{\operatorname{maximize}} \left[L_{\theta_{\mathrm{old}}}(\theta) - C D_{\mathrm{KL}}^{\max}(\theta_{\mathrm{old}}, \theta) \right]$$

4. 보다 practical 한 알고리즘

$$\underset{\theta}{\text{maximize}}\,[L_{\theta_{\text{old}}}(\theta) - C D_{\text{KL}}^{\max}(\theta_{\text{old}}, \theta)] \quad \text{where } C = \frac{4\epsilon\gamma}{(1-\gamma)^2}.$$

- 이런 형태를 Penalty 형태라고 함.
- 그런데 여기서, C가 무척 큰 숫자임.
- 이렇게 될 경우 penalty가 무서워서 step size가 매우 작아질 수 밖에 없음.
- 그래서 한 가지 다른 방법은 다음과 같이 목적함수를 constraint 형태로 변경하는 것임.

$$\underset{\theta}{\text{maximize}}\, L_{\theta_{\text{old}}}(\theta)$$
$$\text{subject to } D_{\text{KL}}^{\max}(\theta_{\text{old}}, \theta) \le \delta$$

4. 보다 practical 한 알고리즘

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

- 다음 문제는 … $D_{\text{KL}}^{\max}(\theta_{\text{old}}, \theta)$ 를 측정할 수 없다는 것임.
- 이 값은 policy가 “모든” states 에서 평가될 수 있다는 가정에 기반한 것인데 이는 현실세계에서 불가능한 가정.
- 따라서 $D_{\text{KL}}^{\max}(\theta_{\text{old}}, \theta)$ 대신 다음을 도입한다.

$$\overline{D}_{\text{KL}}^{\rho}(\theta_1, \theta_2) := \mathbb{E}_{s\sim\rho}\left[D_{\text{KL}}(\pi_{\theta_1}(\cdot|s) \,\|\, \pi_{\theta_2}(\cdot|s))\right]$$

- 이는 heuristic approximation 으로 average KL divergence를 쓰는 방법이다.

$$\underset{\theta}{\text{maximize}}\, L_{\theta_{\text{old}}}(\theta)$$
$$\text{subject to } \overline{D}_{\text{KL}}^{\rho_{\theta_{\text{old}}}}(\theta_{\text{old}}, \theta) \leq \delta.$$

5. 샘플 기반의 objective and constraint estimation

$$\begin{aligned} &\underset{\theta}{\text{maximize}}\, L_{\theta_{\text{old}}}(\theta) \\ &\text{subject to } \overline{D}_{\text{KL}}^{\rho_{\theta_{\text{old}}}}(\theta_{\text{old}}, \theta) \le \delta. \end{aligned}$$

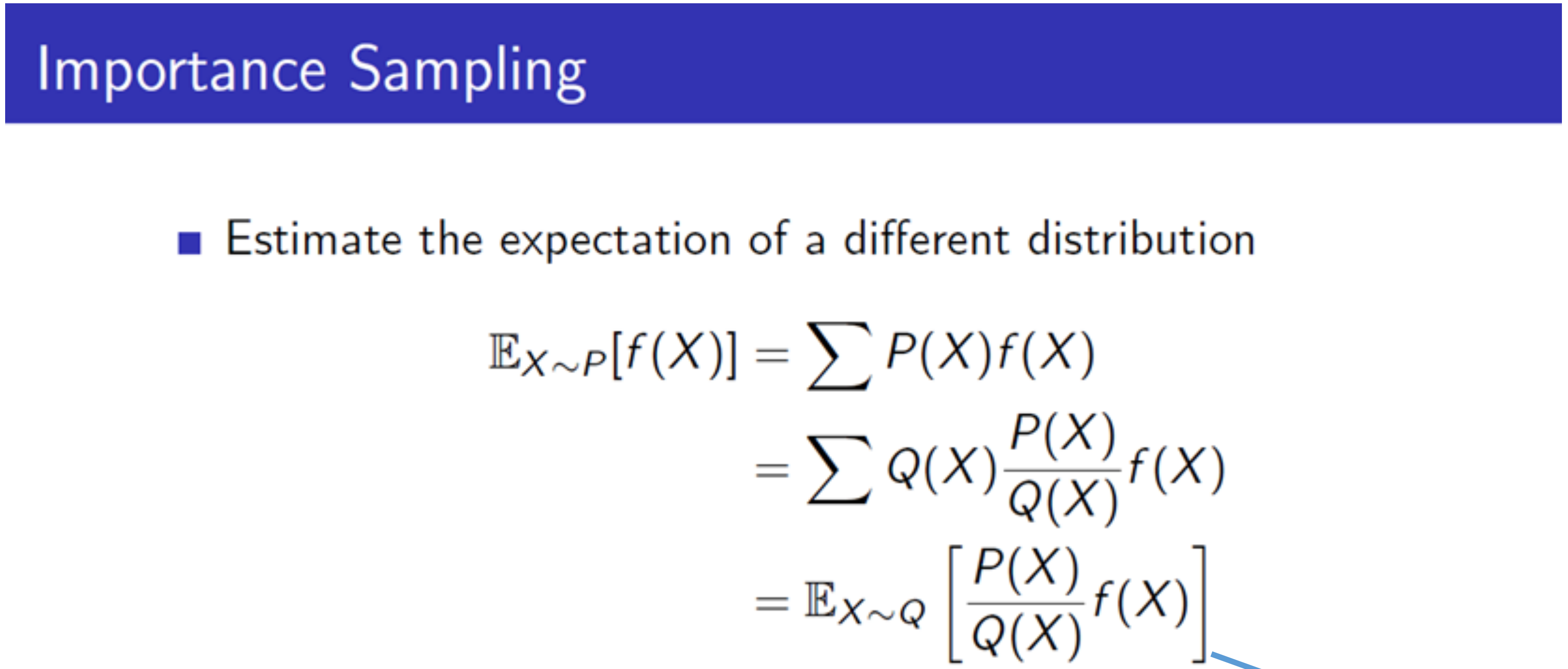
- Objective랑 constraint를 어떻게 샘플 기반의 monte-carlo estimate 으로 대체할 것인가?
- 일단 L을 풀어 써보자

$$\begin{aligned} &\underset{\theta}{\text{maximize}} \sum_s \rho_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a|s) A_{\theta_{\text{old}}}(s,a) \\ &\qquad \text{subject to } \overline{D}_{\text{KL}}^{\rho_{\theta_{\text{old}}}}(\theta_{\text{old}}, \theta) \le \delta. \end{aligned}$$

- 일단 $\sum_s \rho_{\theta_{\text{old}}}(s) [\dots]$ 를 $\frac{1}{1-\gamma} \mathbb{E}_{s \sim \rho_{\theta_{\text{old}}}} [\dots]$로 대체
- 그다음 $A_{\theta_{\text{old}}}$ 를 $Q_{\theta_{\text{old}}}$ 로 대체. (그래도 괜찮은게 objective를 constant 만큼 바꾸기 때문)
- 마지막으로 다음을 이용 $\rightarrow$

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

잠깐 복습!



Importance Sampling

- Estimate the expectation of a different distribution

$$
\begin{aligned}
\mathbb{E}_{X\sim P}[f(X)] &= \sum P(X)f(X) \\
&= \sum Q(X)\frac{P(X)}{Q(X)}f(X) \\
&= \mathbb{E}_{X\sim Q}\left[\frac{P(X)}{Q(X)}f(X)\right]
\end{aligned}
$$

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

q는 sampling distribution. 여기선 old policy를 사용하겠죠?

5. 샘플 기반의 objective and constraint estimation

$$\underset{\theta}{\text{maximize}} \sum_s \rho_{\theta_{\text{old}}}(s) \sum_a \pi_\theta(a|s) A_{\theta_{\text{old}}}(s,a)$$
$$\text{subject to } \overline{D}_{\text{KL}}^{\rho_{\theta_{\text{old}}}}(\theta_{\text{old}}, \theta) \leq \delta.$$

대체! (완전히 equivalent)

$$\underset{\theta}{\text{maximize}}\, \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}, a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{\text{old}}}(s,a)\right]$$
$$\text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s) \,\|\, \pi_\theta(\cdot|s))\right] \leq \delta.$$

5.1 Single Path

In this estimation procedure, we collect a sequence of states by sampling $s_0 \sim \rho_0$ and then simulating the policy $\pi_{\theta_{\text{old}}}$ for some number of timesteps to generate a trajectory $s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T$. Hence, $q(a|s) = \pi_{\theta_{\text{old}}}(a|s)$. $Q_{\theta_{\text{old}}}(s,a)$ is computed at each state-action pair (s_t, a_t) by taking the discounted sum of future rewards along the trajectory.

5. 샘플 기반의 objective and constraint estimation

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

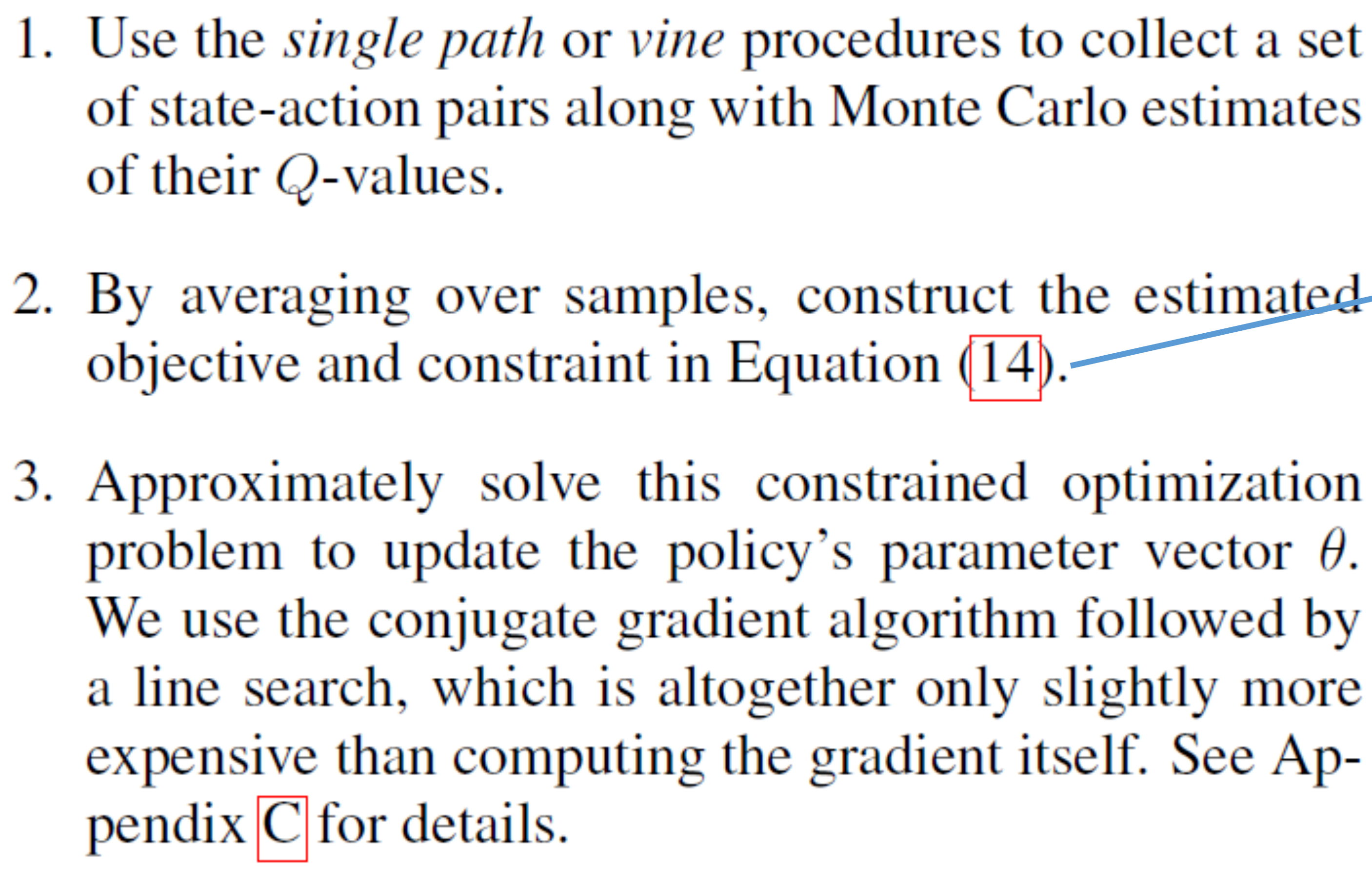
Single Path (Vine 은 pass)

- 이제부터는 그냥 $s_0 \sim \rho_0$ 로 부터 states sequence 들을 샘플링 한다.
- Action은 $\pi_{\theta_{\text{old}}}$ 를 이용해 진행한다. 즉, $q(a|s) = \pi_{\theta_{\text{old}}}(a|s)$
- 다음과 같은 trajectory 들을 계속해서 모은다.

$$s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T$$

- $Q_{\theta_{\text{old}}}(s,a)$ 는 trajectory 동안의 discounted sum of future rewards 로 계산함.
- 이를 이용해 위 식의 objective 와 constraint 를 계산!

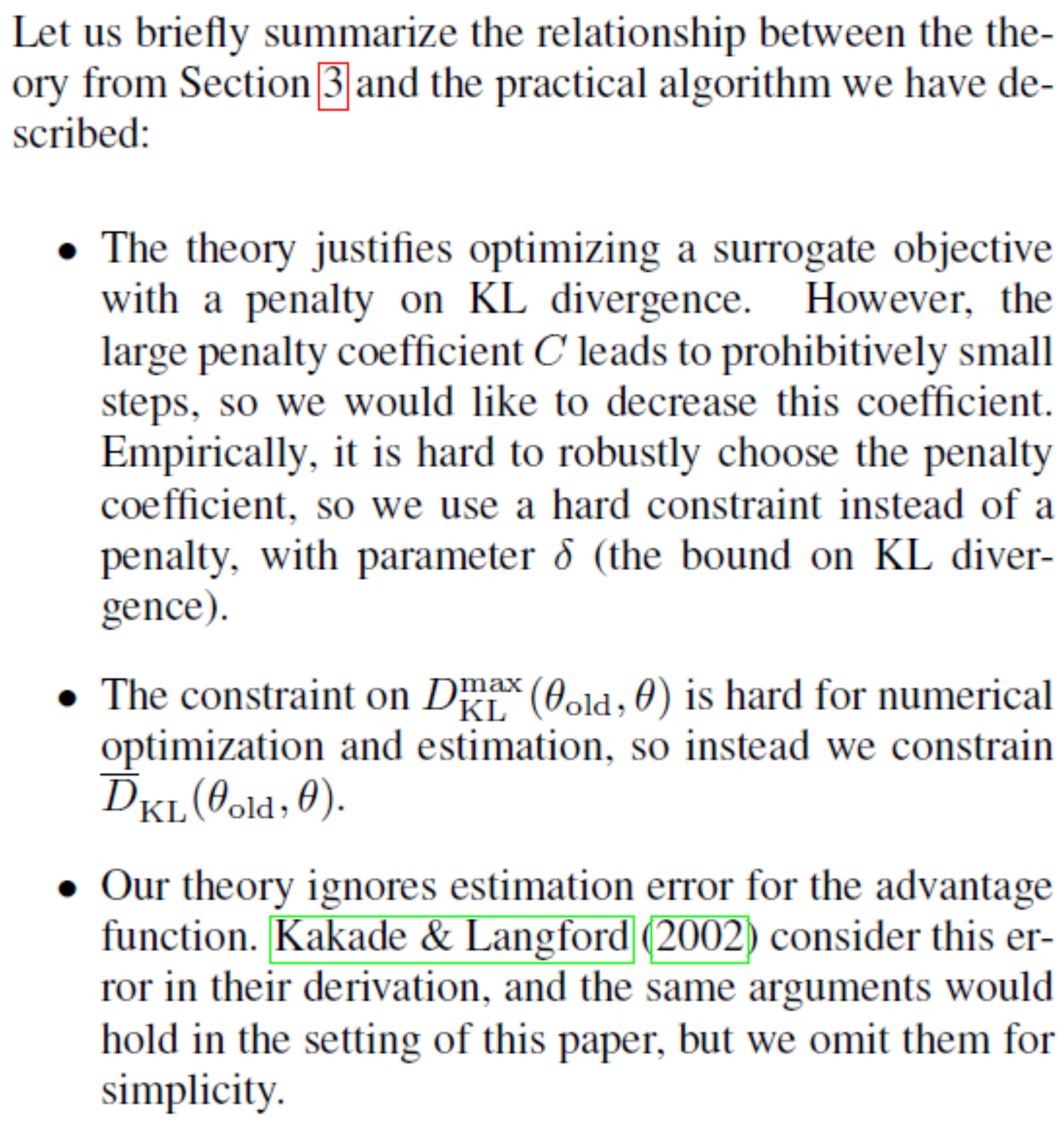
5. 샘플 기반의 objective and constraint estimation



1. Use the *single path* or *vine* procedures to collect a set of state-action pairs along with Monte Carlo estimates of their Q-values.
2. By averaging over samples, construct the estimated objective and constraint in Equation (14).
3. Approximately solve this constrained optimization problem to update the policy's parameter vector θ. We use the conjugate gradient algorithm followed by a line search, which is altogether only slightly more expensive than computing the gradient itself. See Appendix C for details.

$$\underset{\theta}{\text{maximize}}\, \mathbb{E}_{s\sim\rho_{\theta_\text{old}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_\text{old}}(s,a)\right]$$
$$\text{subject to } \mathbb{E}_{s\sim\rho_{\theta_\text{old}}}\left[D_\text{KL}(\pi_{\theta_\text{old}}(\cdot|s)\parallel\pi_\theta(\cdot|s))\right]\le\delta.$$

요약

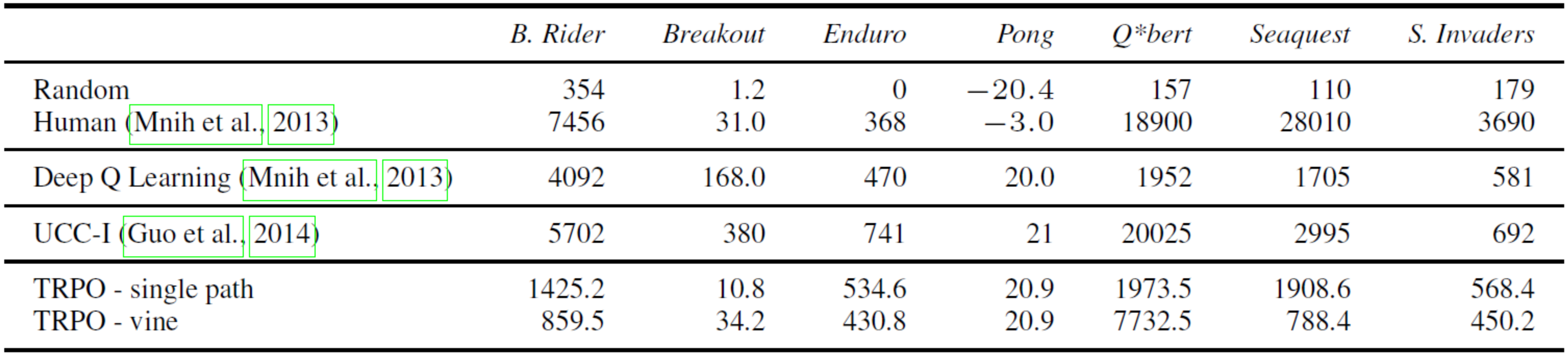


Let us briefly summarize the relationship between the theory from Section 3 and the practical algorithm we have described:

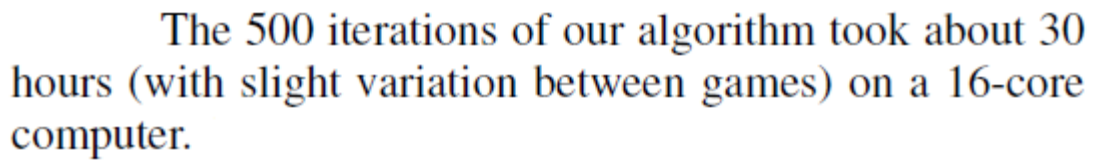
- The theory justifies optimizing a surrogate objective with a penalty on KL divergence. However, the large penalty coefficient C leads to prohibitively small steps, so we would like to decrease this coefficient. Empirically, it is hard to robustly choose the penalty coefficient, so we use a hard constraint instead of a penalty, with parameter δ (the bound on KL divergence).
- The constraint on $D_{\mathrm{KL}}^{\max}(\theta_{\mathrm{old}}, \theta)$ is hard for numerical optimization and estimation, so instead we constrain $\overline{D}_{\mathrm{KL}}(\theta_{\mathrm{old}}, \theta)$.
- Our theory ignores estimation error for the advantage function. Kakade & Langford (2002) consider this error in their derivation, and the same arguments would hold in the setting of this paper, but we omit them for simplicity.

- 이론적으로 KL divergence penalty 를 사용한 surrogate objective 를 최적화 해도 된다는 것을 확인함.
- 그런데 penalty 로 하면 C가 커서 step이 너무 작아져서, penalty 대신 constraint 를 사용함.
- $D_{\mathrm{KL}}^{\max}(\theta_{\mathrm{old}}, \theta)$는 측정하기 어려워서 평균 기반의 $\overline{D}_{\mathrm{KL}}(\theta_{\mathrm{old}}, \theta)$를 사용함.
- Advantage 함수의 측정 오차는 무시하였음.

Atari 실험 결과



	B. Rider	*Breakout*	*Enduro*	*Pong*	*Q*bert*	*Seaquest*	*S. Invaders*
Random	354	1.2	0	−20.4	157	110	179
Human (Mnih et al., 2013)	7456	31.0	368	−3.0	18900	28010	3690
Deep Q Learning (Mnih et al., 2013)	4092	168.0	470	20.0	1952	1705	581
UCC-I (Guo et al., 2014)	5702	380	741	21	20025	2995	692
TRPO - single path	1425.2	10.8	534.6	20.9	1973.5	1908.6	568.4
TRPO - vine	859.5	34.2	430.8	20.9	7732.5	788.4	450.2



The 500 iterations of our algorithm took about 30 hours (with slight variation between games) on a 16-core computer.