Reinforcement Learning

Model-free RL

Александр Костин
telegramm: @Ko3tin
LinkedIn: kostinalexander

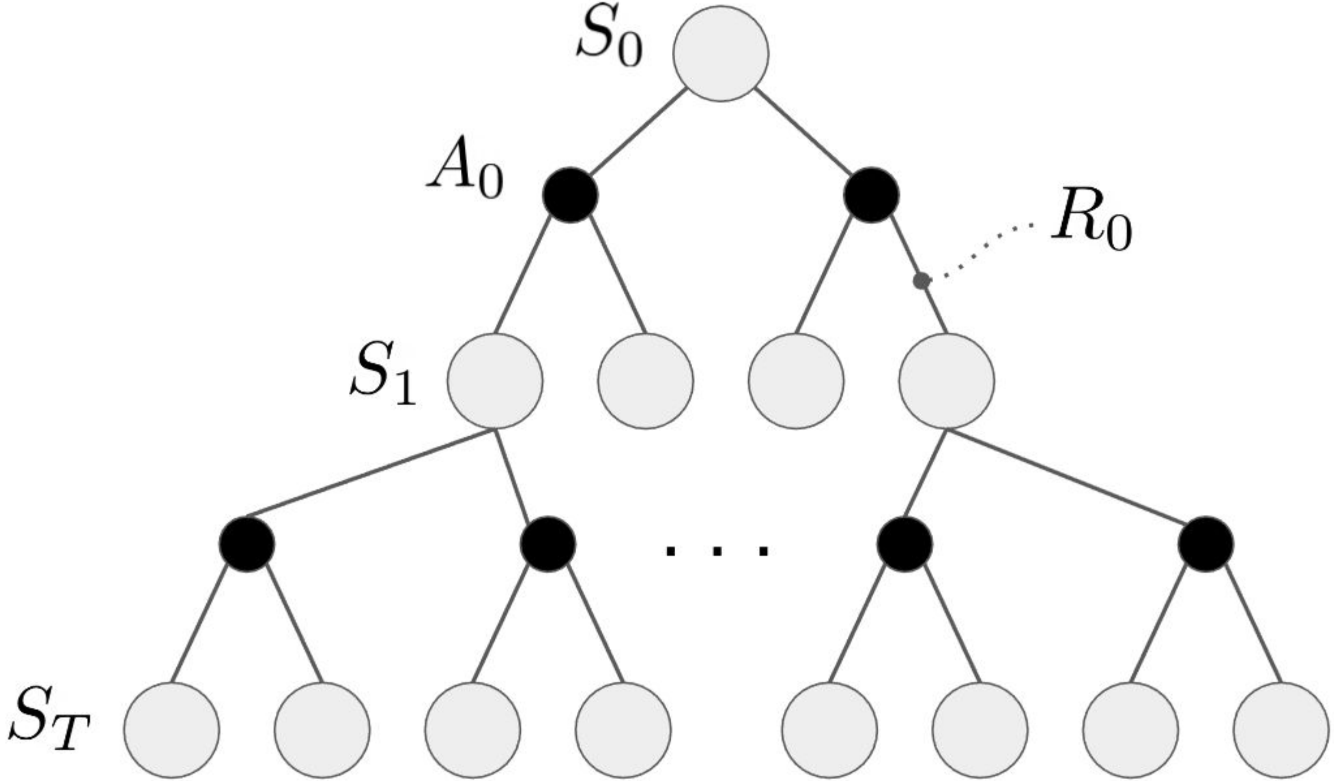
Награда

$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T)$ - траектория

$R = \sum_{t=0}^{T} r_t$ - награда за траекторию

$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \gamma^k r_{t+k} = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$ - return на **t** шаге

Как найти оптимальную политику?



Функции ценности

Ценность состояния **s**:

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,s'} p\big(r, s' \mid s, a\big)\big[r + \gamma V_\pi\big(s'\big)\big]$$

Ценность действия **a** в состоянии **s**:

$$Q_\pi(s, a) = \sum_{r,s'} p\big(r, s' \mid s, a\big)\left[r + \gamma \sum_a \pi\big(a \mid s'\big) Q_\pi\big(s', a\big)\right]$$

Связь Q и V функций:

$$V_\pi(s) = \sum_a \pi(a \mid s) Q_\pi(s, a) = E_{a \sim \pi(. \mid s)}[Q_\pi(s, a)]$$

Операторы Беллмана

Оператор Беллмана для ожидания V:

$$[T^{\pi}V](s) = E_{r,s' \mid s,a=\pi(s)}\left[r + \gamma V\left(s'\right)\right]$$

Оператор Беллмана для ожидания Q:

$$[T^{\pi}Q](s,a) = E_{r,s' \mid s,a}\left[r + \gamma E_{a' \sim \pi(s')}\left[Q\left(s',a'\right)\right]\right]$$

Оператор Беллмана для оптимальности V:

$$[TV](s) = \max_{a} E_{r,s' \mid s,a}\left[r + \gamma V\left(s'\right)\right]$$

Оператор Беллмана для оптимальности Q:

$$[T^{\pi}Q](s,a) = E_{r,s' \mid s,a}\left[r + \gamma \max_{a'}\left[Q\left(s',a'\right)\right]\right]$$

Generalized Policy Iteration

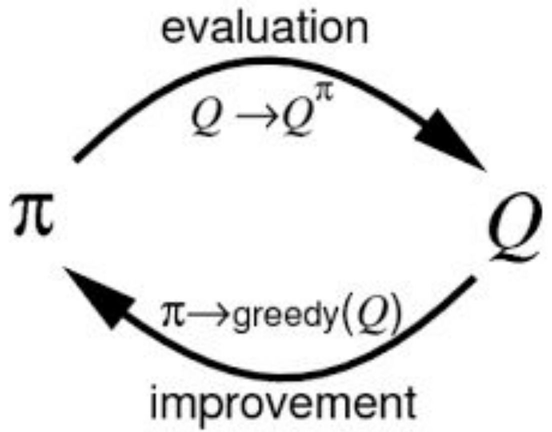
1) Policy evaluation:

$$V_{k+1}^{\pi}(s) = [T^{\pi} V_k](s) = \sum_{a} \pi(a \mid s) \sum_{r,s'} p\big(r, s' \mid s, a\big)\big[r + \gamma V_k^{\pi}\big(s'\big)\big]$$

2) Policy iteration:

$$\pi' = \arg\max_{a} Q^{\pi}(s, a) = \arg\max_{a} \sum_{r,s'} p\big(r, s' \mid s, a\big)\big[r + \gamma V^{\pi}\big(s'\big)\big]$$

Generalized Policy Iteration



$$\pi \geq \pi' \iff V_{\pi}(s) \geq V_{\pi'}(s)\ \forall s \in \mathcal{S}$$

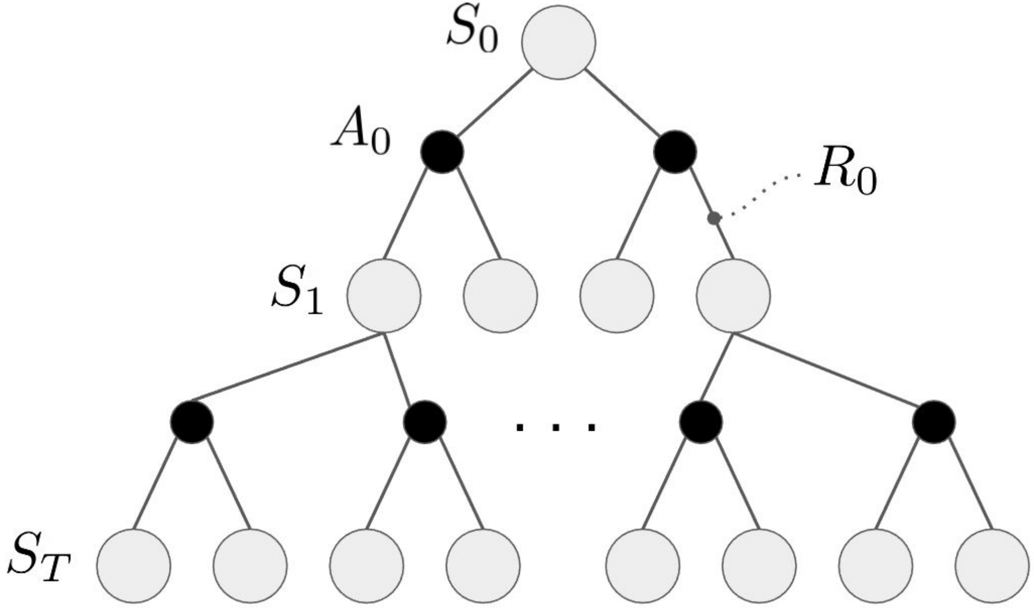
Вопрос

GPI предполагает знание динамики среды **p(r,s’|s,a)**

Что делать, если она неизвестна?

Метод Монте-Карло

Давайте оценивать V(s) как среднее по нескольким траекториям



Метод Монте-Карло

Давайте оценивать V(s) как среднее по нескольким траекториям

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

  $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

  $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

  Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

  $G \leftarrow 0$

  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

    $G \leftarrow \gamma G + R_{t+1}$

    Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

      Append G to $Returns(S_t)$

      $V(S_t) \leftarrow$ average$(Returns(S_t))$

Model-free Control

p(r,s’|s,a) - неизвестен или его трудно получить.

Как в это случае восстановить политику из функций ценности?

1) Допустим Q(s,a) известна. Тогда:

$$\pi(s) = \arg\max_a Q(s, a)$$

2) Допустим V(s) известна. Тогда:

$$\pi(s) = \arg\max_a \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma V(s')\right]$$

Model-free Control

p(r,s'|s,a) - неизвестен или его трудно получить.

Как в это случае восстановить политику из функций ценности?

1) Допустим Q(s,a) известна. Тогда:

$$\pi(s) = \arg\max_a Q(s, a)$$

2) Допустим V(s) известна. Тогда:

$$\pi(s) = \arg\max_a \sum_{r,s'} \cancel{p(r, s' | s, a)} \left[r + \gamma V(s')\right]$$

V(s) бесполезна

Метод Монте-Карло

Тоже самое можно делать и для Q-функции.

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:
- $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
- $Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
- $Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):
- Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
- Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
    - Append G to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Метод Монте-Карло

Тоже самое можно делать и для Q-функции. Однако необходимо, чтобы каждая пара **(s,a)** была посещена бесконечное количество раз в пределе бесконечного количества траекторий.

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:
  $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
  $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
  $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):
  Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
  Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
      $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Exploration vs Exploitation

Для начала рассмотрим **on-policy** методы, которые улучшают или оценивают политику, с помощью которой действия были совершены. В **on-policy** методах политика обычно мягкая:

$$\pi(a \mid s) \geq 0 \; for \, \forall \, s \in S, \, \forall \, a \in A$$

Exploration vs Exploitation

Для начала рассмотрим **on-policy** методы, которые улучшают или оценивают политику, с помощью которой действия были совершены. В **on-policy** методах политика обычно мягкая:

$$\pi(a \mid s) \geq 0 \; for \, \forall \, s \in S, \, \forall \, a \in A$$

Как альтернативу можно рассмотреть ε-жадную политику:

$$\pi(a \mid s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|} & a = a^\star \\ \frac{\epsilon}{|A|} & a \neq a^\star \end{cases}$$

Метод Монте-Карло

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

Метод Монте-Карло

Недостатки:

1) Необходимо дождаться завершения эпизода, чтобы учится на нем
2) Return’ы имеют высокую дисперсию.

TD-learning (Temporal Difference)

TD-learning комбинирует идеи Монте-Карло и динамического программирования:

- Как и Монте-Карло, TD учатся напрямую из собранных траекторий не имея представления о динамике среды
- Как и в динамическом программировании, оценка ценности одной пары **(s,a)** частично строится на оценке следующей пары **(s,a)**.

Общая идея

1) Вспомним уравнение Беллмана для оптимальности V:

$$V(s) = \max_a E_{r,s'}\left[r + \gamma V_\pi\left(s'\right)\right]$$

2) Заменим ожидание **E** на экспоненциальное движущееся среднее:

$$E \leftarrow \alpha \hat{E} + (1-\alpha)E = E + \alpha \boxed{\left(\boxed{\hat{E}} - E\right)},\ \alpha \in (0;1]$$

target TD error

TD prediction

$$V(s_t) \leftarrow \alpha[r_t + \gamma V(s_{t+1})] + (1-\alpha)V(s_t) = V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)]$$

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

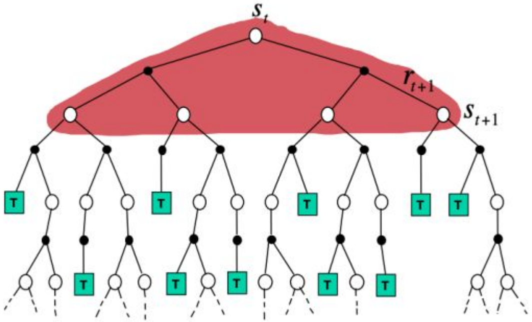
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
    $S \leftarrow S'$
  until S is terminal

TD(0)

Backup diagrams

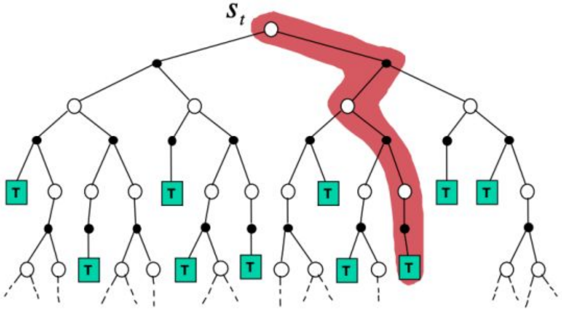
DP

$$v(S_t) \leftarrow \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid A_t \sim \pi(S_t)\right]$$



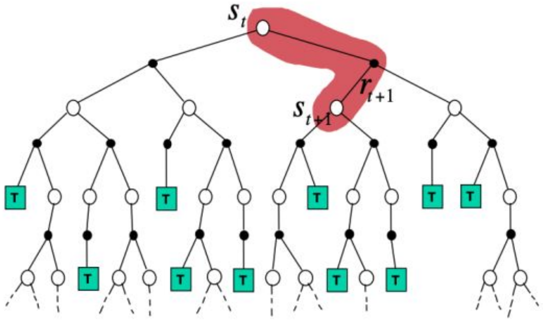
MC

$$v(S_t) \leftarrow v(S_t) + \alpha\left(G_t - v(S_t)\right)$$



TD(0)

$$v(S_t) \leftarrow v(S_t) + \alpha\left(R_{t+1} + \gamma v(S_{t+1}) - v(S_t)\right)$$



Model-free Control

p(r,s’|s,a) - неизвестен или его трудно получить.

Как в это случае восстановить политику из функций ценности?

1) Допустим Q(s,a) известна. Тогда:

$$\pi(s) = \arg\max_a Q(s,a)$$

2) Допустим V(s) известна. Тогда:

$$\pi(s) = \arg\max_a \sum_{r,s'} \cancel{p(r,s'|s,a)} \left[r + \gamma V(s') \right]$$

V(s) бесполезна

SARSA: On-policy TD Control

Применим **TD prediction** к **Q** функции:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Этот алгоритм называется SARSA, потому что для его работы необходимы сэмплы вида (s_t, a_t, r_t, s_{t+1}, a_{t+1}).

SARSA: On-policy TD Control

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Loop for each step of episode:
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

Q-learning: Off-policy TD Control

Теперь вспомним уравнение Беллмана для оптимальности Q функции:

$$Q(s,a) = E_{r,s' \mid s,a}\left[r + \gamma \max_{a'}\left[Q\left(s', a'\right)\right]\right]$$

Применим его к правилу обновления:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

$$\Downarrow$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_t + \gamma \max_{a} Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

Q-learning: Off-policy TD Control

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\textit{terminal}, \cdot) = 0$

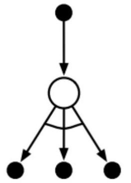
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Expected SARSA

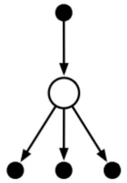
Рассмотрим алгоритм, похожий на Q-learning, но вместо максимума для следующей пары состояние-действие используется ожидание по текущей политике. В остальном метод следует алгоритму Q-learning.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma E_\pi Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] =$$

$$= Q(s_t, a_t) + \left[r_t + \gamma \sum_a \pi(a \mid s_{t+1}) Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right]$$

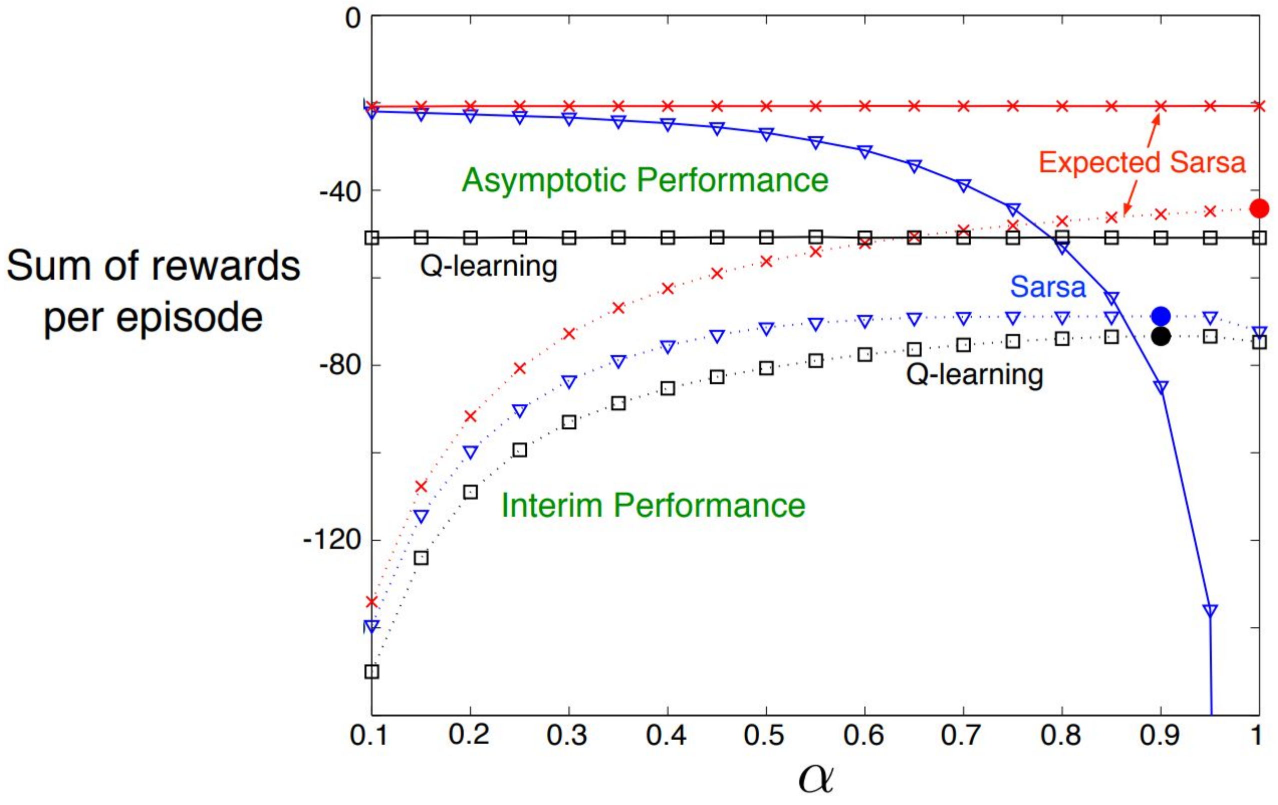


Q-learning

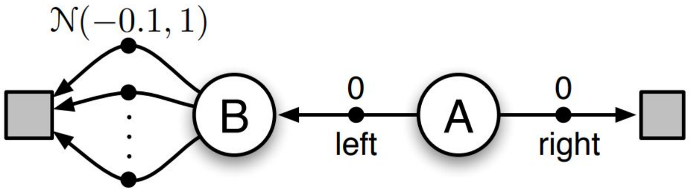


Expected Sarsa

Expected SARSA



Maximization Bias

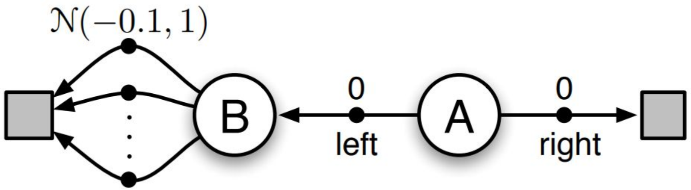


$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big]$$

Maximization Bias

Завышение Q-функции алгоритмами может быть вызвано несколькими причинами:

1) Максимум по оценкам ценностей неявно используется в качестве оценки максимальной ценности, что может привести к существенному положительному смещению
2) Одна и та же оценка используется для выбора максимального действия и его оценки его ценности



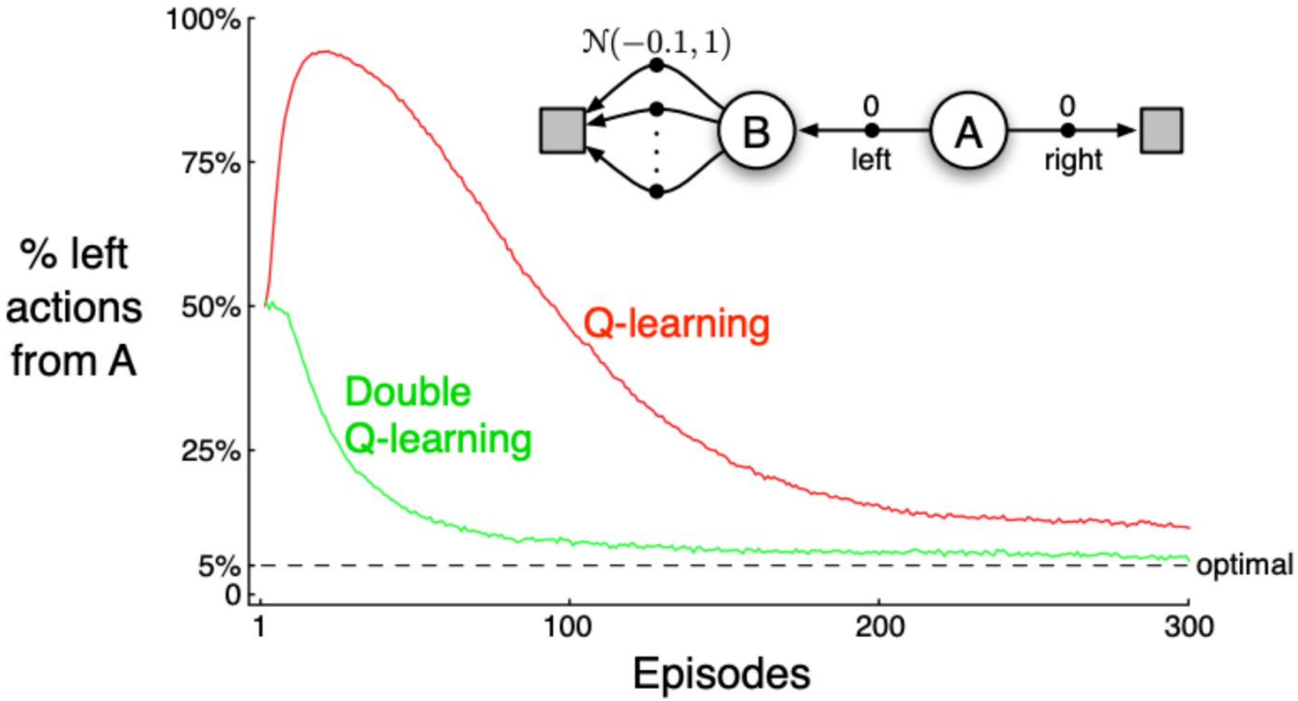
Maximization Bias and Double Q-learning

Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(\textit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:
      $Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$
    else:
      $Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S', a)\big) - Q_2(S, A)\Big)$
    $S \leftarrow S'$
  until S is terminal

Q-Learning vs Double Q-learning



$\varepsilon = 0.1$

$\alpha = 0.1$

$\gamma = 1$

Off-policy vs On-policy

Отличительной чертой on-policy методов является то, что они оценивают ценность действия на основе текущей политики, которая одновременно с этим используется для сбора траекторий.

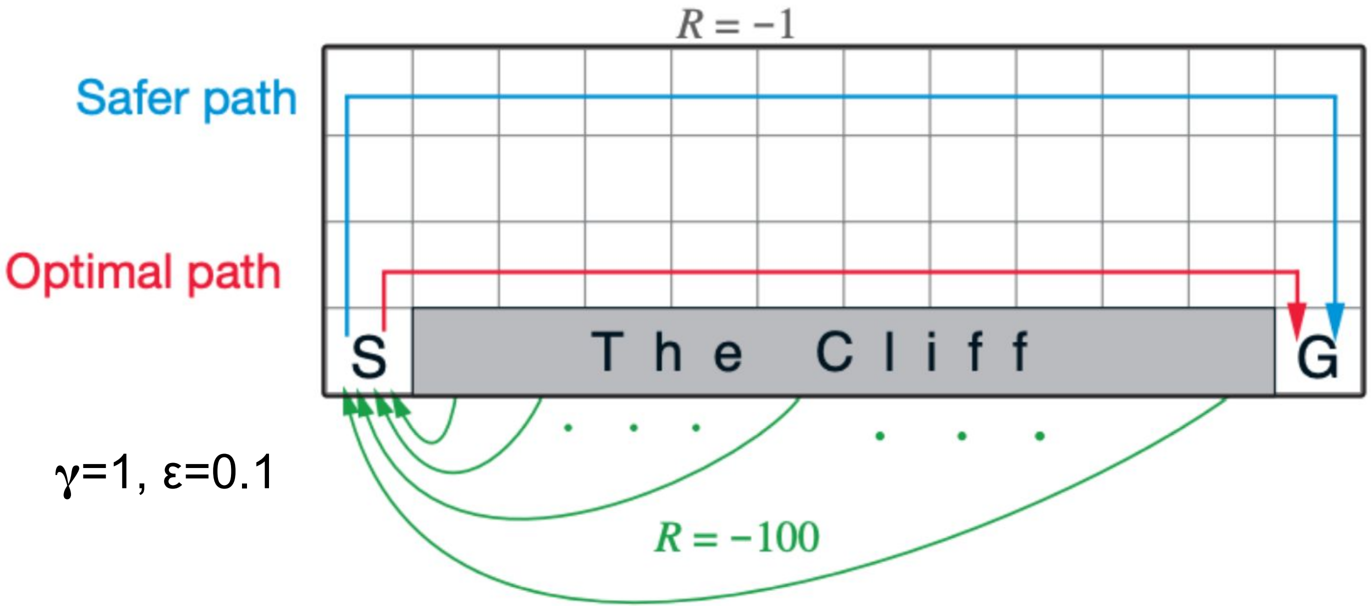
В off-policy методах эти функции разделены:

- Behavior policy собирает траектории
- Target policy оценивает ценность действий

Преимущество такого подхода в том, что target policy может быть детерминированной, а behavior policy продолжать исследовать среду

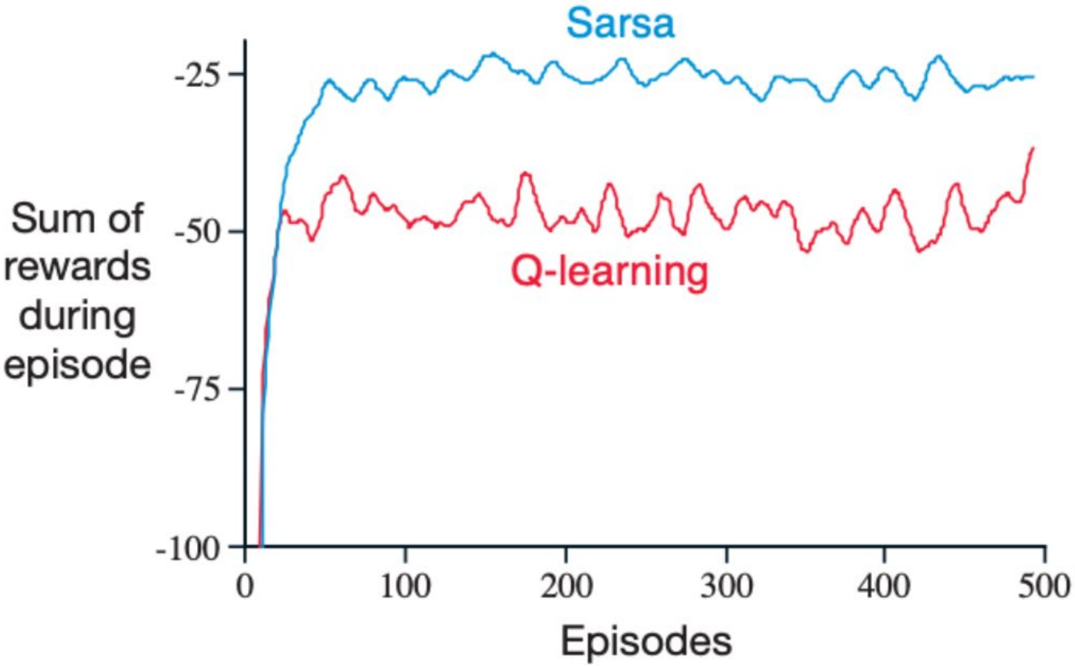
Пример. Cliff World

Какая траектория принадлежит Q-learning, а какая SARSA?



$\gamma=1$, $\varepsilon=0.1$

Пример. Cliff World



Как улучшить?

On-policy vs Off-Policy

On-policy learning:

- Интуитивно понятен
- Агент всегда следует своей же политике
- Не может учится off-policy

Off-policy learning:

- Можно настроить исследование среды
- Можно учиться на экспертных данных, собранном датасете
- Может учиться on-policy

Experience Replay Buffer

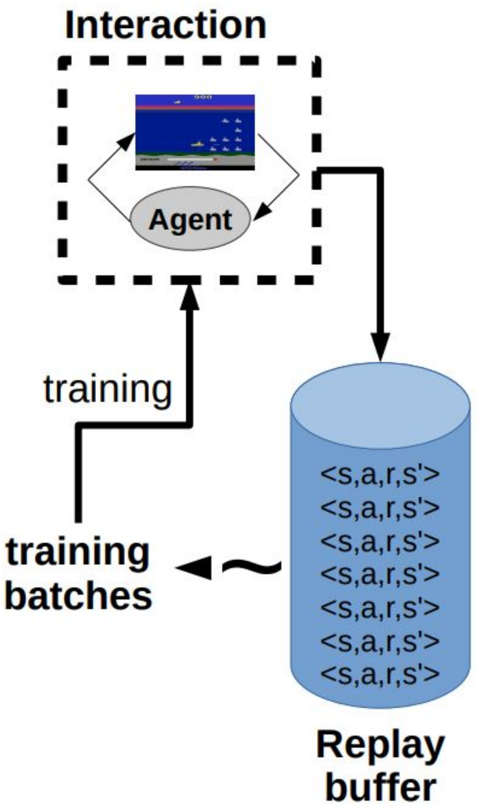
- На каждом шаге сохраняем <s,a,r,s’> в буфер
- Сэмплим n рандомных переходов в батч
- Учимся на батче

Преимущества:

- Не нужно посещать одну и ту же пару (s,a) много раз
- Декоррелирование обучающих сэмплов (пригодится для сеточек)

Недостатки:

- Недоступно для on-policy методов



N-step Bootstrapping

- TD использует оценки, которые могут быть некорректны
- Информация о наградах может распространятся по MDP-дереву медленно
- В MC информация распространяется быстрее, но при этом имеет большую дисперсию
- Нужно что-то среднее

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \gamma^k r_{t+k} = \sum_{k=0}^{T-t} \gamma^k r_{t+k}$$

N-step Control

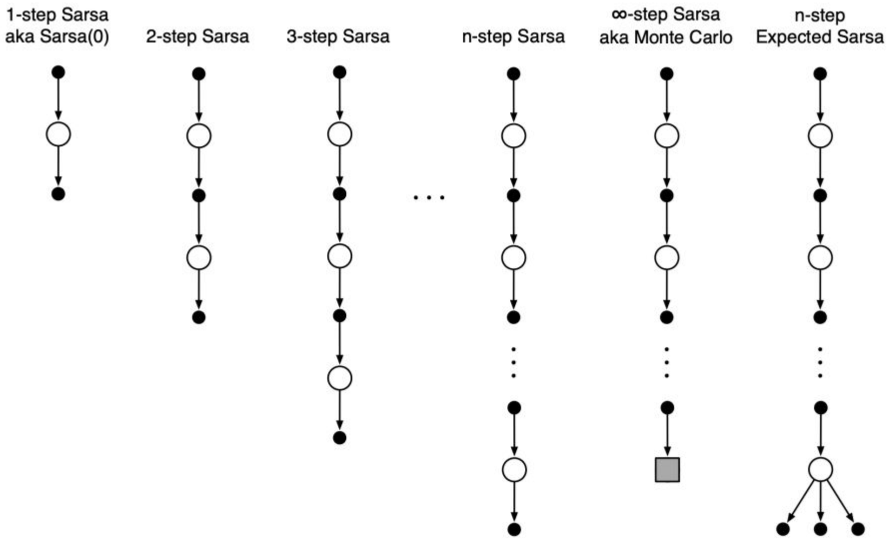
Общая формула:

$$Q(s_t, a_t) \leftarrow \alpha \hat{Q}(s_t, a_t) + (1-\alpha) Q(s_t, a_t) = Q(s_t, a_t) + \alpha \Big(\hat{Q}(s_t, a_t) - Q(s_t, a_t) \Big)$$

1-step SARSA:

$$\hat{Q}(s_t, a_t) = r_t + \gamma Q(s_{t+1}, a_{t+1})$$

N-step SARSA:



N-step Control

Общая формула:

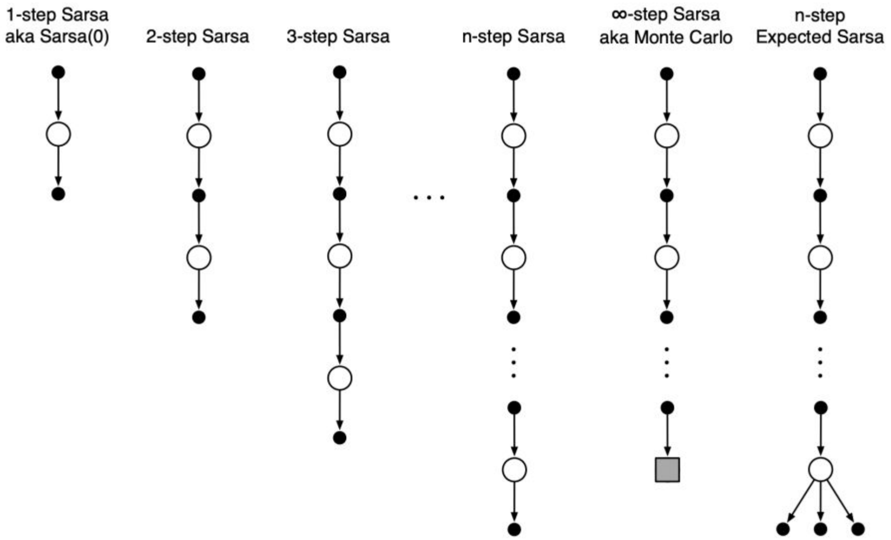
$$Q(s_t, a_t) \leftarrow \alpha \hat{Q}(s_t, a_t) + (1-\alpha) Q(s_t, a_t) = Q(s_t, a_t) + \alpha \Big(\hat{Q}(s_t, a_t) - Q(s_t, a_t) \Big)$$

1-step SARSA:

$$\hat{Q}(s_t, a_t) = r_t + \gamma Q(s_{t+1}, a_{t+1})$$

N-step SARSA:

$$\hat{Q}(s_t, a_t) = \sum_{\tau=t}^{t+n-1} \gamma^{\tau-t} r_t + \gamma^n Q(s_{t+n}, a_{t+n})$$



N-step Control

Общая формула:

$$Q(s_t, a_t) \leftarrow \alpha\hat{Q}(s_t, a_t) + (1-\alpha)Q(s_t, a_t) = Q(s_t, a_t) + \alpha\Big(\hat{Q}(s_t, a_t) - Q(s_t, a_t)\Big)$$

1-step SARSA:

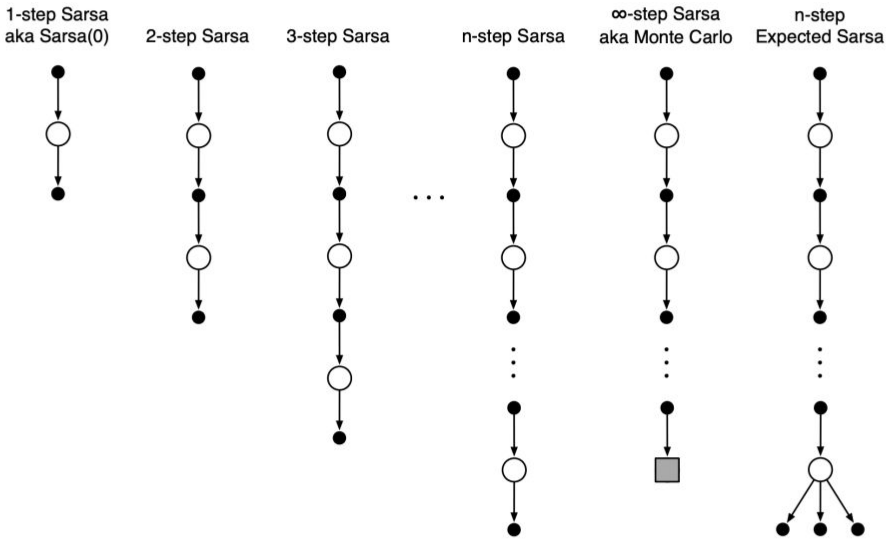
$$\hat{Q}(s_t, a_t) = r_t + \gamma Q(s_{t+1}, a_{t+1})$$

N-step SARSA:

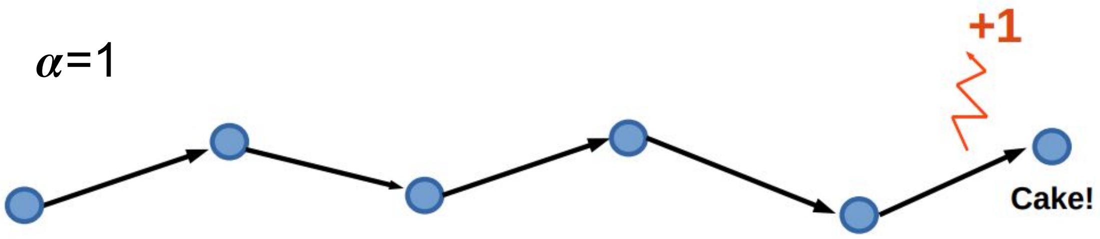
$$\hat{Q}(s_t, a_t) = \sum_{\tau=t}^{t+n-1} \gamma^{\tau-t} r_t + \gamma^n Q(s_{t+n}, a_{t+n})$$

N-step Q-learning:

$$\hat{Q}(s_t, a_t) = \sum_{\tau=t}^{t+n-1} \gamma^{\tau-t} r_t + \gamma^n \max_a Q(s_{t+n}, a)$$



Пример



Сколько итераций потребуется 1-step SARSA для сходимости, с сколько потребуется 5-step SARSA?