

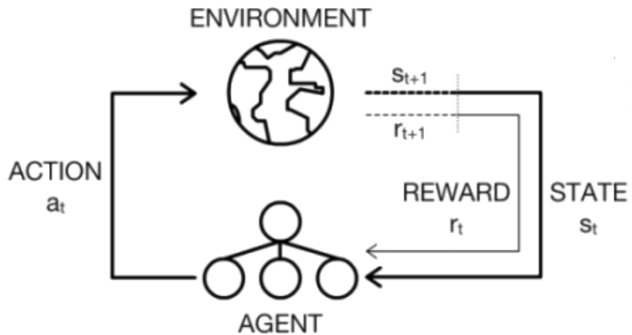
Глубокое обучение и вообще

Бекезин Никита

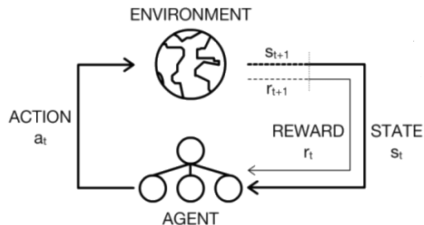
15 декабря 2021 г.

Введение в RL

Reinforcement Learning

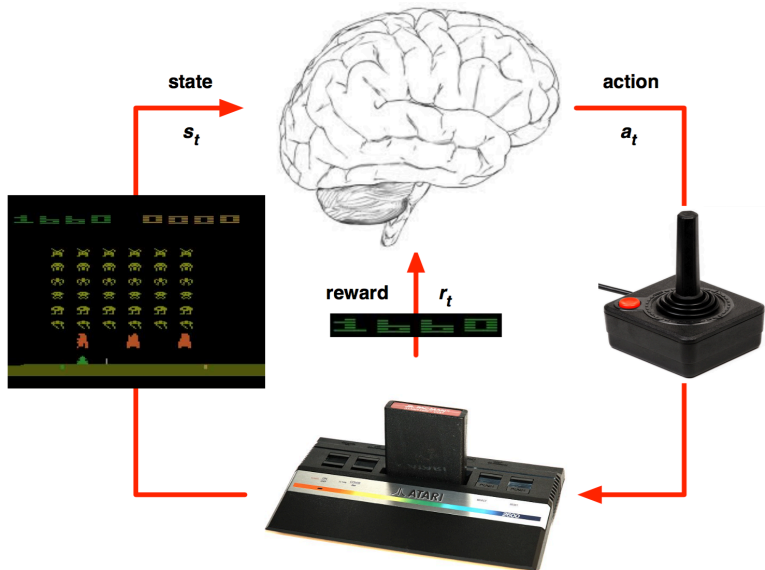


Reinforcement Learning

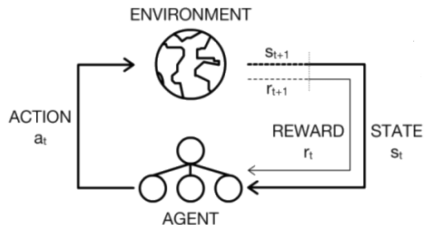


- s_t — состояние среды (state);
- a_t — действия агента (action);
- r_t — награда (reward) за действие;

Reinforcement Learning



Reinforcement Learning



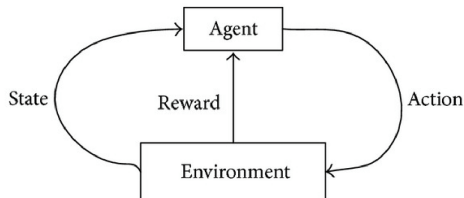
- $v(s)$ — **value function**, по state выдаёт оценку всех будущих reward;
- $p(a | s)$ — **policy function**, выводит вероятность действия в конкретном состоянии так, чтобы вероятность действия, максимизирующего reward была наибольшей
- $Q(s, a)$ — **Q-function**, сообщает reward для действия a в состоянии s .

Reinforcement Learning

- **Идея:** мы пытаемся выразить действия агента с помощью различных функций. Если мы пытаемся каждую функцию представить в виде нейросетки, мы входим в зону deep reinforcement learning.
- В зависимости от того, какую функцию мы оптимизируем, получаем разные алгоритмы

Следующий блок слайдов взят из лекций ШАД по RL

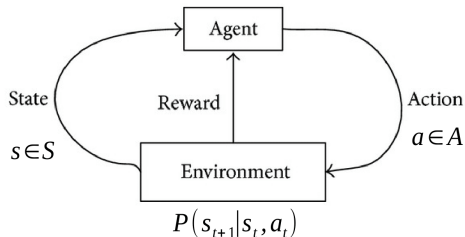
The MDP formalism



Markov Decision Process

- Environment states: $s \in \mathcal{S}$
- Agent actions: $a \in \mathcal{A}$
- Rewards $r \in \mathbb{R}$
- Dynamics: $P(s_{t+1} | s_t, a_t)$

The MDP formalism



Markov Decision Process
Markov assumption

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}) = P(s_{t+1}|s_t, a_t)$$

Total reward



Total reward for session:

$$R = \sum_t r_t$$

Agent's policy:

$$\pi(a|s) = P(\text{take action } a | \text{in state } s)$$

Problem: find policy with highest reward:

$$\pi(a|s) : E_{\pi}[R] \rightarrow \max$$

34

Objective

The easy way:

$E_{\pi} R$ is an expected sum of rewards
that agent with policy π earns per session

Objective

The easy way:

$E_{\pi} R$ is an expected sum of rewards
that agent with policy π earns per session

The hard way:

$$E_{s_0 \sim p(s_0), a_0 \sim \pi(a|s_0), s_1, r_0 \sim P(s', r|s, a)} \dots E_{s_T, r_T \sim P(s', r|s_{T-1}, a_{T-1})} [r_0 + r_1 + r_2 + \dots + r_T]$$

How do we solve it?

General idea:

Play a few sessions

Update your policy

Repeat

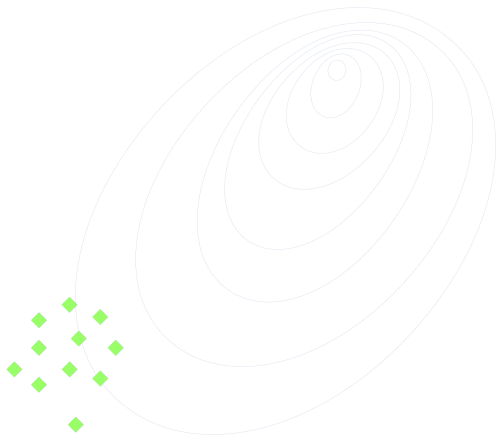
Crossentropy method

Initialize policy

Repeat:

- Sample $N[100]$ sessions
- Pick $M[25]$ best sessions, called **elite** sessions
- Change policy so that it prioritizes actions from elite sessions

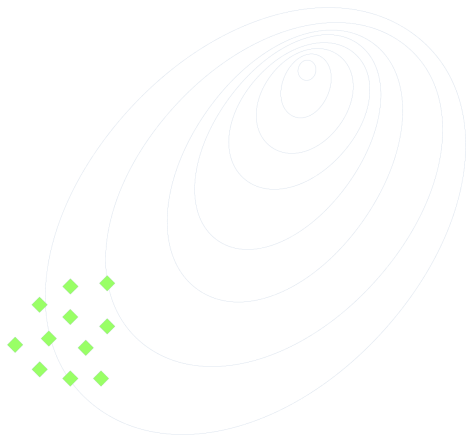
Step-by-step view



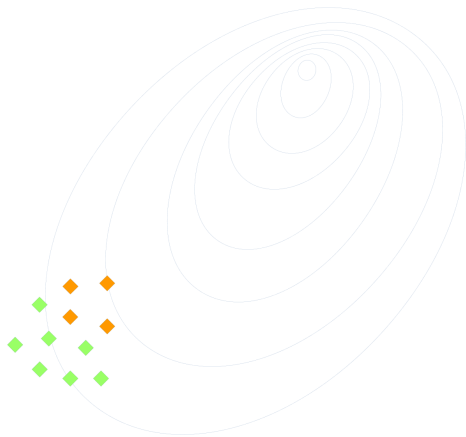
Step-by-step view



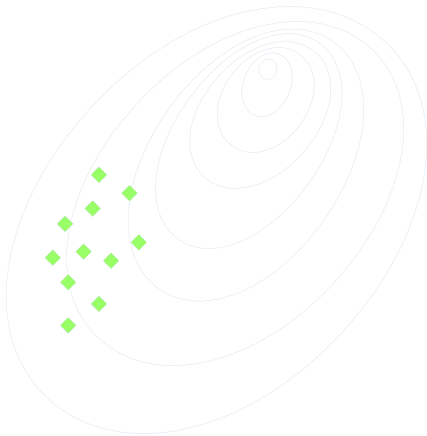
Step-by-step view



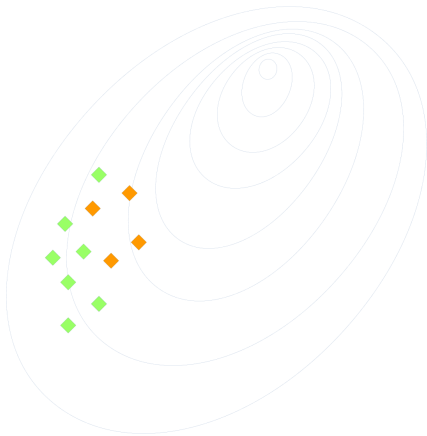
Step-by-step view



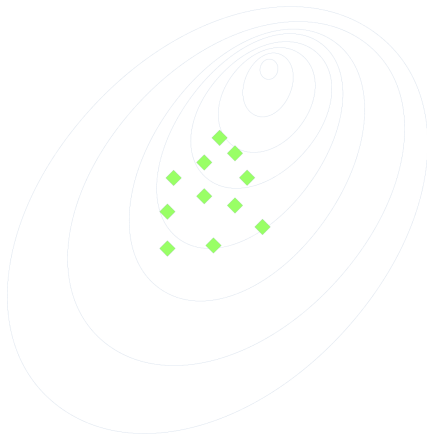
Step-by-step view



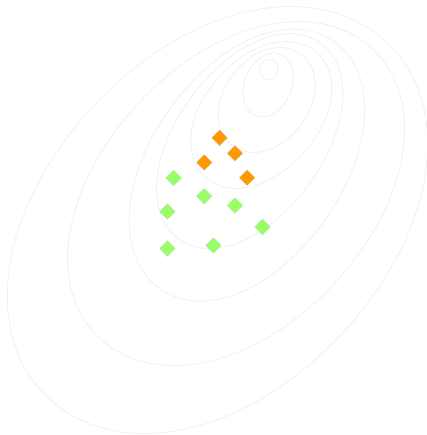
Step-by-step view



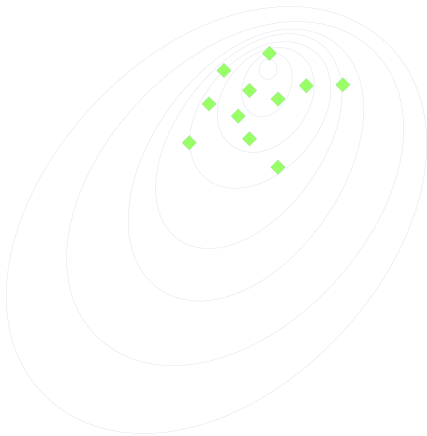
Step-by-step view



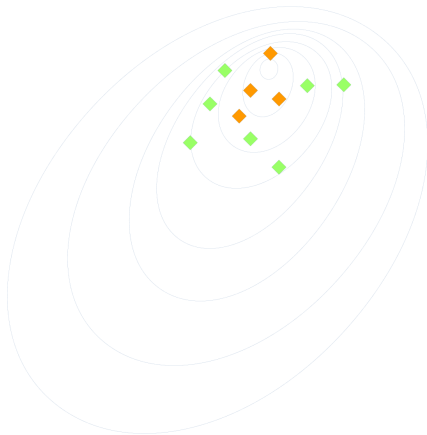
Step-by-step view



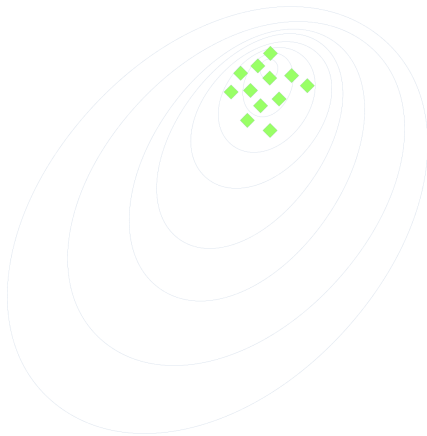
Step-by-step view



Step-by-step view



Step-by-step view



Tabular crossentropy method

- Policy is a matrix

$$\pi(a|s) = A_{s,a}$$

- Sample N games with that policy
- Get M best sessions (elites)

$$Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$$

Tabular crossentropy method

- Policy is a matrix

$$\pi(a|s) = A_{s,a}$$

- Sample N games with that policy
- Take M best sessions (elites)
- Aggregate by states


$$\pi(a|s) = \frac{\sum_{s_t, a_t \in \text{Elite}} [s_t = s][a_t = a]}{\sum_{s_t, a_t \in \text{Elite}} [s_t = s]}$$

Tabular crossentropy method

- Policy is a matrix

$$\pi(a|s) = A_{s,a}$$

- Sample N games with that policy
- Take M best sessions (elite)
- Aggregate by states

$$\pi(a|s) = \frac{\text{took } a \text{ at } s}{\text{was at } s}$$


The diagram shows a right-pointing arrow from the text "In M best games" to the fraction. From the tip of this arrow, two lines branch out to the left, each ending in a small black triangle (head) pointing to the numerator "took a at s" and the denominator "was at s" respectively.

In M best games

Grim reality

If your environment has infinite/large state space



Approximate crossentropy method

- Policy is approximated
 - Neural network predicts $\pi_w(a|s)$ given s
 - Linear model / Random Forest / ...

Can't set $\pi(a|s)$ explicitly

All state-action pairs from M best sessions

$$Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$$

Approximate crossentropy method

Neural network predicts $\pi_w(a|s)$ given s

All state-action pairs from M best sessions

$$Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$$

Maximize likelihood of actions in “best” games

$$\pi = \underset{\pi}{argmax} \sum_{s_i, a_i \in Elite} \log \pi(a_i | s_i)$$

Approximate crossentropy method

- Initialize NN weights $W_0 \leftarrow random$
- Loop:
 - Sample N sessions
 - $Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$
 - $W_{i+1} = W_i + \alpha \nabla \left[\sum_{s_i, a_i \in Elite} \log \pi_{W_i}(a_i | s_i) \right]$

Approximate crossentropy method

- Initialize NN weights `nn = MLPClassifier(...)`
- Loop:
 - Sample N sessions
 - $Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$
 - `nn.fit(elite_states, elite_actions)`

Continuous action spaces

- Continuous state space
- Model $\pi_w(a|s) = N(\mu(s), \sigma^2)$
 - $\mu(s)$ is neural network output
 - σ is a parameter or yet another network output
- Loop:
 - Sample N sessions
 - elite = take M best sessions and concatenate
 - $$W_{i+1} = W_i + \alpha \nabla \left[\sum_{s_i, a_i \in \text{Elite}} \log \pi_{W_i}(a_i | s_i) \right]$$

What changed?

Approximate crossentropy method

- Initialize NN weights `nn = MLPRegressor(...)`
- Loop:
 - Sample N sessions
 - $Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$
 - `nn.fit(elite_states, elite_actions)`

Almost nothing!

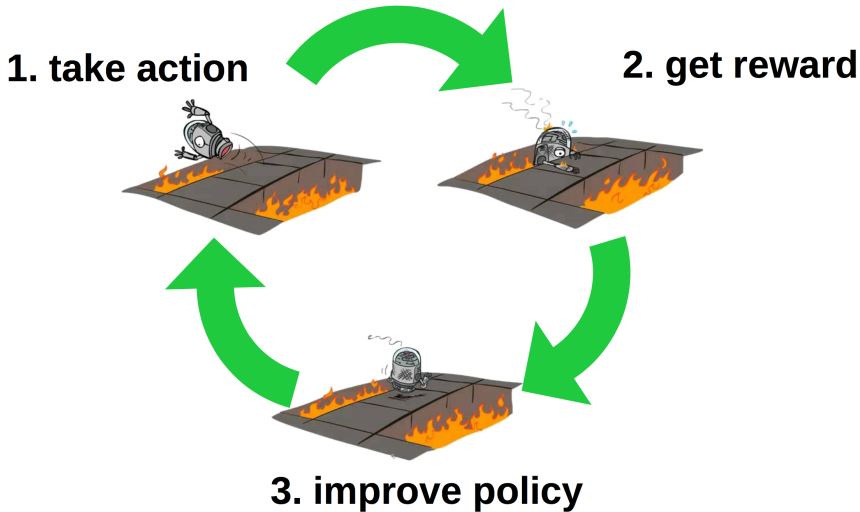
Какой бывает награда

Часто используется такой вид награды: $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$

Фактор дисконтирования γ нужен, чтобы очень далёкое будущее влияло на текущее состояние сети не так сильно, как недалёкое. Когда мы просим сеть разобраться с ближайшим будущим, ей легче разобраться в ситуации и обучение стабильнее.

Иногда агент попадает в очень хорошее состояние и начинает постоянно получать очень большую награду. Из-за этого нейросетке сложно понимать, что 1000 лучше 999. Хочется уменьшить масштаб \Rightarrow центрирование и нормирование. Так, можно отнормировать вектор R_t , полученный в ходе 100 сессий.

Reinforcement Learning - это просто!



Reinforcement Learning - это просто!

