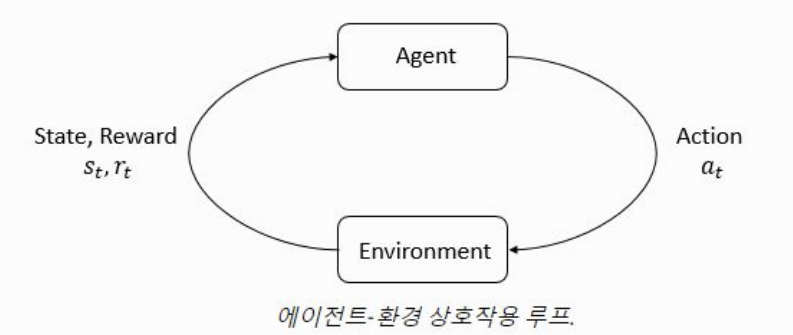
자료 소스 및 목적

- 자료: Spinning Up in Deep RL: https://github.com/openai/spinningup
  - “Constrained Policy Optimization” 의 저자인 OpenAI “Joshua Achiam” 이 작성한 교육자료
  - RL의 안전 전략과 관련된 내용을 고민하고 배경 지식을 획득
  - Python3, OpenAI Gym, OpenMPI, MuJoCo 실습 코드 제공 (PyTorch, Tensorflow)

- 목적:
  - 강화학습에 대한 “안전 전략”을 제공하기 위한 수학적 배경 + 직관적 이해

강화학습의 핵심 개념

- RL은 에이전트들이 시행착오를 통해 어떻게 학습하는지에 대한 연구
  - 행동에 대한 보상 (reward) 혹은 처벌 (constraint / penalty)
  - 환경 (environment) - 에이전트가 상호작용하는 환경을 정의 - MDP*
  - 에이전트 (agent): 환경과의 상호작용을 통해 환경의 상태를 (부분적으로) 관찰 - POMDP* 하여, 다음 행동을 결정함
  - 상태 (state): 환경의 변화 및 에이전트와의 상호작용으로 변하는 현재 환경의 상태
  - 보상 (reward): 현재 환경의 상태가 얼마나 좋거나 나쁜지 알려주는 숫자
  - 행동 (action): 에이전트와 환경의 상호작용으로, 누적 보상을 최대로 하도록 학습



에이전트-환경 상호작용 루프.

강화학습의 강한 제한 요인 - 상태, 관찰, 행동

- 상태 (state) 와 관찰 (observation) - *"환경 인식 범위를 제한"*
  - 세계의 상태에 대한 "완전한 설명"이며, 관찰은 상태에 대한 "부분적인 설명"임
  - 에이전트가 환경의 전체 상태를 관찰할 수 있을 때, "fully observed" MDP
  - 에이전트가 환경의 일부 상태만을 관찰할 수 있을 때, "partially observed" MDP

- 행동 공간 (action space) - *"에이전트의 행동 범위를 제한"*
  - 이산 행동 공간 - 바둑, Atari - 범주형 (categorical) 정책을 채택
    - [Softmax - 분류기]
  - 연속 행동 공간 - 로봇 제어 - 대각 가우시안 (diagonal Gaussian) 정책을 채택
    - 평균 벡터 (μ), 공분산 행렬 (Σ)로 정의 ⇒ multivariate normal distribution
      - VPG / TRPO / PPO 는 standard deviation 에 대한 파라미터로 covariance를 정의
      - 뉴럴 네트워크를 통해 state에 대한 standard deviation을 매핑하는 방법도 있음

$$a = \mu_\theta(s) + \sigma_\theta(s) \odot z$$

$$z \sim \mathcal{N}(0, I)$$

강화학습의 튜닝 - 궤적, 보상, 반환

- 궤적 (trajectory)
  - 세계의 상태와 행동의 순서로, “에피소드” 혹은 “롤아웃”으로 불림

$$\tau = (s_0, a_0, s_1, a_1, ...).$$

- 보상 (reward)과 반환 (return)
  - 보상은 세계의 현재 상태, 방금 취한 행동, 세계의 다음 상태에 따라 달라짐 $r_t = R(s_t, a_t, s_{t+1})$
    - 현재 상태 혹은 상태-행동에 대한 종속성으로 단순화 표현

$$r_t = R(s_t) \qquad r_t = R(s_t, a_t)$$

  - 에이전트는 궤적에 대한 누적 보상을 극대화 하는 것이지만, 계산하는 방법의 차이가 있음

강화학습의 문제 정의

- 강화 학습의 목표는 기대 수익(률)을 극대화 하는 정책을 선택하는 것
  - T 단계 궤적의 확률

$$P(\tau|\pi) = \underbrace{\rho_0(s_0)}_{\text{최초의 상태}} \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t).$$

  - 기대 수익률

$$J(\pi) = \int_{\tau} P(\tau|\pi)R(\tau) = \underset{\tau\sim\pi}{\mathrm{E}}\left[R(\tau)\right].$$

  - 최적의 정책

$$\pi^* = \arg\max_{\pi} J(\pi),$$

강화학습의 문제 정의

- 가치 함수
  - 가치: 상태 혹은 상태-행동 쌍에서 시작하여 특정 정책에 따라 행동할 경우에 대한 기대 수익
  - 주요 기능
    - On-Policy Value Function $V^{\pi}(s)$
      - 특정 상태에서 시작하여 특정 정책에 따라 행동할 경우의 기대 수익

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{\tau\sim\pi}\left[R(\tau)\,|s_0 = s\right]$$

    - On-Policy Action-Value Function $Q^{\pi}(s,a)$
      - 특정 상태과 임의의 행동 (정책을 따르지 않음)에서 시작하여, 특정 정책에 따라 행동할 경우의 기대 수익

$$Q^{\pi}(s,a) = \mathop{\mathrm{E}}_{\tau\sim\pi}\left[R(\tau)\,|s_0 = s, a_0 = a\right]$$

    - Optimal Value Function $V^{*}(s)$
      - 특정 상태에서 시작하여 최적의 정책을 따랐을 때의 기대 수익

$$V^{*}(s) = \max_{\pi} \mathop{\mathrm{E}}_{\tau\sim\pi}\left[R(\tau)\,|s_0 = s\right]$$

    - Optimal Action-Value Function $Q^{*}(s,a)$
      - 특정 상태와 임의의 행동에서 시작하여 최적의 정책으로 행동할 경우의 기대 수익

$$Q^{*}(s,a) = \max_{\pi} \mathop{\mathrm{E}}_{\tau\sim\pi}\left[R(\tau)\,|s_0 = s, a_0 = a\right]$$

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{a\sim\pi}\left[Q^{\pi}(s,a)\right], \quad V^{*}(s) = \max_{a} Q^{*}(s,a).$$

강화학습의 문제 정의: 벨만 방정식

- 개념
  - 출발점의 가치는 그곳에 도달함으로써 기대하는 보상과 다음 도착지에서의 가치를 합친 것임

$$V^{\pi}(s) = \mathop{\mathrm{E}}_{\substack{a\sim\pi \\ s'\sim P}}\left[r(s,a) + \gamma V^{\pi}(s')\right],$$

$$Q^{\pi}(s,a) = \mathop{\mathrm{E}}_{s'\sim P}\left[r(s,a) + \gamma \mathop{\mathrm{E}}_{a'\sim\pi}\left[Q^{\pi}(s',a')\right]\right],$$

Bellman backup

$$V^{*}(s) = \max_{a} \mathop{\mathrm{E}}_{s'\sim P}\left[r(s,a) + \gamma V^{*}(s')\right],$$

$$Q^{*}(s,a) = \mathop{\mathrm{E}}_{s'\sim P}\left[r(s,a) + \gamma \max_{a'} Q^{*}(s',a')\right].$$

- 이득 함수 (Advantage Functions)
  - 특정 상태에서 특정 행동을 취하는 것이 무작위 행동을 선택하는 것보다 얼마나 나은지 정의

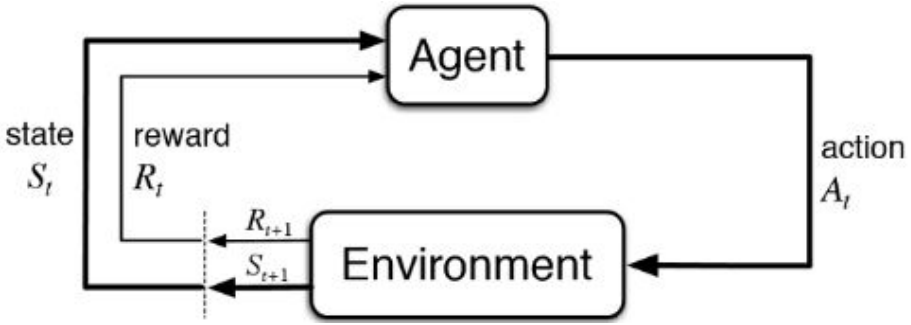
$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s).$$

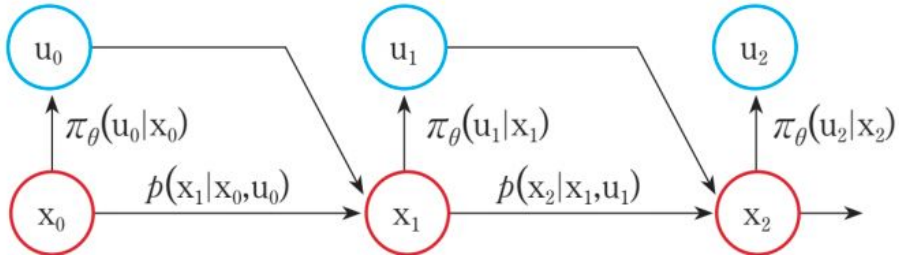
강화학습의 “수학적” 문제 정의 - MDP $\langle S, A, R, P, \rho_0 \rangle$

Markov Property: 전환 (transition) 은 가장 최근의 상태와 행동에만 의존하고, 과거에 종속되지 않음

- 마르코프 결정 프로세스 (Markov Decision Process)
  - 누적 보상을 최대로 하기 위한 최적의 정책을 구하는 것

1. 에이전트는 환경의 상태(S_t)를 측정한다.
2. 현재 상태에서 에이전트의 규칙(정책)에 따라 최적 행동(A_t)를 선택한다.
3. 선택한 행동에 의해 환경의 상태는 다음 상태(S_{t+1})로 전환된다.
4. 다음 상태(S_{t+1})를 바탕으로 에이전트는 새로운 행동(A_{t+1})을 선택한다.
5. 환경으로부터 주어지는 즉각적인 보상(R_{t+1})을 사용해 장기적인 성과를 계산해서 에이전트의 정책을 즉시 또는 주기적으로 계산한다.





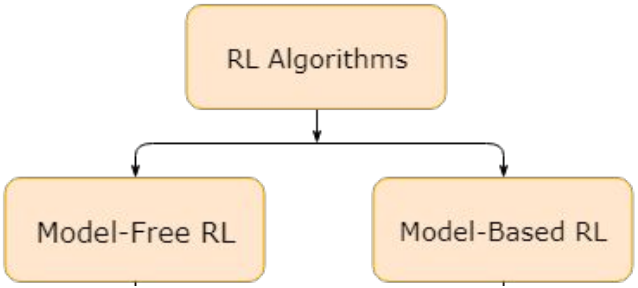
On-Policy vs. Off-Policy

- [정의] 온-폴리시 알고리즘
  - 현재 학습 중인 정책을 기반으로 데이터를 수집하고, 그 정책을 업데이트 하는 방법
  - 에이전트가 현재 사용하는 정책을 유지하며 그 정책을 개선하는 방식
  - 수집한 경험(샘플)을 즉시 활용해 정책을 업데이트하여 안정적인 학습과 탐색-활용 균형 확보
  - VPG (Vanilla Policy Gradient), TRPO, PPO

- [정의] 오프-폴리시 알고리즘
  - 현재 학습 중인 정책과는 다른 정책에 따라 수집한 데이터를 사용해 정책을 업데이트
  - 학습에 사용되는 정책과 데이터 수집 정책이 다를 수 있음
  - 수집한 경험을 재활용할 수 있어 학습 효율이 높으나 학습의 불안정성 증가
  - Q-Learning, DDPG, TD3, SAC

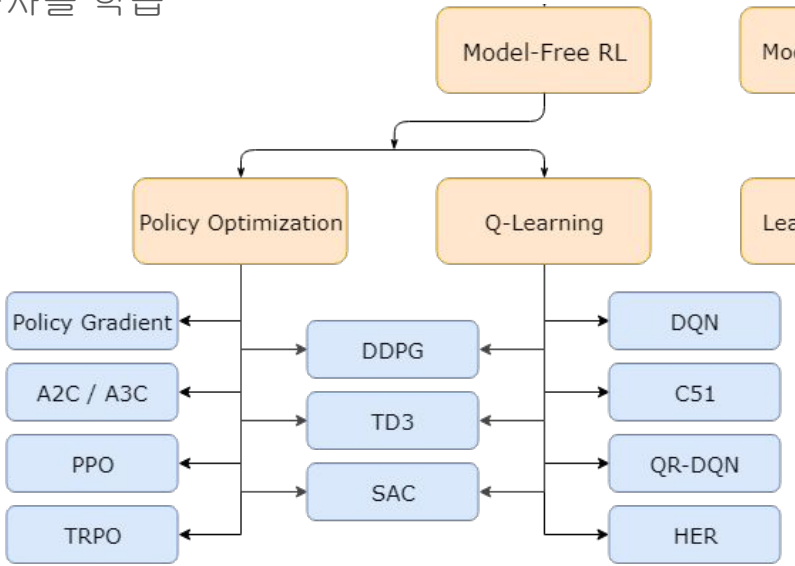
RL 알고리즘의 종류

- Model-Free vs. Model-Based RL
  - 에이전트가 환경 **모델** <상태 전환 및 보상>에 접근하거나 학습할 수 있는지에 대한 분류
    - Model-Based RL
      - 에이전트는 미리 계획한 결과를 학습된 정책으로 정제할 수 있음
      - 모델을 추정하는 과정에서 모델의 “편향”이 발생하여 실제로 동작하지 않음
    - Model-Free RL
      - 대부분의 경우는 환경의 실제 모델을 에이전트가 일반적으로 사용할 수 없음
      - “경험”에 의한 학습을 수행 ⇒ 대부분의 RL 알고리즘



RL 알고리즘의 종류

- Model-Free RL 의 학습 방법
  - 정책 최적화 (Policy Optimization)
    - 정책을 명시적으로 나타낸 매개변수 최적화
    - 대부분 온-폴리시에서 수행
    - 정책 최적화는 온-폴리시 가치 함수에 대한 근사를 학습
    - 안정적 / 신뢰성
  - Q-학습
    - 최적의 액션-가치 함수에 대한 근사를 학습
    - 대부분 오프-폴리시로 수행
    - 불안정 / 샘플 효율성이 높음
  - 두 모드의 보간 방법론
    - DDPG / SAC



성능을 직접 극대화

간접적으로 성능을 극대화 하지만, 업데이트의 결과로 얼마나 많은 변화가 일어나는지에 대한 보수적인 추정치인 대리 목적 함수를 최대화

정책 최적화 알고리즘

- 가장 간단한 정책 그래디언트
  - 확률적 매개변수 정책 π_θ 과 기대 수익률을 최대화$J(\pi_\theta) = \underset{\tau \sim \pi_\theta}{\mathrm{E}}[R(\tau)]$ 하는 것을 목표로 함
  - 가장 간단한 방법으로는 ‘gradient ascent’ 방법 (VPG, TRPO, PPO)

$$\theta_{k+1} = \theta_k + \alpha \, \nabla_\theta J(\pi_\theta)|_{\theta_k} .$$

    - 1) 기대값의 형태를 갖는 정책 성능의 그래디언트를 계산
    - 2) 유한한 수의 기대값의 샘플 추정치를 형성

정책 최적화 알고리즘

1. Probability of a Trajectory

정책 π_θ이 주어졌을때, 에피소드 $\tau = (s_0, a_0, ..., s_{T+1})$의 확률은 다음과 같다.

$$P(\tau|\theta) = \rho_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$$

2. The Log-Derivative Trick

로그 미분 트릭은 $\log x$의 x에 대한 미분이 $\frac{1}{x}$이라는 미적분학의 간단한 정리에 기반한다. 이를 재배열하고 chain rule을 통해 통합하면, 우리는 다음과 같은 식을 얻는다.

$$\nabla_\theta P(\tau|\theta) = P(\tau|\theta)\nabla_\theta \log P(\tau|\theta)$$

3. Log-Prob of a Trajectory

에피소드의 로그 확률은 다음과 같다.

$$\log P(\tau|\theta) = \log \rho_0(s_0) + \sum_{t=0}^{T}(\log P(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t))$$

4. Gradient of Environment Functions

환경은 θ를 매개변수로 가지지 않기 때문에, $\rho_0(s_0), P(s_{t+1}|s_t, a_t), R(\tau)$의 미분값은 0이다.

5. 4를 이용해서 Grad-Log-Prob of a Trajectory 구하기

$$\nabla_\theta P(\tau|\theta) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

1~5를 합치면 다음과 같은 식을 도출할 수 있다.

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \underset{\tau\sim\pi_\theta}{\mathrm{E}}[R(\tau)] \\
&= \nabla_\theta \int_\tau P(\tau|\theta)R(\tau) && \text{Expand expectation} \\
&= \int_\tau \nabla_\theta P(\tau|\theta)R(\tau) && \text{Bring gradient under integral} \\
&= \int_\tau P(\tau|\theta)\nabla_\theta \log P(\tau|\theta)R(\tau) && \text{Log-derivative trick} \\
&= \underset{\tau\sim\pi_\theta}{\mathrm{E}}[\nabla_\theta \log P(\tau|\theta)R(\tau)] && \text{Return to expectation form} \\
\therefore \nabla_\theta J(\pi_\theta) &= \underset{\tau\sim\pi_\theta}{\mathrm{E}}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau)\right] && \text{Expression for grad-log-prob}
\end{aligned}$$

$$\hat{g} = \frac{1}{|\mathcal{D}|} \sum_{\tau\in\mathcal{D}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau),$$

VPG (Vanilla Policy Gradient Algorithm)

정책 기울기의 기본이 되는 핵심 아이디어는 더 높은 수익으로 이어지는 행동의 확률을 높이고, 더 낮은 수익으로 이어지는 행동의 확률을 낮추어 최적의 정책에 도달하는 것입니다.

Algorithm 1 Vanilla Policy Gradient Algorithm

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
6: Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

7: Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

or via another gradient ascent algorithm like Adam.
8: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.
9: **end for**

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t) \right],$$

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_{\theta_k})$$

TRPO (Trust Region Policy Optimization)

TRPO는 성능을 개선하기 위해 가능한 가장 큰 단계를 취하여 정책을 업데이트하는 동시에 새 정책과 이전 정책이 얼마나 가까이 있을 수 있는지에 대한 특별한 제약 조건을 충족

이 제약 조건은 확률 분포 간의 거리를 측정하는 KL-Divergence (비슷하지만 정확히는 아님)로 표현

⇒ policy improvement 단계에서 어떻게 하면 정책을 monotonically improve할 수 있을까?

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}(\theta_k, \theta)$$
$$\text{s.t. } \bar{D}_{KL}(\theta||\theta_k) \leq \delta$$

$$\mathcal{L}(\theta_k, \theta) = \mathop{\mathrm{E}}_{s,a\sim\pi_{\theta_k}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a)\right],$$

$$\bar{D}_{KL}(\theta||\theta_k) = \mathop{\mathrm{E}}_{s\sim\pi_{\theta_k}}\left[D_{KL}\left(\pi_\theta(\cdot|s)||\pi_{\theta_k}(\cdot|s)\right)\right].$$

TRPO (Trust Region Policy Optimization)

Algorithm 1 Trust Region Policy Optimization

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

Hyperparameters: KL-divergence limit δ, backtracking coefficient α, maximum number of backtracking steps K

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

Compute rewards-to-go $\hat{R}_t$.

Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

Use the conjugate gradient algorithm to compute

$$\hat{x}_k \approx \hat{H}_k^{-1} \hat{g}_k,$$

where $\hat{H}_k$ is the Hessian of the sample average KL-divergence.

Update the policy by backtracking line search with

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{\hat{x}_k^T \hat{H}_k \hat{x}_k}} \hat{x}_k,$$

where $j \in \{0, 1, 2, \ldots K\}$ is the smallest value which improves the sample loss and satisfies the sample KL-divergence constraint.

Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

end for

$$\theta_{k+1} = \arg\max_\theta \mathcal{L}(\theta_k, \theta)$$
$$\text{s.t. } \bar{D}_{KL}(\theta||\theta_k) \leq \delta$$

$$\mathcal{L}(\theta_k, \theta) = \mathop{\mathrm{E}}_{s,a \sim \pi_{\theta_k}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a) \right],$$

$$\bar{D}_{KL}(\theta||\theta_k) = \mathop{\mathrm{E}}_{s \sim \pi_{\theta_k}} \left[D_{KL}\left(\pi_\theta(\cdot|s) || \pi_{\theta_k}(\cdot|s) \right) \right].$$