Lecture 18 - Policy gradient

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

DDA4230: Reinforcement Learning
Course Page: [Click]

Policy-based and value-based algorithms

Value-based algorithms include Q-learning, temporal-difference learning, and policy and value iteration

- These algorithms learn the values of actions $V(s)$ or $Q(s,a)$ and then selected action a based on the action values $\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s,a)$;
- The policy does not exist without the action value estimates $Q(s)$.

Policy-based and value-based algorithms

Concerns about value-based methods.

- The vanilla approaches can only address discrete action spaces due to the $\arg\max_{a \in \mathcal{A}}$ operation. However, in practice, the action space is usually continuous.
- Computing the action value functions $Q(s, a)$ for all state-action pair is costly when the action and state spaces are large or continuous.
- The policy of Q-Learning is deterministic and ε-greedy explore can be inefficient.
- It implicitly and indirectly improves the policy by improving the estimates of the values functions. However, we would think intuitively that improving the policy directly would be more efficient.

Policy-based and value-based algorithms

Policy gradient is the canonical approach for policy-based learning.

- Policy-based method directly parameterizes the policy function $\pi_\theta(s)$ without calculating the value functions.
- We use the notation $\boldsymbol{\theta} \in R^d$ for the policy's parameter vector. We then write $\pi(a \mid s, \boldsymbol{\theta}) = \mathbb{P}(a_t = a \mid s_t = s, \boldsymbol{\theta})$ as the probability that action a is taken given that the environment is in state s with parameter θ.
- A value function may still be used to learn the policy parameter, but is not required for action selection (will talk about it later in the actor-critic algorithm).

Policy approximation with parametrization

Discrete Action Space. then a natural way to parameterize a policy is to form parameterized state-action preferences $h(s, a, \boldsymbol{\theta})$ for each (s, a) pair and use a softmax distribution

$$\pi(a \mid s, \boldsymbol{\theta}) = \frac{\exp(h(a, s, \boldsymbol{\theta}))}{\sum_{a'} \exp(h(a', s, \boldsymbol{\theta}))}. \qquad \text{(softmax in action preferences)}$$

- The state-action preference measures how the policy $\pi_{\boldsymbol{\theta}}$ prefer action a given state s. The actions with the highest preferences in each state are given the highest probabilities of being selected.

Policy approximation with parametrization

Discrete Action Space. then a natural way to parameterize a policy is to form parameterized state-action preferences $h(s, a, \boldsymbol{\theta})$ for each (s, a) pair and use a softmax distribution

$$\pi(a \mid s, \boldsymbol{\theta}) = \frac{\exp(h(a, s, \boldsymbol{\theta}))}{\sum_{a'} \exp(h(a', s, \boldsymbol{\theta}))}. \qquad \text{(softmax in action preferences)}$$

- The action preferences $h(a, s, \boldsymbol{\theta})$ can be parameterized arbitrarily. For example, it can simply be the linear combinations of features (as for the feature vectors $x(a, s)$)

$$h(a, s, \boldsymbol{\theta}) = \boldsymbol{\theta}^T x(a, s).$$

Policy approximation with parametrization

Continuous Action Space. The policy can be defined as the normal probability density over a real-valued scalar action, with mean and standard deviation given by parametric function approximators

$$\pi(a \mid s, \boldsymbol{\theta}) = \frac{1}{\sigma(s, \boldsymbol{\theta}_\sigma)\sqrt{2\pi}} \exp\left(-\frac{(a - \mu(s, \boldsymbol{\theta}_\mu))^2}{2\sigma(s, \boldsymbol{\theta}_\sigma)^2}\right).$$

- We divide the policy's parameter vector into two parts, $\boldsymbol{\theta} = [\boldsymbol{\theta}_\mu, \boldsymbol{\theta}_\sigma]$.
- One possible way to parametrize the mean and standard deviation is

$$\mu(s, \boldsymbol{\theta}) = \boldsymbol{\theta}_\mu^T \mathbf{x}_\mu(s), \qquad \sigma(s, \boldsymbol{\theta}) = \exp(\boldsymbol{\theta}_\sigma^T \mathbf{x}_\sigma(s)),$$

where $\mathbf{x}_\sigma(s)$ and $\mathbf{x}_\mu(s)$ are feature vectors.

Policy approximation with parametrization

Advantages of using parametrization

- It handles both discrete and continuous action spaces.
- It could be deterministic or stochastic. If the optimal policy is deterministic, then the preference values $h(a, s, \theta)$ will be driven infinitely higher than all other actions.
- The choice of policy parametrization is sometimes a good way of injecting prior knowledge about the desired form of the policy into the learning.
- Policy gradient has stronger convergence guarantees than value-based method because of the smooth change in the probability.

Policy Gradient Objective

Recall the gradient descent algorithm, $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta_t} + \alpha \widehat{\nabla \boldsymbol{J}(\boldsymbol{\theta})}$ where $\boldsymbol{J}(\boldsymbol{\theta})$ is our objective function and α is the learning rate.

For the episodic case, where the episode terminates at some terminal state set, we define the objective function $\boldsymbol{J}(\boldsymbol{\theta})$ as

$$\boldsymbol{J}(\boldsymbol{\theta}) = V^{\pi_\theta}(s_0) = \sum_{s \in \mathcal{S}} \rho^{\pi_\theta}(s \mid s_0) r(s),$$

where s_0 is the starting state, $V^{\pi_\theta}(s_0)$ is the value function for π_θ, and $r(s) = \mathbb{E}_{a \sim \pi}[\mathcal{R}(s, a)]$ is the expected reward at s following π. The occupancy measure $\rho^{\pi_\theta}(s \mid s_0) = \frac{1}{T} \sum_{t=0}^{T} \mathbb{P}(s_t = s \mid s_0, \pi_\theta)$, where T is a random variable denoting the index of the terminal step.

Policy Gradient Objective

For the continuing case, where the process continues infinitely, we define the objective function $\boldsymbol{J}(\boldsymbol{\theta})$ as the averaged reward over the time steps.

$$
\begin{aligned}
\boldsymbol{J}(\boldsymbol{\theta}) &= \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[r_t \mid s_0, \pi_\theta] \\
&= \lim_{t\to\infty} \mathbb{E}[r_t \mid s_0, \pi_\theta] \\
&= \sum_{s} \rho^{\pi_\theta}(s \mid s_0) r(s) \\
&= V^{\pi_\theta}(s_0),
\end{aligned}
$$

where the occupancy measure $\rho^{\pi_\theta}(s \mid s_0) = \lim_{t\to\infty} \mathbb{P}(s_t = s \mid s_0, \pi_\theta)$ is the stationary distribution of the Markov chain under policy π_θ.

Policy Gradient Objective

For the discounted case where $\gamma < 1$, we define the objective function $\boldsymbol{J}(\boldsymbol{\theta})$ as the expected discounted return

$$\boldsymbol{J}(\boldsymbol{\theta}) = V^{\pi_\theta}(s_0) = \sum_{s \in \mathcal{S}} \rho^{\pi_\theta}(s \mid s_0) r(s),$$

where the occupancy measure $\rho^{\pi_\theta}(s \mid s_0) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s \mid s_0, \pi_\theta)$.

Policy Gradient Objective

The policy gradient theorem states that

$$
\begin{aligned}
\nabla_\theta J(\theta) &\propto \sum_{s\in\mathcal{S}} \rho^{\pi_\theta}(s \mid s_0) \sum_{a\in\mathcal{A}} Q^{\pi_\theta}(s,a) \nabla_\theta \pi_\theta(a \mid s) \\
&= \mathbb{E}_\pi \left[Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a \mid s) \right].
\end{aligned}
$$

Proof of policy gradient theorem

Please refer to the proof of the episodic case in discrete state-action space in the lecture notes.

REINFORCE (episodic Monte-Carlo policy-gradient control)

To compute the gradient $\nabla_\theta \boldsymbol{J}(\theta)$ algorithmically, we can sample N trajectories following the policy π and use the empirical mean to estimate the gradient

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

- For $Q^\pi(s,a)$, we can use return $G_t = \sum \gamma^t r_t$ to estimate.
- For $\nabla_\theta \log \pi_\theta(a \mid s)$, it depends on the form of the policy.

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

REINFORCE (episodic Monte-Carlo policy-gradient control)

Algorithm 1: REINFORCE (Monte-Carlo method)

Initialize the policy parameter θ

for *each episode* **do**

  Sample one trajectory on policy π_θ: $s_0, a_0, r_0, s_1, a_1, \dots, s_T$

  for *each* $t = 0, 1, \dots, T$ **do**

    $G_t \leftarrow \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$

    $\theta \leftarrow \theta + \alpha \gamma^t G_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)$

REINFORCE with baselines

One problem of policy gradient method is high variance. (why? Click to see a very intuitive explanation.) A natural solution is to subtract a baseline $b(s)$ from Q^π, i.e.,

$$\nabla_\theta J(\theta) \propto \sum_{s \in \mathcal{S}} \rho^\pi(s \mid s_0) \sum_{a \in \mathcal{A}} (Q^\pi(s,a) - b(s)) \nabla \pi_\theta(a \mid s) .$$

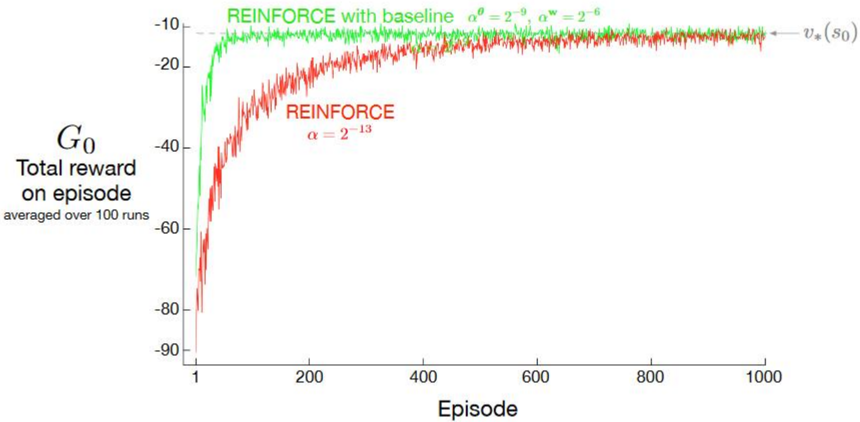
The baseline can be any function, even a random variable, as long as it does not depend on the action a.

$$\sum_a b(s) \nabla \pi(a \mid s, \theta) = b(s) \nabla \sum_a \pi(a \mid s, \theta) = b(s) \nabla 1 = 0 .$$

The expectation value does not change. The update rule that we end up with is a new version of REINFORCE that includes a general baseline

$$\theta \leftarrow \theta + \alpha \gamma^t (G_t - b(s_t)) \nabla_\theta \log \pi_\theta(a_t \mid s_t) .$$

REINFORCE with baselines



REINFORCE with baselines

One natural choice for the baseline is an estimate of the state value $\hat{V}(s, \boldsymbol{w})$, where $\boldsymbol{w} \in \mathbb{R}^d$ is a weight vector to be learned. We can use the same method as we adopted in learning $\boldsymbol{\theta}$ to learn $\boldsymbol{w}$. The complete process is as follows. We have two inputs:

- A differentiable policy parametrization $\pi_\theta(a \mid s)$;
- A differentiable state value function parametrization $\hat{V}(s, \boldsymbol{w})$.

REINFORCE with baselines

Algorithm 2: REINFORCE with baseline

Initialize the policy parameter $\boldsymbol{\theta}$ and $\boldsymbol{w}$ at random.

for *each episode* **do**

  Sample one trajectory under policy π_θ: $s_0, a_0, r_0, s_1, a_1, r_1 \ldots, s_T$

  for *each* $t = 1, 2, \ldots, T$ **do**

    $G_t \leftarrow \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$

    $\delta \leftarrow G_t - \hat{V}(s_t, \boldsymbol{w})$

    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha_{\boldsymbol{w}} \delta \nabla_{\boldsymbol{w}} \hat{V}(s_t, \boldsymbol{w})$

    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \gamma^t \delta \nabla_{\boldsymbol{\theta}} \log \pi_\theta(a_t \mid s_t)$

Question and Answering (Q&A)