Skoltech

Skolkovo Institute of Science and Technology

Introduction to Reinforcement Learning: Deep Policy Gradient

Planning Algorithms in AI

Gonzalo Ferrer
Timur Akhtyamov

Skoltech, 6 December 2021

Summary from L08

- Markov Property and Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$
- The concept of the reward, return and discounted reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$\gamma \in [0, 1]$ is the discount factor, or discount rate. Lower values place more emphasis on immediate rewards

- State-value function (V-function) and Action-value function (Q-function):

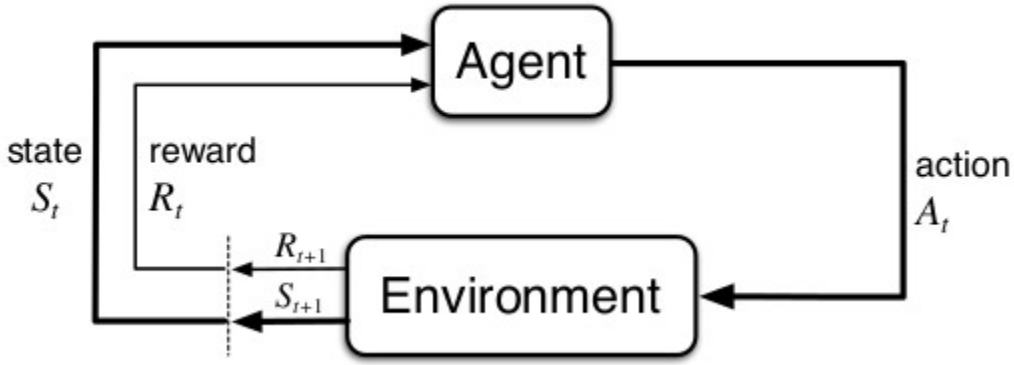
$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right]$$

- Bellman equation, dynamic programming, Value iteration

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(s') \middle| S_t = s, S_{t+1} = s' \right]$$

Reinforcement Learning: basic idea

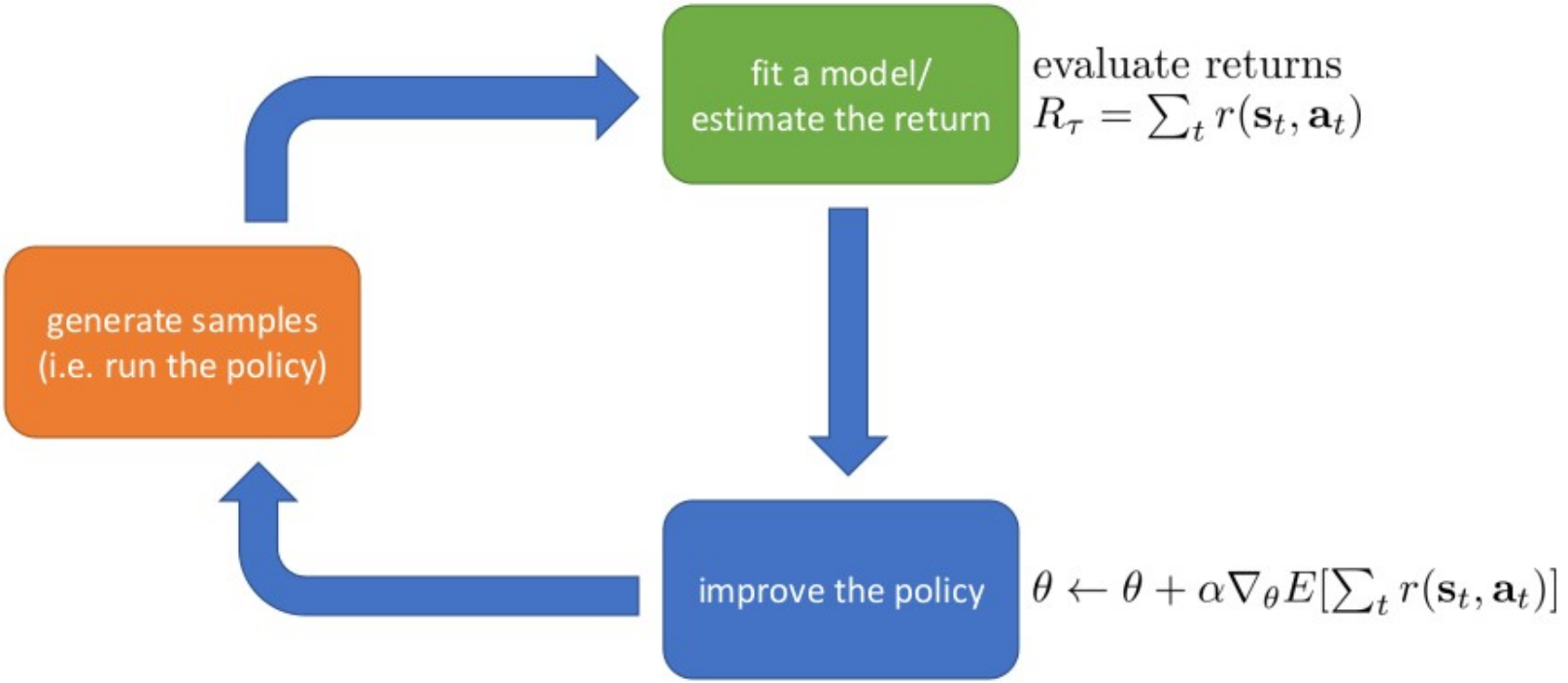
- Let's consider the following scheme from L08



- In the MDP problem, all elements in the tuple where known
- In the *Reinforcement Learning* setting, the **environment** is beyond our knowledge/control and we can only learn based on the **interaction** with it.
- Now, there are many approaches: model-based, estimate value function or q(), or directly estimate the policy function.

Direct Policy Gradient

- Direct optimization of the **parameterized** policy $\pi_\theta(a_t|s_t,\theta)$



Levine “Deep Reinforcement Learning” 2018

Policy Gradient: derivation

$$\theta^* = \arg\max_\theta \underbrace{\mathbb{E}_\pi \left[\sum_t r(s_t, a_t) \right]}_{J(\theta)}$$

$$J(\theta) = \mathbb{E}_\pi \left[G(\tau) \right] = \int \pi_\theta(\tau) G(\tau) d\tau$$

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

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) G(\tau) d\tau = \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) G(\tau) d\tau = \mathbb{E}_\pi \left[\nabla_\theta \log \pi_\theta(\tau) G(\tau) \right]$$

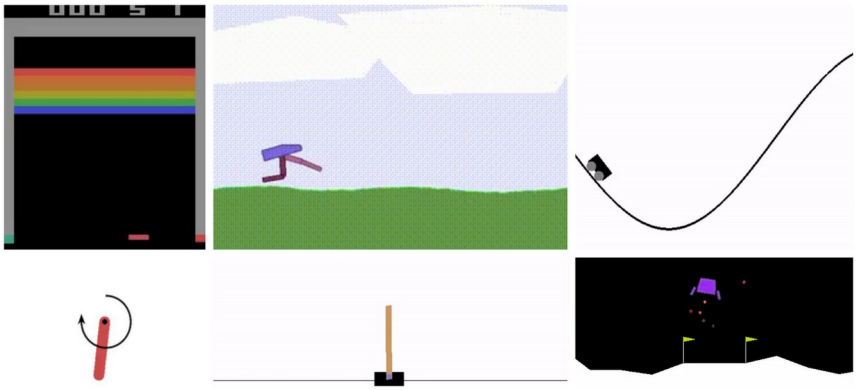
REINFORCE: a Policy Gradient algorithm

- Intuition: make “good” actions more probable
- Usually, converges at least to the local minima
- Unfortunately, often has high variance, thus requires many samples for policy gradient estimation
    - Partially fixed in the extension of the algorithm - *REINFORCE with baselines.*
- How policy can be expressed and parameterized?

Deep Reinforcement Learning

- Any function approximator can be used for parameterizing the policy
- Particularly, deep neural networks can be used
- The usage of deep neural networks for approximation of policy, V- and Q-functions enables the Deep Reinforcement Learning (DRL)
- DRL has many advantages over other approaches:
  - Tackles the *curse of dimensionality* by automatic feature extraction
  - Enables usage of optimization methods applied in Deep Learning
  - Enables usage of any network architecture which allows to use DRL algorithms with various types of data

OpenAI Gym



- The popular package that contains a unified interface for RL environments and implementation of various environments with different tasks
- Included environments have discrete or continuous observation space and discrete or continuous action space
- We are going to implement REINFORCE using neural network as a policy (using PyTorch framework) and see how it works in *CartPole-v0* environment

Policy Gradient: approximation

$$J(\theta) = \mathbb{E}_\pi \left[\sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t})$$

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

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} (\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}))(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t}))$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \left(\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}) \right) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \hat{Q}_{i,t}$$