Soft Actor-Critic: Off-policy Maximum Entropy Deep Reinforcement Learing with Stochasitic Actor

주요 관련 개념

tabular-based v.s continous based

- 전자는 상태/액션 공간이 finite하고 크지 않을 때, exact solution
- 후자는 연속 공간이거나 이산적이라도 무수히 많아서 intractable 할 때, approximate solution

On-policy v.s Off-policy

- On : 하나의 정책으로 학습을 위한 에피소드도 발생시키면서 동시에 학습을 시키기도 함
- Off : 학습 대상 정책과 에피소드 발생 정책이 다르다. 데이터 획득이 용이하나 variance가 심함

Synchronous v.s Asynchronous

- 후자는 복수 개의 학습 에이전트가 공유하는 학습 네트워크를 비동기적으로 업데이트

Sample complexity

- 어떤 학습 성능에 도달하기 위해 필요한 학습데이터량. 많으면 complexity가 높다고 말한다.

Review

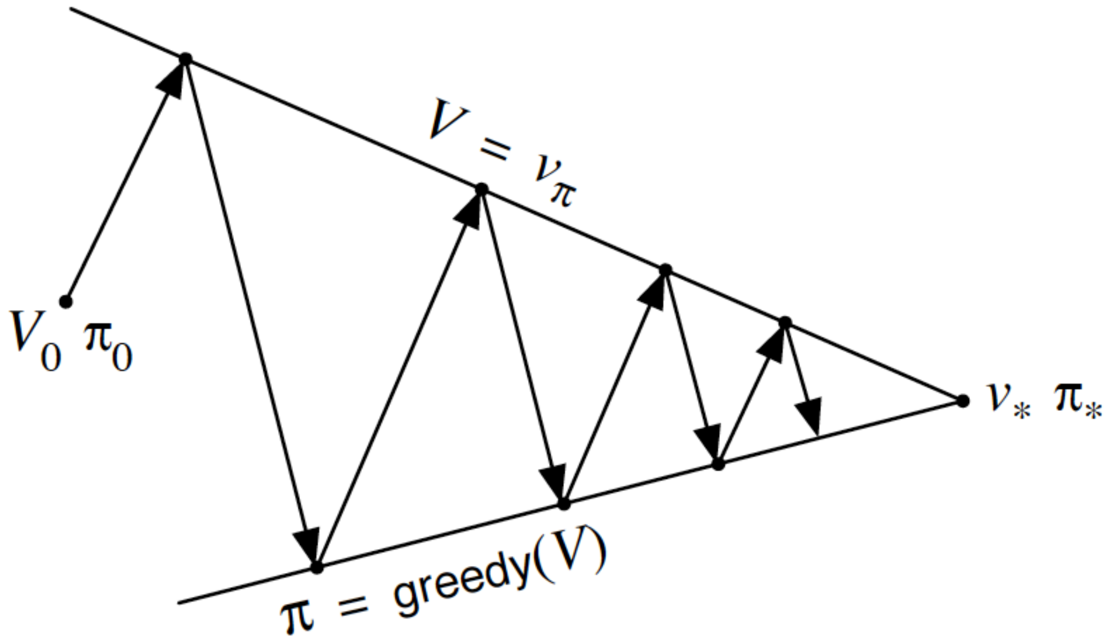
Policy Evalutation and Improvement

- 참고 : 서튼 책
- 정책 평가와 개선을 번갈아. 개선 보장.
- Known enviornment
- tabular only, 모든 상태/액션 공간을 sweep해야 하므로
- converged

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\quad \Delta \leftarrow 0$
   $\quad$ For each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

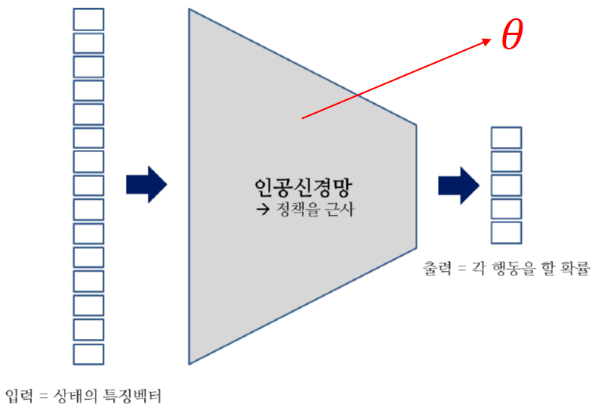
3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad a \leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
   $\quad$ If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return V and π; else go to 2



REINFORCE - MonteCarlo Policy Gradient

- 참고 : RLCode, A3C 쉽고 깊게 이해하기
  - https://www.slideshare.net/WoongwonLee/rlcode-a3c (https://www.slideshare.net/WoongwonLee/rlcode-a3c)
- continous space => Neural Net as policy approximator
- On-policy
- Variance가 높다, on-line 안됨

- 인공신경망으로 정책을 근사 → 정책 신경망



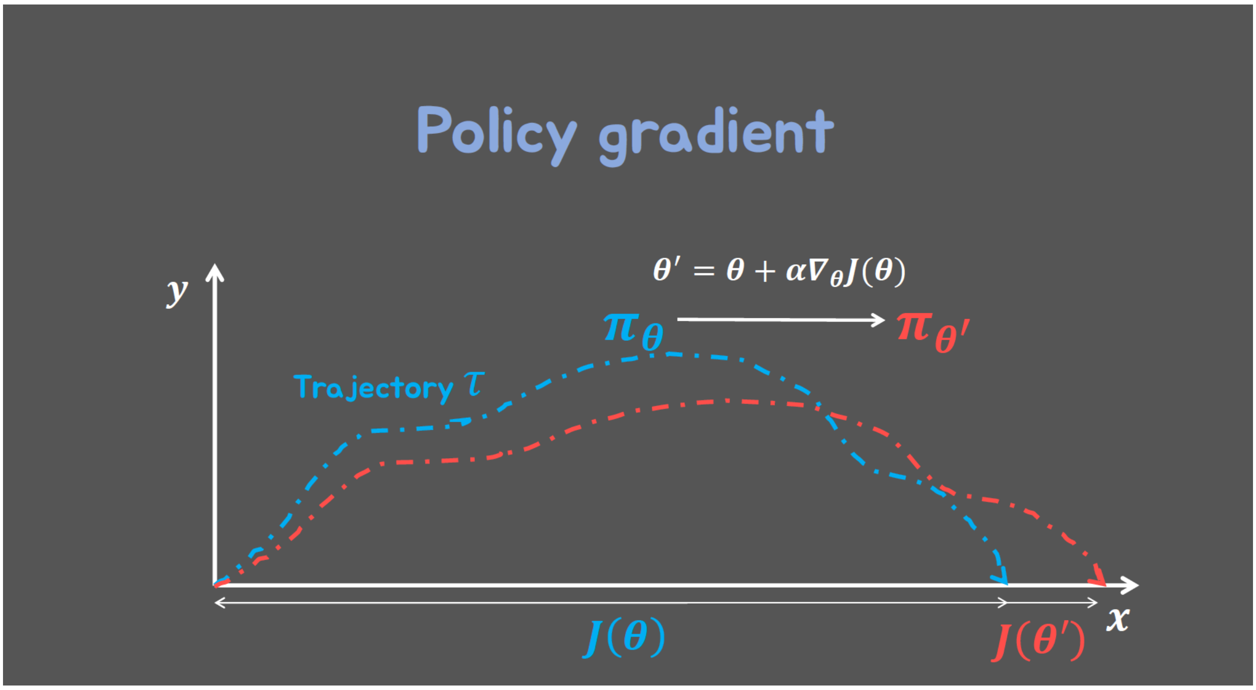
$$\text{정책} \quad \pi_\theta(a|s) = \boldsymbol{P}[A_t = a | S_t = s, \theta]$$

- $J(\theta)$ 를 기준으로 어떻게 θ(정책신경망)을 업데이트할 것인가?

→ θ에 대한 $J(\theta)$의 경사를 따라 올라가다(Gradient ascent)

$$\theta' = \theta + \alpha \nabla_\theta J(\theta)$$

$$\nabla_\theta J(\theta) = \text{Policy gradient}$$



1. 한 에피소드를 현재 정책에 따라 실행

2. Trajectory를 기록

3. 에피소드가 끝난 뒤 G_t 계산

4. Policy gradient를 계산해서 정책 업데이트 Policy gradient $= \sum_{t=0}^{T-1} \nabla_\theta log\pi_\theta(a_t|s_t)\, G_t$

5. (1~4) 반복

에피소드 마다 업데이트 → 몬테카를로 Policy gradient = REINFORCE

Actor Critic

- On-policy, TD-based
- baseline => reduce variance

1. Actor

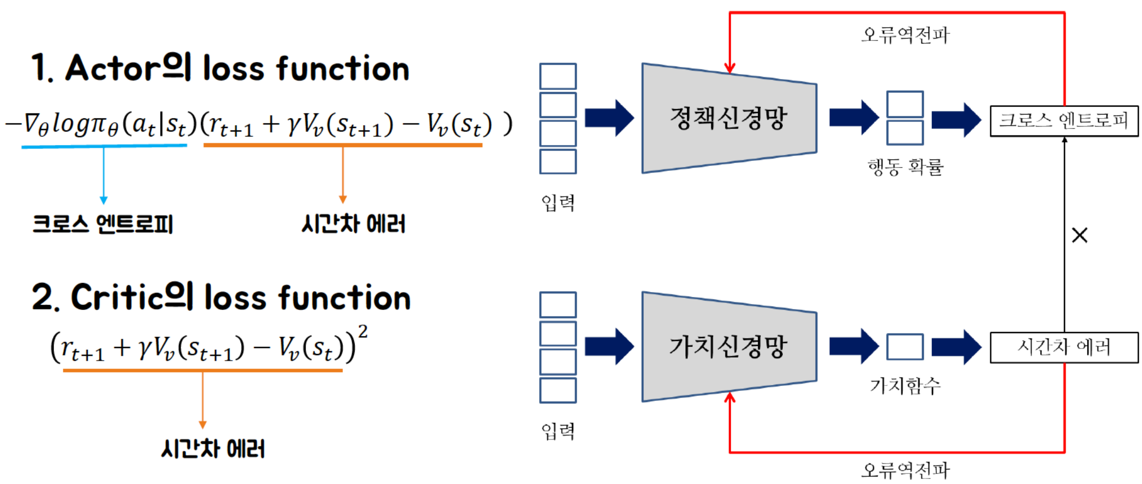
1) 정책을 근사: θ

2) $\nabla_\theta log\pi_\theta(a_t|s_t)(r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)\ )$**로 업데이트**

2. Critic

1) 가치함수(Value function)을 근사: v

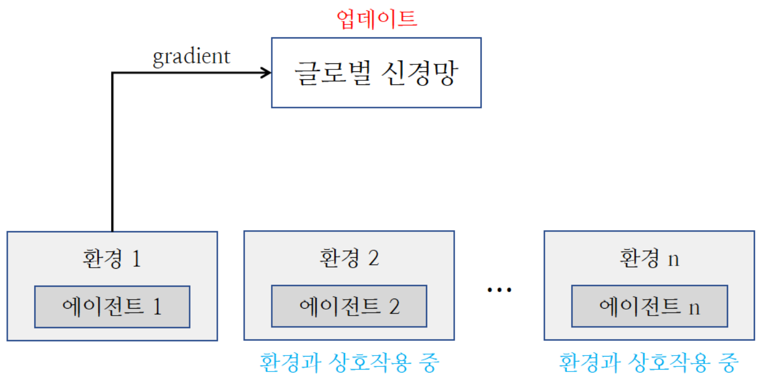
2) $\left(r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)\right)^2$**의 오차함수로 업데이트**



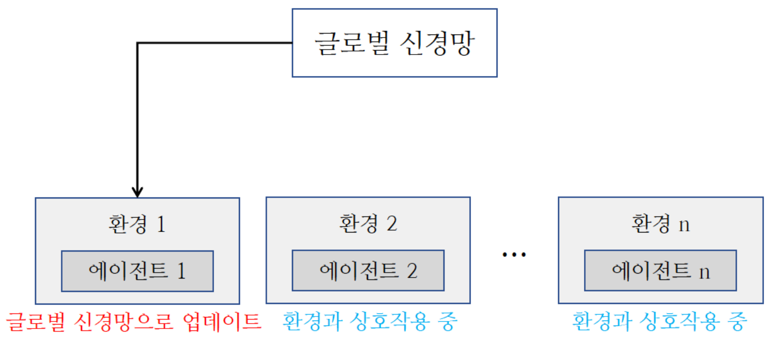
A3C

- 복수 에이전트가 각자의 환경 인스턴스에서 actor-critic 수행
- 글로벌 네트워크에 업데이트 및 동기화
- 기타
  - 20-step loss function
  - maximum entropy based learning objective

비동기적으로 global network를 업데이트: 에이전트 1이 글로벌 신경망을 업데이트



비동기적으로 global network를 업데이트 : 글로벌 신경망으로 에이전트 1을 업데이트



Maximum entropy 기반 RL

- 참고
  - http://bair.berkeley.edu/blog/2017/10/06/soft-q-learning/ (http://bair.berkeley.edu/blog/2017/10/06/soft-q-learning/)
- Optimal policy만 학습하는 목표가 과연 최선인가?

최적화된 학습만 하면 아래처럼 환경이 조금만 바뀌면 기존 최적 정책은 꽝

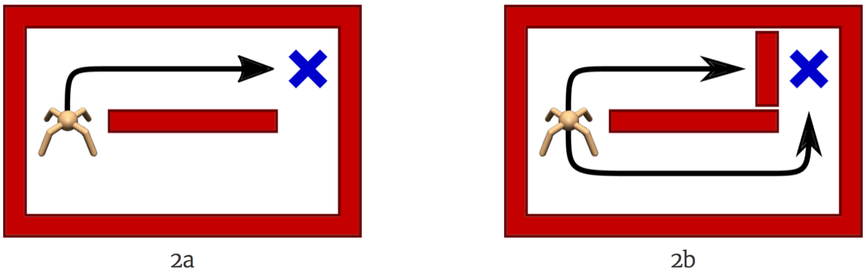


Figure 2: A robot navigating a maze.

아래 왼쪽의 gray처럼 실제는 q-value dist가 multi-modal이나

- 기존 학습 체계는 unimodal max 와 제한된 주변만 고려(weak exploration)
- 반면 오른쪽처럼 더 exploration을 다양하게 해서 multi-modal 을 다 고려하게 학습하는 것이 낫지 않을까?

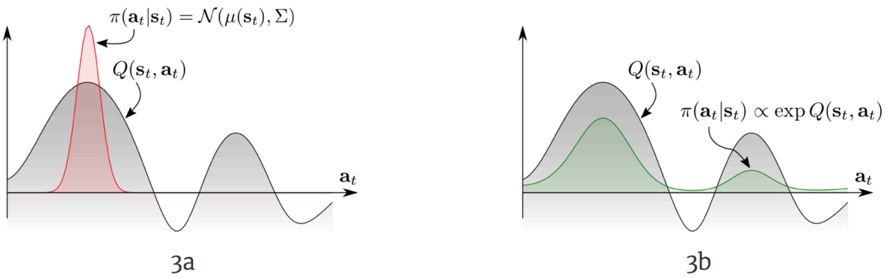


Figure 3: A multimodal Q-function.

이를 위해서 정책 함수를 Boltsman dist형태로

- Q as negative energy
- all policy value are non-zero => all situation considered

$$\pi(\mathbf{a}|\mathbf{s}) \propto \exp Q(\mathbf{s}, \mathbf{a})$$

이는 아래와 같은 Maximum entropy objective를 푸는 것과 동일

$$\pi^*_{\text{MaxEnt}} = \underset{\pi}{\text{argmax}}\, \mathbb{E}_\pi \left[\sum_{t=0}^{T} r_t + \mathcal{H}(\pi(\cdot|\mathbf{s}_t)) \right]$$

Soft Bellman Equation and Soft Q-Learning

$$Q(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}\left[r_t + \gamma\, \text{softmax}_{\mathbf{a}} Q(\mathbf{s}_{t+1}, \mathbf{a})\right]$$

where

$$\text{softmax}_{\mathbf{a}} f(\mathbf{a}) := \log \int \exp f(\mathbf{a})\, d\mathbf{a}$$

참고 : Log-Sum-Exp as approximate max

- https://en.wikipedia.org/wiki/LogSumExp (https://en.wikipedia.org/wiki/LogSumExp)

$$\max\{x_1, \dots, x_n\} \leq LSE(x_1, \dots, x_n) \leq \max\{x_1, \dots, x_n\} + \log(n)$$

참고 : original Q-learning

- Q-learning:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma \max_{a'} Q(s_{t+1}, a')) - Q(s_t, a_t)) \quad (7)$$

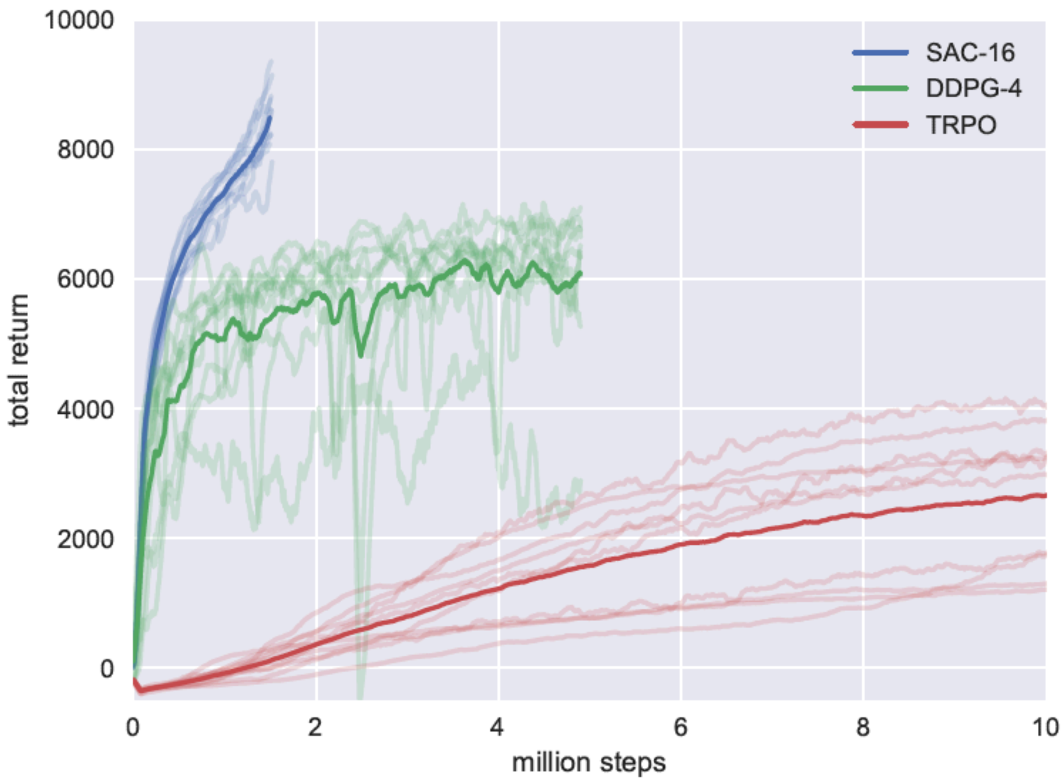
Maximum entropy 기반 체계의 장점과 응용 (see videos)

- better exploration,
- policy transfer between similar tasks,
- new policies to be easily composed from existing policies,
- improves robustness through extensive exploration at training time.

Soft actor-critic

기존 NN기반 policy graident 방법들(ex. A3C, TRPO, DDPG) 문제

- On-policy 기반이라서 그런지 Sample complexity가 높다
    - 대안 : Off-policy
- Variance 가 크다
    - converge가 어렵다. hyper-parameter tuning이 어렵다.
    - 대안 : Maximum entropy based



정리하자면

- actor-critic
- off-policy
- maximum entropy based

on/offpolicy actor-critic

- on
  - Actor가 현재 정책으로 한 건의 state, action pair 발생
  - Critic이 이걸로 바로 비평 : Q - baseline
  - 이 비평을 바탕으로 정책 업데이트
- off
  - Actor가 현재 정책으로 많은 state, action sequence 발생
  - 이를 리플레이 버퍼에 혼합
  - Critic은 리플레이 버퍼에서 퍼온 sample을 바탕으로 비평
  - 이 비평을 바탕으로 정책 업데이트

Soft Q-value, soft value

of Q-functions and value functions, and as we note in Section 4, the soft Q-function of a stochastic policy π satisfies the soft Bellman equation

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathbf{s}}} \left[V^{\pi}(\mathbf{s}_{t+1}) \right], \tag{2}$$

where the soft value $V^{\pi}(\mathbf{s}_t)$ is given by

$$V^{\pi}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi} \left[Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) - \log \pi(\mathbf{a}_t | \mathbf{s}_t) \right]. \tag{3}$$

Soft Bellman Operator 및 Soft Policy Evaluation

$$\mathcal{T}^{\pi} Q = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathbf{s}}, \mathbf{a}_{t+1} \sim \pi} \left[Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \log \pi(\mathbf{a}_{t+1} | \mathbf{s}_{t+1}) \right]. \tag{5}$$

Lemma 1 (Soft Policy Evaluation)**.** *Consider the soft Bellman backup operator $\mathcal{T}^{\pi}$ in Equation 5 and a mapping $Q^k : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and define $Q^{k+1} = \mathcal{T}^{\pi} Q^k$. Then the sequence Q^k will converge to the soft value of π as $k \to \infty$.*

Improved policy projected on parameterized family of policys via KL-divergence

$$\pi_{\text{new}} = \arg \min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}} \left(\pi'(\,\cdot\,|\mathbf{s}_t) \;\|\; \exp\left(Q^{\pi_{\text{old}}}(\mathbf{s}_t, \,\cdot\,) - \log Z^{\pi_{\text{old}}}(\mathbf{s}_t) \right) \right). \tag{6}$$

Soft policy iteration

- tabular에나 적당

Algorithm 1: Soft Policy Iteration

```
while π_new(a_t|s_t) ≠ π_old(a_t|s_t) for some (a_t, s_t) ∈ A × S do
    Q^0 ← Q^{π_old}
    while Q^{k+1}(s_t, a_t) > Q^k(s_t, a_t) for some (a_t, s_t) ∈ A × S do
        Q^{k+1} ← T^{π_new} Q^k
        k ← k + 1
    end
    π_old ← π_new
    π_new ← arg min_{π'∈Π} D_KL(π' || exp(Q^{π_old} − log Z^{π_old}))
end
```

Approximated version

- 정책, Q, 가치, 3개의 Neural Net function approximator
- soft version of graident-based
- off-policy

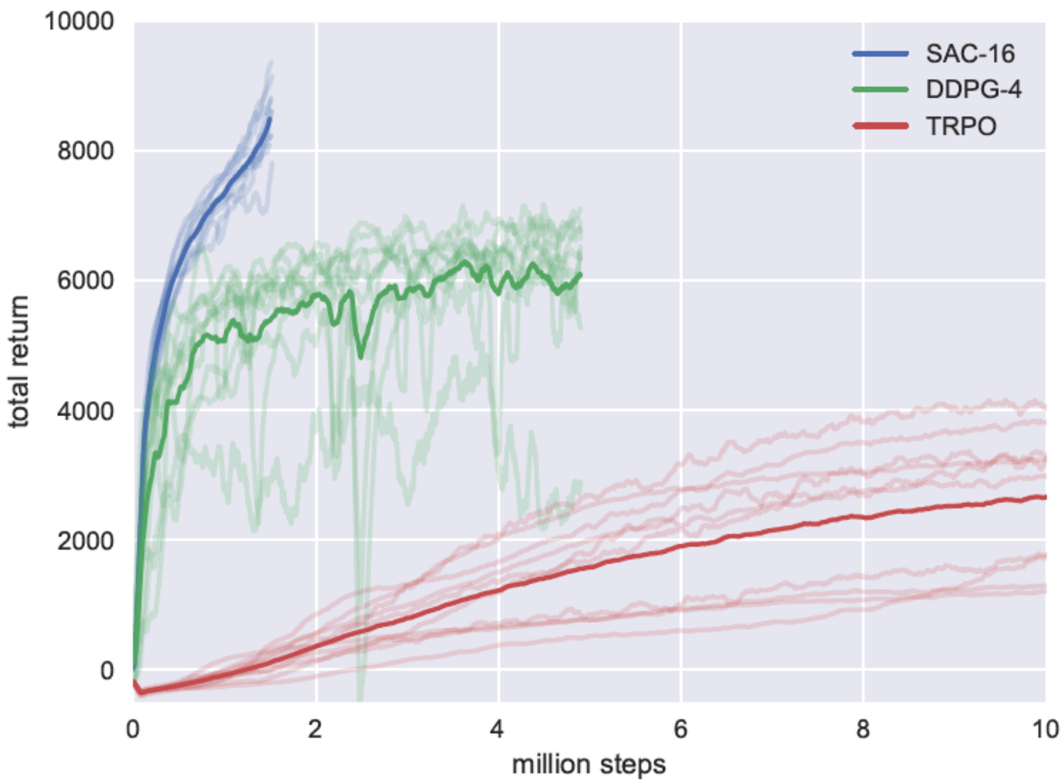
$$J_V(\psi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\frac{1}{2} \left(V_{\psi}(\mathbf{s}_t) - \mathbb{E}_{\mathbf{a}_t \sim \pi_{\phi}} \left[Q_{\theta}(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_{\phi}(\mathbf{a}_t | \mathbf{s}_t) \right] \right)^2 \right], \tag{7}$$

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_{\theta}(\mathbf{s}_t, \mathbf{a}_t) - \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathbf{s}}} \left[V_{\psi}(\mathbf{s}_t) \right] \right) \right)^2 \right], \tag{9}$$

$$J_\pi(\phi) = \mathrm{D}_{\mathrm{KL}}\left(\pi_\phi(\,\cdot\,|\mathbf{s}_t) \,\|\, \exp\left(Q_\theta(\mathbf{s}_t,\,\cdot\,) - \log Z_\theta(\mathbf{s}_t)\right)\right). \tag{11}$$

Algorithm 2: Soft Actor-Critic

```
Initialize parameter vectors ψ, θ, φ.
for each iteration do
    for each environment step do
        a_t ~ π_φ(a_t|s_t)
        s_{t+1} ~ p_s(s_{t+1}|s_t, a_t)
        D ← D ∪ {(s_t, a_t, r(s_t, a_t), s_{t+1})} .
    end
    for each gradient step do
        ψ ← ψ − λ_V ∇̃_ψ J_V(ψ)
        θ ← θ − λ_Q ∇̃_θ J_Q(θ)
        φ ← φ − λ_π ∇̃_φ J_π(φ)
    end
end
```



In [ ]: