

Model Predictive Control and Reinforcement Learning

– Lecture 06: Actor-Critic Methods –

Joschka Boedecker

University of Freiburg

School on Model Predictive Control and Reinforcement Learning
Freiburg, 7-16 October 2026

universität freiburg



NTNU

Norwegian University of
Science and Technology



- ▶ Up to this point, we represented a model or a value function by some parameterized function approximator and extracted the policy implicitly
- ▶ Now, we are going to talk about *Policy Gradient Methods*: methods which consider a parameterized *policy*

$$\pi_{\theta}(a \mid s) = \Pr\{A_t = a \mid S_t = s, \theta_t = \theta\},$$

with parameters θ

- ▶ Policy Gradient Methods are able to represent stochastic policies and scale naturally to very large or continuous action spaces

Preliminaries

- Policy Gradient

- REINFORCE

- Actor-Critic Methods

Deep Actor-Critic Methods

- Proximal Policy Optimization

- Deep Deterministic Policy Gradient

- Soft Actor-Critic

Wrapup



Preliminaries

- Policy Gradient

- REINFORCE

- Actor-Critic Methods

Deep Actor-Critic Methods

- Proximal Policy Optimization

- Deep Deterministic Policy Gradient

- Soft Actor-Critic

Wrapup



- ▶ Remember, we consider a parameterized *policy*

$$\pi_{\boldsymbol{\theta}}(a \mid s) = \Pr\{A_t = a \mid S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\},$$

with parameters $\boldsymbol{\theta}$

- ▶ We update these parameters based on the gradient of some performance measure $J(\boldsymbol{\theta})$ that we want to maximize, i.e. via *gradient ascent*:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)},$$

where $\widehat{\nabla J(\boldsymbol{\theta}_t)} \in \mathbb{R}^d$ is a stochastic estimate whose expectation approximates the gradient of the performance measure w.r.t. $\boldsymbol{\theta}_t$



Policy Objective Functions:

- We consider episodic problems where we define performance as: $J(\boldsymbol{\theta}) = V^{\pi_{\boldsymbol{\theta}}}(S_0)$

Policy Gradient Theorem

For any differentiable policy $\pi(a|s, \boldsymbol{\theta})$ and any of the above policy objective functions, the policy gradient is:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi}[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A | S) Q^{\pi_{\boldsymbol{\theta}}}(S, A)]$$

Reminder: $V^{\pi_{\boldsymbol{\theta}}} = \sum_a \pi_{\boldsymbol{\theta}}(a|s) Q^{\pi_{\boldsymbol{\theta}}}(s, a)$

Proof (episodic case):

$$\begin{aligned}\nabla_{\theta} V^{\pi_{\theta}}(s) &= \nabla_{\theta} \left[\sum_a \pi_{\theta}(a | s) Q^{\pi_{\theta}}(s, a) \right], \quad \text{for all } s \in \mathcal{S} \\&= \sum_a [\nabla_{\theta} \pi_{\theta}(a | s) Q^{\pi_{\theta}}(s, a) + \pi_{\theta}(a | s) \nabla_{\theta} Q^{\pi_{\theta}}(s, a)] \quad (\text{product rule of calculus}) \\&= \sum_a \left[\nabla_{\theta} \pi_{\theta}(a | s) Q^{\pi_{\theta}}(s, a) + \pi_{\theta}(a | s) \nabla_{\theta} \sum_{s', r} P(s', r | s, a) (r + V^{\pi_{\theta}}(s')) \right] \\&= \sum_a \left[\nabla_{\theta} \pi_{\theta}(a | s) Q^{\pi_{\theta}}(s, a) + \pi_{\theta}(a | s) \sum_{s'} P(s' | s, a) \nabla_{\theta} V^{\pi_{\theta}}(s') \right] \\&= \sum_a \left[\nabla_{\theta} \pi_{\theta}(a | s) Q^{\pi_{\theta}}(s, a) + \pi_{\theta}(a | s) \sum_{s'} P(s' | s, a) \sum_{a'} [\nabla_{\theta} \pi_{\theta}(a' | s') Q^{\pi_{\theta}}(s', a') + \right. \\&\quad \left. \pi_{\theta}(a' | s') \sum_{s''} P(s'' | s', a') \nabla_{\theta} V^{\pi_{\theta}}(s'')] \right] \\&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \rightarrow x, k, \pi_{\theta}) \sum_a \nabla_{\theta} \pi_{\theta}(a | x) Q^{\pi_{\theta}}(x, a)\end{aligned}$$

Proof (episodic case):

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} V^{\pi_{\boldsymbol{\theta}}}(s_0) \\&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \rightarrow s, k, \pi_{\boldsymbol{\theta}}) \right) \sum_a \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) Q^{\pi_{\boldsymbol{\theta}}}(s, a) \\&= \sum_s \eta(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) Q^{\pi_{\boldsymbol{\theta}}}(s, a) \\&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) Q^{\pi_{\boldsymbol{\theta}}}(s, a) \\&= \sum_{s'} \eta(s') \sum_s \rho^{\pi}(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) Q^{\pi_{\boldsymbol{\theta}}}(s, a) \\&\propto \sum_s \rho^{\pi}(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) Q^{\pi_{\boldsymbol{\theta}}}(s, a) \\&\quad (\text{Q.E.D.})\end{aligned}$$

- Likelihood ratios exploit the following identity:

$$\begin{aligned} \overbrace{\nabla_{\theta} \pi_{\theta}(a | s)}^{\text{We want the expectation of this}} &= \pi_{\theta}(a | s) \frac{\nabla_{\theta} \pi_{\theta}(a | s)}{\pi_{\theta}(a | s)} \\ &= \underbrace{\pi_{\theta}(a | s) \nabla_{\theta} \log \pi_{\theta}(a | s)}_{\substack{\text{Easy to take the expectation} \\ \text{because we can sample from } \pi!}} \end{aligned}$$

- $\nabla_{\theta} \log \pi_{\theta}(a | s)$ is called the **score function**

Consider a Gaussian policy, where the mean is a linear combination of state features:
 $\pi_{\boldsymbol{\theta}}(a \mid s) \sim \mathcal{N}(s^{\top} \boldsymbol{\theta}, \sigma^2)$, i.e.

$$\pi_{\boldsymbol{\theta}}(a \mid s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \frac{(s^{\top} \boldsymbol{\theta} - a)^2}{\sigma^2}\right)$$

Exercise (5min)

Derive the score function.

Consider a Gaussian policy, where the mean is a linear combination of state features:
 $\pi_{\boldsymbol{\theta}}(a \mid s) \sim \mathcal{N}(s^{\top} \boldsymbol{\theta}, \sigma^2)$, i.e.

$$\pi_{\boldsymbol{\theta}}(a \mid s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \frac{(s^{\top} \boldsymbol{\theta} - a)^2}{\sigma^2}\right)$$

Solution

The log yields

$$\log \pi_{\boldsymbol{\theta}}(a \mid s) = -\frac{1}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} (s^{\top} \boldsymbol{\theta} - a)^2$$

and the gradient

$$\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a \mid s) = -\frac{1}{2\sigma^2} (s^{\top} \boldsymbol{\theta} - a) 2s = \frac{(a - s^{\top} \boldsymbol{\theta})s}{\sigma^2}.$$



- ▶ REINFORCE: Monte Carlo Policy Gradient
- ▶ Builds upon Monte Carlo returns as an unbiased sample of Q^π
- ▶ However, therefore REINFORCE can suffer from high variance

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

Input: a differentiable policy parameterization $\pi(a|s, \theta)$

Algorithm parameter: step size $\alpha > 0$

Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):

 Generate an episode $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \theta)$

 Loop for each step of the episode $t = 0, 1, \dots, T - 1$:

$$\begin{aligned} G &\leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \\ \theta &\leftarrow \theta + \alpha \gamma^t G \nabla \ln \pi(A_t | S_t, \theta) \end{aligned} \tag{G_t}$$

- ▶ Vanilla REINFORCE provides *unbiased* estimates of the gradient $\nabla J(\theta)$, but it can suffer from high variance
- ▶ Goal: reduce variance while remaining unbiased
- ▶ Observation: we can generalize the policy gradient theorem by including an arbitrary *action-independent baseline* $b(s)$, i.e.

$$\begin{aligned}\nabla_{\theta} J(\theta) &\propto \sum_s \rho^{\pi}(s) \sum_a (Q^{\pi}(s, a) - b(s)) \nabla_{\theta} \pi_{\theta}(a | s) \\ &= \sum_s \rho^{\pi}(s) \left[\sum_a Q^{\pi}(s, a) \nabla_{\theta} \pi_{\theta}(a | s) - b(s) \underbrace{\nabla_{\theta} \sum_a \pi_{\theta}(a | s)}_{=0} \right] \\ &= \sum_s \rho^{\pi}(s) \sum_a Q^{\pi}(s, a) \nabla_{\theta} \pi_{\theta}(a | s)\end{aligned}$$

- ▶ Baselines can reduce the variance of gradient estimates significantly!

- ▶ A constant value can be used as a baseline
- ▶ The state-value function can be used as a baseline

Question

Is the Q-function a valid baseline?

Question

Assume an approximation of the state-value function as a baseline. Is REINFORCE then biased?

Indeed, an estimate of the state value function, $\hat{v}(S_t, \mathbf{w})$, is a very reasonable choice for $b(s)$:

REINFORCE with Baseline (episodic), for estimating $\pi_{\theta} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \theta)$

Input: a differentiable state-value function parameterization $\hat{v}(s, \mathbf{w})$

Algorithm parameters: step sizes $\alpha^{\theta} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):

 Generate an episode $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \theta)$

 Loop for each step of the episode $t = 0, 1, \dots, T-1$:

$$G \leftarrow \sum_{k=t+1}^T \gamma^{k-t-1} R_k \quad (G_t)$$

$$\delta \leftarrow G - \hat{v}(S_t, \mathbf{w})$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t, \mathbf{w})$$

$$\theta \leftarrow \theta + \alpha^{\theta} \gamma^t \delta \nabla \ln \pi(A_t | S_t, \theta)$$

Preliminaries

- Policy Gradient

- REINFORCE

- Actor-Critic Methods

Deep Actor-Critic Methods

- Proximal Policy Optimization

- Deep Deterministic Policy Gradient

- Soft Actor-Critic

Wrapup



- ▶ Methods that learn approximations to both policy and value functions and use bootstrapping are called actor-critic methods
 - actor**: learned policy
 - critic**: learned value function (usually a state-value function)

Question

Is REINFORCE-with-baseline considered as an actor-critic method?



- ▶ REINFORCE-with-baseline is unbiased, but tends to learn slowly and has high variance
- ▶ To gain from advantages of TD methods we use actor-critic methods with a bootstrapping critic

One-step actor-critic methods

Replace the full return of REINFORCE with one-step return as follows:

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha (G_{t:t+1} - \hat{v}(S_t, \boldsymbol{w})) \frac{\nabla \pi(A_t | S_t, \boldsymbol{\theta}_t)}{\pi(A_t | S_t, \boldsymbol{\theta}_t)} \\ &= \boldsymbol{\theta}_t + \alpha (R_{t+1} + \gamma \hat{v}(S_{t+1}, \boldsymbol{w}) - \hat{v}(S_t, \boldsymbol{w})) \frac{\nabla \pi(A_t | S_t, \boldsymbol{\theta}_t)}{\pi(A_t | S_t, \boldsymbol{\theta}_t)} \\ &= \boldsymbol{\theta}_t + \alpha \delta_t \frac{\nabla \pi(A_t | S_t, \boldsymbol{\theta}_t)}{\pi(A_t | S_t, \boldsymbol{\theta}_t)}\end{aligned}$$

One-step Actor-Critic (episodic), for estimating $\pi_{\theta} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \theta)$

Input: a differentiable state-value function parameterization $\hat{v}(s, \mathbf{w})$

Parameters: step sizes $\alpha^{\theta} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):

 Initialize S (first state of episode)

$I \leftarrow 1$

 Loop while S is not terminal (for each time step):

$A \sim \pi(\cdot|S, \theta)$

 Take action A , observe S', R

$\delta \leftarrow R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$ (if S' is terminal, then $\hat{v}(S', \mathbf{w}) \doteq 0$)

$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S, \mathbf{w})$

$\theta \leftarrow \theta + \alpha^{\theta} I \delta \nabla \ln \pi(A|S, \theta)$

$I \leftarrow \gamma I$

$S \leftarrow S'$



Preliminaries

Policy Gradient

REINFORCE

Actor-Critic Methods

Deep Actor-Critic Methods

Proximal Policy Optimization

Deep Deterministic Policy Gradient

Soft Actor-Critic

Wrapup

- ▶ Motivation: how can we take the biggest possible improvement step on a policy using the data we currently have, without stepping so far that we accidentally cause performance collapse?
- ▶ We collect data with $\pi_{\theta_{\text{old}}}$
- ▶ And we want to optimize some objective to get a new policy π_{θ}
- ▶ In PPO, we *ignore* the change in state distribution and optimize a **surrogate objective**:

$$\begin{aligned} J_{\text{old}}(\theta) &= \mathbb{E}_{S \sim \rho^{\pi_{\theta_{\text{old}}}}, A \sim \pi_{\theta}} [\mathcal{A}^{\pi_{\theta_{\text{old}}}}(S, A)] \\ &= \mathbb{E}_{(S, A) \sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_{\theta}}{\pi_{\theta_{\text{old}}}} \mathcal{A}^{\pi_{\theta_{\text{old}}}}(S, A) \right] = \mathbb{E}_{(S, A) \sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_{\theta}}{\pi_{\theta_{\text{old}}}} (q^{\pi_{\theta_{\text{old}}}}(S, A) - v^{\pi_{\theta_{\text{old}}}}(S, A)) \right] \end{aligned}$$

- ▶ Improvement Theory: $\eta(\pi_{\theta}) \geq J_{\text{old}}(\theta) - c \cdot \max_s D_{\text{KL}}(\pi_{\theta_{\text{old}}} || \pi_{\theta})$
- ▶ If we keep the KL-divergence between our old and new policies small, optimizing the surrogate is close to optimizing $\eta(\pi_{\theta})$!

- ▶ Adaptive Penalty Surrogate Objective:

$$\mathbb{E}_{(S,A) \sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_{\theta}}{\pi_{\theta_{\text{old}}}} \mathcal{A}^{\pi_{\theta_{\text{old}}}}(S, A) - \beta D_{\text{KL}}(\pi_{\theta_{\text{old}}} || \pi_{\theta}) \right]$$

- ▶ Clipped Surrogate Objective:

$$\mathbb{E}_{(S,A) \sim \pi_{\theta_{\text{old}}}} \left[\min \left(\frac{\pi_{\theta}}{\pi_{\theta_{\text{old}}}} \mathcal{A}^{\pi_{\theta_{\text{old}}}}(S, A), \text{clip}\left(\frac{\pi_{\theta}}{\pi_{\theta_{\text{old}}}}, 1 - \epsilon, 1 + \epsilon\right) \mathcal{A}^{\pi_{\theta_{\text{old}}}}(S, A) \right) \right]$$

Algorithm 1 PPO-Clip

- 1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
- 4: Compute rewards-to-go $\hat{R}_t$.
- 5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k} .
- 6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

- 7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

- 8: **end for**
-

credits: <https://spinningup.openai.com/en/latest/algorithms/ppo.html>



- ▶ DDPG is an actor-critic method (*Continuous DQN*)
- ▶ Recall the DQN-target: $Y_j = R_j + \gamma \max_a Q_{\mathbf{w}^-}(S_{j+1}, a)$
- ▶ In case of continuous actions, the maximization step is not trivial
- ▶ Therefore, we approximate deterministic actor μ representing the $\arg \max_a Q_{\mathbf{w}}(S_{j+1}, a)$ by a neural network and update its parameters following the

Deterministic Policy Gradient Theorem

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\mu} [\nabla_{\boldsymbol{\theta}} \mu_{\boldsymbol{\theta}}(S) \nabla_a Q_{\mathbf{w}}(S, a)|_{a=\mu(S)}] .$$

- ▶ Exploration by adding Gaussian noise to the output of μ



- ▶ The Q-function is fitted to the adapted TD-target:

$$Y_j = R_j + \gamma Q_{\mathbf{w}^-}(S_{j+1}, \mu_{\boldsymbol{\theta}^-}(S_{j+1}))$$

- ▶ The parameters of the target networks of the actor $\boldsymbol{\theta}^-$ and the critic $\mathbf{w}^-$ are then adjusted with a soft update

$$\mathbf{w}^- \leftarrow (1 - \tau)\mathbf{w}^- + \tau\mathbf{w} \text{ and } \boldsymbol{\theta}^- \leftarrow (1 - \tau)\boldsymbol{\theta}^- + \tau\boldsymbol{\theta}$$

with $\tau \in (0, 1]$

- ▶ DDPG used to be very popular and still is the basis for more SOTA actor-critic algorithms
- ▶ However, it can be quite unstable and sensitive to its hyperparameters

Algorithm 1: DDPG

Initialize replay memory D to capacity N

Initialize critic Q and actor μ with random weights

for episode $i = 1, \dots, M$ **do**

for $t = 1, \dots, T$ **do**

 select action $A_t = \mu(s_t, \theta) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma)$

 Store transition (S_t, A_t, S_{t+1}, R_t) in D

 Sample minibatch of transitions (S_j, A_j, R_j, S_{j+1}) from D

 Set $y_j = \begin{cases} R_j & \text{if } S_{j+1} \text{ is terminal} \\ R_j + \gamma Q(S_{j+1}, \mu(S_{j+1}, \theta^-), \mathbf{w}^-) & \text{else} \end{cases}$

 Update the parameters of Q according to the TD-error

 Update the parameters of μ according to:

$$\nabla_{\theta} J \approx \frac{1}{N} \sum_j \nabla_a Q_{\mathbf{w}}(S_j, a)|_{a=\mu(S_j)} \nabla_{\theta} \mu_{\theta}(S_j)$$

 Adjust the parameters of the target networks via a soft update

- ▶ Soft Actor-Critic: entropy-regularized value-learning
- ▶ The policy is trained to maximize a trade-off between expected return and entropy ($H(p) = \mathbb{E}_{x \sim p}[-\log p(x)]$), a measure of randomness in the policy:

$$\pi_* = \arg \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} R_{t+1} + \alpha H(\pi(\cdot | S_t = s_t)) \right]$$

- ▶ The value functions are then defined as:

$$V^{\pi}(s) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} R_{t+1} + \alpha H(\pi(\cdot | S_t = s_t)) | S_0 = s \right]$$

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} R_{t+1} + \alpha \sum_{t=1}^{\infty} \gamma^t H(\pi(\cdot | S_t = s_t)) | S_0 = s, A_0 = a \right]$$

- ▶ And their relation as: $V^{\pi}(s) = \mathbb{E}_{\pi} [Q^{\pi}(s, a)] + \alpha H(\pi(\cdot | S_t = s))$

- ▶ The corresponding Bellman equation for Q^π is

$$\begin{aligned} Q^\pi(s_t, a_t) &= \mathbb{E}_\pi \left[R_{t+1} + \gamma (Q^\pi(S_{t+1}, A_{t+1}) + \alpha H(\pi(\cdot | S_{t+1}))) \right] \\ &= \mathbb{E}_\pi [R_{t+1} + \gamma V^\pi(S_{t+1})]. \end{aligned}$$

- ▶ Loss for the Q-networks:

$$L(\mathbf{w}_i, \mathcal{D}) = \mathbb{E}_{(S, A, R, S') \sim \mathcal{D}} \left[\left(Q_{\mathbf{w}_i}(S, A) - y(R, S') \right)^2 \right]$$

where the target is:

$$y(r, s') = r + \gamma \left(\min_{j=1,2} Q_{\mathbf{w}_j^-}(s', \tilde{A}') - \alpha \log \pi_\theta(\tilde{A}' | s') \right), \quad \tilde{A}' \sim \pi_\theta(\cdot | s')$$

- ▶ We want to find a policy which maximizes expected future return and expected future entropy, i.e. which maximizes $V^\pi(s)$:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_{A \sim \pi(\cdot | s)} [Q^\pi(s, A)] + \alpha H(\pi(\cdot | s)) \\ &= \mathbb{E}_{A \sim \pi(\cdot | s)} [Q^\pi(s, A) - \alpha \log \pi(A | s)] \end{aligned}$$

- ▶ To optimize the policy despite the sampling of actions, we make use of the *reparameterization trick*:

$$\tilde{A}_\theta(s, \xi) = \tanh(\mu_\theta(s) + \sigma_\theta(s) \odot \xi), \quad \xi \sim \mathcal{N}(0, I)$$

- ▶ We can thus rewrite the expectation from above as:

$$\mathbb{E}_{A \sim \pi_\theta} [Q^{\pi_\theta}(s, A) - \alpha \log \pi_\theta(A | s)] = \mathbb{E}_{\xi \sim \mathcal{N}(0, I)} \left[Q^{\pi_\theta}(s, \tilde{A}_\theta(s, \xi)) - \alpha \log \pi_\theta(\tilde{A}_\theta(s, \xi) | s) \right]$$

- ▶ Final policy loss is then:

$$\max_{\theta} \mathbb{E}_{s, \xi} \left[\min_{j=1,2} Q_{\mathbf{w}_j^-}(s, \tilde{A}_\theta(s, \xi)) - \alpha \log \pi_\theta(\tilde{A}_\theta(s, \xi) | s) \right]$$

Algorithm 1 Soft Actor-Critic

```

1: Input: initial policy parameters  $\theta$ , Q-function parameters  $\phi_1, \phi_2$ , empty replay buffer  $\mathcal{D}$ 
2: Set target parameters equal to main parameters  $\phi_{\text{targ},1} \leftarrow \phi_1, \phi_{\text{targ},2} \leftarrow \phi_2$ 
3: repeat
4:   Observe state  $s$  and select action  $a \sim \pi_\theta(\cdot|s)$ 
5:   Execute  $a$  in the environment
6:   Observe next state  $s'$ , reward  $r$ , and done signal  $d$  to indicate whether  $s'$  is terminal
7:   Store  $(s, a, r, s', d)$  in replay buffer  $\mathcal{D}$ 
8:   If  $s'$  is terminal, reset environment state.
9:   if it's time to update then
10:    for  $j$  in range(however many updates) do
11:      Randomly sample a batch of transitions,  $B = \{(s, a, r, s', d)\}$  from  $\mathcal{D}$ 
12:      Compute targets for the Q functions:
    
```

$$y(r, s', d) = r + \gamma(1 - d) \left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s') \right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s')$$

```

13:    Update Q-functions by one step of gradient descent using
    
```

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_{\phi_i}(s, a) - y(r, s', d))^2 \quad \text{for } i = 1, 2$$

```

14:    Update policy by one step of gradient ascent using
    
```

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s)|s) \right),$$

where $\tilde{a}_\theta(s)$ is a sample from $\pi_\theta(\cdot|s)$ which is differentiable wrt θ via the reparametrization trick.

```

15:    Update target networks with
    
```

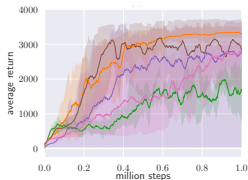
$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho) \phi_i \quad \text{for } i = 1, 2$$

```

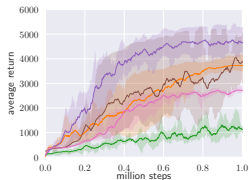
16:  end for
17: end if
18: until convergence
    
```

credits: <https://spinningup.openai.com/en/latest/algorithms/sac.html>

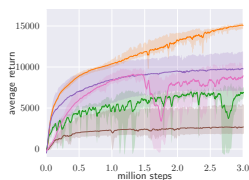
► Performance comparison from (Haarnoja et al., 2018):



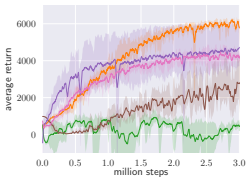
(a) Hopper-v1



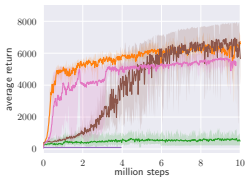
(b) Walker2d-v1



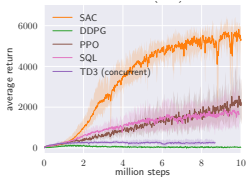
(c) HalfCheetah-v1



(d) Ant-v1



(e) Humanoid-v1



(f) Humanoid (rllab)



Having heard this lecture, you can now...

- ▶ understand policy gradient methods and derive the policy gradient theorem
- ▶ design and implement actor-critic methods that combine policy and value function learning
- ▶ apply state-of-the-art algorithms (PPO, DDPG, SAC) to continuous control problems

If you want to get an even more detailed overview about the current SOTA, you can have a look at `Stable Baselines3`, which is a good start for training your own RL agents:

`https://github.com/DLR-RM/stable-baselines3`