ml-mipt-advanced

Lecture 10. Policy gradient methods.

Nikolay Karpachev
8.04.2024

Value-based vs. Policy-based RL

- Last lecture:
  - Approximate V and Q functions

$$V_\theta(s) \approx V^\pi(s)$$
$$Q_\theta(s,a) \approx Q^\pi(s,a)$$

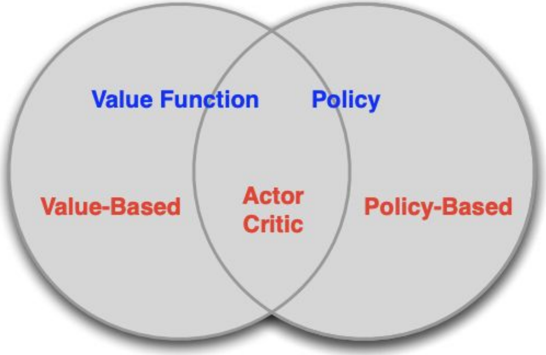
Policy induced from Q-values

- Current lecture:
  - Directly parametrize the policy

$$\pi_\theta(s,a) = \mathbb{P}\left[a \mid s, \theta\right]$$

Value-based vs. Policy-based RL

1. Value-based
   a. Learnt value function
   b. Implicit policy (e.g. eps-greedy)
2. Policy-based
   a. No value function
   b. Learnt policy
3. Actor-critic
   a. Learnt value function
   b. Learnt policy



Policy-based RL

Advantages:

- Better convergence properties
- Effective in continuous action spaces
- Can learn any kind of stochasticity

Disadvantages:

- Converge to a local optimum
- High variance in single evaluation runs

Policy-based RL

left or **right**?

Value-based RL

What's **Q(s,right)** under gamma=0.99?

Value-based RL: Approximation error

DQN is trained to minimize

$$L \approx E[Q(s_t, a_t) - (r_t + \gamma \cdot max_{a'} Q(s_{t+1}, a'))]^2$$

Simple 2-state world

	True	(A)	(B)
Q(s0,a0)	*1*	*1*	*2*
Q(s0,a1)	*2*	*2*	*1*
Q(s1,a0)	*3*	*3*	*3*
Q(s1,a1)	*100*	*50*	*100*
		better policy	**less MSE**

Policy Gradient

- Given: policy parametrization
- Find: best parameters set

Q. What is the objective function?

Policy Gradient

- Given: policy parametrization
- Find: best parameters set

Q. What is the objective function?

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

Episodic environments

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

Continual environments

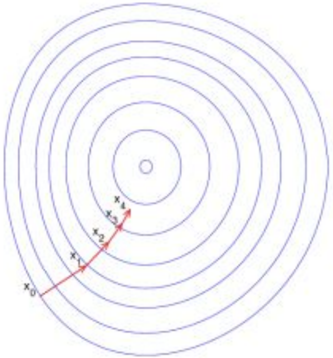
Policy Gradient Optimization

- $J(\theta)$ - policy objective function

- Gradient ascent in parameters space w.r.t. J

$$\Delta\theta = \alpha \nabla_\theta J(\theta)$$

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$



Policy Gradient Optimization

$$J(\theta) = \sum_{s \in \mathcal{S}} d^\pi(s) V^\pi(s) = \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) Q^\pi(s, a)$$

Precise estimating gradient is difficult

- stationary distribution is not known
- computationally expensive in high-dimensional spaces
- intractable in continous action spaces

Policy Gradient Approximation

Idea 1: Numeric approximation

Finite differences on each axis:

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

Policy Gradient Approximation

Idea 1: Numeric approximation

Finite differences on each axis:

$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

- Noisy estimate
- Computationally expensive (n runs)
- Works for arbitrary (even non-differentiable) policies

Policy Gradient Approximation

Idea 2: Monte-Carlo estimation

$$\mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[r] = \sum_{s \in \mathcal{S}} d_{\pi_\theta}(s) \sum_{a \in \mathcal{A}} \pi(a|s;\theta) R(s,a)$$

Even if intractable, a function can be estimated:

- *if represented as an expectation over trajectories **from the current policy***

Policy Gradient Approximation

Log-derivative trick:

$$
\begin{aligned}
\nabla_\theta \pi_\theta(s,a) &= \pi_\theta(s,a)\frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} \\
&= \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)
\end{aligned}
$$

Policy Gradient Approximation

Algorithm: Monte-Carlo Policy Gradient (REINFORCE)

$$
\begin{aligned}
\mathcal{J}(\theta) &= \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \pi(a|s;\theta) Q_\pi(s,a) \\
\nabla\mathcal{J}(\theta) &= \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \nabla\pi(a|s;\theta) Q_\pi(s,a) \\
&= \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \pi(a|s;\theta) \frac{\nabla\pi(a|s;\theta)}{\pi(a|s;\theta)} Q_\pi(s,a) \\
&= \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \pi(a|s;\theta) \nabla \ln \pi(a|s;\theta) Q_\pi(s,a) \\
&= \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|s;\theta) Q_\pi(s,a)]
\end{aligned}
$$

Policy Gradient Approximation

Algorithm: Monte-Carlo Policy Gradient (REINFORCE)

$$\begin{aligned}
\mathcal{J}(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi(a|s;\theta) Q_\pi(s,a) \\
\nabla \mathcal{J}(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \nabla \pi(a|s;\theta) Q_\pi(s,a) \\
&= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi(a|s;\theta) \frac{\nabla \pi(a|s;\theta)}{\pi(a|s;\theta)} Q_\pi(s,a) \\
&= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi(a|s;\theta) \nabla \ln \pi(a|s;\theta) Q_\pi(s,a) \\
&= \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|s;\theta) Q_\pi(s,a)]
\end{aligned}$$

by log-derivative trick

Policy Gradient Approximation

Algorithm: Monte-Carlo Policy Gradient (REINFORCE)

$$\mathcal{J}(\theta) = \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi(a|s;\theta) Q_\pi(s,a)$$

$$\begin{aligned}
\nabla \mathcal{J}(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \nabla \pi(a|s;\theta) Q_\pi(s,a) \\
&= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi(a|s;\theta) \frac{\nabla \pi(a|s;\theta)}{\pi(a|s;\theta)} Q_\pi(s,a) \\
&= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi(a|s;\theta) \nabla \ln \pi(a|s;\theta) Q_\pi(s,a) \\
&= \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|s;\theta) Q_\pi(s,a)]
\end{aligned}$$

by log-derivative trick

this expectation can be estimated by samples of episodes

Policy Gradient Approximation

Policy Gradient Theorem:

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|s,\theta) Q_\pi(s,a)]$$

Monte-Carlo Policy Gradient: REINFORCE

function REINFORCE

    Initialise θ arbitrarily

    for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**

        for $t = 1$ to $T - 1$ **do**

            $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$

        end for

    end for

    return θ

end function

Monte-Carlo Policy Gradient: REINFORCE

Problems:

- Gradient update involves Q-function from single transition
- Which is noisy

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|s,\theta) Q_\pi(s,a)]$$

Hence,

- High variance in individual gradient steps
- Slower (and more unstable) convergence

Actor-Critic Policy Gradient

Idea: introduce critic model to estimate Q-fuction

$$Q_w(s, a) \approx Q^{\pi_\theta}(s, a)$$

Actor-critic algorithm

1. **Actor** - policy itself, selects (samples) actions
2. **Critic** - value function (Q, V) prediction

Actor-Critic Policy Gradient

REINFORCE:

monte-carlo sampled policy gradient

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|s,\theta) Q_\pi(s,a)]$$

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$$

Actor-Critic:

estimated policy gradient

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)\right]$$

$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)$$

Actor-Critic Policy Gradient

Q.: How to train critic model?

Actor-Critic Policy Gradient

Q.: How to train critic model?

A.: The same way as in DQN

- Policy Evaluation
- MC / TD / TD(l) - whatever

Actor-Critic Policy Gradient

function QAC
    Initialise s, θ
    Sample $a \sim \pi_\theta$
    for each step **do**
        Sample reward $r = \mathcal{R}_s^a$; sample transition $s' \sim \mathcal{P}_{s,\cdot}^a$
        Sample action $a' \sim \pi_\theta(s', a')$
        $\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$
        $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$
        $w \leftarrow w + \beta \delta \phi(s, a)$
        $a \leftarrow a', s \leftarrow s'$
    end for
end function

Actor-Critic Policy Gradient

function QAC
  Initialise s, θ
  Sample $a \sim \pi_\theta$
  for each step **do**
    Sample reward $r = \mathcal{R}_s^a$; sample transition $s' \sim \mathcal{P}_{s,\cdot}^a$
    Sample action $a' \sim \pi_\theta(s', a')$
    $\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$
    $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$
    $w \leftarrow w + \beta \delta \phi(s, a)$
    $a \leftarrow a', s \leftarrow s'$
  end for
end function

Actor: *updates \theta by policy gradient*

Critic: *updates \w by linear TD(0)*

Advantage Actor-Critic

Idea: we can reduce variance in monte-carlo sampling estimates

Definition: Baseline B(s, a) w.r.t. policy is such a function that

$$
\begin{aligned}
\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) B(s)\right] &= \sum_{s\in\mathcal{S}} d^{\pi_\theta}(s) \sum_{a} \nabla_\theta \pi_\theta(s,a) B(s) \\
&= \sum_{s\in\mathcal{S}} d^{\pi_\theta} B(s) \nabla_\theta \sum_{a\in\mathcal{A}} \pi_\theta(s,a) \\
&= 0
\end{aligned}
$$

Advantage Actor-Critic

Idea: we can reduce variance in monte-carlo sampling estimates

Definition: Baseline B(s, a) w.r.t. policy is such a function that

$$\begin{aligned}
\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) B(s)\right] &= \sum_{s \in \mathcal{S}} d^{\pi_\theta}(s) \sum_{a} \nabla_\theta \pi_\theta(s,a) B(s) \\
&= \sum_{s \in \mathcal{S}} d^{\pi_\theta} B(s) \nabla_\theta \sum_{a \in \mathcal{A}} \pi_\theta(s,a) \\
&= 0
\end{aligned}$$

Any function that does not depend on a is a baseline !

Advantage Actor-Critic

Idea: we can reduce variance in monte-carlo sampling estimates

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|s,\theta) Q_\pi(s,a)]$$

$$\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) B(s)\right] = 0$$

If we subtract baseline B(s) from Q under monte-carlo sample

1. **expectation does not change**
2. **if B(s) correlates with Q(s, a), variance decreases**

Advantage Actor-Critic

Idea: we can reduce variance in monte-carlo sampling estimates

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|s,\theta) Q_\pi(s,a)]$$

$$\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) B(s)\right] = 0$$

If we subtract baseline B(s) from Q under monte-carlo sample

(1) **expectation does not change**
(2) **if B(s) correlates with Q(s, a), variance decreases**

variance decreases -> more stable updates

Advantage Actor-Critic (A2C)

Common choice for baseline function: state value function

$$B(s) = V^{\pi_\theta}(s)$$

"advantage" function

$$A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ A^{\pi_\theta}(s,a)\right]$$

Outro

- Policy-based vs. Value-based RL
- Policy Gradient Estimation
  - Numerical: finite differences
  - Monte-Carlo: gradient as an expectation -> sampling
- REINFORCE
- Actor-Critic
- A2C

Acknowledgements

This lecture uses materials from

(1) RL Lectures by David Siver (licensed CC-BY-NC 4.0)

(2) Practical_RL lectures by Yandex Data School (Unlicense license)

Questions?