Deep Learning

Lecture 10

reused a parts of a good (theory) course at HSE University (http://wiki.cs.hse.ru/Reinforcement_learning_2022_2023)

Lecturer: Daniil Tiapkin (https://d-tiapkin.github.io/)

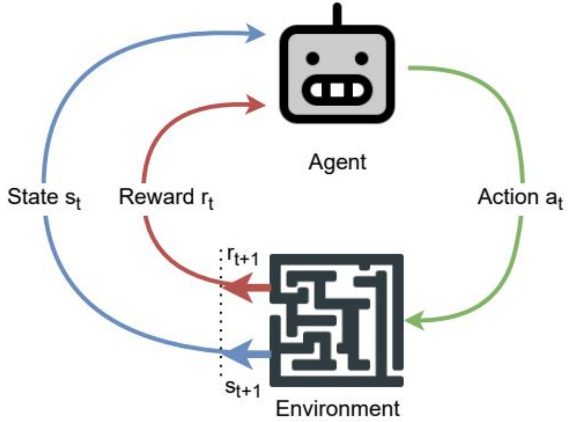
In previous lecture

- Markov Decision Processes
- Value Iteration -> Least-Squares Value Iteration
- Exploration-Exploitation Tradeoff
- Experience Replay (Replay Buffer)
- Deep Q-Network (DQN)
- Atari & Procgen Benchmarks

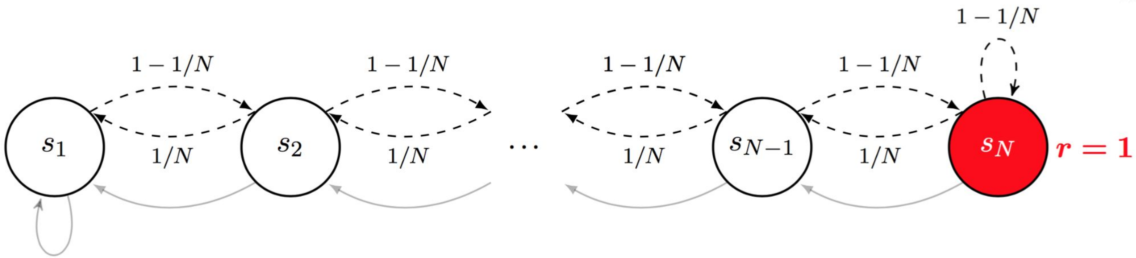
Recap: RL and Markov Decision Process

Markov Decision Process is a 5-tuple (*S, A, P, R,* γ)

- *S* - state space;
- *A* - action space;
- *P(s' | s,a)* - transition probability kernel;
- *R(s,a)* - reward distribution (with a mean reward *r(s,a)*);
- γ - discounting factor



Policy π: rule to choose next action given current state



MDP Example: Chain

Recap: Value function and Q-function

Goal of the agent: find a policy $\boldsymbol{\pi}$ that maximizes the expected sum of rewards:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right] \qquad Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a\right],$$

$$r_t \sim \mathrm{R}(\cdot|s_t, a_t),\, s_{t+1} \sim \mathrm{P}(\cdot|s_t, a_t),\, a_t \sim \pi(\cdot|s_t).$$

A policy that attains maximum for each states is called *optimal.*

RL Objective

- Goal of RL – find good policy, so let us parameterize policies!

$$\pi_\theta(a|s) = \frac{\exp(f_\theta(s,a))}{\sum_{a' \in \mathcal{A}} \exp(f_\theta(s,a'))}$$

- Let s_0 be an initial state. Then our objective is $J(\theta) = V^{\pi_\theta}(s_0)$

Overall, we recast RL problem as **optimization problem**

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

Q: Why this problem is difficult?

Policy Optimization – first difficulties

- Recall the definition of value:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right],$$

- Policy is **hidden** inside the expectation in a very non-direct way
- How to compute gradients with respect to the policy to perform gradient ascent (for example)?

$$\theta_{t+1} = \theta_t + \alpha_t \nabla J(\theta_t)$$

Policy Gradient Theorem

Theorem 15 (Policy Gradient Theorem). *Let* $M = (\mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{R}, \gamma)$ *be discounted MDP. Let* $B\colon \mathcal{S} \to \mathbb{R}$ *be any bounded function on states (so called baseline). Then the gradient of value function with respect to parameter* θ *is equal to*

$$\nabla_\theta V^{\pi_\theta}(s_0) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot (Q^{\pi_\theta}(s_t, a_t) - B(s_t))\right].$$

- We will discuss what is baseline and how to choose it later.
  Now just think that B = 0.

Proof of Policy Gradient Theorem, pt.1

By Bellman equations $$V^{\pi_\theta}(s_0) = \sum_{a_0 \in \mathcal{A}} \pi_\theta(a_0|s_0) Q^{\pi_\theta}(s, a),$$

By chain rule:

$$\nabla_\theta V^{\pi_\theta}(s_0) = \sum_{a_0 \in \mathcal{A}} [\nabla_\theta \pi_\theta(a_0|s_0) Q^{\pi_\theta}(s_0, a_0)) + \pi_\theta(a_0|s_0) \cdot \nabla Q^{\pi_\theta}(s_0, a_0)].$$

“Log-derivative trick”: $$\nabla_\theta \log \pi_\theta(a|s) = \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)},$$

Overall:

$$\nabla_\theta V^{\pi_\theta}(s_0) = \mathbb{E}_{a_0 \sim \pi_\theta(\cdot|s_0)} [\nabla_\theta \log \pi_\theta(a_0|s_0) \cdot Q^{\pi_\theta}(s_0, a_0) + \nabla Q^{\pi_\theta}(s_0, a_0)].$$

Proof of Policy Gradient Theorem, pt.2

Bellman equations:

$$Q^\pi(s,a) = r(s,a) + \gamma \mathrm{P} V^\pi(s,a),$$

$$V^\pi(s) = \sum_{a \in \mathcal{A}} Q^\pi(s,a)\pi(a,s)$$

As a result:

$$\nabla_\theta Q^{\pi_\theta}(s_0,a_0) = \nabla_\theta\big[r(a_0,s_0) + \gamma \mathbb{E}_{s_1 \sim \mathrm{P}(\cdot|s_0,a_0)}[V^{\pi_\theta}(s_1)]\big] = \gamma \mathbb{E}_{s_1 \sim \mathrm{P}(\cdot|s_0,a_0)}[\nabla_\theta V^{\pi_\theta}(s_1)],$$

Plug-in in derivative for value:

$$\nabla_\theta V^{\pi_\theta}(s_0) = \mathbb{E}_{a_0 \sim \pi_\theta(\cdot|s_0), s_1 \sim \mathrm{P}(\cdot|s_0,a_0)}[\nabla_\theta \log \pi_\theta(a_0|s_0) \cdot Q^{\pi_\theta}(s_0,a_0) + \gamma \nabla_\theta V^{\pi_\theta}(s_1)].$$

Rolling-out we obtain

$$\nabla_\theta V^{\pi_\theta}(s_0) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot Q^{\pi_\theta}(s_t,a_t)\right],$$

Proof of Policy Gradient Theorem, pt.3

To finish the proof, we have to show $\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot B(s_t)\right] = 0.$

$$
\begin{aligned}
\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot B(s_t)\right] &= \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t \sim (\mathrm{P}^{\pi_\theta})^t(\cdot|s_0), a \sim \pi_\theta(\cdot|s_t)}[\nabla_\theta \log \pi_\theta(a|s_t) \cdot B(s_t)] \\
&= \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t \sim (\mathrm{P}^{\pi_\theta})^t(\cdot|s_0)}\left[\sum_{a \in \mathcal{A}} \pi_\theta(a|s_t) \nabla_\theta \log \pi_\theta(a|s_t) \cdot B(s_t)\right] \\
&= \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t \sim (\mathrm{P}^{\pi_\theta})^t(\cdot|s_0)}\left[\sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(a|s_t) \cdot B(s_t)\right] \\
&= \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t \sim (\mathrm{P}^{\pi_\theta})^t(\cdot|s_0)}\left[\nabla_\theta \underbrace{\left(\sum_{a \in \mathcal{A}} \pi_\theta(a|s_t)\right)}_{1} \cdot B(s_t)\right] = 0.
\end{aligned}
$$

Policy Gradient Theorem

Theorem 15 (Policy Gradient Theorem). *Let $M = (\mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{R}, \gamma)$ be discounted MDP. Let $B\colon \mathcal{S} \to \mathbb{R}$ be any bounded function on states (so called baseline). Then the gradient of value function with respect to parameter θ is equal to*

unknown!

$$\nabla_\theta V^{\pi_\theta}(s_0) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot (Q^{\pi_\theta}(s_t, a_t) - B(s_t))\right].$$

unknown!

- **Q:** How to apply it in the real life?

Policy Gradient Estimation

- Assume that we used a current policy to obtain trajectory

$$(s_0, a_0, r_0, \ldots, s_t, a_t, r_t, \ldots).$$

- Estimate Q-value: $$Q^{\pi_\theta}(s_t, a_t) \approx G_t = \sum_{k=t}^{\infty} \gamma^{k-t} r_k.$$

- Estimate outer expectation $$\hat{\nabla} J(\theta) = \sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot G_t.$$

Lemma 5. *$\hat{\nabla} J(\theta)$ is an unbiased estimate of $\nabla J(\theta)$.*

REINFORCE

Algorithm that uses the following gradient estimate to perform SGD is called REINFORCE

Algorithm 6 REINFORCE

Input: MDP $M = (\mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{R}, H)$;
Initialize: θ_0;
for $k = 0, 1, \ldots,$ **do**
  Play policy π_{θ_k} and receive a trajectory $s_0^k, a_0^k, r_0^k, \ldots, s_T^k, a_T^k, r_T^k$.
  Compute estimates of Q-function $G_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$ for all $t = 0, \ldots, T$;
  Compute estimate of policy gradient $\hat{\nabla} J(\theta_k) = \sum_{t=0}^{T} \gamma^t \nabla_\theta \pi_\theta(a_t|s_t) \cdot G_t$;
  Perform a stochastic gradient step $\theta_{k+1} = \theta_k + \alpha_k \hat{\nabla} J(\theta_k)$.
end for

Remark. Algorithm can utilize only trajectories that were collected during the training, such algorithms are called *on-policy.*

Algorithms that can utilize data obtained by other policy are called *off-policy*.

Problems of REINFORCE

- Inefficient sample utilization;
  - Requires fast simulator!

- Large variance that leads to slow convergence;
  - Can be handled somehow by working with several parallel environments and collecting trajectories.

Variance Reduction: Actor-Critic Algorithm

Idea 1. Let’s estimate Q-value smarter, we have Bellman equations for it!

$$Q^\pi(s,a) = r(s,a) + \gamma \mathrm{P} V^\pi(s,a),$$
$$V^\pi(s) = \sum_{a \in \mathcal{A}} Q^\pi(s,a)\pi(a,s)$$

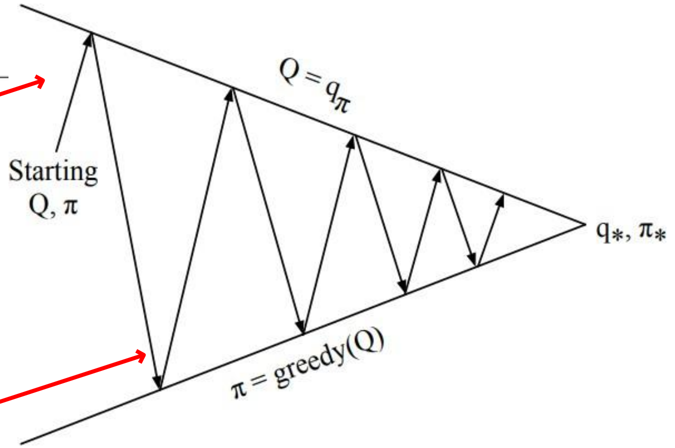
We can do it in DQN-fashion! $Q_\psi \approx Q^{\pi_\theta},$

$$\hat{\nabla}_{\mathrm{AC}} J(\theta) = \sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot Q_\psi(s_t, a_t)$$

Remember this guy? Connection to policy iteration

Algorithm 1 Policy Iteration for discounted MDPs

Input: MDP $M = (\mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{R}, \gamma)$, the immediate reward function r, iterations budget T
Initialize: π^0 as some set of policies;
for $t \in [T]$ **do**
  Compute Q^{π^t} by solving Bellman equations (see Theorem 4);
  Find π^{t+1} as a greedy policy w.r.t. Q^{π^t}.
end for
Output: estimate of optimal policy π^T.



Policy Evaluation steps

Policy Improvement steps

Variance Reduction: baseline selection

Idea 2. Use baseline!

Theorem 15 (Policy Gradient Theorem). *Let $M = (\mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{R}, \gamma)$ be discounted MDP. Let $B\colon \mathcal{S} \to \mathbb{R}$ be any bounded function on states (so called baseline). Then the gradient of value function with respect to parameter θ is equal to*

$$\nabla_\theta V^{\pi_\theta}(s_0) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot (Q^{\pi_\theta}(s_t, a_t) - B(s_t))\right].$$

Natural choice: B(s) is value function!

(this choice is not unique and is not optimal!)

Advantage Actor Critic (A2C)

Remark 2. The baseline is needed for variance reduction purposes. The most common choice is $B(s) = V^{\pi_\theta}(s)$ that leads to the following view on policy gradient theorem

$$\nabla_\theta V^{\pi_\theta}(s_0) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot A^{\pi_\theta}(s_t, a_t)\right],$$

where $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ is *advantage* function.

Q: How to estimate advantage function?

Advantage estimation

First, let us provide unbiased estimate:

$$
\begin{aligned}
A^{\pi_\theta}(s_t, a_t) &= Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t) \\
&= \mathbb{E}_{r_t \sim \mathrm{R}(s_t, a_t) s'_t \sim \mathrm{P}(s_t, a_t)}\left[r_t + \gamma V^{\pi_\theta}(s'_t)\right] - V^{\pi_\theta}(s_t) \\
&\approx r_t + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)
\end{aligned}
$$

In this estimation we need a network only for V-function, that is usually much easier to learn! We can do it through Bellman equations and optimizing TD loss

$$
\mathcal{L}_{\text{critic}}(\psi) = \sum_{t=1}^{T} \left(V_\psi(s_t) - r_t - \gamma V_{\tilde{\psi}}(s_{t+1}) \right)^2
$$

target network, usually just stopgrad here

A2C as Policy Iteration

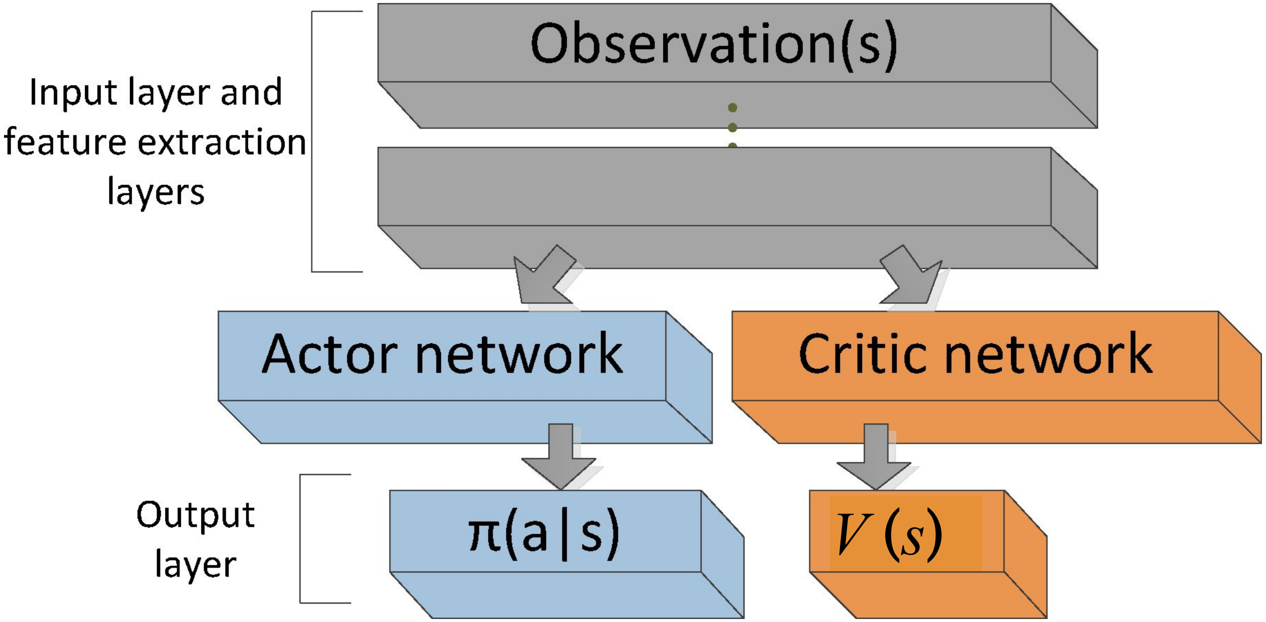
Policy Improvement: step by

$$\hat{\nabla}_{\mathrm{A2C}} J(\theta) = \sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot (r_t + \gamma V_\psi(s_{t+1}) - V_\psi(s_t))$$

Policy evaluation: gradient step on the following loss

$$\mathcal{L}_{\mathrm{critic}}(\psi) = \sum_{t=1}^{T} \Big(V_\psi(s_t) - r_t - \gamma V_{\tilde{\psi}}(s_{t+1}) \Big)^2$$

Architecture Details



source: AI Masters RL course (https://ozonmasters.ru/reinforcementlearning)

Exploration-Exploitation Trade-off

For DQN we need exploration to satisfy constraints on the replay buffer generation distribution.

Q: Do we need additional exploration for policy gradient methods?

We just reformulate RL problem as an optimization problem and perform SGD!

$$\max_{\theta} J(\theta)$$

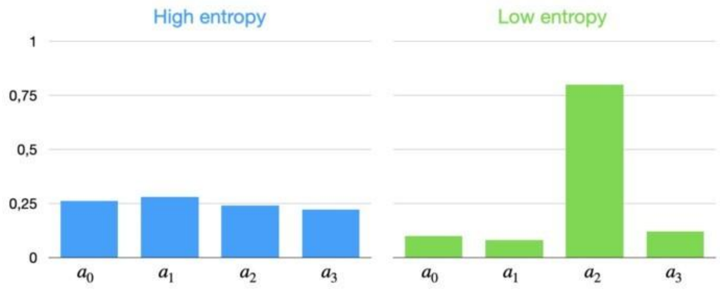
Exploration for Policy-Gradient methods

A: We still need exploration since we can stuck in a local optimum!

One common way: add negative entropy to the loss function for actor network to maximize it!

$$\mathcal{H}(\pi_\theta(s_t)) = \sum_{a \in \mathcal{A}} \pi_\theta(a|s_t) \log\left(\frac{1}{\pi_\theta(a|s_t)}\right)$$

- Encourages exploration;
- Introduces bias, so a coefficient should be small!



Advanced topics on Policy Gradient methods

- Generalized Advantage Estimation (GAE)
- Introduction of a small delay into on-policy generation
  - Proximal Policy Optimization (PPO);
  - Trust-Region Policy Optimization (TRPO);
  - Asynchronous Advantage Actor Critic (A3C) and Impala;
  - Mirror Descent Policy Optimization (MDPO);
- Different types of baselines:
  - Hindsight Credit Assignment;
  - Action-dependent baselines;

- IMPLEMENTATION MATTERS IN DEEP POLICY GRADIENTS: A CASE STUDY ON PPO AND TRPO

  https://arxiv.org/abs/2005.12729

Recap for RL

- What is RL?
- Markov Decision Process;
- Why don’t we use it everywhere and where it is useful;
- V-function, Q-function;
- Value Iteration, Policy iteration;
- Least-Squared Value Iteration;
- Exploration-Exploitation trade-off;
- Experience Replay (Replay Buffer);
- Deep Q-Network (DQN);
- Policy Gradient Theorem, log-derivative trick;
- REINFORCE;
- Actor-Critic algorithm and A2C;