Soft Actor-Critic (SAC)

Insoon Yang

Department of Electrical and Computer Engineering
Seoul National University

Control + Optimization Research Lab

Review: Max-entropy stochastic control

$$\max_{\pi} \ \mathbb{E}^{\pi}\left[\sum_t [r(s_t, u_t) + \alpha \underbrace{H(\pi_t(\cdot|s_t))}_{\text{entropy of } \pi}] \right]$$

Review: Max-entropy stochastic control

$$\max_{\pi} \ \mathbb{E}^{\pi}\left[\sum_t [r(s_t, u_t) + \alpha \underbrace{H(\pi_t(\cdot|s_t))}_{\text{entropy of } \pi}]\right]$$

- Bellman equation:

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'}[V^*(s')]$$
$$V^*(s) = \alpha \log \int \exp\left(\frac{1}{\alpha} Q^*(s,a)\right) da$$

Review: Max-entropy stochastic control

$$\max_{\pi} \ \mathbb{E}^{\pi}\left[\sum_t [r(s_t,u_t) + \alpha \underbrace{H(\pi_t(\cdot|s_t))}_{\text{entropy of } \pi}]\right]$$

- Bellman equation:

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'}[V^*(s')]$$
$$V^*(s) = \alpha \log \int \exp\left(\frac{1}{\alpha} Q^*(s,a)\right) da$$

- Optimal policy:

$$\pi^*(a|s) = \exp\left(\frac{1}{\alpha}[Q^*(s,a) - V^*(s)]\right)$$

Desired features in RL

- Exploration

Desired features in RL

- Exploration
- Sample efficiency

Desired features in RL

- Exploration
- Sample efficiency
- Stable convergence

Desired features in RL

- Exploration
- Sample efficiency
- Stable convergence
- Little hyperparameter tuning

Desired features in RL

- Exploration
- Sample efficiency
- Stable convergence
- Little hyperparameter tuning

Q) Can we use max-entropy method in RL to have all the desired features?

Soft Actor-Critic (SAC)

Soft Actor-Critic:
Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor

Tuomas Haarnoja[1] Aurick Zhou[1] Pieter Abbeel[1] Sergey Levine[1]

Idea:

Soft Actor-Critic (SAC)

Soft Actor-Critic:
Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor

Tuomas Haarnoja[1] **Aurick Zhou**[1] **Pieter Abbeel**[1] **Sergey Levine**[1]

Idea:

- Max-entropy + off-policy actor-critic

Soft Actor-Critic (SAC)

Soft Actor-Critic:
Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor

Tuomas Haarnoja[1] Aurick Zhou[1] Pieter Abbeel[1] Sergey Levine[1]

Idea:

- Max-entropy + off-policy actor-critic

Advantages:

- Exploration
- Sample efficiency
- Stable convergence
- Little hyperparameter tuning

From soft policy iteration to SAC

From soft policy iteration to SAC

Q) What's the idea of policy iteration?

From soft policy iteration to SAC

Q) What's the idea of policy iteration?

1. Soft policy evaluation:
   Evaluate Q^{π}

From soft policy iteration to SAC

Q) What's the idea of policy iteration?

1. Soft policy evaluation:
   Evaluate Q^π
2. Soft policy improvement:
   Update π

From soft policy iteration to SAC

Q) What's the idea of policy iteration?

1. Soft policy evaluation:
   Evaluate Q^π

2. Soft policy improvement:
   Update π

From soft policy iteration to SAC

Q) What's the idea of policy iteration?

1. Soft policy evaluation:
   Evaluate Q^π
2. Soft policy improvement:
   Update π

- It's just a max-entropy variant of standard PI

Soft policy evaluation

- Modified Bellman operator:

$$T^\pi Q(s,a) := r(s,a) + \gamma \mathbb{E}_{s'}[V(s')],$$

where $V(s) := \mathbb{E}_{a\sim\pi(\cdot|s)}\big[Q(s,a) \qquad \underbrace{-\log\pi(a|s)}_{\text{takes into account entropy}} \qquad \big]$

Soft policy evaluation

- Modified Bellman operator:

$$T^{\pi}Q(s,a) := r(s,a) + \gamma \mathbb{E}_{s'}[V(s')],$$

where $V(s) := \mathbb{E}_{a\sim\pi(\cdot|s)}\Big[Q(s,a) \underbrace{- \log \pi(a|s)}_{\text{takes into account entropy}}\Big]$

- Repeatedly apply T^{π} to Q_k:

$$Q_{k+1} \leftarrow T^{\pi}Q_k$$

Soft policy evaluation

- Modified Bellman operator:

$$T^\pi Q(s,a) := r(s,a) + \gamma \mathbb{E}_{s'}[V(s')],$$

where $V(s) := \mathbb{E}_{a\sim\pi(\cdot|s)}\big[Q(s,a) \quad \underbrace{-\log\pi(a|s)}_{\text{takes into account entropy}} \quad \big]$

- Repeatedly apply T^π to Q_k:

$$Q_{k+1} \leftarrow T^\pi Q_k$$

- **Result: Q_k converges to Q^π!**

Soft policy improvement

- If no constraint on policies,

$$\pi_{new}(a|s) := \exp\left(\frac{1}{\alpha}[Q^{\pi_{old}}(s,a) - V^{\pi_{old}}(s)]\right)$$

Soft policy improvement

- If no constraint on policies,

$$\pi_{new}(a|s) := \exp\left(\frac{1}{\alpha}[Q^{\pi_{old}}(s,a) - V^{\pi_{old}}(s)]\right)$$

- When constraints need to be satisfied (i.e., $\pi \in \Pi$),

Soft policy improvement

- If no constraint on policies,

$$\pi_{new}(a|s) := \exp\left(\frac{1}{\alpha}[Q^{\pi_{old}}(s,a) - V^{\pi_{old}}(s)]\right)$$

- When constraints need to be satisfied (i.e., $\pi \in \Pi$),
  Information projection:

$$\pi_{new}(\cdot|s) \in \arg\min_{\pi'(\cdot|s)\in\Pi} D_{KL}\left(\pi'(\cdot|s) \parallel \underbrace{\exp\left(\frac{1}{\alpha}[Q^{\pi_{old}}(s,\cdot) - V^{\pi_{old}}(s)]\right)}_{\text{target policy}}\right)$$

Soft policy improvement

- If no constraint on policies,

$$\pi_{new}(a|s) := \exp\left(\frac{1}{\alpha}[Q^{\pi_{old}}(s,a) - V^{\pi_{old}}(s)]\right)$$

- When constraints need to be satisfied (i.e., $\pi \in \Pi$),
  Information projection: $\boxed{\text{constraint내에서 target policy와 가장 가까운 policy를 pick (KL divergence 로 측정)}}$

$$\pi_{new}(\cdot|s) \in \arg\min_{\pi'(\cdot|s)\in\Pi} D_{KL}\left(\pi'(\cdot|s) \;\|\; \underbrace{\exp\left(\frac{1}{\alpha}[Q^{\pi_{old}}(s,\cdot) - V^{\pi_{old}}(s)]\right)}_{\text{target policy}}\right)$$

- **Result: Monotonic improvement on policy!**
  (π_{new} better than π_{old})

Model-free version of soft PI: Soft actor-critic

- Soft PI: max-entropy variant of PI

Model-free version of soft PI: Soft actor-critic

- Soft PI: max-entropy variant of PI
- Soft actor-critic (SAC): max-entropy variant of actor-critic

critic & actor: evaluation & improvement

Model-free version of soft PI: Soft actor-critic

- Soft PI: max-entropy variant of PI
- Soft actor-critic (SAC): max-entropy variant of actor-critic
  - Soft critic:
    evaluates soft Q-function Q_ϕ of policy π

Model-free version of soft PI: Soft actor-critic

- Soft PI: max-entropy variant of PI
- Soft actor-critic (SAC): max-entropy variant of actor-critic
  - Soft critic:
    evaluates soft Q-function Q_ϕ of policy π
  - Soft actor:
    improves max-entropy policy using critic's evaluation

Step I: Soft critic

Critic: evaluates soft Q-function Q_ϕ of policy π

Step I: Soft critic

Critic: evaluates soft Q-function Q_ϕ of policy π

Q) How?

Step I: Soft critic

Critic: evaluates soft Q-function Q_ϕ of policy π

Q) How?

Idea: DQN + Max-entropy

Step I: Soft critic

Critic: evaluates soft Q-function Q_ϕ of policy π

Q) How?

Idea: DQN + Max-entropy

- Training soft Q-function:

Step I: Soft critic

Critic: evaluates soft Q-function Q_ϕ of policy π

Q) How?

Idea: DQN + Max-entropy

- Training soft Q-function:

$$\min_{\phi} \; J_Q(\phi) := \mathbb{E}_{(s_t,a_t)}\left[\frac{1}{2}(Q_\phi(s_t,a_t) - \underbrace{[r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}}[V_{\phi^-}(s_{t+1})]]}_{=: y_t^- \text{ target}})^2\right]$$

Step I: Soft critic

Critic: evaluates soft Q-function Q_ϕ of policy π

Q) How?

Idea: DQN + Max-entropy

- Training soft Q-function:

$$\min_\phi \; J_Q(\phi) := \mathbb{E}_{(s_t,a_t)}\left[\frac{1}{2}(Q_\phi(s_t,a_t) - \underbrace{[r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}}[V_{\phi^-}(s_{t+1})]]}_{=: y_t^- \text{ target}})^2\right]$$

- Stochastic gradient:

Step I: Soft critic

Critic: evaluates soft Q-function Q_ϕ of policy π

Q) How?

Idea: DQN + Max-entropy

- Training soft Q-function:

$$\min_{\phi} J_Q(\phi) := \mathbb{E}_{(s_t,a_t)}\left[\frac{1}{2}\Big(Q_\phi(s_t,a_t) - \underbrace{[r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}}[V_{\phi^-}(s_{t+1})]]}_{=:y_t^- \text{ target}}\Big)^2\right]$$

- Stochastic gradient:

w/ : soft Actor - critic
w/o: DDPG

$$\hat{\nabla}_\phi J_Q(\phi) = \nabla_\phi Q_\phi(s_t,a_t) \times \\ \Big[Q_\phi(s_t,a_t) - [r(s_t,a_t) + \gamma \underbrace{(Q_{\phi^-}(s_{t+1},a_{t+1}) - \alpha \log(\pi_\theta(a_{t+1}|s_{t+1}))}_{\text{sample estimate of } \mathbb{E}_{s_{t+1}}[V_{\phi^-}(s_{t+1})]}\Big]$$

Step II: Soft actor

Actor: updates policy using critic's evaluation

Step II: Soft actor

Actor: updates policy using critic's evaluation

Q) How?

Step II: Soft actor

Actor: updates policy using critic's evaluation

Q) How?

Idea: Policy gradient + Max-entropy:

Step II: Soft actor

Actor: updates policy using critic's evaluation

Q) How?

Idea: Policy gradient + Max-entropy:

- Minimizing the expected KL-divergence:

$$\min_{\theta} D_{KL}\left(\pi_\theta(\cdot|s_t) \parallel \underbrace{\exp\left(\frac{1}{\alpha}[Q_\phi(s_t,\cdot) - V_\phi(s_t)]\right)}_{\text{target policy}}\right),$$

Step II: Soft actor

Actor: updates policy using critic's evaluation

Q) How?

Idea: Policy gradient + Max-entropy:

- Minimizing the expected KL-divergence:

information projection of target
policy to policy network

$$\min_{\theta} D_{KL}\left(\pi_\theta(\cdot|s_t) \parallel \underbrace{\exp\left(\frac{1}{\alpha}[Q_\phi(s_t,\cdot) - V_\phi(s_t)]\right)}_{\text{target policy}}\right),$$

which is equivalent to

$$\min_{\theta} \; J_\pi(\theta) := \mathbb{E}_{s_t}\left[\mathbb{E}_{a_t \sim \pi_\theta(\cdot|s_t)}[\alpha \log \pi_\theta(a_t|s_t) - Q_\phi(s_t, a_t)]\right]$$

얘는 확률분포 - - > 네트워크 필요

Reparametrization trick

- Reparameterize the policy as

$$a_t := f_\theta(\epsilon_t; s_t),$$

where ϵ_t is an input noise with some fixed distribution (e.g., Gaussian)

Reparametrization trick

- Reparameterize the policy as

$$a_t := f_\theta(\epsilon_t; s_t),$$

where ϵ_t is an input noise with some fixed distribution (e.g., Gaussian)

- Benefit: lower variance

Reparametrization trick

- Reparameterize the policy as

$$a_t := f_\theta(\epsilon_t; s_t),$$

where ϵ_t is an input noise with some fixed distribution (e.g., Gaussian)

- Benefit: lower variance
- Rewrite the policy optimization problem as

$$\min_\theta \; J_\pi(\theta) := \mathbb{E}_{s_t, \epsilon_t}\left[\alpha \log \pi_\theta(f_\theta(\epsilon_t; s_t) | s_t) - Q_\phi(s_t, f_\theta(\epsilon_t; s_t))\right]$$

Reparametrization trick

stochastic actor

- Reparameterize the policy as

$$a_t := f_\theta(\epsilon_t; s_t),$$

네트워크 안쓰고 random하게 action 추출

where ϵ_t is an input noise with some fixed distribution (e.g., Gaussian)

- Benefit: lower variance
- Rewrite the policy optimization problem as

$$\min_\theta J_\pi(\theta) := \mathbb{E}_{s_t, \epsilon_t}\left[\alpha \log \pi_\theta(f_\theta(\epsilon_t; s_t)|s_t) - Q_\phi(s_t, f_\theta(\epsilon_t; s_t))\right]$$

- Stochastic gradient:

$$\begin{aligned}\hat{\nabla}_\theta J_\pi(\theta) &= \nabla_\theta \alpha \log \pi_\theta(a_t|s_t) \\ &\quad + [\nabla_a \alpha \log \pi_\phi(a_t|s_t) - \nabla_a Q_\phi(s_t, a_t)]\nabla_\theta f_\theta(\epsilon_t; s_t),\end{aligned}$$

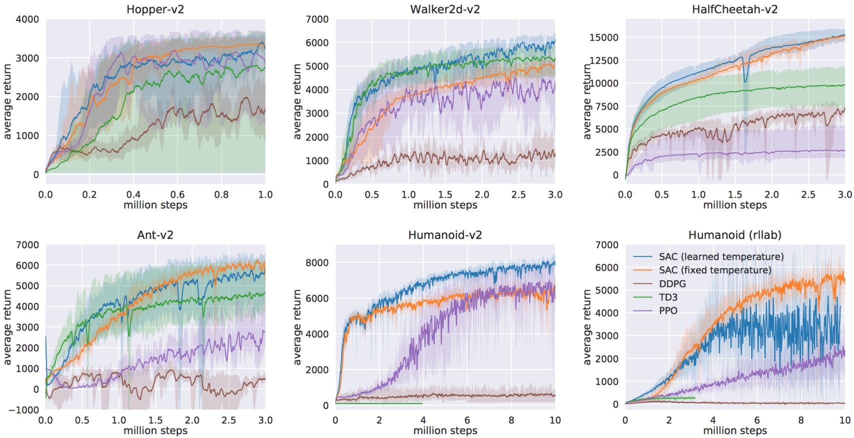
where a_t is evaluated at $f_\theta(\epsilon_t; s_t)$

Putting everything together: Soft Actor-Critic (SAC)

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

Results



- Soft actor-critic (blue and yellow) performs **consistently** across all tasks and **outperforming** both on-policy and off-policy methods in the most challenging tasks.

Dynamixel Claw task from vision

- The robot must rotate the valve so that the colored peg faces the right.
- The video embedded in the bottom right corner shows the frames as seen by the policy

Testing robustness of the learned policy against visual perturbations

- The robot must rotate the valve so that the colored peg faces the right.

Train the Minitaure robot to walk in 2 hours

Which algorithm should I use?

Which algorithm should I use?

- Guideline from Sergey Levine (UC Berkeley)