DDA4230 Reinforcement learning	Policy gradient
Lecture 17	
Lecturer: Guiliang Liu	*Scribe: Ke Li*

1 Goal of this lecture

Starting from this lecture we discuss general reinforcement learning that can have continuous state and action spaces.

In this lecture we discuss Monte-Carlo methods via policy function approximation and policy gradient.

Suggested reading: Chapter 13 of *Reinforcement learning: An introduction*;

2 Policy-based and value-based algorithms

2.1 Value-based methods

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 $Q(s)$.

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 implied by Q-Learning is deterministic and ϵ-greedy exploration can be quite 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.

These motivate policy-based methods.

2.2 Policy-based methods

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).

3 Policy approximation with parametrization

3.1 Discrete case

If the action space is discrete, 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

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

- 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.
- 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 $x(a, s)$, see Chapter 9 in Sutton and Barto's Book)

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

3.2 Continuous case

For continuous case, 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

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

  where $\boldsymbol{x}_{\boldsymbol{\sigma}}(s)$ and $\boldsymbol{x}_{\boldsymbol{\mu}}(s)$ are feature vectors discussed in Chapter 9 in Sutton and Barto's book.

So far, we have constructed the policy function approximation for both discrete and continuous cases.

3.3 Advantages of using parametrization

- It handles both discrete and continuous action spaces.
- It could be deterministic or stochastic. Action preference $h(a, s, \boldsymbol{\theta})$ enables the selection of actions with arbitrary probabilities. If the optimal policy is deterministic, then the preferences value 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 reinforcement learning system.
- Policy gradient has stronger convergence guarantees than value based method because of the smooth change in the probability.

4 Policy gradient

4.1 Policy gradient theorem

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. The objective $\boldsymbol{J}(\boldsymbol{\theta})$ have a variety of definitions, including the episodic setting, the continuing setting, and the discounted continuing setting. These settings will correspond to a varieties of definitions of the occupancy measure.

We take discrete state and action spaces as an example. For continuous spaces, under mild assumptions we can replace the summation over actions and states by the integration over actions and states.

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.

For the continuing case, where the process continues infinitely, we define the objective

function $\boldsymbol{J(\theta)}$ as the averaged reward over the time steps.

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

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.

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

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

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

The policy gradient theorem states that

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

4.2 Proof of policy gradient theorem

We prove the episodic case and leave other cases and the continuous setting to the reader. For the proof of the continuing case, see Chapter 13.6 in Sutton and Barto's book. When the context is clear we write π_θ into π.

Proof: We have

$$\begin{aligned}
&\nabla_\theta V^\pi(s) \\
&= \nabla_\theta \Big(\sum_{a\in\mathcal{A}} \pi_\theta(a \mid s) Q^\pi(s,a) \Big) \\
&= \sum_{a\in\mathcal{A}} \Big(\nabla_\theta \pi_\theta(a \mid s) Q^\pi(s,a) + \pi_\theta(a \mid s) \nabla_\theta Q^\pi(s,a) \Big) && \text{Derivative product rule.} \\
&= \sum_{a\in\mathcal{A}} \Big(\nabla_\theta \pi_\theta(a \mid s) Q^\pi(s,a) + \pi_\theta(a \mid s) \nabla_\theta \sum_{s',r} \mathbb{P}(s', r \mid s, a)(r + V^\pi(s')) \Big) && \text{Expand } Q^\pi \text{ with future state value.} \\
&= \sum_{a\in\mathcal{A}} \Big(\nabla_\theta \pi_\theta(a \mid s) Q^\pi(s,a) + \pi_\theta(a \mid s) \sum_{s',r} \mathbb{P}(s', r \mid s, a) \nabla_\theta V^\pi(s') \Big) && \mathbb{P}(s',r|s,a) \text{ and } r \text{ are independent of } \theta \\
&= \sum_{a\in\mathcal{A}} \Big(\nabla_\theta \pi_\theta(a \mid s) Q^\pi(s,a) + \pi_\theta(a \mid s) \sum_{s'} \mathbb{P}(s' \mid s, a) \nabla_\theta V^\pi(s') \Big) && \text{Because } \mathbb{P}(s'|s,a) = \textstyle\sum_r \mathbb{P}(s',r|s,a)
\end{aligned}$$

Then,

$$\nabla_\theta V^\pi(s) = \sum_{a\in\mathcal{A}} \left(\nabla_\theta \pi_\theta(a \mid s) Q^\pi(s,a) + \pi_\theta(a \mid s) \sum_{s'} \mathbb{P}(s' \mid s, a) \nabla_\theta V^\pi(s') \right).$$

This equation has a nice recursive form (see the red parts!) and the future state value function $V^\pi(s')$ can be repeated unrolled by following the same equation.

We consider the following visitation sequence and label the probability of transitioning from state s to state x with policy π_θ after k step as $\rho^\pi(s \to x, k)$

$$s \xrightarrow{a\sim\pi_\theta(\cdot|s)} s' \xrightarrow{a\sim\pi_\theta(\cdot|s')} s'' \xrightarrow{a\sim\pi_\theta(\cdot|s'')} \dots .$$

- When $k = 0$, $\rho^\pi(s \to s, k = 0) = 1$;
- When $k = 1$, we scan through all possible actions and sum up the transition probabilities to the target state: $\rho^\pi(s \to s', k = 1) = \sum_{a\in\mathcal{A}} \pi_\theta(a \mid s)\mathbb{P}(s' \mid s, a)$;
- Imagine that the goal is to go from state s to x after $k + 1$ steps, we can write it as $\rho^\pi(s \to x, k + 1) = \sum_{s'} \rho^\pi(s \to s', k)\rho^\pi(s' \to x, 1)$. That is, we first arrive at s' after k step and we go 1 step further.

Then we go back to unroll the recursion of $\nabla_\theta V^\pi(s)$. Denote

$$\phi(s) = \sum_{a\in\mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) Q^\pi(s,a)$$

for simplicity. We have

$$\begin{aligned}
&\nabla_\theta V^\pi(s) \\
=&\phi(s) + \sum_a \pi_\theta(a \mid s) \sum_{s'} \mathbb{P}(s' \mid s, a) \nabla_\theta V^\pi(s') \\
=&\phi(s) + \sum_{s'} \sum_a \pi_\theta(a \mid s) \mathbb{P}(s' \mid s, a) \nabla_\theta V^\pi(s') \\
=&\phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \nabla_\theta V^\pi(s') \\
=&\phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) (\phi(s') + \sum_{s''} \rho^\pi(s' \to s'', 1) \nabla_\theta V^\pi(s'')) \\
=&\phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \phi(s') + \sum_{s''} \rho^\pi(s \to s'', 2) \nabla_\theta V^\pi(s'') \\
=&\phi(s) + \sum_{s'} \rho^\pi(s \to s', 1) \phi(s') + \sum_{s''} \rho^\pi(s \to s'', 2) \phi(s'') + \sum_{s'''} \rho^\pi(s \to s''', 3) \nabla_\theta V^\pi(s''') \\
=&\dots \text{Repeatedly unroll the parts of } \nabla_\theta V^\pi(\cdot) \\
=&\sum_{x\in\mathcal{S}} \sum_{k=0}^{\infty} \rho^\pi(s \to x, k) \phi(x) .
\end{aligned}$$

By plugging it into the objective function $\boldsymbol{J}(\theta)$, we have

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta V^\pi(s_0) && \text{Starting from a random state } s_0 \\
&= \sum_s \sum_{k=0}^{\infty} \rho^\pi(s_0 \to s, k)\phi(s) && \text{Denote } \eta(s)=\textstyle\sum_{k=0}^{\infty} \rho^\pi(s_0 \to s,k) \\
&= \sum_s \eta(s)\phi(s) \\
&= \Big(\sum_s \eta(s)\Big) \sum_s \frac{\eta(s)}{\sum_s \eta(s)}\phi(s) && \text{Normalize } \eta(s), s\in\mathcal{S} \text{ to be a probability distribution} \\
&\propto \sum_s \frac{\eta(s)}{\sum_s \eta(s)}\phi(s) && \textstyle\sum_s \eta(s) \text{ is a constant} \\
&= \sum_s \rho^\pi(s \mid s_0) \sum_a \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a)\,. && \rho^\pi(s|s_0)=\frac{\eta(s)}{\sum_s \eta(s)} \text{ is the stationary distribution}
\end{aligned}
$$

The policy gradient can then be written as

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

where $\mathbb{E}_\pi$ refers to trajectory sampling $\mathbb{E}_{s\sim d_\pi, a\sim\pi_\theta}$ according to π. □

For the proof in the continuous case, see Chapter 13.6 in Sutton and Barto's book.

5 Policy gradient algorithms

5.1 REINFORCE (episodic Monte-Carlo policy-gradient control)

Now we discuss how 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.

The process is pretty straightforward. The discount γ^t in the last step is usually omitted in practice.

Algorithm 1: REINFORCE (Monte-Carlo method)

Initialize the policy parameter θ
for *each episode* **do**
  Sample one trajectory on policy π_θ: $s_0, a_0, r_0, s_1, a_1, \ldots, s_T$
  for *each* $t = 0, 1, \ldots, T$ **do**
    $G_t \leftarrow \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$
    $\theta \leftarrow \theta + \alpha \gamma^t G_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)$

5.2 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

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

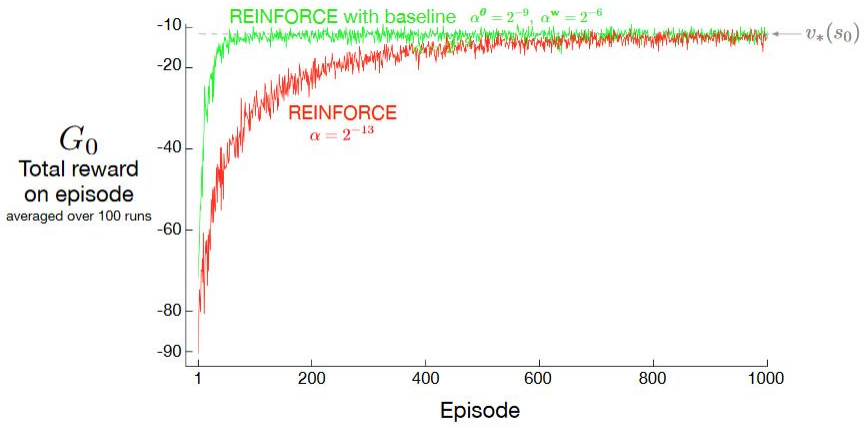

Figure 1: Comparison of REINFORCE and REINFORCE with baseline

As shown in the figure above (source: Sutton and Barto's book, Chapter 13), adding a baseline can learn *much faster*. This post nicely explained why a baseline works for reducing the variance.

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 $\hat{V}(s, \boldsymbol{w})$.

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**
    $G_t \leftarrow \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$
    $\delta \leftarrow G_t - \hat{V}(s_t, \boldsymbol{w})$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha_{\boldsymbol{w}} \delta \nabla_{\boldsymbol{w}} \hat{V}(s_t, \boldsymbol{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \gamma^t \delta \nabla_{\boldsymbol{\theta}} \log \pi_\theta(a_t \mid s_t)$

References

[1] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. MIT Press. 2018.

[2] John Schulman, et al. "High-dimensional continuous control using generalized advantage estimation." ICLR 2016.

[3] David Silver, et al. "Deterministic policy gradient algorithms." ICML. 2014.

[4] Adrien, E. Intuitive explanation of policy gradient. 2018. Available here.

[5] Lilian, W. Policy gradient algorithms. 2018. Available here.

Acknowledgement

This lecture notes partially use material from *Reinforcement learning: An introduction.*