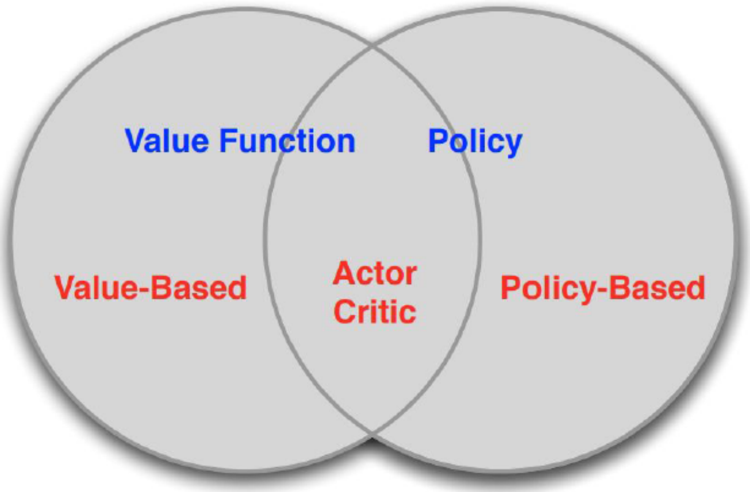
Policy-Based Reinforcement Learning

Wenjun Zeng

Reading Group of Ye Lab, UM

May 25, 2019

Categories of RL



- Value-based: Learnt value function, implicit policy
  - Q-learning, DQN, SARSA
- Policy-based: No value function, learnt policy
  - REINFORCE, TRPO (Trust region policy optimization), PPO (Proximal)
- Actor-Critic: Learnt value function, learnt policy
  - A2C, A3C

Contents

I. Basics

Markov Decision Processes (MDP)

- State space $\mathcal{S}$, $s \in \mathcal{S}$
- Action space $\mathcal{A}$, $a \in \mathcal{A}$
- Policy π, $a = \pi(s)$ for deterministic case or $a \sim \pi(a|s)$ for stochastic case
- Reward $R(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- Transition Dynamics P: $\mathcal{S} \times \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$

  $p(s'|s, a)$ is the probability that the agent reaches state s' from s after taking action a.

Objective of MDP

Denoting s_t, a_t, and $r_t = r(s_t, a_t)$ as the state, action, and reward of $t = 1, 2, \cdots$, MDP aims at finding policy to maximize the expectation of

$$\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t)$$

with $\gamma \in [0, 1)$ the discount factor.

Reinforcement Learning:

- Model-based: State transition $p(s'|s, a)$ is explicitly known.
- Model-free: unknown.

II. Policy Gradient

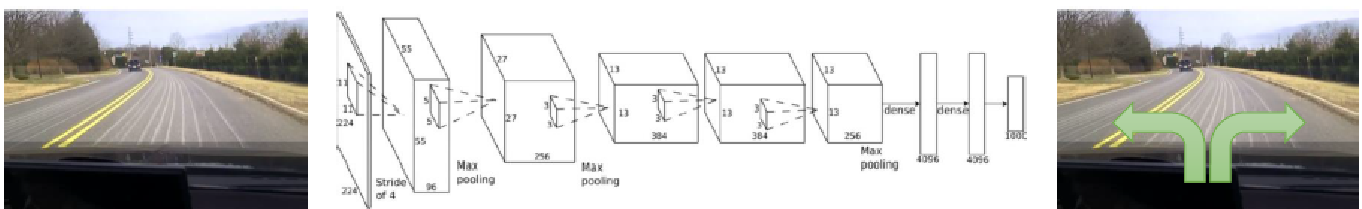
Parameterized Policy

Recall a policy $\pi(\cdot)$

- ❑ $a = \pi(s)$ for deterministic case
- ❑ $a \sim \pi(a|s)$ for stochastic case, i.e., $\pi(a|s)$ is a probability distribution

Anyway, the policy is a function. Thus, finding the optimal policy is a problem of functional optimization (not easy to handle).

Parameterized policy: $\pi(\cdot) = \pi_\theta(\cdot)$ with $\theta \in \mathbb{R}^m$. For example, use a neural network



$\mathbf{s}_t$ $\qquad$ $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ $\qquad$ $\mathbf{a}_t$

In this case, θ is the weights of the DNN.

Policy (functional) optimization $\rightarrow$ parameter optimization

Policy Optimization

Denote $\tau = (s_1, a_1, s_2, a_2, \cdots, s_T, a_T)$,

$$p_\theta(\tau) = p_\theta(s_1, a_1, \cdots, s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

The objective function of RL is written as

$$J(\theta) = \mathbb{E}\left[\sum_{t=1}^{T} \gamma^t r(s_t, a_t) | \pi_\theta\right] = \mathbb{E}_{\tau \sim p_\theta(\tau)} [r(\tau)]$$

where $r(\tau) = \sum_{t=1}^{T} \gamma^t r(s_t, a_t)$. Our goal is

$$\max_\theta J(\theta)$$

Gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Policy Gradient

How to compute the gradient $\nabla_\theta J(\theta)$?

$$J(\theta) = \int_\tau p_\theta(\tau) r(\tau) \, \mathrm{d}\tau$$

Differentiate $J(\theta)$ w.r.t. θ yields

$$\nabla_\theta J(\theta) = \int_\tau \nabla_\theta p_\theta(\tau) r(\tau) \, \mathrm{d}\tau$$

Difficult to go on? We resort to a nice trick

$$\nabla_\theta p_\theta(\tau) = p_\theta(\tau) \frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} = p_\theta(\tau) \nabla_\theta \log p_\theta(\tau)$$

and we obtain

$$\begin{aligned} \nabla_\theta J(\theta) &= \int_\tau p_\theta(\tau) r(\tau) \nabla_\theta \log p_\theta(\tau) \, \mathrm{d}\tau \\ &= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[r(\tau) \nabla_\theta \log p_\theta(\tau) \right] \end{aligned}$$

Policy Gradient Estimator

Remember

$$p_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

Hence

$$\log p_\theta(\tau) = \log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \sum_{t=1}^{T} \log p(s_{t+1}|s_t, a_t)$$

and

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} r(s_t, a_t) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right]$$

Use N samples to estimate $\nabla_\theta J(\theta)$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=1}^{T} r(s_t^i, a_t^i) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(a_t^i | s_t^i \right) \right]$$

III. REINFORCE Algorithm

1. Initialize θ

2. Sample $\left\{\tau^i = (s_1^i, a_1^i, \cdots, s_T^i, a_T^i)\right\}_{i=1}^N$ from $\pi_\theta(a_t|s_t)$ (run the policy)

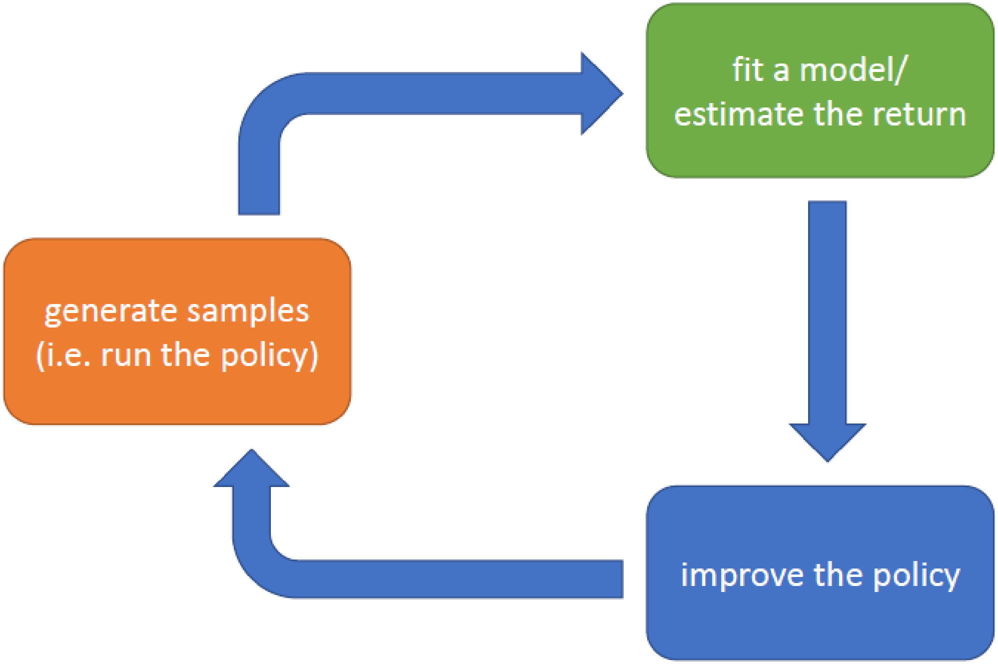
3. Estimate gradient

$$\nabla_\theta J(\theta) \approx \sum_{i=1}^{N} \left[\sum_{t=1}^{T} r(s_t^i, a_t^i) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(a_t^i | s_t^i\right) \right]$$

4. Update θ

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Understanding REINFORCE (I)



Understanding REINFORCE (II)

Recall

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Maximum likelihood (ML):

$$\nabla_\theta J_{\mathrm{ML}}(\theta) = \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(a_t^i|s_t^i\right) = \sum_{i=1}^{N} \nabla_\theta \log p_\theta(\tau^i)$$

Compare

$$\nabla_\theta J(\theta) = \sum_{i=1}^{N} \left[\sum_{t=1}^{T} r(s_t^i, a_t^i) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(a_t^i|s_t^i\right)\right] = \sum_{i=1}^{N} r(\tau^i) \nabla_\theta \log p_\theta(\tau^i)$$

REINFORCE is very similar to ML! Use return $r(\tau^i)$ to reweight.

- High return is made more likely
- Low return is made less likely
- Simply formalize the notion of "trial-and-error"

Viewpoint: RL is in fact like supervised learning (Dr. Jiayu Zhou, MSU).

Advantages and Disadvantages

Advantages:

1. Effective in high-dimensional or continuous action spaces
2. Can learn stochastic policies

Disadvantages:

1. Typically converge to a local rather than global optimum
2. Policy gradient estimation is typically inefficient and high variance

IV. Variance Reduction

Causality: Policy at t' cannot affect reward at t when $t' > t$.

Modify

$$\nabla_\theta J(\theta) = \sum_{i=1}^{N} \left[\sum_{t=1}^{T} r(s_t^i, a_t^i) \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(a_t^i | s_t^i \right) \right]$$

into $\Rightarrow$

$$\nabla_\theta J(\theta) = \sum_{i=1}^{N} \left[\sum_{t=1}^{T} \left(\sum_{t' \geq t}^{T} r(s_{t'}^i, a_{t'}^i) \right) \left(\nabla_\theta \log \pi_\theta a_t^i | s_t^i \right) \right]$$

where $\sum_{t' \geq t}^{T} r(s_{t'}^i, a_{t'}^i)$ is the reward to go.

Or into

$$\nabla_\theta J(\theta) = \sum_{i=1}^{N} \left[\sum_{t=1}^{T} \left(\sum_{t' \geq t}^{T} \gamma^{t'-t} r(s_{t'}^i, a_{t'}^i) \right) \left(\nabla_\theta \log \pi_\theta a_t^i | s_t^i \right) \right]$$

Variance Reduction: Baseline

Idea: Introduce a baseline function dependent on the state:

$$\nabla_\theta J(\theta) = \sum_{i=1}^{N} \left[\sum_{t=1}^{T} \left(\sum_{t' \geq t}^{T} \gamma^{t'-t} r(s_{t'}^i, a_{t'}^i) - b(s_t) \right) \nabla_\theta \log \pi_\theta \left(a_t^i | s_t^i \right) \right]$$

We are allowed to do this because subtracting a baseline is unbiased in expectation. In expectation, the gradient estimator remains unchanged.

Simple baseline: Average reward, not the best but pretty good (S. Levine's Lecture 2018).

V. Actor-Critic

Key idea: Choose a better baseline that push up the probability of an action from a state, if this action was better than the expected value of what we should get from that state.

What does this remind us of?

Value function (How good is a state?)

$$V^{\pi_\theta}(s) = \mathbb{E}\left[\sum_t \gamma^t r(s_t, a_t) \middle| s_1 = s, \pi_\theta\right]$$

is the expected return from following the policy from state s.

Q-function (How good is a state-action pair?)

$$Q^{\pi_\theta}(s, a) = \mathbb{E}\left[\sum_t \gamma^t r(s_t, a_t) \middle| s_1 = s, a_1 = a, \pi_\theta\right]$$

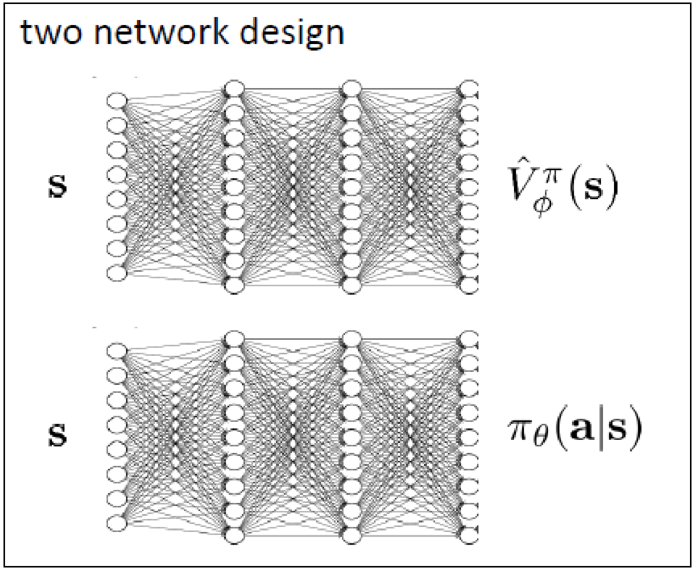
is the expected return for executing a particular action a at a given state s.

Choose $V_\phi(s_t) \approx V^{\pi_\theta}(s_t)$ as the baseline, i.e., we have

$$\nabla_\theta J(\theta) = \sum_{i=1}^{N} \left[\sum_{t=1}^{T} \left(\sum_{t' \geq t}^{T} \gamma^{t'-t} r(s_{t'}^i, a_{t'}^i) - V_\phi(s_t) \right) \nabla_\theta \log \pi_\theta \left(a_t^i | s_t^i \right) \right]$$

Problem: we do not know $V_\phi(\cdot)$.

Parameterize it use a neural network and learn it! Combine Policy Gradient and value learning by training both an actor (policy) and a critic (value function).



Actor-Critic Algorithm

Initialize policy parameter θ and critic parameter ϕ

For iteration $= 1, 2, \cdots$ **do**

Sample N episodes under current policy

$\Delta\theta \leftarrow \mathbf{0}$

For $i = 1, 2, \cdots, N$ **do**

For $t = 1, 2, \cdots, T$ **do**

$A_t^i = \sum_{t' \geq t}^{T} \gamma^{t'-t} r(s_{t'}^i, a_{t'}^i) - V_\phi(s_t)$

$\Delta\theta \leftarrow \Delta\theta + \nabla_\theta \log \pi_\theta \left(a_t^i | s_t^i\right)$

End For

End For

$\Delta\phi \leftarrow \sum_i \sum_t \nabla_\phi (A_t^i)^2$

$\theta \leftarrow \theta + \alpha \Delta\theta$

$\phi \leftarrow \phi - \beta \Delta\phi$

End For

Advantage Actor-Critic (A2C)

Use both value function and Q-function.

We are happy with an action a_t in a state s_t if

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$$

is large. On the contrary, we are unhappy with an action if it is small. Intuitively, $A^\pi(s_t, a_t)$ means how much better it is to take a specific action compared to the average, general action at the given state. We call this value the advantage value.

Gradient estimator

$$\nabla_\theta J(\theta) = \sum_{i=1}^{N} \left[\sum_{t=1}^{T} A^\pi(s_t^i, a_t^i) \nabla_\theta \log \pi_\theta \left(a_t^i | s_t^i \right) \right]$$

Problem: we do not know $Q(\cdot)$ and $V(\cdot)$. Do we need two neural networks?

No!

Advantage Actor-Critic (A2C)

Note that we have the relationship between Q and V

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|a_t, s_t)}[V(s_{t+1})] \approx r(s_t, a_t) + \gamma V(s_{t+1})$$

Thus, we rewrite the advantage as

$$A_\phi(s_t, a_t) \approx r(s_t, a_t) + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

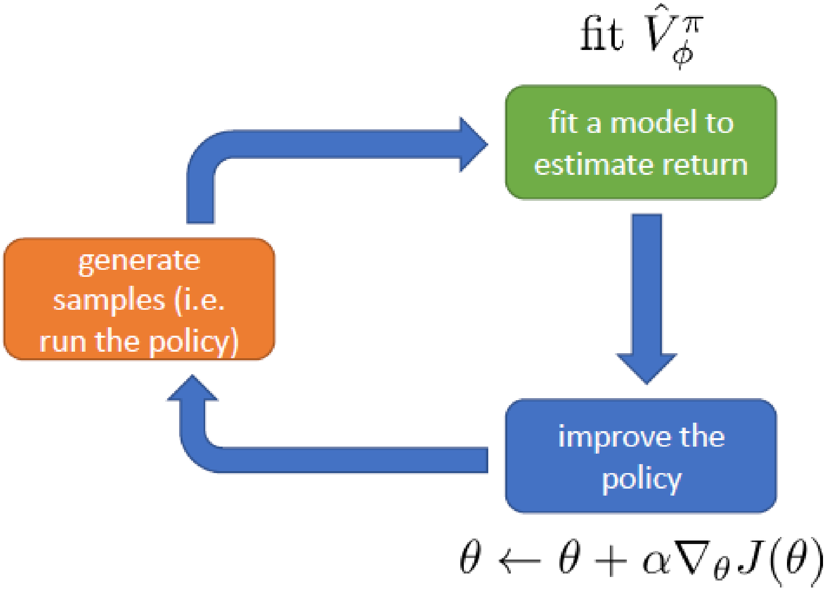
$V_\phi(s_t)$ is learned by

$$\min_\phi A_\phi^2(s_t, a_t)$$

All other steps of A2C remain the same as the Actor-Critic.

Understanding Actor-Critic

- The actor decides which action to take (actor: policy)
- The critic tells the actor how good its action was and how it should adjust (critic: value/Q-function)



References of Policy Gradient

- Classic papers
  - Williams (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning: introduces REINFORCE algorithm
  - Baxter & Bartlett (2001). Infinite-horizon policy-gradient estimation: temporally decomposed policy gradient (not the first paper on this! see actor-critic section later)
  - Peters & Schaal (2008). Reinforcement learning of motor skills with policy gradients: very accessible overview of optimal baselines and natural gradient
- Deep reinforcement learning policy gradient papers
  - Levine & Koltun (2013). Guided policy search: deep RL with importance sampled policy gradient (unrelated to later discussion of guided policy search)
  - Schulman, L., Moritz, Jordan, Abbeel (2015). Trust region policy optimization: deep RL with natural policy gradient and adaptive step size
  - Schulman, Wolski, Dhariwal, Radford, Klimov (2017). Proximal policy optimization algorithms: deep RL with importance sampled policy gradient

References of Actor-Critic

- Classic papers
  - Sutton, McAllester, Singh, Mansour (1999). Policy gradient methods for reinforcement learning with function approximation: actor-critic algorithms with value function approximation
- Deep reinforcement learning actor-critic papers
  - Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, Kavukcuoglu (2016). Asynchronous methods for deep reinforcement learning: A3C -- parallel online actor-critic
  - Schulman, Moritz, L., Jordan, Abbeel (2016). High-dimensional continuous control using generalized advantage estimation: batch-mode actor-critic with blended Monte Carlo and function approximator returns
  - Gu, Lillicrap, Ghahramani, Turner, L. (2017). Q-Prop: sample-efficient policy-gradient with an off-policy critic: policy gradient with Q-function control variate

Thank you!