Reinforcement Learning

Practical #08

SAPIENZA
UNIVERSITÀ DI ROMA

Exercise

Today we will implement the Soft Actor Critic (SAC) algorithm[1].

We will test the implementation on the Pendulum environment.

[1] https://arxiv.org/pdf/1801.01290.pdf

Why SAC

- **On-policy** algorithms (TRPO, PPO, A2C) can be very **sample inefficient**, because they need **completely new samples** after each policy update. In contrast, Q-learning based off-policy methods are able to learn efficiently from past samples using **experience replay buffers**
- SAC optimizes a **stochastic policy** in an **off-policy** way

SAC

- SAC is an **off-policy** algorithm
- The original version (that we will implement) can only be used for environments with **continuous action spaces**, there is an alternate version that can handle discrete action spaces
- A central feature of SAC is **entropy regularization.** The policy is trained to maximize a trade-off between expected return and entropy.

Entropy maximization

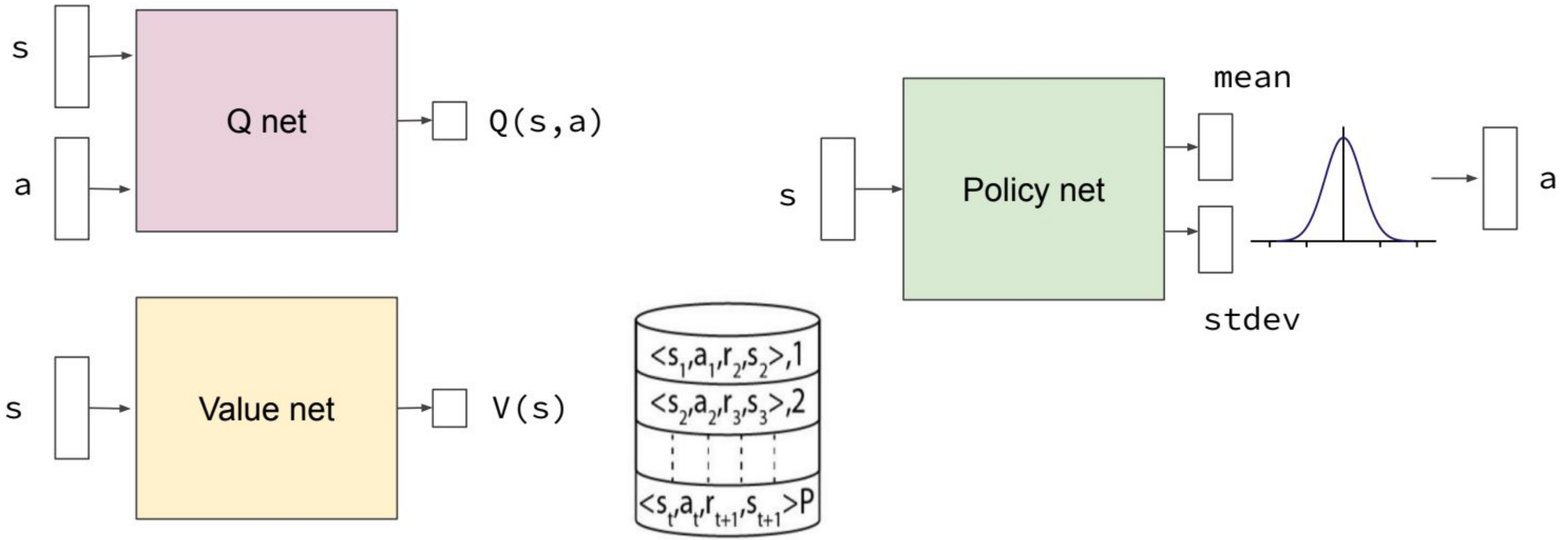
Instead of only maximizing the lifetime rewards, SAC seeks to also maximize the entropy of the policy.

High entropy in our policy encourage **exploration**.

Avoid collapse into repeatedly selecting a particular action that could exploit some inconsistency in the approximated Q function.

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\,\cdot\,|\mathbf{s}_t)) \right]. \quad (1)$$

SAC elements



SAC 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

Loss function V network

We update the Value function weights by minimizing

$$J_V(\psi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}}\left[\frac{1}{2}\left(V_\psi(\mathbf{s}_t) - \mathbb{E}_{\mathbf{a}_t \sim \pi_\phi}\left[Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right]\right)^2\right] \quad (5)$$

entropy

SAPIENZA
UNIVERSITÀ DI ROMA

Loss function Q network

We update the Q function weights by minimizing

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(\mathbf{s}_t,\mathbf{a}_t) - \hat{Q}(\mathbf{s}_t,\mathbf{a}_t)\right)^2\right]$$

where

$$\hat{Q}(\mathbf{s}_t,\mathbf{a}_t) = r(\mathbf{s}_t,\mathbf{a}_t) + \gamma\,\mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[V_{\bar{\psi}}(\mathbf{s}_{t+1})\right]$$ if s is not terminal

$$\hat{Q}(\mathbf{s}_t,\mathbf{a}_t) = r(\mathbf{s}_t,\mathbf{a}_t)$$ if s is terminal

Loss function policy network

We update the policy function weights by minimizing

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\mathrm{D}_{\mathrm{KL}} \left(\pi_\phi(\,\cdot\,|\mathbf{s}_t) \,\middle\|\, \frac{\exp\left(Q_\theta(\mathbf{s}_t,\,\cdot\,)\right)}{Z_\theta(\mathbf{s}_t)} \right) \right]$$

make the distribution of the Policy function look more like the distribution of the exponentiation of our Q Function normalized by another function Z

Reparameterization trick

Reparametrization trick is used to make sure the sampling from the policy is differentiable, so there are no problems in backpropagating the errors

$$\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t)$$

We parametrize the action in this way, where epsilon is a noise vector sampled from a Gaussian distribution

Loss function Policy network with reparametrization trick

The loss function for the network becomes

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

Reparametrization trick