DDA4230 Reinforcement learning Interconnections between policy and value

Lecture 19

Lecturer: Guiliang Liu *Scribe: Baoxiang Wang*

1 Goal of this lecture

In this lecture we revisit policy-based and value-based methods. As we have discussed in the beginning of discrete MDPs, the two types of methods given known model are in a primal-dual relationship. We seek analogous of such interconnections in RL with continuous spaces and deep RL.

Suggested reading: Reference in the lecture notes.

2 Recap: Policy gradient and Q-learning

In policy gradient, we can sample multiple trajectories following the policy π and use the empirical mean to estimate the gradient

$$\nabla_\phi J(\phi) = \mathbb{E}_\pi[Q^\pi(s,a)\nabla_\phi \log \pi_\phi(a \mid s)] \,.$$

Algorithm 1: REINFORCE with baseline

Initialize the policy parameter ϕ and θ at random.
for *each episode* **do**
  Sample one trajectory under policy π_ϕ: $s_0, a_0, r_0, s_1, a_1, r_1 \ldots, s_T$
  for *each* $t = 1, 2, \ldots, T$ **do**
    $G_t \leftarrow \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$
    $\delta \leftarrow G_t - \hat{V}(s_t, \theta)$
    $\theta \leftarrow \theta + \alpha_\theta \delta \nabla_\theta \hat{V}(s_t, \theta)$
    $\phi \leftarrow \phi + \alpha_\phi \gamma^t \delta \nabla_\phi \log \pi_\phi(a_t \mid s_t)$

In Q-learning, we use stochastic gradient descent to find a local minimum of the squared error by sampling the gradients with respect to the parameters θ and updating θ as

$$\Delta\theta = -\frac{1}{2}\alpha_\theta \nabla_\theta J(\theta) = \alpha_\theta \mathbb{E}[(Q(s,a) - \hat{Q}(s,a,\theta))\nabla_\theta \hat{Q}(s,a,\theta)] \,.$$

3 Stein's identity

Stein's identity is a commonly used lemma that describes the property of the score function. Under the context of reinforcement learning, it states that given a policy $\pi(a \mid s)$,

$$\mathbb{E}_\pi[\nabla_a \log \pi(a \mid s)\phi(s,a) + \nabla_a \phi(s,a)] = 0$$

Algorithm 2: Deep Q-learning

Initialize replay memory D with a fixed capacity
Initialize action value function $\hat{Q}$ with random weights θ
Initialize target action value function $\hat{Q}$ with weights $\theta^- = \theta$
for *episode* $k = 1, \ldots, K$ **do**
  Observe initial frame x_1 and pre-process frame to get state s_1
  for *time step* $t = 1, \ldots, T$ **do**
    Select action $a_t = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg\max_a \hat{Q}(s_t, a, \theta) & \text{otherwise} \end{cases}$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Pre-process s_t, x_{t+1} to get s_{t+1}, and store transition (s_t, a_t, r_t, s_{t+1}) in D
    Sample uniformly a random minibatch of N transitions $\{(s_j, a_j, r_j, s_{j+1})\}_{j \in [N]}$ from D
    Set $y_j = r_j$ if episode ends at step $j+1$, otherwise set $y_j = r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a', \theta^-)$
    Perform a stochastic gradient descent step on $J(\theta) = \frac{1}{N} \sum_{j=1}^{N} (y_j - \hat{Q}(s_j, a_j, \theta))^2$ with respect to θ
    Every C steps reset $\theta^- = \theta$

holds for an arbitrary state-action function $\phi(s, a)$. An example is that when we set $\phi(s, a) = b(s)$ that does not depend on a, the identity is reduced to $\mathbb{E}_\pi[\nabla_a \log \pi(a \mid s) b(s)] = 0$, which guarantees that the baseline in policy gradient is unbiased. One could also let $\phi(s, a) = Q(s, a)$, such that

$$\mathbb{E}_\pi[\nabla_a \log \pi(a \mid s) Q(s, a) + \nabla_a Q(s, a)] = 0 \,.$$

Without loss of generality assume that the policy $\pi \sim \mathcal{N}(\mu, \sigma)$ follows a Gaussian distribution. A reparametrization of π is to isolate the stochasticity in π into a standard normal random variable $\xi \sim \mathcal{N}(0, 1)$, such that $a = f_\theta(s, \xi)$ and therefore

$$\pi(a \mid s) = \int f_\theta(s, \xi) \mathbb{P}(\xi) d\xi \,.$$

With Stein's identity, we could show that for an arbitrary ϕ

$$\mathbb{E}[\nabla_\theta \log \pi(a \mid s) \phi(s, a)] = \mathbb{E}[\nabla_\theta f_\theta(s, \xi) \nabla_a \phi(s, a)] \,.$$

Therefore, the policy gradient estimator

$$\mathbb{E}[\nabla_\theta \log \pi(a \mid s) Q(s, a)] = \mathbb{E}[\nabla_\theta f_\theta(s, \xi) \nabla_a Q(s, a)] \,,$$

which indicates that the right hand side term is also an estimate of the policy gradient. When f does not depend on ξ, this estimator is known as the *deep deterministic policy gradient*.

4 Actor–critic methods

In practice, most of the variance is from the Monte-Carlo estimation G_t of $Q(s_t, a_t)$. An estimation with much lower variance can be obtained by estimating a parametrized $Q(s, a)$ and bootstrap the estimation into the policy gradient. This results in a biased estimator but with much lower variance. One way to estimate the value function is the temporal-difference method, which has been discussed in previous lectures. With this bootstrap, the methods is called actor-critic.

Actor-critic methods consist of two models.

- The critic updates the value function parameters $\boldsymbol{w}$.
- The actor updates the policy parameters $\boldsymbol{\theta}$ in the direction suggested by the critic.

Note that although the REINFORCE with baseline method learns both a policy and a state value function, we do not consider it to be an actor–critic method because its state value function is used only as a baseline instead of a critic.

One-step actor–critic methods replace the full return of REINFORCE with the one-step return and use a learned state value function as the baseline, as

$$\begin{aligned}\boldsymbol{\theta_{t+1}} &= \boldsymbol{\theta_t} + \alpha_{\boldsymbol{\theta}}(G_t - \hat{V}(s_t, \boldsymbol{w}))\nabla \log \pi_\theta(a_t \mid s_t) \\ &= \boldsymbol{\theta_t} + \alpha_{\boldsymbol{\theta}}(r_t + \gamma \hat{V}(s_{t+1}, \boldsymbol{w}) - \hat{V}(s_t, \boldsymbol{w}))\nabla \log \pi_\theta(a_t \mid s_t)\,.\end{aligned}$$

This algorithm then takes two inputs: a differentiable policy parametrized by $\pi_\theta(a \mid s)$ and a differentiable state value function parametrized by $\hat{V}(s, \boldsymbol{w})$.

Algorithm 3: One-step actor–critic (episodic)

Initialize the policy parameter $\boldsymbol{\theta}$ and $\boldsymbol{w}$ at random. **for** *each episode* **do**
  Initialize s_0, the first state of each episode
  for *each* $t = 0, 1, \ldots, T-1$ **do**
    sample $a \sim \pi(a \mid s_t, \boldsymbol{\theta})$
    take action a and observe s', r
    $\delta \leftarrow r + \gamma \hat{V}(s', \boldsymbol{w}) - \hat{V}(s, \boldsymbol{w})$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha_{\boldsymbol{w}} \delta \nabla_{\boldsymbol{w}} \hat{V}(s, \boldsymbol{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \delta \nabla_{\boldsymbol{\theta}} \log \pi(a \mid s, \boldsymbol{\theta})$
    $s' \leftarrow s$

5 Soft actor-critic

Model-free deep RL methods are notoriously expensive in terms of their sample complexity. Even relatively simple tasks can require millions of steps of data collection, and complex behaviors with high-dimensional observations might need substantially more. Meanwhile, these methods are often brittle with respect to their hyperparameters: learning rates, exploration constants, and other settings must be set carefully for different problem settings to achieve good results. One cause for the poor sample efficiency of deep RL methods is on-policy learning: some of the most commonly used deep RL algorithms, such as

TRPO [SLA+15], PPO [SWD+17] or A3C [MBM+16], require new samples to be collected for each gradient step. This quickly becomes extravagantly expensive, as the number of gradient steps and samples per step needed to learn an effective policy increases with task complexity. Off-policy algorithms aim to reuse past experience. This is not directly feasible with conventional policy gradient formulations, but is relatively straightforward for Q-learning based methods [MKS+15]. Unfortunately, the combination of off-policy learning and high-dimensional, nonlinear function approximation with neural networks presents a major challenge for stability and convergence [MSB+09]. This challenge is further exacerbated in continuous state and action spaces, where a separate actor network is often used to perform the maximization in Q-learning. A commonly used algorithm in such settings, deep deterministic policy gradient (DDPG) [LHP+16], provides for sample-efficient learning but is notoriously challenging to use due to its extreme brittleness and hyperparameter sensitivity [DCH+16, HIB+18].

In this section, we devise an off-policy maximum entropy actor-critic algorithm, called soft actor-critic (SAC) [HZH+18], which provides for both sample-efficient learning and stability. This algorithm extends readily to very complex, high-dimensional tasks, such as the Humanoid benchmark [DCH+16] with 21 action dimensions, where off-policy methods such as DDPG typically struggle to obtain good results. SAC also avoids the complexity and potential instability associated with approximate inference in prior off-policy maximum entropy algorithms based on soft Q-learning [HTAL17]. We discuss a convergence proof for policy iteration in the maximum entropy framework, and then introduce the SAC algorithm based on an approximation to this procedure that can be practically implemented with deep neural networks.

5.1 Soft policy iteration

Standard RL maximizes the expected sum of rewards $\sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[\mathcal{R}(s_t,a_t)]$. We will consider a more general maximum entropy objective, which favors stochastic policies by augmenting the objective with the expected entropy of the policy over $\rho_\pi(s_t)$, as

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[\mathcal{R}(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot \mid s_t))]\,. \tag{1}$$

The temperature parameter α determines the relative importance of the entropy term against the reward, and thus controls the stochasticity of the optimal policy. The maximum entropy objective differs from the standard maximum expected reward objective used in conventional reinforcement learning, though the conventional objective can be recovered in the limit as $\alpha \to 0$. For the rest of this lecture notes, we will omit writing the temperature explicitly, as it can always be subsumed into the reward by scaling it by α^{-1}.

We will begin by deriving soft policy iteration, a general algorithm for learning optimal maximum entropy policies that alternates between policy evaluation and policy improvement in the maximum entropy framework. Our derivation is based on a tabular setting, to enable theoretical analysis and convergence guarantees, and we extend this method into the general continuous setting in the next section. We will show that soft policy iteration converges to the optimal policy within a set of policies which might correspond, for instance, to a set of parameterized densities.

In the policy evaluation step of soft policy iteration, we wish to compute the value of a policy π according to the maximum entropy objective in Equation (1). For a fixed policy, the soft Q-value can be computed iteratively, starting from any function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and repeatedly applying a modified Bellman backup operator $\mathcal{B}^\pi$ given by

$$\mathcal{B}^\pi Q(s_t, a_t) = \mathcal{R}(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathbb{P}}[V(s_{t+1})], \tag{2}$$

where

$$V(s_t) = \mathbb{E}_{a_t \sim \pi}[Q(s_t, a_t) - \log \pi(a_t \mid s_t)] \tag{3}$$

is the soft state value function. We can obtain the soft value function for any policy π by repeatedly applying $\mathcal{B}^\pi$ as formalized below.

Lemma 1 (Soft policy evaluation) *Consider the soft Bellman backup operator $\mathcal{B}^\pi$ in Equation (2) and a mapping $Q^0 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define $Q^{k+1} = \mathcal{B}^\pi Q^k$. Then the sequence Q^k will converge to the soft Q-value of π as $k \to \infty$.*

Proof: Define the entropy augmented reward as $r_\pi(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim \mathbb{P}}[\mathcal{H}(\pi(\cdot \mid \mathbb{P}))]$ and rewrite the update rule as

$$Q(s_t, a_t) \leftarrow r_\pi(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathbb{P}, a_{t+1} \sim \pi}[Q(s_{t+1}, a_{t+1})]$$

and apply the standard convergence results for policy evaluation [SB18]. The assumption $|\mathcal{A}| < \infty$ is required to guarantee that the entropy augmented reward is bounded. □

In the policy improvement step, the policy is updated towards the exponential of the new Q-function. This particular choice of update can be guaranteed to result in an improved policy in terms of its soft value. To account for the constraint that $\pi \in \Pi$, the improved policy is projected into such a desired set of policies. While in principle one could choose any projection, it will turn out to be convenient to use the information projection defined in terms of the Kullback-Leibler divergence. In the other words, in the policy improvement step, for each state, we update the policy according to

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} d_{\text{KL}}\Big(\pi'(\cdot \mid s_t) \parallel \frac{\exp\left(Q^{\pi_{\text{old}}}(s_t, \cdot)\right)}{Z^{\pi_{\text{old}}}(s_t)}\Big). \tag{4}$$

The partition function $Z^{\pi_{\text{old}}}(s_t)$ normalizes the distribution, and while it is intractable in general, it does not contribute to the gradient with respect to the new policy and can thus be ignored, as noted in the next section. For this projection, we can show that the new, projected policy has a higher value than the old policy with respect to the objective in Equation (1). We formalize this result in Lemma 2.

Lemma 2 (Soft policy improvement) *Let $\pi_{\text{old}} \in \Pi$ and let π_{new} be the optimum of the minimization problem defined in Equation (4). Then, $Q^{\pi_{\text{new}}}(s_t, a_t) \geq Q^{\pi_{\text{old}}}(s_t, a_t)$ for all $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$ when $|\mathcal{A}| < \infty$.*

Proof: Let $\pi_{\text{old}} \in \Pi$ and let $Q^{\pi_{\text{old}}}$ and $V^{\pi_{\text{old}}}$ be the corresponding soft action value function and soft state value function, and let π_{new} be defined as

$$\begin{aligned}\pi_{\text{new}}(\cdot \mid s_t) &= \arg\min_{\pi' \in \Pi} d_{\text{KL}}\Big(\pi'(\cdot \mid s_t) \parallel \exp\left(Q^{\pi_{\text{old}}}(s_t, \cdot) - \log Z^{\pi_{\text{old}}}(s_t)\right)\Big) \\ &= \arg\min_{\pi' \in \Pi} J_{\pi_{\text{old}}}(\pi'(\cdot \mid s_t))\,,\end{aligned}$$

where the second equation is our definition of J. It must be the case that $J_{\pi_{\text{old}}}(\pi_{\text{new}}(\cdot \mid s_t)) \leq J_{\pi_{\text{old}}}(\pi_{\text{old}}(\cdot \mid s_t))$, since we can always choose $\pi_{\text{new}} = \pi_{\text{old}} \in \Pi$. Hence

$$\mathbb{E}_{a_t \sim \pi_{\text{new}}}[\log \pi_{\text{new}}(a_t \mid s_t) - Q^{\pi_{\text{old}}}(s_t, a_t) + \log Z^{\pi_{\text{old}}}(s_t)] \leq \mathbb{E}_{a_t \sim \pi_{\text{old}}}[\log \pi_{\text{old}}(a_t \mid s_t) - Q^{\pi_{\text{old}}}(s_t, a_t) + \log Z^{\pi_{\text{old}}}(s_t)]\,.$$

Since the partition function $Z^{\pi_{\text{old}}}$ depends only on the state, the inequality reduces to

$$\mathbb{E}_{a_t \sim \pi_{\text{new}}}[Q^{\pi_{\text{old}}}(s_t, a_t) - \log \pi_{\text{new}}(a_t \mid s_t)] \geq V^{\pi_{\text{old}}}(s_t)\,. \tag{5}$$

Next, consider the soft Bellman equation

$$\begin{aligned}Q^{\pi_{\text{old}}}(s_t, a_t) &= r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathbb{P}}[V^{\pi_{\text{old}}}(s_{t+1})] \\ &\leq r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathbb{P}}[\mathbb{E}_{a_{t+1} \sim \pi_{\text{new}}}[Q^{\pi_{\text{old}}}(s_{t+1}, a_{t+1}) - \log \pi_{\text{new}}(a_{t+1} \mid s_{t+1})]] \\ &\vdots \\ &\leq Q^{\pi}_{\text{new}}(s_t, a_t)\,,\end{aligned}$$

where we have repeatedly expanded $Q^{\pi_{\text{old}}}$ on the RHS by applying the soft Bellman equation and the bound in Equation (5). Convergence to $Q^{\pi_{\text{new}}}$ follows from Lemma 1. □

The full soft policy iteration algorithm alternates between the soft policy evaluation and the soft policy improvement steps, and it will provably converge to the optimal maximum entropy policy among the policies in Π, as shown in the below lemma.

Lemma 3 (Soft policy iteration) *Repeated application of soft policy evaluation and soft policy improvement from any $\pi \in \Pi$ converges to a policy π^* such that $Q^{\pi^*}(s_t, a_t) \geq Q^{\pi}(s_t, a_t)$ for all $\pi \in \Pi$ and $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, assuming that $|\mathcal{A}| < \infty$.*

Proof: Let π_i be the policy at iteration i. By Lemma 2, the sequence Q^{π_i} is monotonically increasing. Since Q^{π} is bounded above for $\pi \in \Pi$ (both the reward and entropy are bounded), the sequence converges to some π^*. We will still need to show that π^* is indeed optimal. At convergence, it must be the case that $J_{\pi^*}(\pi^*(\cdot \mid s_t)) < J_{\pi^*}(\pi(\cdot \mid s_t))$ for all $\pi \in \Pi$, $\pi \neq \pi^*$. Using the same iterative argument as in the proof of Lemma 2, we get $Q^{\pi^*}(s_t, a_t) > Q^{\pi}(s_t, a_t)$ for all $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, that is, the soft value of any other policy in Π is lower than that of the converged policy. Hence π^* is optimal in Π. □

Although this algorithm will provably find the optimal solution, we can perform it in its exact form only in the tabular case. Therefore, we will next approximate the algorithm for continuous domains, where we need to rely on a function approximator to represent the Q-values, and running the two steps until convergence would be computationally too expensive. The approximation gives rise to a new practical algorithm, called soft actor-critic.

5.2 SAC algorithm for deep RL

As discussed above, large continuous domains require us to derive a practical approximation to soft policy iteration. To that end, we will use function approximators for both the Q-function and the policy, and instead of running evaluation and improvement to convergence, alternate between optimizing both networks with stochastic gradient descent. We will consider a parameterized state value function $V_\psi(s_t)$, soft Q-function $Q_\theta(s_t, a_t)$, and a tractable policy $\pi_\phi(a_t \mid s_t)$. The parameters of these networks are ψ, θ, and ϕ. For example, the value functions can be modeled as expressive neural networks, and the policy as a Gaussian with mean and covariance given by neural networks. We will next derive update rules for these parameter vectors.

The state value function approximates the soft value. There is no need in principle to include a separate function approximator for the state value, since it is related to the Q-function and the policy according to Equation (3). This quantity can be estimated from a single action sample from the current policy without introducing a bias, but in practice, including a separate function approximator for the soft value can stabilize training and is convenient to train simultaneously with the other networks.

Update $V_\psi(s)$ The soft value function is trained to minimize the squared residual error

$$J_V(\psi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\frac{1}{2} \left(V_\psi(s_t) - \mathbb{E}_{a_t \sim \pi_\phi} [Q_\theta(s_t, a_t) - \log \pi_\phi(a_t \mid s_t)] \right)^2 \right] , \tag{6}$$

where $\mathcal{D}$ is the distribution of previously sampled states and actions, or a replay buffer. The gradient of Equation (6) can be estimated with an unbiased estimator

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(s_t) \left(V_\psi(s_t) - Q_\theta(s_t, a_t) + \log \pi_\phi(a_t \mid s_t) \right) , \tag{7}$$

where the actions are sampled according to the current policy instead of the replay buffer and the action value $Q_\theta(s_t, a_t)$ can be replaced by its Monte-Carlo sample.

Update $Q_\theta(s, a)$ The soft Q-function parameters can be trained to minimize the soft Bellman residual

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

where

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathbb{P}} [V_{\bar{\psi}}(s_{t+1})] ,$$

which again can be optimized with stochastic gradients

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(a_t, s_t) \left(Q_\theta(s_t, a_t) - r(s_t, a_t) - \gamma V_{\bar{\psi}}(s_{t+1}) \right) .$$

The update makes use of a target value network $V_{\bar{\psi}}$, where $\bar{\psi}$ can be an exponentially moving average of the value network weights, which has been shown to stabilize training [MKS+15]. Alternatively, we can update the target weights to match the current value function weights periodically.

Update $\pi_\phi(a \mid s)$ Finally, the policy parameters can be learned by directly minimizing the expected KL-divergence in Equation (4)

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\left(\pi_\phi(\cdot \mid s_t) \parallel \frac{\exp\left(Q_\theta(s_t, \cdot)\right)}{Z_\theta(s_t)} \right) \right] . \tag{8}$$

There are several options for minimizing J_π. A typical solution for policy gradient methods is to use the likelihood ratio gradient estimator (REINFORCE [Wil92]), which does not require backpropagating the gradient through the policy and the target density networks. However, in our case, the target density is the Q-function, which is represented by a neural network and can be differentiated. It is thus convenient to apply the reparameterization trick instead, resulting in an estimator with a lower variance. To that end, we reparameterize the policy using a neural network transformation

$$a_t = f_\phi(\epsilon_t; s_t) ,$$

where ϵ_t is an input noise vector, sampled from some fixed distribution, such as a spherical Gaussian. We can now rewrite the objective in Equation (8) as

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}} \left[\log \pi_\phi(f_\phi(\epsilon_t; s_t) \mid s_t) - Q_\theta(s_t, f_\phi(\epsilon_t; s_t)) \right] , \tag{9}$$

where π_ϕ is defined implicitly in terms of f_ϕ. We can approximate the gradient of Equation (9) with

$$\widehat{\nabla_\phi J_\pi(\phi)} = \nabla_\phi \log \pi_\phi(a_t \mid s_t) + (\nabla_{a_t} \log \pi_\phi(a_t \mid s_t) - \nabla_{a_t} Q(s_t, a_t)) \nabla_\phi f_\phi(\epsilon_t; s_t) , \tag{10}$$

where a_t is evaluated at $f_\phi(\epsilon_t; s_t)$. This unbiased gradient estimator extends the DDPG style policy gradients [LHP+16] to any tractable stochastic policy.

SAC algorithm The algorithm also makes use of two Q-functions to mitigate maximization bias in the policy improvement step that is known to degrade the performance of value-based methods [Has10, FHM18]. In particular, we parameterize two Q-functions, with parameters θ_1 and θ_2, and train them independently to optimize $J_Q(\theta_1)$ and $J_Q(\theta_2)$, respectively. We then use the minimum of the Q-functions for the value gradient in Equation (7) and policy gradient in Equation (10), as proposed by [FHM18].

The method alternates between collecting experience from the environment with the current policy and updating the function approximators using the stochastic gradients from batches sampled from a replay buffer. In practice, a single environment step is taken followed by one or several gradient steps. The complete algorithm is described in Algorithm 4.

6 Interconnection between policy-based and value-based learning

With Equation (10) pointing out the interconnection between an action-dependent term and DDPG via a reparameterized policy, this connects learning via trial and error and via the gradient of the action value function. Previous work, including entropy-based policies [HTAL17], interpolated policy gradient [GLG+17b, GLG+17a] and action-dependent baselines [WRD+18, LFM+18, GCW+18, TBG+18] also built up preliminary understanding of this topic, for which we leave them to the readers to explore.

Algorithm 4: Soft actor-critic (SAC)

Initialize parameter vectors ψ, $\bar{\psi}$, θ_1, θ_2, ϕ
for *each iteration* **do**
  for *each environment step* **do**
    $a_t \sim \pi_\phi(a_t \mid s_t)$
    $s_{t+1} \sim \mathbb{P}(s_{t+1} \mid s_t, a_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$
  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 $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$

Acknowledgement

This lecture notes use material from the original SAC paper.

References

[DCH+16] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[FHM18] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596, 2018.

[GCW+18] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.

[GLG+17a] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *International Conference on Learning Representations*, 2017.

[GLG+17b] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, Bernhard Schölkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3849–3858, 2017.

[Has10] Hado van Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, volume 2, pages 2613–2621, 2010.

[HIB+18] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[HTAL17] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361, 2017.

[HZH+18] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[LFM+18] Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. Action-dependent control variates for policy optimization via Stein identity. In *International Conference on Learning Representations*, 2018.

[LHP+16] 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. In *International Conference on Learning Representations*, 2016.

[MBM+16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[MSB+09] Hamid Reza Maei, Csaba Szepesvari, Shalabh Bhatnagar, Doina Precup, David Silver, and Richard S Sutton. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*, pages 1204–1212, 2009.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[SLA+15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[TBG+18] George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard E Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning. In *International Conference on Machine Learning*, 2018.

[Wil92] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[WRD+18] Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M. Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. In *International Conference on Learning Representations*, 2018.