Theory and Methods for Reinforcement Learning

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 9: Actor-Critic Methods for Deep RL

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

EE-618 (Spring 2020)

License Information for Reinforcement Learning Slides

Outline

- This class:
  1. Actor-Critic Methods for Deep RL
- Next class:
  1. Inverse Reinforcement Learning

Recommended reading

- I. Grondman, et al, *A survey of actor-critic reinforcement learning: Standard and natural policy gradients*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42.6 (2012): 1291-1307.
- Chapter 13 in S. Sutton, and G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.

Motivation

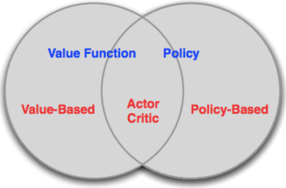
Motivation

Monte-Carlo policy gradient method suffers from high variance. How can we speed up learning? Learn the value function along with the policy!

Recap

- Discounted return: $R_t = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau$
- State-action value function: $Q^\pi(s,a) = \mathbb{E}\left[R_t \mid s_t = s, a_t = a, \pi\right]$
- State value function: $V^\pi(s) = \mathbb{E}_{a \sim \pi(s)}\left[Q^\pi(s,a)\right]$
- Advantage function: $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$
- Optimal state-action value function: $Q^*(s,a) = \max_\pi Q^\pi(s,a)$
- Optimal policy: $\pi^*(s) = \arg\max_{a'} Q^*(s,a')$
- Optimal state value function: $V^*(s) = \max_a Q^*(s,a)$
- Bellman expectation: $Q^\pi(s,a) = \mathbb{E}_{s'}\left[r + \gamma \mathbb{E}_{a' \sim \pi(s')}\left[Q^\pi(s',a')\right] \mid s, a, \pi\right]$
- Bellman optimality: $Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a') \mid s, a\right]$

Three Approaches to RL



- Policy based learning (actor-only methods)
- Value based learning (critic-only methods)
- Actor-Critic learning

Three Approaches to RL

- Actor-only Methods
  - work with a parameterized family of policies
  - explicitly learn policy $\pi_\theta(a \mid s)$ that implicitly maximize reward over all policies
  - a spectrum of continuous actions can be generated
  - policy gradient methods suffer from high variance in the estimates of the gradient
- Critic-only Methods
  - use temporal difference learning or Bellman optimality relationship
  - have a lower variance in the estimates of expected returns
  - derive a policy by selecting greedy actions
  - learn value function $Q^w(s,a)$ and from there infer policy $\pi(s) = \arg\max_a Q^w(s,a)$
  - undermines the ability of using continuous actions

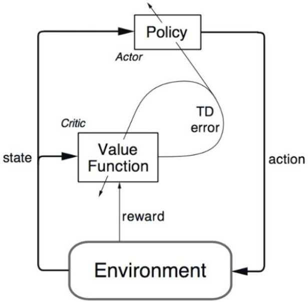
Three Approaches to RL

- Actor-Critic Methods
  - combine the advantages of actor-only and critic-only methods
  - parameterized actor: computing continuous actions without the need for optimization procedures on a value function
  - critic: supplies the actor with low-variance knowledge of the performance
  - lower variance is traded for a larger bias at the start of learning when the critic's estimates are far from accurate
  - actor-critic methods usually have good convergence properties, in contrast to critic-only methods [4]

Actor-Critic Architectures

- Actor-critic algorithms maintain two sets of parameters:
  - critic parameters w to approximate action-value function under current policy
  - actor (policy) parameters θ

- Actor-critic algorithms follow an approximate policy gradient:
  - critic updates Q-function parameters w like in policy evaluation
  - actor updates policy gradient θ in direction suggested by critic



Stochastic On-Policy Search

• Parametrize a stochastic policy $\pi_\theta : \mathcal{S} \to \Delta\mathcal{A}$

• Notation: the parameter θ would be omitted for the policy π_θ when the policy is present in the subscript or superscript of other functions.

• The on-policy objective function:

$$\begin{aligned} J(\theta) &= V^\pi(s_0) \\ &= \sum_{s \in \mathcal{S}} d^\pi(s) V^\pi(s) \\ &= \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) Q^\pi(s, a) \end{aligned}$$

where

- $d^\pi(s)$ is the stationary distribution of Markov chain for π_θ
- $d^\pi(s) = \lim_{t\to\infty} \mathbb{P}[S_t = s \mid s_0, \pi_\theta]$ is the probability that $S_t = s$ when starting from s_0 and following policy π_θ for t steps.

Stochastic On-Policy Gradient Theorem

Theorem

For any differentiable policy $\pi_\theta(a \mid s)$*, we have*

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} Q^\pi(s,a)\, \pi_\theta(a \mid s) \\
&\propto \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} Q^\pi(s,a)\, \nabla_\theta \pi_\theta(a \mid s) \\
&= \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s)\, Q^\pi(s,a) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \\
&= \mathbb{E}_{s \sim d^\pi, a \sim \pi_\theta}\left[Q^\pi(s,a)\, \nabla_\theta \log \pi_\theta(a \mid s)\right] \\
&= \mathbb{E}_\pi\left[Q^\pi(s,a)\, \nabla_\theta \log \pi_\theta(a \mid s)\right]
\end{aligned}$$

Stochastic Off-Policy Search

- Advantages of off-policy methods:
  1. The off-policy approach does not require full trajectories and can reuse any past episodes (experience replay) for much better sample efficiency.
  2. The sample collection follows a behavior policy different from the target policy, bringing better exploration.

- The off-policy objective function (with behavior policy $\beta(a \mid s)$):

$$J(\theta) = \sum_{s \in \mathcal{S}} d^{\beta}(s) \sum_{a \in \mathcal{A}} Q^{\pi}(s, a) \pi_{\theta}(a \mid s) = \mathbb{E}_{s \sim d^{\beta}}\left[\sum_{a \in \mathcal{A}} Q^{\pi}(s, a) \pi_{\theta}(a \mid s)\right],$$

where

- $d^{\beta}(s) = \lim_{t \to \infty} \mathbb{P}[S_t = s \mid s_0, \beta]$
- Q^{π} is the action-value function estimated with regard to the target policy π_{θ}

Stochastic Off-Policy Gradient Theorem

$$
\begin{aligned}
\nabla_\theta J_\beta(\theta) &= \nabla_\theta \mathbb{E}_{s\sim d^\beta}\left[\sum_{a\in\mathcal{A}} Q^\pi(s,a)\,\pi_\theta(a \mid s)\right] \\
&= \mathbb{E}_{s\sim d^\beta}\left[\sum_{a\in\mathcal{A}} Q^\pi(s,a)\,\nabla_\theta\pi_\theta(a \mid s) + \nabla_\theta Q^\pi(s,a)\,\pi_\theta(a \mid s)\right] \\
&\approx \mathbb{E}_{s\sim d^\beta}\left[\sum_{a\in\mathcal{A}} Q^\pi(s,a)\,\nabla_\theta\pi_\theta(a \mid s)\right] \\
&= \sum_{s\in\mathcal{S}} d^\beta(s) \sum_{a\in\mathcal{A}} \beta(a \mid s)\,\frac{\pi_\theta(a \mid s)}{\beta(a \mid s)} Q^\pi(s,a)\,\frac{\nabla_\theta\pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \\
&= \mathbb{E}_{s\sim d^\beta,a\sim\beta}\left[\frac{\pi_\theta(a \mid s)}{\beta(a \mid s)} Q^\pi(s,a)\,\nabla_\theta\log\pi_\theta(a \mid s)\right] \\
&= \mathbb{E}_\beta\left[\frac{\pi_\theta(a \mid s)}{\beta(a \mid s)} Q^\pi(s,a)\,\nabla_\theta\log\pi_\theta(a \mid s)\right]
\end{aligned}
$$

- This is a good approximation since it can preserve the set of local optima to which gradient ascent converges [1].

REINFORCE (Monte-Carlo Policy Gradient)

- Vanilla policy gradient update has no bias but high variance.

- Relies on an estimated return by Monte-Carlo methods using episode samples to update the policy parameter θ.

- Since $Q^{\pi}(S_t, A_t) = \mathbb{E}_{\pi}[G_t \mid S_t, A_t]$, we have

$$\begin{aligned} \nabla_{\theta} J(\theta) &= \mathbb{E}_{\pi}\left[Q^{\pi}(s, a) \nabla_{\theta} \log \pi_{\theta}(a \mid s)\right] \\ &= \mathbb{E}_{\pi}\left[G_t \nabla_{\theta} \log \pi_{\theta}(A_t \mid S_t)\right] \end{aligned}$$

- It relies on a full trajectory and that's why it is a Monte-Carlo method.

REINFORCE: Monte Carlo Policy Gradient [7]

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π^*

Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\theta \in \mathbb{R}^d$
for each episode **do**
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$ following $\pi(\cdot \mid \cdot, \theta)$
  for each step of the episode $t = 0, 1, \ldots, T-1$ **do**
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $\theta \leftarrow \theta + \alpha \gamma^t G \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}$
  end for
end for

• The use of stochastic gradient method ensures the convergence to a local optimum when choosing a decreasing step α_t such that $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$.

Stochastic Actor-Critic Algorithms

- An actor adjusts the parameters θ of the stochastic policy $\pi_\theta(s)$ by stochastic gradient ascent

- A critic estimates the action-value function $Q^w(s,a) \approx Q^\pi(s,a)$ using an appropriate policy evaluation algorithm such as temporal-difference learning.

Compatible Function Approximation: Bias in AC

- Approximating the policy gradient introduces bias
- A biased policy gradient may not find the right solution
- Luckily, if we choose value function approximation carefully, then we can avoid bias
- If the following two conditions are satisfied:
  1. Value function approximator is compatible to the policy

$$\nabla_w Q^w(s,a) = \nabla_\theta \log \pi_\theta(a \mid s)$$

  2. Value function parameters w minimize the mean-squared error

$$\nabla_w \mathbb{E}_{s\sim d^\pi, a\sim\pi_\theta}\left[\left(Q^w(s,a) - Q^\pi(s,a)\right)^2\right] = 0$$

- Then the policy gradient is without bias [6]:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim d^\pi, a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s)\, Q^w(s,a)\right]$$

Action-Value Actor-Critic (QAC)

Action-Value Actor-Critic Algorithm

Algorithm parameters: learning rates $\alpha_w, \alpha_\theta > 0$.
Initialize s, θ, w at random; sample $a \sim \pi_\theta(a \mid s)$.
for $t = 0, 1, \dots, T$ **do**
  Sample reward $r_t \sim R(s, a)$ and next state $s' \sim P(s' \mid s, a)$
  Then sample the next action $a' \sim \pi_\theta(a' \mid s')$
  Update the policy parameters:

$$\theta \leftarrow \theta + \alpha_\theta Q^w(s, a) \nabla_\theta \log \pi_\theta(a \mid s)$$

  Compute the correction (TD error) for action-value at time t:

$$\delta_t = r_t + \gamma Q^w\left(s', a'\right) - Q^w(s, a)$$

  Use it to update the parameters of action-value function:

$$w \leftarrow w + \alpha_w \delta_t \nabla_w Q^w(s, a)$$

end for

Advantage Actor Critic (AAC or A2C)

- In this critic Advantage value function is used:

$$A^{\pi_\theta}(s,a) \;=\; Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

- The advantage function can significantly reduce variance of policy gradient

- So the critic should really estimate the advantage function, for instance, estimating both $V(s)$ and Q using two function approximators and two parameter vectors:

$$\begin{aligned} V^{\pi_\theta}(s) &\approx V^{v}(s) \\ Q^{\pi_\theta}(s,a) &\approx Q^{w}(s,a) \\ A(s,a) &= Q^{w}(s,a) - V^{v}(s) \end{aligned}$$

- And updating both value functions by e.g. TD learning

Policy Gradient Method with Deterministic Policy

- Deterministic policy parametrization $\mu_\theta : \mathcal{S} \to \mathcal{A}$
- The on-policy objective function:

$$J(\mu_\theta) = \int_{\mathcal{S}} d^\mu(s) Q^\mu(s, \mu_\theta(s)) = \mathbb{E}_{s\sim d^\mu}\left[Q^\mu(s, \mu_\theta(s))\right]$$

- In continuous action spaces, greedy policy improvement becomes problematic, requiring a global maximisation at every step.
- Instead, a simple and computationally attractive alternative is to move the policy in the direction of the gradient of Q, rather than globally maximising Q.
- Specifically, for each visited state s, the policy parameters θ^{k+1} are updated in proportion to the gradient $\nabla_\theta Q^{\mu_k}(s, \mu_\theta(s))$:

$$\begin{aligned}\theta^{k+1} &= \theta^k + \alpha \mathbb{E}_{s\sim d^{\mu^k}}\left[\nabla_\theta Q^{\mu_k}(s, \mu_\theta(s))\right] \\ &= \theta^k + \alpha \mathbb{E}_{s\sim d^{\mu^k}}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu^k}(s,a)\big|_{a=\mu_\theta(s)}\right]\end{aligned}$$

Policy Gradient Method with Deterministic Policy

Theorem (on-policy)

For any differentiable policy μ_θ, we have

$$\nabla_\theta J(\mu_\theta) \; = \; \mathbb{E}_{s\sim d^\mu}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right],$$

where d^{μ_θ} is the discounted state visitation frequency under policy μ_θ.

Theorem (relationship with stochastic policy gradient)

Consider a stochastic policy $\pi_{\mu_\theta,\sigma}$ such that $\pi_{\mu_\theta,\sigma}(a|s) = \nu_\sigma(\mu_\theta(s),a)$ where σ is a parameter controlling the variance. Then under some regularity assumptions over ν_σ and the MDP, we have

$$\lim_{\sigma\to 0^+} \nabla J(\pi_{\mu_\theta,\sigma}) = \nabla J(\mu_\theta).$$

Deterministic Actor-Critic Algorithms

- On-Policy Deterministic Actor-Critic (SARSA updates)

$$\begin{aligned} \delta_t &= r_t + \gamma Q^w(s_{t+1}, a_{t+1}) - Q^w(s_t, a_t) \\ w_{t+1} &= w_t + \alpha_w \delta_t \nabla_w Q^w(s_t, a_t) \\ \theta_{t+1} &= \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) \nabla_a Q^w(s_t, a)|_{a=\mu_\theta(s_t)} \end{aligned}$$

- The off-policy performance objective:

$$J_\beta\left(\mu_\theta\right) = \int_{\mathcal{S}} d^\beta(s) V^\mu\left(s\right) = \int_{\mathcal{S}} d^\beta(s) Q^\mu\left(s, \mu_\theta(s)\right) = \mathbb{E}_{s \sim d^\beta}\left[Q^\mu\left(s, \mu_\theta(s)\right)\right]$$

- The off-policy gradient:

$$\nabla_\theta J_\beta(\mu_\theta) \approx \mathbb{E}_{s \sim d^\beta}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a)|_{a=\mu_\theta(s)}\right]$$

- Off-Policy Deterministic Actor-Critic (Q-learning updates)

$$\begin{aligned} \delta_t &= r_t + \gamma Q^w(s_{t+1}, \mu_\theta(s_{t+1})) - Q^w(s_t, a_t) \\ w_{t+1} &= w_t + \alpha_w \delta_t \nabla_w Q^w(s_t, a_t) \\ \theta_{t+1} &= \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) \nabla_a Q^w(s_t, a_t)|a = \mu_\theta(s) \end{aligned}$$

Compatible Function Approximation

• We find a critic $Q^w(s,a)$ such that the gradient $\nabla_a Q^\mu(s,a)$ can be replaced by $\nabla_a Q^w(s,a)$, without affecting the deterministic policy gradient.

Theorem

A function approximator $Q^w(s,a)$ *is compatible with a deterministic policy* $\mu_\theta(s)$, $\nabla_\theta J_\beta(\theta) = \mathbb{E}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^w(s,a)|_{a=\mu_\theta(s)}\right]$, *if*

1. $\nabla_a Q^w(s,a)|_{a=\mu_\theta(s)} = \nabla_\theta \mu_\theta(s)^\top w$ *and*
2. w *minimises the mean-squared error,* $\mathrm{MSE}(\theta, w) = \mathbb{E}\left[\epsilon(s;\theta,w)^\top \epsilon(s;\theta,w)\right]$ *where* $\epsilon(s;\theta,w) = \nabla_a Q^w(s,a)|_{a=\mu_\theta(s)} - \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}$

• For any deterministic policy $\mu_\theta(s)$, there always exists a compatible function approximator of the form

$$Q^w(s,a) \;=\; (a - \mu_\theta(s))^\top \nabla_\theta \mu_\theta(s)^\top w + V^v(s),$$

where $V^v(s)$ may be any differentiable baseline function that is independent of the action a.

DDPG: Deep Deterministic Policy Gradient [5]

- DDPG is an extension of Q-learning for continuous action spaces.
- Therefore, it is an off-policy algorithm (we can use ER!)
- It is also an actor-critic algorithm (has networks Q_ϕ and π_θ)
- Uses Q and π target networks for stability.
- Differently from other critic algorithms, policy is deterministic
- Noise added for exploration: $a_t = \mu_\theta(s_t) + \xi$, where $\xi \sim \mathcal{N}(0, \sigma I)$
- Q_ϕ network is trained using standard loss function:

$$L(\phi, \mathcal{D}) = \underset{(s,a,r,s')\sim\mathcal{D}}{\mathbb{E}} \left[\left(Q_\phi(s,a) - \left\{ r + \gamma Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s')) \right\} \right)^2 \right]$$

- As action is deterministic and continuous (NN), we can easily follow the gradient in policy network to increase future reward:

$$\nabla_\theta \underset{s\sim\mathcal{D}}{\mathbb{E}} \left[Q_\phi(s, \mu_\theta(s)) \right] \approx \frac{1}{N} \sum \nabla_a Q_\phi(s,a) \nabla_\theta \mu_\theta(s)$$

DDPG algorithm [5]

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

TD3: Twin Delayed DDPG [2]

- DDPG brittle with respect to hyperparameters and other kinds of tuning.
- TD3 is an off-policy algorithm.
- TD3 can only be used for environments with continuous action spaces.
- Similar to DDPG but with the following changes:
  1. *Clipped action exploration or target policy smoothing*: noise added like DDPG but noise bounded to fixed range

$$a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{\text{Low}}, a_{\text{High}}\right), \quad \epsilon \sim \mathcal{N}(0, \sigma)$$

  2. *Pessimistic Double-Q Learning*: It uses two (twin) Q networks and uses the pessimistic one for current state for updating the network

$$L(\phi_i, \mathcal{D}) = \underset{(s,a,r,s')\sim\mathcal{D}}{\mathbb{E}}\left[\left(Q_{\phi_i}(s,a) - \left\{r + \gamma \min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s'))\right\}\right)^2\right]$$

  3. *Delayed Policy Updates*: Updates of Critic are more frequent than of policy (e.g. 2 or 3 times)

TD3 algorithm [2]

Algorithm 1 TD3

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** T **do**
  Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'
  Store transition tuple (s, a, r, s') in $\mathcal{B}$

  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
  Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
  if t mod d **then**
    Update ϕ by the deterministic policy gradient:
    $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    Update target networks:
    $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
  end if
end for

SAC: Soft Actor Critic [3]

- Policy Entropy-regularized: we will look for *maximum entropy* policies with given data (in SAC we go back to stochastic π).

$$\mathcal{H}\left(\pi(\cdot \mid s)\right) = \mathop{\mathbb{E}}_{a \sim \pi(s)} \left[-\log \pi(a \mid s)\right]$$

- So we search for policy:

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \left\{ R(s_{t+1}) + \alpha \mathcal{H}\left(\pi(\cdot \mid s_t)\right) \right\} \right]$$

where α is the trade-off between reward and entropy.

- Entropy enforces exploration, so no need to add noise to actions.

- Usually α decreases during learning and is disabled to test performance.

SAC: Soft Actor Critic [3]

- Let's define value functions in this case:

$$
\begin{aligned}
V^{\pi}(s) &= \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}\left\{R(s_{t+1})+\alpha\mathcal{H}\left(\pi(\cdot\mid s_t)\right)\right\}\Big|s_0=s\right]\\
Q^{\pi}(s,a) &= \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}R(s_{t+1})+\alpha\sum_{t=1}^{\infty}\gamma^{t}\mathcal{H}\left(\pi(\cdot\mid s_t)\right)\Big|s_0=s,a_0=a\right]
\end{aligned}
$$

- So Bellman equations can be written as:

$$
\begin{aligned}
V^{\pi}(s) &= \mathop{\mathbb{E}}_{\tau\sim\pi}\left[Q^{\pi}(s,a)+\alpha\mathcal{H}\left(\pi(\cdot\mid s)\right)\right]\\
Q^{\pi}(s,a) &= \mathop{\mathbb{E}}_{s'\sim P,a'\sim\pi}\left[R(s,a,s')+\gamma\left\{Q^{\pi}(s',a')+\alpha\mathcal{H}\left(\pi(\cdot\mid s')\right)\right\}\right]\\
&= \mathop{\mathbb{E}}_{s'\sim P}\left[R(s,a,s')+\gamma V^{\pi}(s')\right]
\end{aligned}
$$

SAC: Soft Actor Critic [3]

- Architecture: Networks and loss functions for each one:
  - Q-value functions: $Q_{\theta_1}(s,a)$, $Q_{\theta_2}(s,a)$ (twin like TD3)

$$L(\theta_i, \mathcal{D}) = \underset{(s,a,r,s')\sim\mathcal{D}}{\mathbb{E}}\left[\left(Q_{\theta_i}(s,a) - \left\{r + \gamma V_{\psi_{\mathrm{targ}}}(s')\right\}\right)^2\right]$$

  - Value functions $V_\psi(s)$, $V_{\psi_{\mathrm{targ}}}(s)$

$$L(\psi, \mathcal{D}) = \underset{s\sim\mathcal{D}, a\sim\pi_\phi}{\mathbb{E}}\left[\left(V_\psi(s) - \left\{\min_{i=1,2} Q_{\theta_i}(s,a) - \alpha \log \pi_\phi(a \mid s)\right\}\right)^2\right]$$

  - Policy $\pi_\phi(a \mid s)$. Maximize

$$\underset{a\sim\pi}{\mathbb{E}}\left[Q^\pi(s,a) - \alpha \log \pi(a \mid s)\right]$$

  which maximize V value function ... but how to compute gradients?

Reparametrization Trick

- Problematic because in ∇_ϕ, expectation follow stochastic π_ϕ.

$$\mathop{\mathbb{E}}_{a\sim\pi_\phi}\left[Q^{\pi_\phi}(s,a) - \alpha\log\pi_\phi(a\mid s)\right]$$

- Use a reparametrizarion trick. It can be done when we define the stochastic π_ϕ as Gaussian by adding noise to the action:

$$\tilde{a}_\phi(s,\xi) \;=\; \tanh\left(\mu_\phi(s) + \sigma_\phi(s)\odot\xi\right), \quad \xi\sim\mathcal{N}(0,I)$$

- Now we can rewrite the term as:

$$\mathop{\mathbb{E}}_{a\sim\pi_\phi}\left[Q^{\pi_\phi}(s,a) - \alpha\log\pi_\phi(a\mid s)\right] \;=\; \mathop{\mathbb{E}}_{\xi\sim\mathcal{N}}\left[Q^{\pi_\phi}(s,\tilde{a}_\phi(s,\xi)) - \alpha\log\pi_\phi(\tilde{a}_\phi(s,\xi)\mid s)\right]$$

- Now we can optimize the policy according to

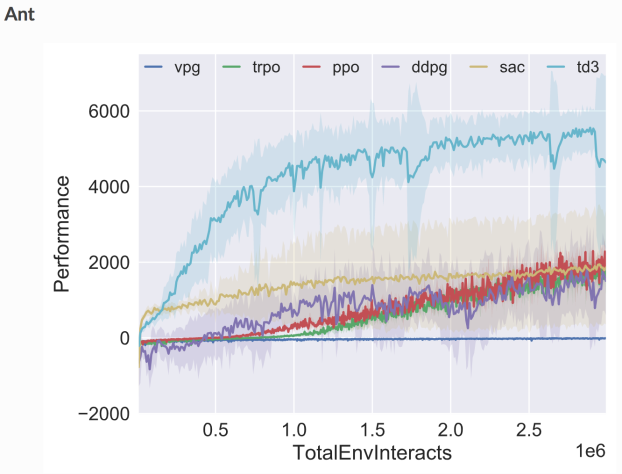
$$\max_\phi \mathop{\mathbb{E}}_{s\sim\mathcal{D},\xi\sim\mathcal{N}}\left[Q_{\theta_1}(s,\tilde{a}_\phi(s,\xi)) - \alpha\log\pi_\phi(\tilde{a}_\phi(s,\xi)\mid s)\right]$$

Soft Actor Critic Algorithm

Algorithm 1 Soft Actor-Critic

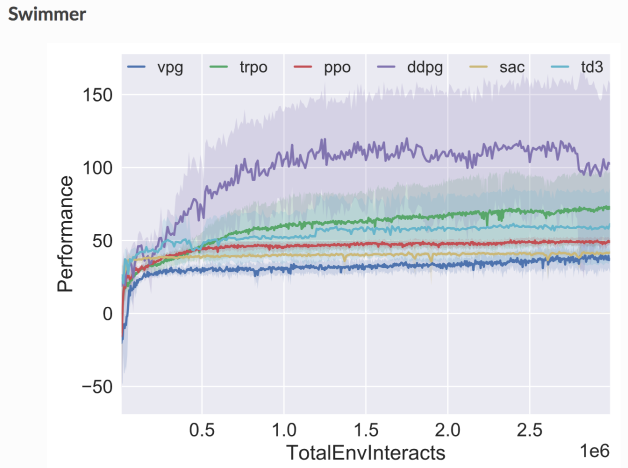
Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
  end for
end for

Performance Comparison



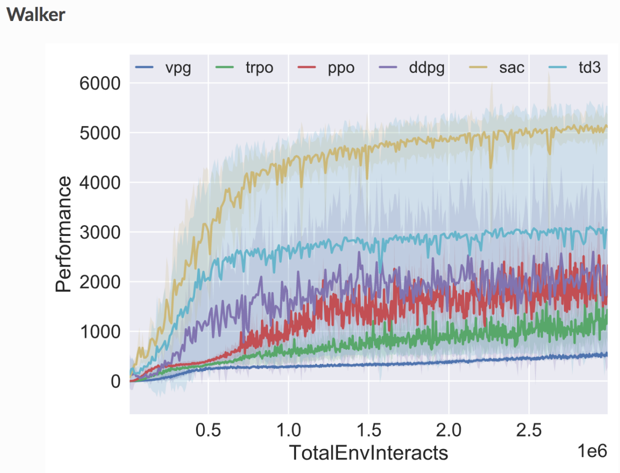
3M timestep benchmark for Ant-v2.

Performance Comparison



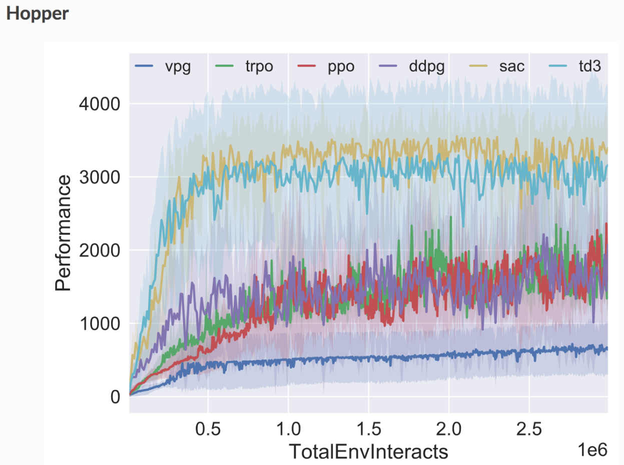
3M timestep benchmark for Swimmer-v2.

Performance Comparison



3M timestep benchmark for Walker2d-v2.

Performance Comparison



3M timestep benchmark for Hopper-v2.

Deep RL Algorithms

- https://github.com/openai/spinningup
- https://www.cs.upc.edu/ mmartin/URL/MindmapRLAlgorithms.pdf

References

[1] Thomas Degris, Martha White, and Richard S Sutton.
Off-policy actor-critic.
arXiv preprint arXiv:1205.4839, 2012.

[2] Scott Fujimoto, Herke Hoof, and David Meger.
Addressing function approximation error in actor-critic methods.
In *International Conference on Machine Learning*, pages 1582–1591, 2018.

[3] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine.
Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.
In *International Conference on Machine Learning*, pages 1856–1865, 2018.

[4] Vijay R Konda and John N Tsitsiklis.
On actor-critic algorithms.
SIAM journal on Control and Optimization, 42(4):1143–1166, 2003.

[5] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra.
Continuous control with deep reinforcement learning.
arXiv preprint arXiv:1509.02971, 2015.

[6] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour.
Policy gradient methods for reinforcement learning with function approximation.
In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[7] Ronald J Williams.
Simple statistical gradient-following algorithms for connectionist reinforcement learning.
Machine learning, 8(3-4):229–256, 1992.