ECEN 743: Reinforcement Learning

Advanced Actor-Critic Algorithms

Dileep Kalathil
Assistant Professor
Department of Electrical and Computer Engineering
Texas A&M University

Soft Actor Critic (SAC) Algorithm

References

- (SAC Paper) Haarnoja, et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", *International Conference on Machine Learning (ICML)*, 2018.
- (Updated SAC and explanation) Haarnoja, et al. "Soft actor-critic algorithms and applications", 2018.
- (Soft Q-learning Paper) Haarnoja, et al. "Reinforcement learning with deep energy-based policies", *International Conference on Machine Learning (ICML)*, 2017.
- (Convergence analysis of entropy regularized NPG) Cen et al. "Fast global convergence of natural policy gradient methods with entropy regularization", *Operations Research*, 2022.

Entropy Regularized RL

- Standard RL objective: $\max_{\pi} \quad \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ where $a_t \sim \pi(s_t, \cdot)$

Entropy Regularized RL

- Standard RL objective: $\max_\pi \quad \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ where $a_t \sim \pi(s_t, \cdot)$
- Entropy regularized RL objective

$$\max_\pi \quad \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) - \alpha \log \pi(s_t, a_t))], \quad \text{where, } a_t \sim \pi(s_t, \cdot)$$

Entropy Regularized RL

- Standard RL objective: $\max_\pi \quad \mathbb{E}[\sum_{t=0}^\infty \gamma^t r(s_t, a_t)]$ where $a_t \sim \pi(s_t, \cdot)$
- Entropy regularized RL objective

$$\max_\pi \quad \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) - \alpha \log \pi(s_t, a_t))], \quad \text{where, } a_t \sim \pi(s_t, \cdot)$$

- Entropy of a distribution p: $H(p) = -\sum_x p(x) \log p(x)$

Entropy Regularized RL

- Standard RL objective: $\max_\pi \quad \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ where $a_t \sim \pi(s_t, \cdot)$
- Entropy regularized RL objective

$$\max_\pi \quad \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) - \alpha \log \pi(s_t, a_t)], \quad \text{where, } a_t \sim \pi(s_t, \cdot)$$

- Entropy of a distribution p: $H(p) = -\sum_x p(x) \log p(x)$
- So, this is also called maximum entropy RL:

$$\max_\pi \quad \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) + \alpha H(\pi(s_t, \cdot))], \quad \text{where, } a_t \sim \pi(s_t, \cdot)$$

Why max-entropy?

- Maximum entropy formulation provides a significant improvement in exploration and robustness
- It gives nicer objective functions from an optimization perspective

Soft Values

- Standard value function $V_{\pi,\mu} = \mathbb{E}_{s\sim\mu}[V_\pi(s)]$

Soft Values

- Standard value function $V_{\pi,\mu} = \mathbb{E}_{s\sim\mu}[V_\pi(s)]$
- Soft value function

$$V^{\mathsf{s}}_{\pi,\mu} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) - \alpha \log(\pi(s_t, \cdot))], \quad \text{where, } a_t \sim \pi(s_t, \cdot), s_0 \sim \mu$$

Soft Values

- Standard value function $V_{\pi,\mu} = \mathbb{E}_{s\sim\mu}[V_\pi(s)]$
- Soft value function

$$V^{\mathsf{s}}_{\pi,\mu} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) - \alpha \log(\pi(s_t, \cdot))], \quad \text{where, } a_t \sim \pi(s_t, \cdot), s_0 \sim \mu$$

- Denoting $H(\pi, \mu) = \mathbb{E}_{s\sim\rho_{\pi,\mu}}[H(\pi(s, \cdot))]$, we get

$$V^{\mathsf{s}}_{\pi,\mu} = V_{\pi,\mu} + \alpha H(\pi, \mu)$$

Soft Values

- Standard value function $V_{\pi,\mu} = \mathbb{E}_{s\sim\mu}[V_\pi(s)]$
- Soft value function

$$V^{\mathsf{s}}_{\pi,\mu} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) - \alpha \log(\pi(s_t, \cdot))], \quad \text{where, } a_t \sim \pi(s_t, \cdot), s_0 \sim \mu$$

- Denoting $H(\pi, \mu) = \mathbb{E}_{s\sim\rho_{\pi,\mu}}[H(\pi(s, \cdot))]$, we get

$$V^{\mathsf{s}}_{\pi,\mu} = V_{\pi,\mu} + \alpha H(\pi, \mu)$$

- Denoting π^* and $\pi^{*\mathsf{s}}$ as the optimal policies of standard and regularized objectives, we get

$$V_{\pi^*,\mu} \leq V^{\mathsf{s}}_{\pi^{*\mathsf{s}},\mu} \leq V_{\pi^*,\mu} + \frac{\alpha \log|\mathcal{A}|}{(1-\gamma)}$$

Soft Values

- Standard value function $V_{\pi,\mu} = \mathbb{E}_{s\sim\mu}[V_\pi(s)]$
- Soft value function

$$V^{\mathsf{s}}_{\pi,\mu} = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t (r(s_t,a_t) - \alpha \log(\pi(s_t,\cdot))], \quad \text{where, } a_t \sim \pi(s_t,\cdot), s_0 \sim \mu$$

- Denoting $H(\pi,\mu) = \mathbb{E}_{s\sim\rho_{\pi,\mu}}[H(\pi(s,\cdot))]$, we get

$$V^{\mathsf{s}}_{\pi,\mu} = V_{\pi,\mu} + \alpha H(\pi,\mu)$$

- Denoting π^* and $\pi^{*\mathsf{s}}$ as the optimal policies of standard and regularized objectives, we get

$$V_{\pi^*,\mu} \leq V^{\mathsf{s}}_{\pi^{*\mathsf{s}},\mu} \leq V_{\pi^*,\mu} + \frac{\alpha \log|\mathcal{A}|}{(1-\gamma)}$$

- So, $\pi^{*\mathsf{s}}$ could also be nearly optimal in terms of the standard value function, as long as the regularization parameter α is chosen to be sufficiently small

Soft Policy Evaluation

- Define the soft policy evaluation operator T_π^{s} as

$$T_\pi^{\mathsf{s}} Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)}[V(s')], \text{ where, } V(s) = \mathbb{E}_{a \sim \pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)]$$

Soft Policy Evaluation

- Define the soft policy evaluation operator T_π^{s} as

$$T_\pi^{\mathsf{s}} Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)}[V(s')], \text{ where, } V(s) = \mathbb{E}_{a \sim \pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)]$$

Lemma

1. *T_π^{s} is a contraction mapping*
2. *The iteration $Q_{k+1} = T_\pi^{\mathsf{s}} Q_k$ will converge to the soft Q-value function Q_π^{s}, which satisfy the consistency equation*

$$Q_\pi^{\mathsf{s}}(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)}[V_\pi^{\mathsf{s}}(s')], \text{ where, } V_\pi^{\mathsf{s}}(s) = \mathbb{E}_{a \sim \pi(s,\cdot)}[Q_\pi^{\mathsf{s}}(s,a) - \alpha \log \pi(s,a)]$$

Form of the Optimal Soft Policy

- Optimal soft policy has the softmax form: $\pi^{*\mathsf{s}}(s,a) \propto \exp(Q^{*\mathsf{s}}(s,a)/\alpha)$

Form of the Optimal Soft Policy

- Optimal soft policy has the softmax form: $\pi^{*\mathsf{s}}(s,a) \propto \exp(Q^{*\mathsf{s}}(s,a)/\alpha)$
- More precisely,

$$\pi^{*\mathsf{s}}(s,a) = \exp\left(\frac{1}{\alpha}(Q^{*\mathsf{s}}(s,a) - V^{*\mathsf{s}}(s))\right), \text{ and,}$$

$$V^{*\mathsf{s}}(s) = \alpha \log\left(\sum_a \exp\left(Q^{*\mathsf{s}}(s,a)/\alpha\right)\right)$$

Softmax Policy: Intuition

- We want to solve $\max_\pi \ \mathbb{E}_{a\sim\pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)]$

Softmax Policy: Intuition

- We want to solve $\max_\pi \; \mathbb{E}_{a\sim\pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)]$

$$\mathbb{E}_{a\sim\pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)] = \alpha \sum_a \pi(s,a) \log\left(\frac{\exp(Q(s,a)/\alpha)}{\pi(s,a)}\right)$$

Softmax Policy: Intuition

- We want to solve $\max_\pi \; \mathbb{E}_{a\sim\pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)]$

$$\begin{aligned}\mathbb{E}_{a\sim\pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)] &= \alpha \sum_a \pi(s,a) \log\left(\frac{\exp(Q(s,a)/\alpha)}{\pi(s,a)}\right) \\ &\leq \alpha \log\left(\sum_a \pi(s,a) \frac{\exp(Q(s,a)/\alpha)}{\pi(s,a)}\right)\end{aligned}$$

Softmax Policy: Intuition

- We want to solve $\max_{\pi} \; \mathbb{E}_{a\sim\pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)]$

$$\begin{aligned}
\mathbb{E}_{a\sim\pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)] &= \alpha \sum_a \pi(s,a) \log\left(\frac{\exp(Q(s,a)/\alpha)}{\pi(s,a)}\right) \\
&\leq \alpha \log\left(\sum_a \pi(s,a) \frac{\exp(Q(s,a)/\alpha)}{\pi(s,a)}\right) \\
&= \alpha \log\left(\sum_a \exp(Q(s,a)/\alpha)\right)
\end{aligned}$$

Softmax Policy: Intuition

- We want to solve $\max_\pi \ \mathbb{E}_{a\sim\pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)]$

$$\begin{aligned}
\mathbb{E}_{a\sim\pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)] &= \alpha \sum_a \pi(s,a) \log\left(\frac{\exp(Q(s,a)/\alpha)}{\pi(s,a)}\right) \\
&\leq \alpha \log\left(\sum_a \pi(s,a) \frac{\exp(Q(s,a)/\alpha)}{\pi(s,a)}\right) \\
&= \alpha \log\left(\sum_a \exp(Q(s,a)/\alpha)\right)
\end{aligned}$$

Now, notice that the inequality in the above steps becomes equality if $\pi \propto \exp(Q(s,a)/\alpha)$.

Softmax Policy: Intuition

- We want to solve $\max_\pi \ \mathbb{E}_{a\sim\pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)]$

$$\begin{aligned}
\mathbb{E}_{a\sim\pi(s,\cdot)}[Q(s,a) - \alpha \log \pi(s,a)] &= \alpha \sum_a \pi(s,a) \log\left(\frac{\exp(Q(s,a)/\alpha)}{\pi(s,a)}\right) \\
&\leq \alpha \log\left(\sum_a \pi(s,a)\frac{\exp(Q(s,a)/\alpha)}{\pi(s,a)}\right) \\
&= \alpha \log\left(\sum_a \exp(Q(s,a)/\alpha)\right)
\end{aligned}$$

Now, notice that the inequality in the above steps becomes equality if $\pi \propto \exp(Q(s,a)/\alpha)$.
Homework: Try to solve the optimization directly and get the softmax solution

Soft Bellman Operator

- Consider the natural extension of the standard Bellman operator

$$T^{\mathsf{s}}Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)}[\max_{\pi(s',\cdot)} \mathbb{E}_{a' \sim \pi(s',\cdot)}[Q(s',a') - \alpha \log \pi(s',a')]]$$

Soft Bellman Operator

- Consider the natural extension of the standard Bellman operator

$$T^{\mathsf{s}}Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)}[\max_{\pi(s',\cdot)} \mathbb{E}_{a' \sim \pi(s',\cdot)}[Q(s',a') - \alpha \log \pi(s',a')]]$$

- From the previous slide, this is equivalent to

$$T^{\mathsf{s}}Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)}[\alpha \log(\sum_{a'} \mathsf{exp}(Q(s',a')/\alpha))]$$

Soft Bellman Operator

- Consider the natural extension of the standard Bellman operator

$$T^{\mathsf{s}}Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)}[\max_{\pi(s',\cdot)} \mathbb{E}_{a' \sim \pi(s',\cdot)}[Q(s',a') - \alpha \log \pi(s',a')]]$$

- From the previous slide, this is equivalent to

$$T^{\mathsf{s}}Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(s,a)}[\alpha \log(\sum_{a'} \mathsf{exp}(Q(s',a')/\alpha))]$$

Proposition

1. T^{s} *is a contraction w.r.t.* $\|\cdot\|_\infty$ *with a contraction coefficient* γ
2. $Q^{*\mathsf{s}}$ *is the unique fixed point of* T^{s}

Soft Policy Iteration

- Similar to the standard policy iteration. Start with an initial policy π_0. For each k, perform soft policy evaluation and soft policy update

Soft Policy Iteration

- Similar to the standard policy iteration. Start with an initial policy π_0. For each k, perform soft policy evaluation and soft policy update
- Soft policy evaluation: Given the policy π_k, compute $Q^{\mathsf{s}}_{\pi_k}$

Soft Policy Iteration

- Similar to the standard policy iteration. Start with an initial policy π_0. For each k, perform soft policy evaluation and soft policy update
- Soft policy evaluation: Given the policy π_k, compute $Q^{\mathsf{s}}_{\pi_k}$
- Soft policy update: Given Q^{s}_{π}, update the policy to get $\pi_{k+1}(s,a) = \frac{1}{Z_{\pi_k}(s)} \exp(Q^{\mathsf{s}}_{\pi_k}(s,a)/\alpha)$

Soft Policy Iteration

- Similar to the standard policy iteration. Start with an initial policy π_0. For each k, perform soft policy evaluation and soft policy update
- Soft policy evaluation: Given the policy π_k, compute $Q^{\mathsf{s}}_{\pi_k}$
- Soft policy update: Given Q^{s}_{π}, update the policy to get $\pi_{k+1}(s,a) = \frac{1}{Z_{\pi_k}(s)} \exp(Q^{\mathsf{s}}_{\pi_k}(s,a)/\alpha)$

Proposition

1. *Monotone improvement:*

$$V^{\mathsf{s}}_{\pi_{k+1}}(s) - V^{\mathsf{s}}_{\pi_k}(s) = \mathbb{E}_{s' \sim \rho_{\pi_{k+1},\delta(s)}}[\frac{\alpha}{(1-\gamma)} D_{\mathrm{KL}}(\pi_{k+1}(s,\cdot), \pi_k(s,\cdot))]$$

Soft Policy Iteration

- Similar to the standard policy iteration. Start with an initial policy π_0. For each k, perform soft policy evaluation and soft policy update
- Soft policy evaluation: Given the policy π_k, compute $Q^{\mathsf{s}}_{\pi_k}$
- Soft policy update: Given Q^{s}_{π}, update the policy to get $\pi_{k+1}(s,a) = \frac{1}{Z_{\pi_k}(s)} \exp(Q^{\mathsf{s}}_{\pi_k}(s,a)/\alpha)$

Proposition

1. *Monotone improvement:*

$$V^{\mathsf{s}}_{\pi_{k+1}}(s) - V^{\mathsf{s}}_{\pi_k}(s) = \mathbb{E}_{s' \sim \rho_{\pi_{k+1},\delta(s)}}[\frac{\alpha}{(1-\gamma)} D_{\mathrm{KL}}(\pi_{k+1}(s,\cdot), \pi_k(s,\cdot))]$$

2. *Convergence rate:*

$$\left\|Q^{*\mathsf{s}} - Q^{\mathsf{s}}_{\pi_k}\right\|_\infty \leq \gamma^k \left\|Q^{*\mathsf{s}} - Q^{\mathsf{s}}_{\pi_0}\right\|_\infty$$

$$\left\|\log \pi^{*\mathsf{s}} - \log \pi_{k+1}\right\|_\infty \leq \frac{2}{\alpha}\gamma^k \left\|Q^{*\mathsf{s}} - Q^{\mathsf{s}}_{\pi_0}\right\|_\infty$$

Off-policy Learning

- Policy optimization algorithms such as PG, TRPO, PPO are on-policy learning algorithms

Off-policy Learning

- Policy optimization algorithms such as PG, TRPO, PPO are on-policy learning algorithms
- On-policy algorithms require new samples to be collected for every update of the policy; Data collected from previous iterates is discarded in on-policy algorithms

Off-policy Learning

- Policy optimization algorithms such as PG, TRPO, PPO are on-policy learning algorithms
- On-policy algorithms require new samples to be collected for every update of the policy; Data collected from previous iterates is discarded in on-policy algorithms
- On-policy data collection is expensive (in terms of the number of gradient steps and samples per step needed); this is exacerbated in high dimensional complex tasks

Off-policy Learning

- Policy optimization algorithms such as PG, TRPO, PPO are on-policy learning algorithms
- On-policy algorithms require new samples to be collected for every update of the policy; Data collected from previous iterates is discarded in on-policy algorithms
- On-policy data collection is expensive (in terms of the number of gradient steps and samples per step needed); this is exacerbated in high dimensional complex tasks
- Off-policy learning algorithms can reuse the data from all learning episodes (recall the replay buffer in DQN)

Off-policy Learning

- Policy optimization algorithms such as PG, TRPO, PPO are on-policy learning algorithms
- On-policy algorithms require new samples to be collected for every update of the policy; Data collected from previous iterates is discarded in on-policy algorithms
- On-policy data collection is expensive (in terms of the number of gradient steps and samples per step needed); this is exacerbated in high dimensional complex tasks
- Off-policy learning algorithms can reuse the data from all learning episodes (recall the replay buffer in DQN)
- SAC is an off-policy policy optimization algorithm

Soft Actor-Critic Algorithm: Q-Value update

- Function approximation for soft Q-function Q_θ and policy π_ϕ

Soft Actor-Critic Algorithm: Q-Value update

- Function approximation for soft Q-function Q_θ and policy π_ϕ
- Instead of running evaluation for each policy π_k and performing improvement using $Q^{\mathsf{s}}_{\pi_k}$, SAC alternate between optimizing θ, ϕ networks with stochastic gradient descent

Soft Actor-Critic Algorithm: Q-Value update

- Function approximation for soft Q-function Q_θ and policy π_ϕ
- Instead of running evaluation for each policy π_k and performing improvement using $Q^{\mathsf{s}}_{\pi_k}$, SAC alternate between optimizing θ, ϕ networks with stochastic gradient descent
- Soft Q-function parameter update: Recall that $V^{\mathsf{s}}_{\pi}(s) = \mathbb{E}_{a \sim \pi(s,\cdot)}[Q^{\mathsf{s}}_{\pi}(s,a) - \alpha \log \pi(s,a)]$. So, we define the loss function $L_Q(\theta)$ as

$$L_Q(\theta) = (1/2)\mathbb{E}_{(s,a,r,s') \sim \mathcal{D}}\Big[\Big(Q_\theta(s,a) - \Big(r + \gamma \mathbb{E}_{a' \sim \pi_\phi(s,\cdot)}[Q_{\bar{\theta}}(s',a') - \alpha \log \pi(s',a')]\Big)\Big)^2\Big],$$

where, the target $\bar{\theta}$ is an exponentially moving average of past soft Q-function weights.

Soft Actor-Critic Algorithm: Q-Value update

- Function approximation for soft Q-function Q_θ and policy π_ϕ
- Instead of running evaluation for each policy π_k and performing improvement using $Q^{\mathsf{s}}_{\pi_k}$, SAC alternate between optimizing θ, ϕ networks with stochastic gradient descent
- Soft Q-function parameter update: Recall that $V^{\mathsf{s}}_\pi(s) = \mathbb{E}_{a\sim\pi(s,\cdot)}[Q^{\mathsf{s}}_\pi(s,a) - \alpha \log \pi(s,a)]$. So, we define the loss function $L_Q(\theta)$ as

$$L_Q(\theta) = (1/2)\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\Big[\Big(Q_\theta(s,a) - \big(r + \gamma \mathbb{E}_{a'\sim\pi_\phi(s,\cdot)}[Q_{\bar\theta}(s',a') - \alpha \log \pi(s',a')]\big)\Big)^2\Big],$$

where, the target $\bar\theta$ is an exponentially moving average of past soft Q-function weights.
- The gradient of $L_Q(\theta)$ estimated using a single sample (s,a,r,s') and $a' \sim \pi_\phi(s',\cdot)$ as

$$\widehat{\nabla}_\theta L_Q(\theta) = \nabla_\theta Q_\theta(s,a)\left(Q_\theta(s,a) - \big(r + \gamma \mathbb{E}_{a'\sim\pi_\phi(s,\cdot)}[Q_{\bar\theta}(s',a') - \alpha \log \pi(s',a')]\big)\right)$$

Soft Actor-Critic Algorithm: Policy update

- Policy parameter update: SAC updates policy using the loss function

$$L_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\left[D_{\mathrm{KL}}(\pi_\phi(s,\cdot), \frac{1}{Z_\theta(s)}\exp(Q_\theta(s,\cdot)/\alpha)\right]$$

Soft Actor-Critic Algorithm: Policy update

- Policy parameter update: SAC updates policy using the loss function

$$L_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\left[D_{\mathrm{KL}}(\pi_\phi(s,\cdot), \frac{1}{Z_\theta(s)}\exp(Q_\theta(s,\cdot)/\alpha)\right]$$

- This can be rewritten as

$$L_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{a\sim\pi_\phi(s,\cdot)}\left[\alpha\log\pi_\phi(s,a) - Q_\theta(s,a)\right]$$

Soft Actor-Critic Algorithm: Policy update

- Policy parameter update: SAC updates policy using the loss function

$$L_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\left[D_{\mathrm{KL}}(\pi_\phi(s,\cdot), \frac{1}{Z_\theta(s)}\exp(Q_\theta(s,\cdot)/\alpha)\right]$$

- This can be rewritten as

$$L_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{a\sim\pi_\phi(s,\cdot)}\left[\alpha\log\pi_\phi(s,a) - Q_\theta(s,a)\right]$$

- Use $a = f_\phi(\epsilon; s)$, where ϵ is an input noise vector sampled from a fixed distribution such as Gaussian

Soft Actor-Critic Algorithm: Policy update

- Policy parameter update: SAC updates policy using the loss function

$$L_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\left[D_{\mathrm{KL}}(\pi_\phi(s,\cdot), \frac{1}{Z_\theta(s)}\exp(Q_\theta(s,\cdot)/\alpha)\right]$$

- This can be rewritten as

$$L_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{a\sim\pi_\phi(s,\cdot)}\left[\alpha\log\pi_\phi(s,a) - Q_\theta(s,a)\right]$$

- Use $a = f_\phi(\epsilon; s)$, where ϵ is an input noise vector sampled from a fixed distribution such as Gaussian
- Then, we can rewrite the above loss function as

$$L_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{\epsilon\sim\mathcal{N}}\left[\log\pi_\phi(s, f_\phi(\epsilon; s)) - Q_\theta(s, f_\phi(\epsilon; s))\right]$$

Soft Actor-Critic Algorithm: Policy update

- Policy parameter update: SAC updates policy using the loss function

$$L_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\left[D_{\mathrm{KL}}(\pi_\phi(s,\cdot), \frac{1}{Z_\theta(s)}\exp(Q_\theta(s,\cdot)/\alpha)\right]$$

- This can be rewritten as

$$L_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{a\sim\pi_\phi(s,\cdot)}\left[\alpha\log\pi_\phi(s,a) - Q_\theta(s,a)\right]$$

- Use $a = f_\phi(\epsilon; s)$, where ϵ is an input noise vector sampled from a fixed distribution such as Gaussian
- Then, we can rewrite the above loss function as

$$L_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{\epsilon\sim\mathcal{N}}\left[\log\pi_\phi(s, f_\phi(\epsilon; s)) - Q_\theta(s, f_\phi(\epsilon; s))\right]$$

- The gradient of $L_\pi(\phi)$ estimated using a single sample $s\sim D$ and $\epsilon\sim\mathcal{N}$ as

$$\widehat{\nabla}_\phi L_\pi(\phi) = \nabla_\phi\log\pi_\phi(s,a) + (\nabla_a\log\pi_\phi(s,a) - \nabla_a Q_\theta(s,a))\nabla_\phi f_\phi(s,a)|_{a=f_\phi(\epsilon;s)}$$

Soft Actor-Critic Algorithm: Double Q-Network

- Recall the double DQN algorithm!
- Over estimation bias in standard Q-learning: Using the same network to select the max action and estimate its value will lead to overestimating the Q-value
- SAC overcome this issue by training two Q-functions, with parameters θ_1 and θ_2. The minimum of the the soft Q-functions is then used for the stochastic gradient updates (more on this later)

Soft Actor-Critic Algorithm

Algorithm 1 Soft Actor-Critic

Input: θ_1, θ_2, ϕ ▷ Initial parameters
  $\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$ ▷ Initialize target network weights
  $\mathcal{D} \leftarrow \emptyset$ ▷ Initialize an empty replay pool
  for each iteration **do**
    for each environment step **do**
      $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$ ▷ Sample action from the policy
      $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ ▷ Sample transition from the environment
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$ ▷ Store the transition in the replay pool
    end for
    for each gradient step **do**
      $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$ ▷ Update the Q-function parameters
      $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Update policy weights
      $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ ▷ Adjust temperature
      $\bar{\theta}_i \leftarrow \tau\theta_i + (1 - \tau)\bar{\theta}_i$ for $i \in \{1, 2\}$ ▷ Update target network weights
    end for
  end for
Output: θ_1, θ_2, ϕ ▷ Optimized parameters

SAC Performance