Lecture 1: Introduction to Reinforcement learning. Cross-Entropy Method.

Anton Plaksin

Markov Decision Process

Markov Property

$$\mathbb{P}[S_{t+1}|S_t, A_t] = \mathbb{P}[S_{t+1}|S_0, A_0, S_1, A_1 \dots, S_t, A_t]$$

$$\mathbb{P}[R_t|S_t, A_t] = \mathbb{P}[R_t|S_0, A_0, S_1, A_1 \dots, S_t, A_t] = 1$$

Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{P}_0, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a state space
- $\mathcal{A}$ is an action space
- $\mathcal{P}$ is a transition probability function

$$\mathcal{P}(s'|s, a) = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]$$

- $\mathcal{P}_0$ — an initial state probability function
- $\mathcal{R}$ — a reward function

$$\mathcal{R}(s, a) = R_t \quad \Leftrightarrow \quad \mathbb{P}[R_t|S_t = s, A_t = a] = 1$$

- $\gamma \in [0, 1]$ — discount coefficient

MDP with final states

Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{P}_0, \mathcal{R}, \gamma \rangle$

The agent's goal is to maximize

$$G = \sum_{t=0}^{\infty} \gamma^t R_t$$

Markov Decision Process $\langle \mathcal{S}, \mathcal{S}_F, \mathcal{A}, \mathcal{P}, \mathcal{P}_0, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}_F$ — a set of final states

The agent's goal is to maximize

$$G = \sum_{t=0}^{T} \gamma^t R_t, \quad \text{if} \quad S_T \in \mathcal{S}_F$$

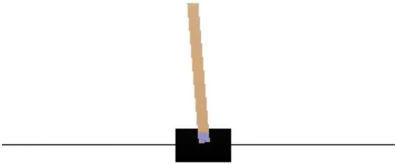
or

$$G = \sum_{t=0}^{\infty} \gamma^t R_t \quad \text{if} \quad S_t \notin \mathcal{S}_F$$

Example: Breakout Atari Game

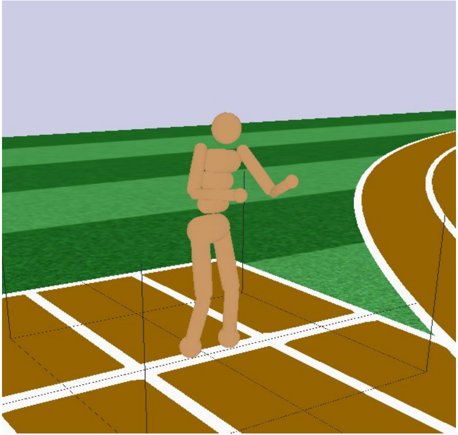
- States: pixels
- Actions: $\rightarrow$, $\leftarrow$, «0»
- Rewards: points
- Final states: when the ball falls down

Example: Cartpole



- States: $\mathbb{R}^4$
- Actions: $\rightarrow$, $\leftarrow$, «0»
- Rewards: $+1$ for each step
- Final states: when the pole falls down

Example: Humanoid



- States: $\mathbb{R}^{26}$
- Actions: $\mathbb{R}^{6}$
- Rewards: $+1$ for each step
- Final states: when the Humanoid falls down

OpenAI Gym Interface

initial_stete = env.reset()

- initial_stete — an initial state $S_0 \sim \mathcal{P}_0$
- env.state = initial_stete

next_stete, reward, done, info = env.step(action)

- action — a current action A_t
- next_stete — a next state $S_{t+1} \sim \mathcal{P}(S_{t+1}|S_t, A_t)$
- reward — a current reward $R_t = \mathcal{R}(S_t, A_t)$
- done — the inclusion $S_{t+1} \in \mathcal{S}_F$ holds or not
- info — an additional information
- env.state = next_stete

Stochastic policy

$$\pi(a|s) \in [0,1], \quad a \in \mathcal{A}, \quad s \in \mathcal{S}$$

- Set π
- Agent starts from the initial state $S_0 \sim \mathcal{P}_0$
- acts $A_0 \sim \pi(\cdot|S_0)$
- gets the reward $R_0 = \mathcal{R}(S_0, A_0)$ and goes to the next state $S_1 \sim \mathcal{P}(\cdot|S_0, A_0)$
- acts $A_1 \sim \pi(\cdot|S_1)$
- gets the reward $R_1 = \mathcal{R}(S_1, A_1)$ and goes to the next state $S_2 \sim \mathcal{P}(\cdot|S_1, A_1)$
- $\ldots$
- $\tau = \{S_0, A_0, S_1, A_1, S_2, A_2, \ldots\}, \quad G(\tau) = \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(S_t, A_t)$

The Reinforcement Learning problem

$$\mathbb{E}_\pi[G] \quad \longrightarrow \quad \max_\pi$$

What is $\mathbb{E}_\pi[G]$?

$$\tau = \{S_0, A_0, S_1, A_1, S_2, A_2, \ldots\}$$

$$\begin{aligned}
\mathbb{P}(\tau) &= \mathbb{P}(S_0)\mathbb{P}(A_0|S_0)\mathbb{P}(S_1|S_0, A_0) \\
&\times \mathbb{P}(A_1|S_1)\mathbb{P}(S_2|S_1, A_1) \\
&\times \mathbb{P}(A_2|S_2)\mathbb{P}(S_3|S_2, A_2) \\
&\times \cdots
\end{aligned}$$

$$\mathbb{P}(\tau|\pi) := \mathcal{P}_0(S_0) \prod_{t=0}^{\infty} \pi(A_t|S_t)\mathcal{P}(S_{t+1}|S_t, A_t)$$

$$\mathbb{E}_\pi[G] = \sum_\tau G(\tau)\mathbb{P}(\tau|\pi) \quad \text{or} \quad \mathbb{E}_\pi[G] = \int_\tau G(\tau)\mathbb{P}(\mathrm{d}\tau|\pi)$$

How to calculate $\mathbb{E}_\pi[G]$?

If $\mathcal{P}$, $\mathcal{P}_0$, π are deterministic

$$\mathbb{E}_\pi[G] = G(\tau)$$

General case

$$\mathbb{E}_\pi[G] \approx \frac{1}{K}\sum_{k=1}^{K} G(\tau_k), \quad \tau_k \sim \{\mathcal{P}, \mathcal{P}_0, \pi\}$$

Markov Decision Process

Markov Property

$$\mathbb{P}[S_{t+1}|S_t, A_t] = \mathbb{P}[S_{t+1}|S_1, A_1, S_2, A_2 \ldots, S_t, A_t]$$

$$\mathbb{P}[R_t|S_t, A_t] = \mathbb{P}[R_t|S_1, A_1, S_2, A_2 \ldots, S_t, A_t] = 1$$

Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ — a finite $(|\mathcal{S}| = n)$ state space
- $\mathcal{A}$ — a finite $(|\mathcal{A}| = m)$ action space
- $\mathcal{P}$ — a deterministic transition probability function

$$\mathcal{P}(s'|s, a) = \mathbb{P}[S_{t+1} = s'|S_t = s, A_t = a]$$

- $\mathcal{P}_0$ — a deterministic initial state function
- $\mathcal{R}$ — a reward function

$$\mathcal{R}(s, a) = R_t \quad \Leftrightarrow \quad \mathbb{P}[R_t|S_t = s, A_t = a] = 1$$

- $\gamma \in [0, 1]$ — discount coefficient

Reinforcement learning as an optimization problem

State and action spaces

$$\mathcal{S} = \{1, 2, \dots n\}, \quad \mathcal{A} = \{1, 2, \dots m\}$$

Policy

$$\pi(a|s) = \Pi_{a,s}, \quad a \in \mathcal{A}, \quad s \in \mathcal{A}$$

Finite-dimensional optimization problem

$$\max_{\Pi} f(\Pi),$$

где $f(\Pi) = \mathbb{E}_{\pi}[G]$

Cross-Entropy Method. General scheme

On each iteration:

- Policy evaluation. Seeking $\mathrm{E}_\pi[G]$
- Policy improvement. Seeking $\pi' \geq \pi$ $(\mathrm{E}_{\pi'}[G] \geq \mathrm{E}_\pi[G])$

Quantile (Percentile)

Let $q \in (0,1)$. q-quantile of the numbers $G_1, G_2, \ldots, G_K$ is a number γ_q such that

$$\frac{|\{G_k, k \in \overline{1,K} \colon G_k \leq \gamma_q\}|}{|\{G_k, k \in \overline{1,K}\}|} \geq q$$

$$\frac{|\{G_k, k \in \overline{1,K} \colon G_k \geq \gamma_q\}|}{|\{G_k, k \in \overline{1,K}\}|} \geq 1 - q$$

Let $p \in [0, 100]$. p-percentile is $(p/100)$-quantile

Cross-Entropy Method

Let π_0 be an initial (uniform) policy, N be a number of iterations, $q \in (0,1)$ — parameter for defining elite trajectories.
For each $n \in \overline{0,N}$, do

- (Policy evaluation) Acting in accordance with the current policy π_n, get K trajectories τ_k, $k \in \overline{1,K}$ and total rewards $G(\tau_k)$. Evaluate π_n:

$$\mathbb{E}_{\pi_n}[G] \approx V_{\pi_n} := \frac{1}{K}\sum_{k=1}^{K} G(\tau_k)$$

- (Policy improvement) Select «elite» trajectories $\mathcal{T}_n = \{\tau_k, k \in \overline{1,K} \colon G(\tau_k) > \gamma_q\}$ (γ_q — q-quantile of the numbers $G(\tau_k)$, $k \in \overline{1,K}$). If $\mathcal{T}_n \neq \emptyset$, then update policy as

$$\pi_{n+1}(a|s) = \frac{\text{number of pairs}(a|s)\text{ in trajectories from } \mathcal{T}_n}{\text{number of } s \text{ in trajectories from } \mathcal{T}_n}$$

What are the weaknesses of the algorithm?

- Requires a large number of sessions
- The policy update is highly dependent on randomness
- Problems with the stochastic environments
- State and action spaces must be finite

Weakness: The policy update is highly dependent on randomness

Solution:

- Laplace smoothing

$$\pi_{n+1}(a|s) = \frac{|(a|s) \in \mathcal{T}_n| + \lambda}{|s \in \mathcal{T}_n| + \lambda|\mathcal{A}|}, \quad \lambda > 0$$

- Policy smoothing

$$\pi_{n+1}(a|s) \leftarrow \lambda\pi_{n+1}(a|s) + (1-\lambda)\pi_n(a|s), \quad \lambda \in (0,1]$$

Weakness: Problems with the stochastic environments

Solution:

By stochastic policy π_n, sample deterministic policies $\pi_{n,m}$, $m \in \overline{1, M}$. According to them, get trajectories $\tau_{m,k}$, $m \in \overline{1, M}$, $k \in \overline{1, K}$. Define

$$V_{\pi_{n,m}} = \frac{1}{K} \sum_{k=1}^{K} G(\tau_{m,k})$$

Select «elite» trajectories $\mathcal{T}_n = \{\tau_{m,k}, m \in \overline{1, M}, k \in \overline{1, K} \colon V_{\pi_{n,m}} > \gamma_q\}$ (γ_q — q-quantile of the numbers $V_{\pi_{n,m}}$, $m \in \overline{1, M}$).