Soft Actor-Critic Algorithms and Applications

Presenter: Zizhao Wang

10/12/2021
Motivation

Sample Efficiency
Reinforcement Learning (RL) can solve complex decision making and control problems, but it is notoriously sample-inefficient.

Vinyals et al., 2019
“During training, each agent experienced up to **200 years** of real-time StarCraft play.”

[Grandmaster level in StarCraft II using multi-agent reinforcement learning](#)
Motivation

Sample Efficiency

Reinforcement Learning (RL) can solve complex decision making and control problems, but it is notoriously sample-inefficient.

Levine et al., 2016

- 14 robot arms learning to grasp in parallel
- Observing over 800,000 grasp attempts (3000 robot-hours of practice), we can see the beginnings of intelligent reactive behaviors.

[Link: Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection]
Motivation

Robustness to Hyperparameters

RL performance is brittle to hyperparameters, which needs laborious tuning.

Henderson et al., 2017

Reward scaling has a significant impact on DDPG performance.

(DDPG will be introduced soon)
Problem Setting

Markov Decision Process (MDP)

State Space \( s_t \in S \)

Action Space \( a_t \in A \)

Transition Probability \( p(s_{t+1}|s_t, a_t) \)

Reward \( r(s_t, a_t) \)

Policy \( \pi(\cdot|s_t) \)

Trajectory \( \tau \)
Context

RL Ranking on Sample Efficiency

- model-based

learns a model of the environment + use the learned model to imagine interactions
Context

RL Ranking on Sample Efficiency

- model-based
- model-free
  - off-policy
    - can learns from data generated by a different policy

off-policy RL
Context

RL Ranking on Sample Efficiency

- model-based
- model-free
  - off-policy
    - can learn from data generated by a different policy
  - on-policy
    - must learn from data generated by the current policy

off-policy RL

on-policy RL
Context

RL Ranking on Sample Efficiency

- model-based
- model-free
  - off-policy
    can learn from data generated by a different policy
  - on-policy
    must learn from data generated by the current policy
Context

Actor-Critic

- Critic: estimates future return of state-action pair
  \[ Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s = s_0, a = a_0 \right] \]
  \[ Q^\pi(s, a) = \mathbb{E}_{s' \sim \pi, a' \sim \pi(\cdot \mid s')} \left[ r(s, a) + \gamma Q^\pi(s', a') \right] \]

- Actor: adjusts policy to maximize critic's estimated future return
  \[ \pi(s) = \text{arg max}_a Q^\pi(s, a) \]
Context

Actor-Critic

- Critic: \( Q^\pi(s, a) = \mathbb{E}_{s' \sim p, a' \sim \pi(\cdot|s')} [r(s, a) + \gamma Q^\pi(s', a')] \)
- Actor: \( \pi(s) = \arg \max_a Q^\pi(s, a) \)

Deep Deterministic Policy Gradient (DDPG)

- Critic
  \( L_Q = \mathbb{E}_{(s, a, r, s') \sim D} [(Q(s, a) - (r + \gamma Q(s', \pi(s'))))^2] \)
- Actor
  \( L_\pi = -\mathbb{E}_{s \sim D} [Q(s, \pi(s))] \)
Context

Actor-Critic

- Critic: $Q^\pi(s, a) = \mathbb{E}_{s' \sim p, a' \sim \pi(\cdot|s')} [r(s, a) + \gamma Q^\pi(s', a')]$
- Actor: $\pi(s) = \arg \max_a Q^\pi(s, a)$

Deep Deterministic Policy Gradient (DDPG)

- Critic
  $$L_Q = \mathbb{E}_{(s, a, r, s') \sim D} [(Q(s, a) - (r + \gamma Q(s', \pi(s'))))^2]$$
- Actor
  $$L_\pi = -\mathbb{E}_{s \sim D} [Q(s, \pi(s))]$$
- Deterministic policy makes training unstable and brittle to hyperparameters.
Context

Standard RL
Reward: \( r(s, a) \)

Maximum Entropy RL
reward: \( r(s, a) + \alpha \mathcal{H}(\pi(\cdot|s)) \)
Context

Standard RL
Reward: $r(s, a)$

Maximum Entropy RL
reward: $r(s, a) + \alpha \mathcal{H}(\pi(\cdot|s))$

Entropy $\mathcal{H}$: measure of uncertainty

$$
\mathcal{H}(p) = - \int p(x) \log p(x) dx = - \mathbb{E}_{x \sim p} [\log p(x)]
$$

![Graph showing entropy as a function of p]
Context

Standard RL

reward: $r(s, a)$

optimal policy:

$$\pi^* = \arg \max \sum_t \mathbb{E}[\gamma^t r(s_t, a_t)]$$

Maximum Entropy RL

reward: $r(s, a) + \alpha \mathcal{H}(\pi(\cdot|s))$

optimal policy:

$$\pi^* = \arg \max \sum_t \mathbb{E}[\gamma^t (r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)))]$$
Context

**Standard RL**

reward: \( r(s, a) \)

optimal policy:

\[
\pi^* = \arg\max \sum_t \mathbb{E}[\gamma^t r(s_t, a_t)]
\]

**Maximum Entropy RL**

reward: \( r(s, a) + \alpha \mathcal{H}(\pi(\cdot | s)) \)

optimal policy:

\[
\pi^* = \arg\max \sum_t \mathbb{E}[\gamma^t (r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot | s_t)))]
\]

Why is it helpful?

- encourages exploration
- enables multi-modal action selection
Context

Standard RL

reward: $r(s, a)$

optimal policy:

$$\pi^* = \arg \max \sum_t \mathbb{E}[\gamma^t r(s_t, a_t)]$$

Maximum Entropy RL

reward: $r(s, a) + \alpha \mathcal{H}(\pi(\cdot | s))$

optimal policy:

$$\pi^* = \arg \max \sum_t \mathbb{E}[\gamma^t (r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot | s_t)))]$$

Why is it helpful?

- encourages exploration
- enables multi-modal action selection
Method

Actor-Critic (DDPG)

reward: \( r(s, a) \)

Q-value:

\[
Q^\pi(s_0, a_0) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]
\]

Soft Actor-Critic

reward: \( r(s, a) + \alpha \mathcal{H}(\pi(\cdot|s)) \)

Q-value:

\[
Q^\pi(s_0, a_0) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) + \alpha \sum_{t=1}^{\infty} \gamma^t \mathcal{H}(\pi(\cdot|s_t)) \right]
\]
Method

**Actor-Critic (DDPG)**

**reward:** $r(s, a)$

**Q-value:**

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

**Q-value updates:**

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

**Soft Actor-Critic**

**reward:** $r(s, a) + \alpha \mathcal{H}(\pi(\cdot|s))$

**Q-value:**

$$Q^\pi(s_0, a_0) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) + \alpha \sum_{t=1}^{\infty} \gamma^t \mathcal{H}(\pi(\cdot|s_t)) \right]$$

**Q-value updates:**

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim p, a' \sim \pi} \left[ r(s, a) + \gamma (Q^\pi(s', a') + \alpha \mathcal{H}(\pi(\cdot|s'))) \right]$$
Method

Actor-Critic (DDPG)

reward: \( r(s, a) \)

Q-value:

\[
Q^\pi(s_0, a_0) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]
\]

Q-value updates:

\[
Q^\pi(s, a) = \mathbb{E}_{s' \sim p_{a'}} \left[ r(s, a) + \gamma Q^\pi(s', a') \right]
\]

policy improvement:

\[
\pi_{\text{new}} = \arg \max_{\pi} Q^\pi_{\text{old}}(s_t, \pi(\cdot|s_t))
\]

Soft Actor-Critic

reward: \( r(s, a) + \alpha \mathcal{H}(\pi(\cdot|s)) \)

Q-value:

\[
Q^\pi(s_0, a_0) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) + \alpha \sum_{t=1}^{\infty} \gamma^t \mathcal{H}(\pi(\cdot|s_t)) \right]
\]

Q-value updates:

\[
Q^\pi(s, a) = \mathbb{E}_{s' \sim p_{a'}} \left[ r(s, a) + \gamma (Q^\pi(s', a') + \alpha \mathcal{H}(\pi(\cdot|s'))) \right]
\]

policy improvement:

\[
\pi_{\text{new}} = \arg \min_{\pi} D_{KL} \left( \pi(\cdot|s_t) \| \frac{\exp(\frac{1}{\alpha} Q^\pi_{\text{old}}(s_t, \cdot))}{Z^\pi_{\text{old}}(s_t)} \right)
\]
Method

Actor-Critic (DDPG)

\[ \pi_{\text{new}} = \arg \max_{\pi} Q^{\pi_{\text{old}}}(s_t, \pi(\cdot|s_t)) \]

Soft Actor-Critic

\[ \pi_{\text{new}} = \arg \min_{\pi} D_{\text{KL}} \left( \pi(\cdot|s_t) \parallel \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(s_t, \cdot)\right)}{Z^{\pi_{\text{old}}}(s_t)} \right) \]
Method

Proof of Convergence

Assumption: finite state and action space $|S| < \infty, |A| < \infty$

1. If we update Q-value as follows, it will converge to $Q^\pi$ as $k \to \infty$.

$$Q^{k+1}(s, a) = \mathbb{E}_{s' \sim p} \left[ r(s, a) + \gamma (Q^k(s', a') + \alpha \mathcal{H}(\pi(\cdot | s')) ) \right]$$
Method

Proof of Convergence

Assumption: finite state and action space $|S| < \infty, |A| < \infty$

1. If we update Q-value as follows, it will converge to $Q^\pi$ as $k \to \infty$.

$$Q^{k+1}(s, a) = \mathbb{E}_{s' \sim p} \left[ r(s, a) + \gamma (Q^k(s', a') + \alpha \mathcal{H}(\pi(\cdot|s'))) \right]$$

2. If we update the policy as follows, then $Q^{\pi_{\text{new}}}(s_t, a_t) \geq Q^{\pi_{\text{old}}}(s_t, a_t), \forall (s_t, a_t) \in S \times A$.

$$\pi_{\text{new}} = \arg \min_{\pi} D_{KL} \left( \pi(\cdot|s_t) \parallel \frac{\exp \left( \frac{1}{\alpha} Q^{\pi_{\text{old}}}(s_t, \cdot) \right)}{Z^{\pi_{\text{old}}}(s_t)} \right)$$
Method

Proof of Convergence

Assumption: finite state and action space $|S| < \infty, |A| < \infty$

1. If we update Q-value as follows, it will converge to $Q^\pi$ as $k \to \infty$.

$$Q^{k+1}(s, a) = \mathbb{E}_{s' \sim p, a' \sim \pi} \left[ r(s, a) + \gamma (Q^k(s', a') + \alpha H(\pi(\cdot|s'))) \right]$$

2. If we update the policy as follows, then $Q^{\pi_{\text{new}}}(s_t, a_t) \geq Q^{\pi_{\text{old}}}(s_t, a_t), \forall (s_t, a_t) \in S \times A$.

$$\pi_{\text{new}} = \arg\min_{\pi} D_{KL} \left( \pi(\cdot|s_t) \parallel \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(s_t, \cdot)\right)}{Z^{\pi_{\text{old}}}(s_t)} \right)$$

3. If we repeat step 1 and 2, we will find the optimal policy $\pi^*$ such that $Q^{\pi^*}(s_t, a_t) \geq Q^\pi(s_t, a_t)$ for any policy $\pi$ and $\forall (s_t, a_t) \in S \times A$. 
Method

Proof of Convergence: Are they realistic?

Assumption: finite state and action space $|S| < \infty, |A| < \infty$

1. If we update Q-value as follows, it will converge to $Q^\pi$ as $k \to \infty$.

   $$Q^{k+1}(s,a) = \mathbb{E}_{s',a' \sim p\pi}[r(s,a) + \gamma (Q^k(s',a') + \alpha \mathcal{H}(\pi(\cdot|s')))]$$

2. If we update the policy as follows, then $Q^{\pi_{new}}(s_t,a_t) \geq Q^{\pi_{old}}(s_t,a_t), \forall (s_t,a_t) \in S \times A$.

   $$\pi_{new} = \arg \min_{\pi} D_{KL}\left(\pi(\cdot|s_t) \parallel \frac{\exp(\frac{1}{\alpha} Q^{\pi_{old}}(s_t, \cdot))}{Z_{\pi_{old}}(s_t)}\right)$$

3. If we repeat step 1 and 2, we will find the optimal policy $\pi^*$ such that $Q^{\pi^*}(s_t,a_t) \geq Q^{\pi}(s_t,a_t)$ for any policy $\pi$ and $\forall (s_t,a_t) \in S \times A$. 
Method

Proof of Convergence: Are they realistic?

Assumption: finite state and action space $|S| < \infty, |A| < \infty$

1. If we update Q-value as follows, it will converge to $Q^\pi$ as $k \to \infty$.

$$Q^{k+1}(s, a) = \mathbb{E}_{s' \sim p \atop a' \sim \pi} \left[ r(s, a) + \gamma (Q^k(s', a') + \alpha H(\pi(\cdot | s')) \right]$$

2. If we update the policy as follows, then $Q^{\pi_{\text{new}}}(s_t, a_t) \geq Q^{\pi_{\text{old}}}(s_t, a_t), \forall (s_t, a_t) \in S \times A$.

$$\pi_{\text{new}} = \arg\min_{\pi} D_{KL} \left( \pi(\cdot | s_t) \parallel \frac{\exp \left( \frac{1}{\alpha} Q^{\pi_{\text{old}}}(s_t, \cdot) \right)}{Z^{\pi_{\text{old}}}(s_t)} \right)$$

3. If we repeat step 1 and 2, we will find the optimal policy $\pi^*$ such that $Q^{\pi^*}(s_t, a_t) \geq Q^\pi(s_t, a_t)$ for any policy $\pi$ and $\forall (s_t, a_t) \in S \times A$. For how many step? Possibly a lot.
Method

Proof of Convergence: Are they realistic?

Assumption: finite state and action space $|S| < \infty, |A| < \infty$

1. If we update Q-value as follows, it will converge to $Q^\pi$ as $k \to \infty$.

$$Q^{k+1}(s, a) = \mathbb{E}_{s' \sim p} \left[ r(s, a) + \gamma (Q^k(s', a') + \alpha \mathcal{H}(\pi(\cdot | s'))) \right]$$

2. If we update the policy as follows, then $Q^{\pi_{\text{new}}}(s_t, a_t) \geq Q^{\pi_{\text{old}}}(s_t, a_t), \forall (s_t, a_t) \in S \times A$.

$$\pi_{\text{new}} = \arg\min_{\pi} D_{KL} \left( \pi(\cdot | s_t) \parallel \frac{\exp(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(s_t, \cdot))}{Z^{\pi_{\text{old}}}(s_t)} \right)$$

3. If we repeat step 1 and 2, we will find the optimal policy $\pi^*$ such that $Q^{\pi^*}(s_t, a_t) \geq Q^{\pi}(s_t, a_t)$ for any policy $\pi$ and $\forall (s_t, a_t) \in S \times A$. For how many step? Possibly a lot.

Not applicable to many robotics problems (continuous state/action) and not computationally tractable.
Method

Implementation (Practical Approximation)

1. critic training
   - convergence → gradient descent
     
     \[
     Q^{k+1}(s, a) = \mathbb{E}_{s' \sim p} \left[ r(s, a) + \gamma (Q^k(s', a') + \alpha \mathcal{H}(\pi(\cdot|s'))) \right], k \to \infty
     \]

     \[
     L_Q = \mathbb{E}_{(s, a, r) \sim D} \left[ \left( Q(s, a) - \left( r + \gamma \mathbb{E}_{s' \sim p} \left[ \mathbb{E}_{a' \sim \pi(\cdot|s')} [Q(s', a')] + \alpha \mathcal{H}(\pi(\cdot|s')) \right] \right) \right]^2
     \]
Method

Implementation (Practical Approximation)

1. critic training
   - convergence → gradient descent
     \[ Q^{k+1}(s, a) = \mathbb{E}_{s' \sim p} \left[ r(s, a) + \gamma (Q^k(s', a') + \alpha \mathcal{H}(\pi(\cdot|s'))) \right], k \to \infty \]
   - remove expectation
     \[ L_Q = \mathbb{E}_{(s, a, r) \sim D} \left( \left( Q(s, a) - \left( r + \gamma \mathbb{E}_{s' \sim p} \left[ Q(s', a') + \alpha \mathcal{H}(\pi(\cdot|s')) \right] \right) \right)^2 \right] \]
     \[ = \mathbb{E}_{(s, a, r) \sim D} \left( Q(s, a) - \left( r + \gamma \mathbb{E}_{s' \sim p} [Q(s', a') + \alpha \mathcal{H}(\pi(\cdot|s'))] \right) \right)^2, a' \sim \pi(\cdot|s') \]
Method

Implementation (Practical Approximation)

1. critic training
   - convergence → gradient descent
     \[
     Q^{k+1}(s, a) = \mathbb{E}_{s' \sim p} \left[ r(s, a) + \gamma \left( Q^k(s', a') + \alpha \mathcal{H}(\pi(\cdot|s')) \right) \right], k \to \infty
     \]
   - remove expectation
     - next action
     - entropy
     \[
     \mathcal{H}(p) = -\mathbb{E}_{x \sim p} [\log p(x)]
     \]
Method

Implementation (Practical Approximation)

1. critic training
   - convergence → gradient descent
     \[
     Q^{k+1}(s, a) = \mathbb{E}_{s' \sim p} \left[ r(s, a) + \gamma \left( Q^k(s', a') + \alpha \mathcal{H}(\pi(\cdot | s')) \right) \right], k \to \infty
     \]
   - remove expectation
     - next action
     - entropy
       \[
       \mathcal{H}(p) = -\mathbb{E}_{x \sim p}[\log p(x)]
       \]
     - next state

\[
L_Q = \mathbb{E}_{(s, a, r) \sim D} \left[ \left( Q(s, a) - \left( r + \gamma \mathbb{E}_{s' \sim p} \left[ Q(s', a') + \alpha \mathcal{H}(\pi(\cdot | s')) \right] \right) \right)^2 \right]
\]

\[
= \mathbb{E}_{(s, a, r) \sim D} \left[ \left( Q(s, a) - \left( r + \gamma \mathbb{E}_{s' \sim p} \left[ Q(s', a') + \alpha \mathcal{H}(\pi(\cdot | s')) \right] \right) \right)^2 \right], a' \sim \pi(\cdot | s')
\]

\[
= \mathbb{E}_{(s, a, r) \sim D} \left[ \left( Q(s, a) - \left( r + \gamma \mathbb{E}_{s' \sim p} \left[ Q(s', a') - \alpha \log \pi(a' | s') \right] \right) \right)^2 \right]
\]

\[
= \mathbb{E}_{(s, a, r, s') \sim D} \left[ \left( Q(s, a) - \left( r + \gamma Q(s', a') - \alpha \log \pi(a' | s') \right) \right)^2 \right]
\]
Method

Implementation (Practical Approximation)

2. actor

training

- convergence → gradient descent
- rewrite KL divergence

\[ D_{KL}(p||q) = \mathbb{E}_{x \sim p(\cdot)} \left[ \log \frac{p(x)}{q(x)} \right] \]

\[ \pi_{new} = \arg \min_{\pi} D_{KL} \left( \pi(\cdot|s_t) \left\| \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{old}}(s_t, \cdot)\right)}{Z^{\pi_{old}}(s_t)} \right) \right) \]

\[ L_{\pi} = \mathbb{E}_{s \sim D} \left[ \mathbb{E}_{a' \sim \pi(\cdot|s')} \left[ \log \frac{\pi(a'|s')}{\exp\left(\frac{1}{\alpha} Q(s', a')/Z^{\pi}\right)} \right] \right] \]
Method

Implementation (Practical Approximation)

2. actor

training

- convergence → gradient descent
- rewrite KL divergence

\[
\pi_{\text{new}} = \arg \min_{\pi} D_{\text{KL}} \left( \pi(\cdot|s_t) \bigg|\bigg| \frac{\exp\left(\frac{1}{\alpha} Q_{\text{old}}(s_t, \cdot)\right)}{Z_{\text{old}}(s_t)} \right)
\]

\[
L_\pi = \mathbb{E}_{s \sim D} \left[ \mathbb{E}_{a' \sim \pi(\cdot|s')} \left[ \log \frac{\pi(a'|s')}{\exp\left(\frac{1}{\alpha} Q(s', a')/Z_\pi\right)} \right] \right]
\]

\[
= \mathbb{E}_{s \sim D} \left[ \mathbb{E}_{a' \sim \pi(\cdot|s')} \left[ \log \pi(a'|s') - \frac{1}{\alpha} Q(s', a') + \log Z_\pi \right] \right]
\]
Method

Implementation (Practical Approximation)

2. actor

- training
  - convergence → gradient descent
  - rewrite KL divergence
  - scale loss by $\alpha$ and omit constant

\[
\pi_{\text{new}} = \arg \min_{\pi} D_{KL} \left( \pi(\cdot | s_t) \mid \frac{\exp(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(s_t, \cdot))}{Z^{\pi_{\text{old}}}(s_t)} \right)
\]

\[
L_{\pi} = \mathbb{E}_{s \sim D} \left[ \mathbb{E}_{a' \sim \pi(\cdot | s')} \left[ \log \frac{\pi(a' | s')}{\exp(\frac{1}{\alpha} Q(s', a') / Z^{\pi})} \right] \right]
\]

\[
= \mathbb{E}_{s \sim D} \left[ \mathbb{E}_{a' \sim \pi(\cdot | s')} \left[ \log \pi(a' | s') - \frac{1}{\alpha} Q(s', a') + \log Z^{\pi} \right] \right]
\]

\[
= \mathbb{E}_{s \sim D} \left[ \alpha \log \pi(a' | s') - Q(s', a') \right]
\]
Method

Implementation (Practical Approximation)

2. actor

- convergence → gradient descent
- rewrite KL divergence
- scale loss by $\alpha$ and omit constant
- remove expectation over action

$$\pi_{\text{new}} = \arg \min_\pi D_{\text{KL}} \left( \pi(\cdot | s_t) \left| \left| \frac{\exp \left( \frac{1}{\alpha} Q^{\pi^\text{old}}(s_t, \cdot) \right)}{Z^{\pi^\text{old}}(s_t)} \right) \right)$$

$$L_\pi = \mathbb{E}_{s \sim D} \left[ \mathbb{E}_{a' \sim \pi(\cdot | s')} \left[ \log \frac{\pi(a' | s')}{\exp \left( \frac{1}{\alpha} Q(s', a') \right) / Z^\pi} \right] \right]$$

$$= \mathbb{E}_{s \sim D} \left[ \mathbb{E}_{a' \sim \pi(\cdot | s')} \left[ \log \pi(a' | s') - \frac{1}{\alpha} Q(s', a') + \log Z^\pi \right] \right]$$

$$= \mathbb{E}_{s \sim D} \left[ \mathbb{E}_{a' \sim \pi(\cdot | s')} [\alpha \log \pi(a' | s') - Q(s', a')] \right]$$

$$= \mathbb{E}_{s \sim D} [\alpha \log \pi(a' | s') - Q(s', a')]$$
Method

Implementation (Practical Approximation)

2. actor

training: gradient descent with $L_\pi = \mathbb{E}_{s \sim D} [\alpha \log \pi(a' | s') - Q(s', a')]$, $a' \sim \pi(\cdot | s')$

design choice: policy as normal distribution

- but normal distribution is unimodal, it loses the declared multi-modal advantages.
Method

Automatic Entropy Adjustment

Choosing the optimal $\alpha$ is not trivial

- Recall for maximum entropy RL, the reward is $r(s, a) + \alpha \mathcal{H}(\pi(\cdot|s))$. 
Method

Automatic Entropy Adjustment

Choosing the optimal $\alpha$ is not trivial

- Recall for maximum entropy RL, the reward is $r(s, a) + \alpha \mathcal{H}(\pi(.|s))$.
- So $\alpha$ depends on the magnitude of $r$, but $r$ can vary a lot across
  - different tasks
  - different policies as the policy gets improved
Method

Automatic Entropy Adjustment

Choosing the optimal $\alpha$ is not trivial

- Recall for maximum entropy RL, the reward is $r(s, a) + \alpha H(\pi(\cdot | s))$.
- So $\alpha$ depends on the magnitude of $r$, but $r$ can vary a lot across
  - different tasks
  - different policies as the policy gets improved
- Ideal policy entropy should be
  - stochastic enough for exploration in uncertain regions
  - deterministic enough for performance in learned regions
Method

Automatic Entropy Adjustment

Choosing the optimal $\alpha$ is not trivial

- Recall for maximum entropy RL, the reward is $r(s, a) + \alpha \mathcal{H}(\pi(\cdot | s))$.
- So $\alpha$ depends on the magnitude of $r$, but $r$ can vary a lot across
  - different tasks
  - different policies as the policy gets improved
- Ideal policy entropy should be
  - stochastic enough for exploration in uncertain regions
  - deterministic enough for performance in learned regions
- Only constrain the **average** entropy across states
Method

Automatic Entropy Adjustment

Only constrain the average entropy across states

\[ L_\alpha = \mathbb{E}_{s \sim D} [\alpha (\mathcal{H}(\pi(\cdot|s)) - \bar{\mathcal{H}})] \]

\[ = \mathbb{E}_{s \sim D} [\alpha (- \log \pi(a|s) - \bar{\mathcal{H}})], a \sim \pi(\cdot|s) \]

- \( \bar{\mathcal{H}} \): target entropy, \( \alpha \) always > 0
- increase \( \alpha \) if \( \mathcal{H}(\pi(\cdot|s')) < \bar{\mathcal{H}} \), decrease otherwise.
Method

Overall Algorithm

\textbf{Input:} \( \theta_1, \theta_2, \phi \)
\[ \theta_1 \leftarrow \theta_1, \theta_2 \leftarrow \theta_2 \]
\[ \mathcal{D} \leftarrow \emptyset \]

\textbf{for} each iteration \textbf{do}
\textbf{for} each environment step \textbf{do}
\[ a_t \sim \pi_{\phi}(a_t|s_t) \]
\[ s_{t+1} \sim p(s_{t+1}|s_t, a_t) \]
\[ \mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\} \]
\textbf{end for}
\textbf{end for}

\textbf{for} each gradient step \textbf{do}
\[ \theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i) \text{ for } i \in \{1, 2\} \]
\[ \phi \leftarrow \phi - \lambda_{\pi} \nabla_{\phi} J_\pi(\phi) \]
\[ \alpha \leftarrow \alpha - \lambda_{\alpha} \nabla_{\alpha} J(\alpha) \]
\[ \theta_i \leftarrow \tau \theta_i + (1 - \tau) \hat{\theta}_i \text{ for } i \in \{1, 2\} \]
\textbf{end for}

\textbf{end for}

\textbf{Output:} \( \theta_1, \theta_2, \phi \)

\[ \begin{align*}
\triangleright & \hspace{0.5cm} \text{Initial parameters} \\
\triangleright & \hspace{0.5cm} \text{Initialize target network weights} \\
\triangleright & \hspace{0.5cm} \text{Initialize an empty replay pool} \\
\triangleright & \hspace{0.5cm} \text{Sample action from the policy} \\
\triangleright & \hspace{0.5cm} \text{Sample transition from the environment} \\
\triangleright & \hspace{0.5cm} \text{Store the transition in the replay pool} \\
\triangleright & \hspace{0.5cm} \text{Update the Q-function parameters} \\
\triangleright & \hspace{0.5cm} \text{Update policy weights} \\
\triangleright & \hspace{0.5cm} \text{Adjust temperature} \\
\triangleright & \hspace{0.5cm} \text{Update target network weights} \\
\triangleright & \hspace{0.5cm} \text{Optimized parameters}
\end{align*} \]
Experiment

Simulated Benchmarks

State: joint value
Action: joint torque
Metric: average return
Experiment

Baselines

- SAC with learned $\alpha$
- SAC with fixed $\alpha$
- DDPG (off-policy, deterministic policy)
- TD3 (DDPG with engineering improvements)
- PPO (on-policy)
Experiment

Baselines

- SAC with learned $\alpha$
- SAC with fixed $\alpha$
- DDPG (off-policy, deterministic policy)
- TD3 (DDPG with engineering improvements)
- PPO (on-policy)

Hypotheses

Compared to baselines, if SAC has better

- sample-efficiency: learning speed + final performance
- stability: performance on hard tasks where hyperparameters tuning is challenging
Experiment
Experiment

Simulated Benchmarks

- Easy tasks (hopper, walker)
  - all algorithm performs comparably except for DDPG
Experiment

Simulated Benchmarks

- Normal tasks (half cheetah, ant)
  - SAC > TD3 > DDPG & PPO in both learning speed and final performance
Experiment

Simulated Benchmarks

- Hard tasks (humanoid)
  - DDPG & its variant TD3 fail to learn
  - SAC learns much faster than PPO
Experiment

Simulated Benchmarks

- Larger variance with the automatic temperature adjustment
Experiment

Real World Quadrupedal Locomotion

State: low-dimensional

Challenges: sample efficiency & generalization to unseen environments

Training: 160k steps (2 hours)
Experiment

Real World Quadrupedal Locomotion

Train on Flat

Test on Slope

Test with Obstacles

Test with Stairs
Experiment

Real World Manipulation

State: hand joint angles + image / ground truth valve position

Challenges: precepting the valve position from images

Comparison (using ground truth valve position): SAC (3 hrs) vs PPO (7.4 hrs)
Limitations

● Multi-modal policy
  ○ Though it is declared that maximum entropy RL can benefit from multi-modal policy, SAC chooses to use a unimodal policy (normal distribution).
  ○ All experiments don’t require multi-modal behaviors to finish.

● Hyperparameter tuning
  ○ Target entropy $\overline{H}$ brings a new hyperparameter to tune.
  ○ Average entropy constraints do not provide the desired exploration + exploitation balance.
  ○ For manipulation tasks that require accurate control, even averaged entropy regularization still hurts the performance.
Future Work and Extended Readings

● Learn multi-modal policy rather than unimodal policy
  ○ Reinforcement Learning with Deep Energy-Based Policies

● Improve sample-efficiency with auxiliary tasks
  ○ Improving Sample Efficiency in Model-Free Reinforcement Learning from Images
  ○ CURL: Contrastive Unsupervised Representations for Reinforcement Learning

● Improve generalizability by learning tasks-relevant features
  ○ Learning Invariant Representations for Reinforcement Learning without Reconstruction
  ○ Learning Task Informed Abstractions
Summary

● Problem: sample-efficient RL with automatic hyperparameter tuning

● Limitations of prior work:
  ○ On-policy RL is sample-inefficient
  ○ DDPG that uses deterministic policy is brittle to hyperparameters

● Key insights of the proposed work
  ○ Maximal entropy RL encourages exploration and robust to environments & hyperparameters.
  ○ Use the average entropy across states as regularization.

● State-of-the-art sample efficiency and generalizabilities on simulated and real world tasks.