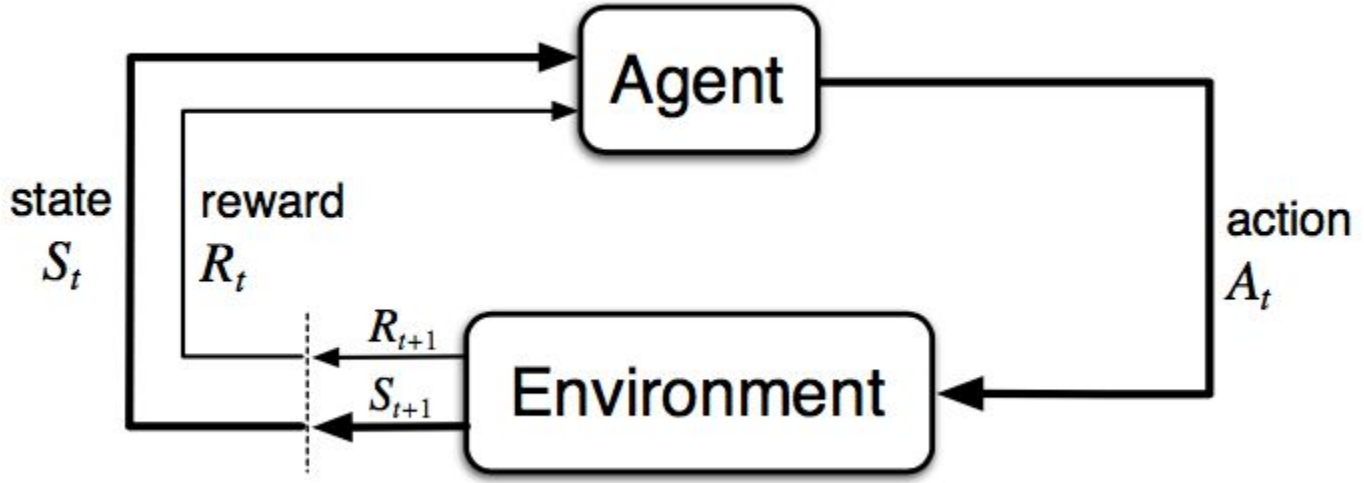
Q-learning

Гришанин Виктор

Марковский процесс принятия решений



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

Доход агента

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

$$0 \leq \gamma \leq 1$$

γ показывает важность последующих наград в данный момент

Policy функция

$$\pi(a|s) = P(a_t = a|s_t = s)$$

$$s \in S, a \in A(s)$$

Задача: найти такую policy, чтобы максимизировать награду

State-value и action-value функции

- State-value функция

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

- Action-value функция

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

Bellman expectation equation (для state-value)

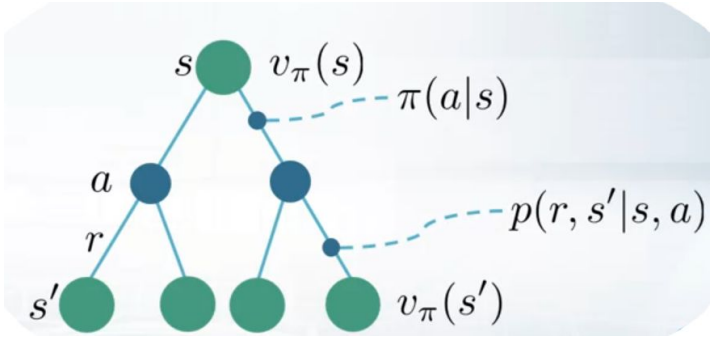
$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

$$= \mathbb{E}_\pi[R_t + \gamma G_{t+1} | S_t = s]$$

$$= \sum_a \pi(a|s) \sum_{r,s'} p(r, s'|s, a)[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']]$$

$$= \sum_a \pi(a|s) \sum_{r,s'} p(r, s'|s, a)(r + \gamma v_\pi(s')]$$

$$= \mathbb{E}_\pi[R_t + \gamma v_\pi(S_{t+1}) | S_t = s]$$



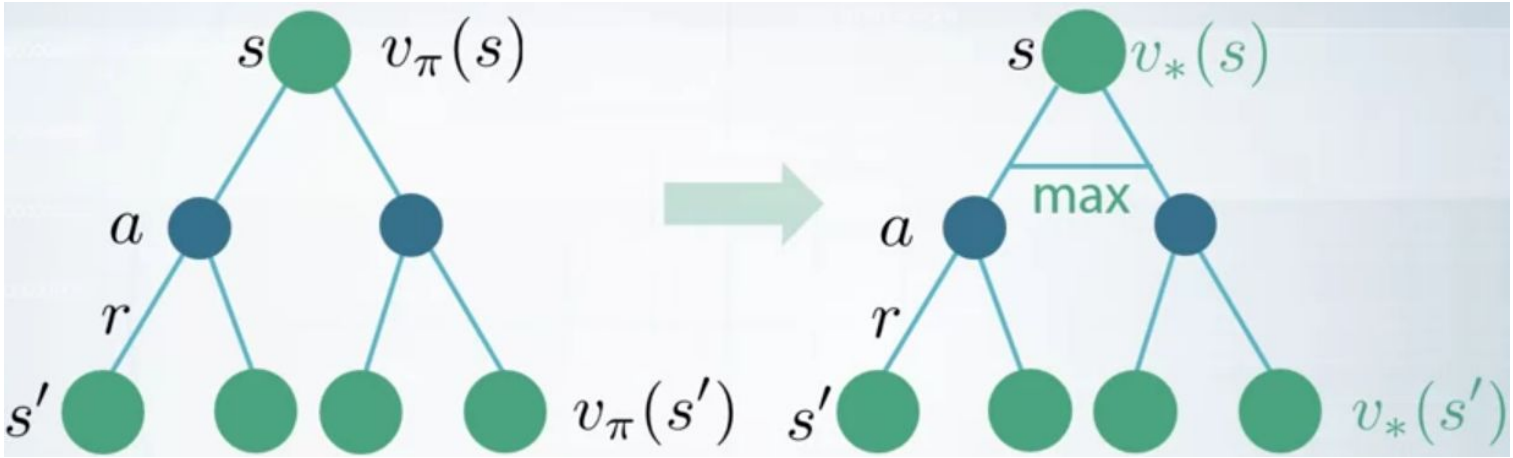
Оптимальность (policy-функции)

$$\pi \geq \pi' \Leftrightarrow v_\pi(s) \geq v_{\pi'}(s), \forall s$$

π^* - оптимальная политика (лучше или равна любой другой)

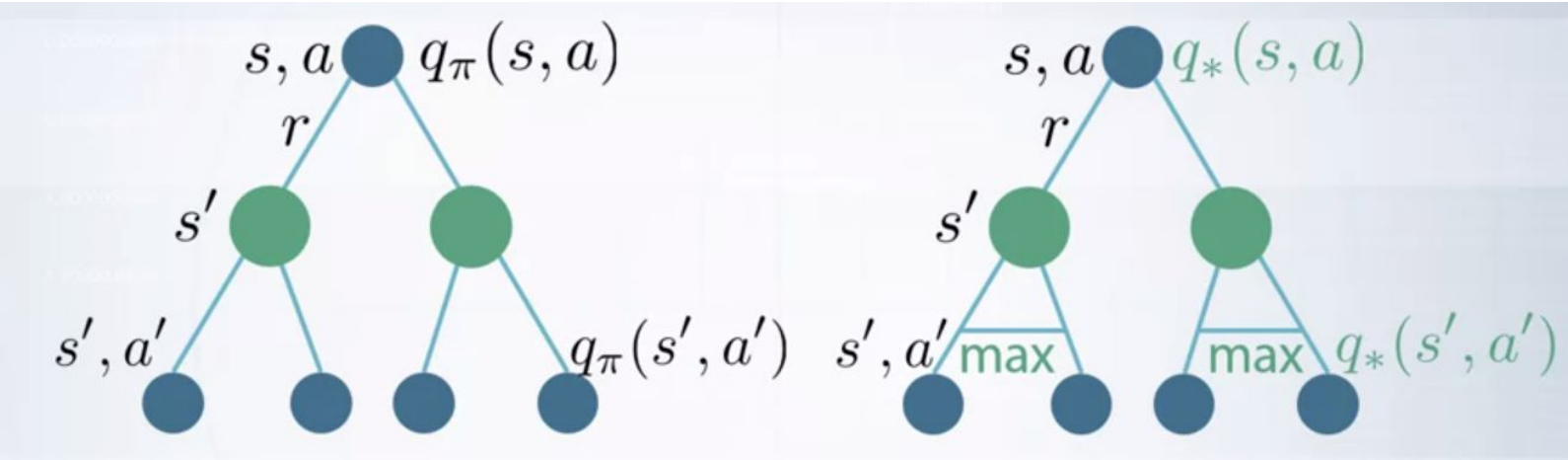
$$v_*(s) = \max_\pi v_\pi(s) \quad q_*(s,a) = \max_\pi q_\pi(s,a)$$

Bellman optimality equation (для state-value)



$$v_*(s) = \max_a \sum_{r,s'} p(r, s'|s, a)[r + \gamma v_*(s')] = \max_a \mathbb{E}_\pi[R_t + \gamma v_*(S_{t+1})|S_t = s, A_t = a]$$

Bellman optimality equation (для action-value)



$$q_*(s,a) = \mathbb{E}_\pi[R_t + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a] = \sum_{r,s'} p(r, s' | s, a)[r + \gamma \max_{a'} q_*(s', a')]$$

On-policy vs off-policy

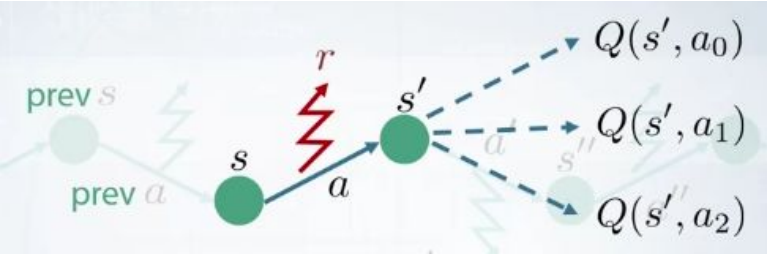
- On-policy метод: пытается улучшить policy, которой придерживаются (behavior policy = target policy)

  Примеры: Monte-Carlo, SARSA

- Off-policy метод: с помощью behavior policy оптимизирует target policy (при этом target policy != behavior policy)

  Примеры: Q-learning, Expected SARSA

Q-learning



- Инициализация Q(s,a) нулями
- Выбор s
- Цикл:
  - Выбрать действие a из s с помощью политики разведки
  - После совершения действия a получить r и s’
  - Посчитать Q для следующего состояния:
    - $\hat{Q}(s,a) = r(s,a) + \gamma \max_{a_i} Q(s', a_i)$
    - Обновить:

$$Q(s,a) \leftarrow \alpha \hat{Q}(s,a) + (1-\alpha) Q(s,a)$$

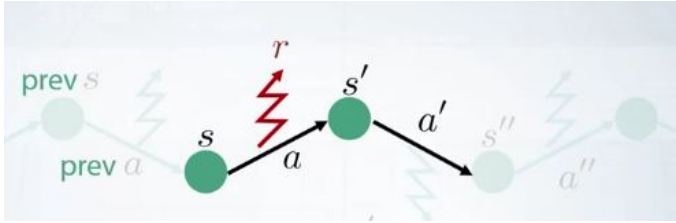
$$s \leftarrow s'$$

Q-learning vs SARSA

SARSA и Expected value SARSA

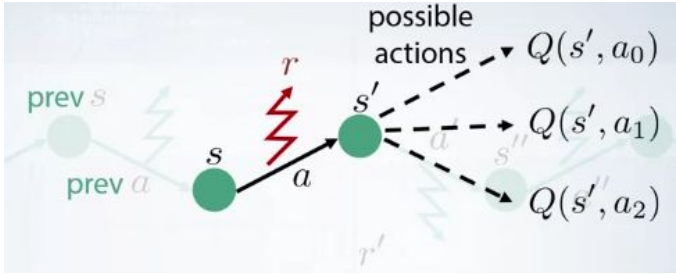
SARSA:

$$\hat{Q}(s,a) = r(s,a) + \gamma Q(s',a')$$



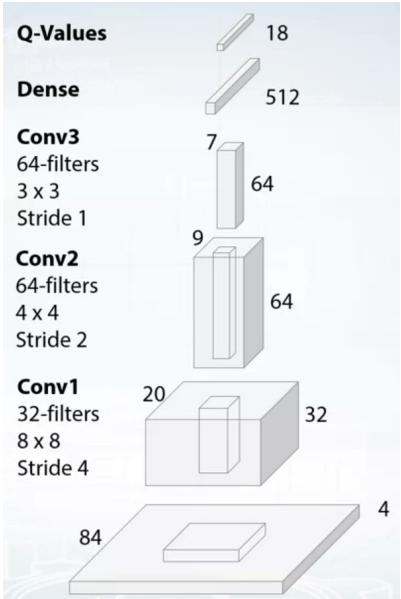
Expected value SARSA:

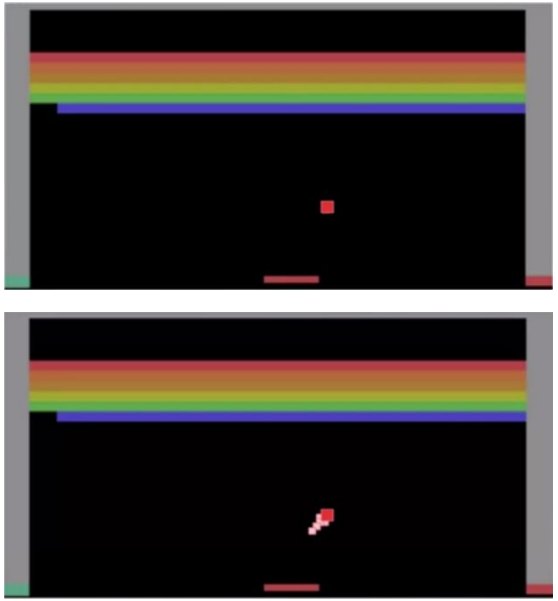
$$\hat{Q}(s,a) = r(s,a) + \gamma \mathbb{E}_{a_i} Q(s',a_i)$$



Q-learning vs SARSA

Deep Q-learning





$$w \leftarrow w + \alpha[R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, w) - \hat{q}(S_t, A_t, w)]\nabla \hat{q}(S_t, A_t, w)$$

Deep Q-learning: решение проблем

- Корреляция в последовательности данных
  - Experience replay
- Нестабильность распределения данных из-за политики
  - Target networks
- Нестабильные градиенты
  - Reward clipping

Experience replay

Идея: использование буфера с хранящимися в нём последними наблюдениями (кортежами (s, a, r, s')).

В цикле:

- Сэмплируем несколько кортежей из буфера
- Обучаем Q-function с помощью этого мини-батча
- Совершить действия с помощью эпсилон-жадной политики
- Добавить новые наблюдения в буфер

Experience replay

Преимущества:

- Помогает с коррелированными данными
- Повышает скорость обучения
- Можно обучать параллельно

Недостатки:

- Использование большого количества памяти
- Можно улучшить сэмплирование из пула

Источники

https://coursera.org/share/570bb19a8524e932915958726d2a3615 - курс Practical reinforcement learning

https://www.researchgate.net/publication/50247491_An_Analysis_of_Q-Learning_Algorithms_with_Strategies_of_Reward_Function - Manju, Punithavalli. An Analysis of Q-Learning Algorithms with Strategies of Reward Function, 2011

http://incompleteideas.net/book/RLbook2020.pdf - Sutton, Barto. Reinforcement Learning: An Introduction, 2018