TD-ОБУЧЕНИЕ

Сергей Николенко

СПбГУ — Санкт-Петербург

29 апреля 2022 г.

Random facts:

- В Японии 29 апреля — День Сёва (день рождения императора Хирохито); он открывает «Золотую неделю» праздников: День Конституции (3 мая), День зелени (День основния государства, 4 мая), Праздник детей (5 мая); жизнь в Японии в эту неделю замирает
- 29 апреля 1627 г. кардинал Ришельё создал Компанию Ста акционеров для колонизации Канады, а 29 апреля 1770 г. Джеймс Кук впервые высадился в Австралии
- 29 апреля 1925 г. Николай Бухарин впервые употребил в докладе на XIV партконференции фразу «генеральная линия партии», а 29 апреля 1933 г. при ЦК ВКП(б) была создана Центральная комиссия по чистке партии
- 29 апреля 1945 г. Адольф Гитлер женился на Еве Браун, а уже на следующий день покончил жизнь самоубийством
- 29 апреля 1961 г. Леонид Рогозов, хирург 6-й Советской антарктической экспедиции, успешно вырезал сам себе аппендикс
- 29 апреля 1967 г. Мухаммед Али был лишён боксёрского титула, потому что накануне отказался от призыва в армию

TD-обучение

TD-обучение

- Общий принцип TD-обучения: давайте обучать оценки состояний на основе обученных нами ранее оценок для последующих состояний.
- $TD(0)$-обучение: инициализировать $V(s)$ и π произвольно, затем на каждом эпизоде обучения:
  - инициализировать s;
  - для каждого шага t в эпизоде:
    - выбрать A_t в состоянии S_t по стратегии π;
    - сделать A_t, пронаблюдать результат R_{t+1} и следующее состояние S_{t+1};
    - $V(S_t) := V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$.

TD-обучение

- Здесь по сути методы Монте-Карло и TD-обучение расходятся в том, как строить оценку для $V(s)$:
  - MC-методы оценивают $V(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$, собирая статистику из G_t;
  - а TD-методы оценивают на один шаг вперёд как $V(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s \right]$.
- Смысл TD-обучения в том, чтобы использовать уже обученные закономерности для поиска более глубоких закономерностей.
- В результате обучение получится целенаправленным, обучается гораздо быстрее, чем другие стратегии.

TD-обучение

- Здесь тоже есть on-policy и off-policy варианты

Алгоритм Sarsa (on-policy TD control):

- инициализировать случайно $Q(s, a)$;
- повторять до сходимости:
  - инициализировать S_0, выбрать A_0 по стратегии, полученной из Q (например, по ϵ-жадной стратегии);
  - для каждого шага в эпизоде $t = 0, \dots, T$:
    - сделать действие A_t, получить награду R_{t+1}, перейти в состояние S_{t+1};
    - выбрать A_{t+1} по стратегии, полученной из Q (например, по ϵ-жадной стратегии);
    - обновить Q:

$$Q(S_t, A_t) := Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right)$$

- Этот алгоритм умеет искать оптимальную *мягкую* π (т.е. опять нужно самому исследовать)

TD-обучение

- A off-policy вариант называется Q-обучение; он ещё проще, и это была очень мощная идея, которая до сих пор определяет многое в RL (Watkins, 1989)

Алгоритм Q-learning (off-policy TD control):

- инициализировать случайно $Q(s, a)$;
- повторять до сходимости:
  - инициализировать S_0
  - для каждого шага в эпизоде $t = 0, \dots, T$:
    - выбрать A_t по стратегии, полученной из Q (например, по ϵ-жадной стратегии);
    - сделать действие A_t, получить награду R_{t+1}, перейти в состояние S_{t+1};
    - обновить Q:

$$Q(S_t, A_t) := Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right)$$

- Этот алгоритм умеет искать оптимальную *жёсткую* π_*, делая ходы по мягкой стратегии

Обобщения и дополнительные замечания

n-шаговые алгоритмы

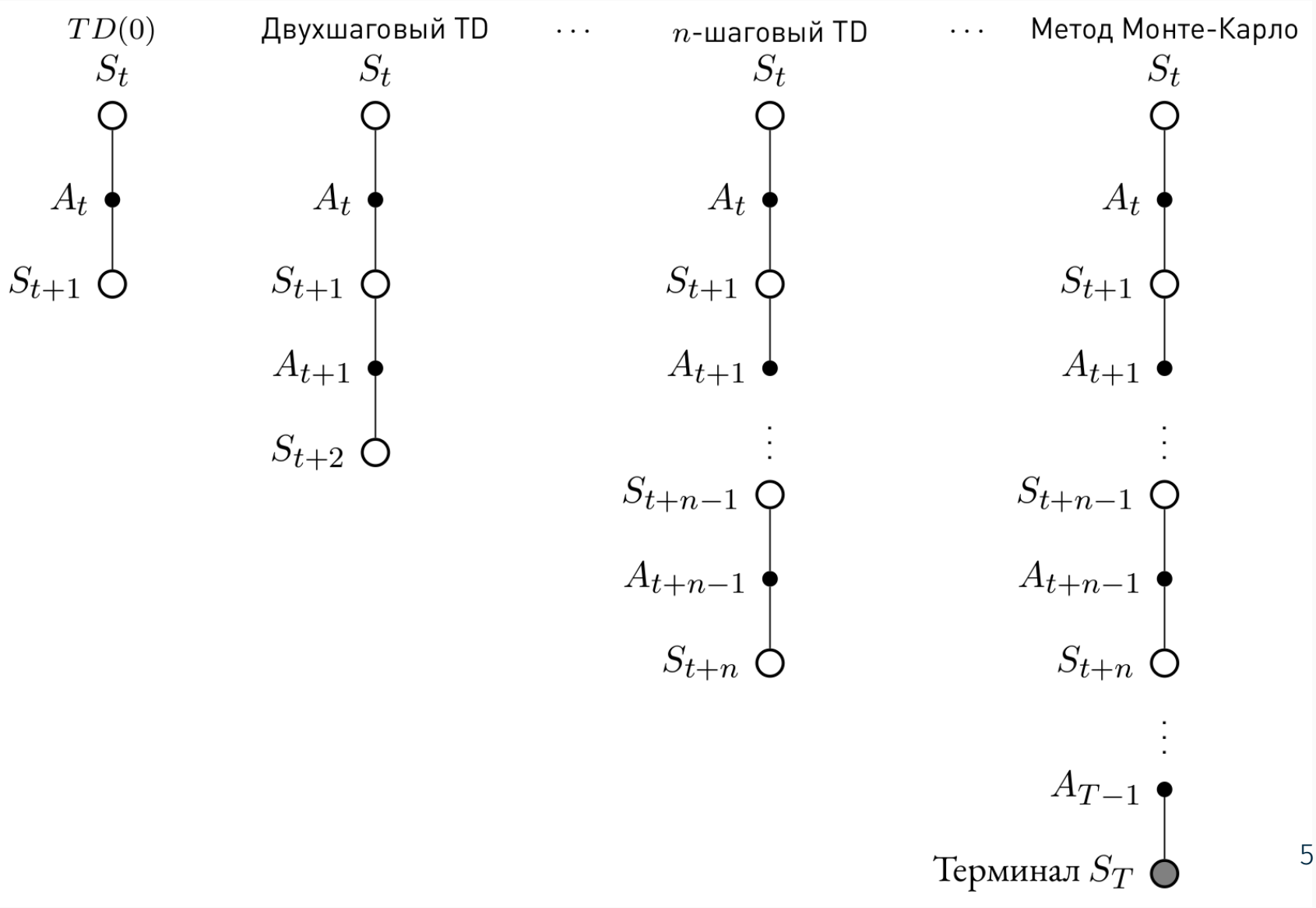
- Мы говорили о TD-алгоритмах, которые оценивают $G_t = R_{t+1} + \gamma V_t(S_{t+1})$.
- Но можно же и дальше развернуть:

$$
\begin{aligned}
G_{t:t+2} &= R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+1}) \\
G_{t:t+3} &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V_{t+2}(S_{t+3}) \\
&\dots = \dots \\
G_{t:t+n} &= R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})
\end{aligned}
$$

- На бесконечности это сливается с методами Монте-Карло, потому что там мы обновляем на основе награды за весь эпизод, от конца к началу

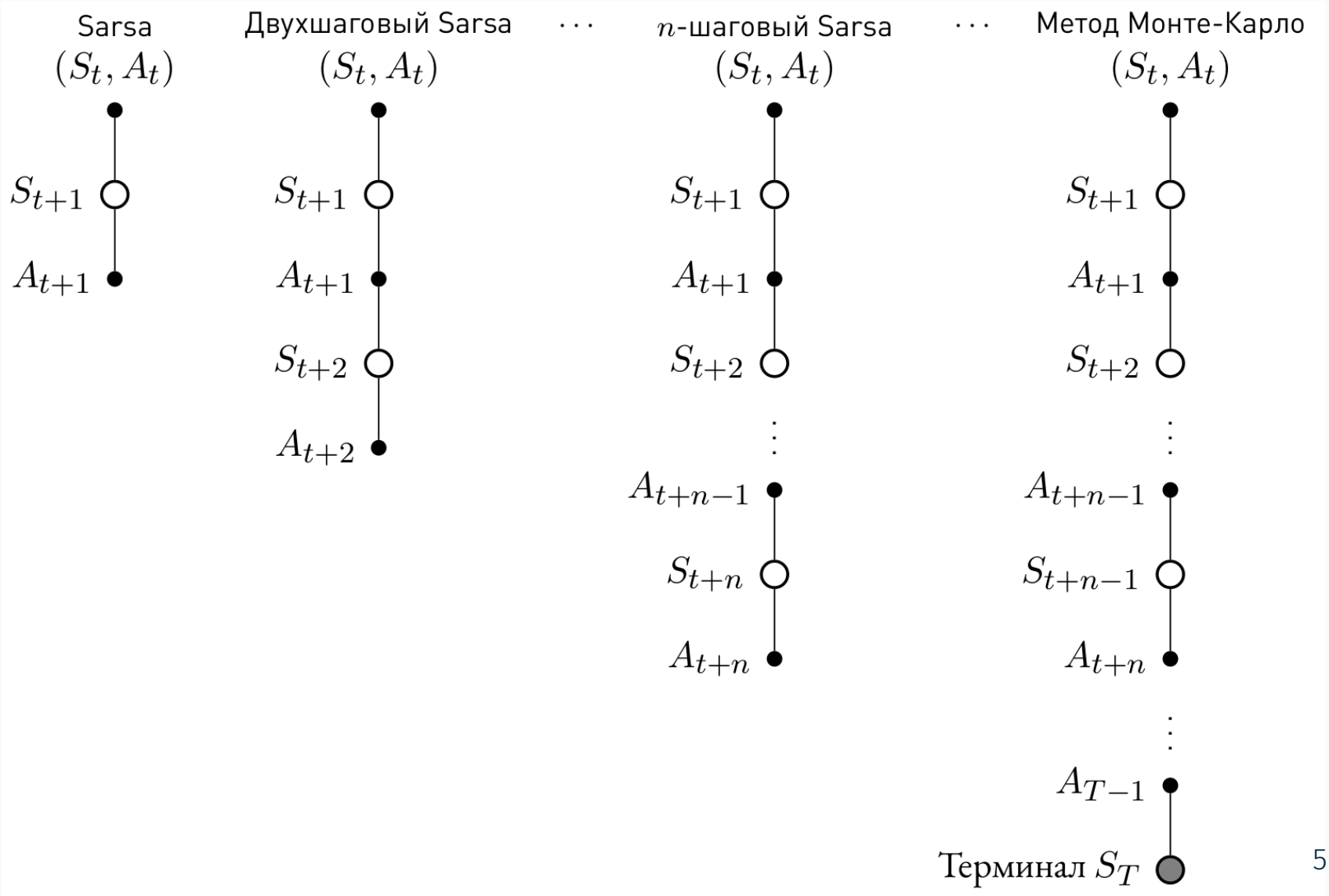
n-шаговые алгоритмы

- Обобщение TD-обучения: n-шаговые методы



n-шаговые алгоритмы

- Обобщение TD-управления: n-шаговые методы



n-шаговые алгоритмы

- И теперь появляются n-шаговые варианты всех алгоритмов.

Алгоритм n-step Sarsa (on-policy TD control):

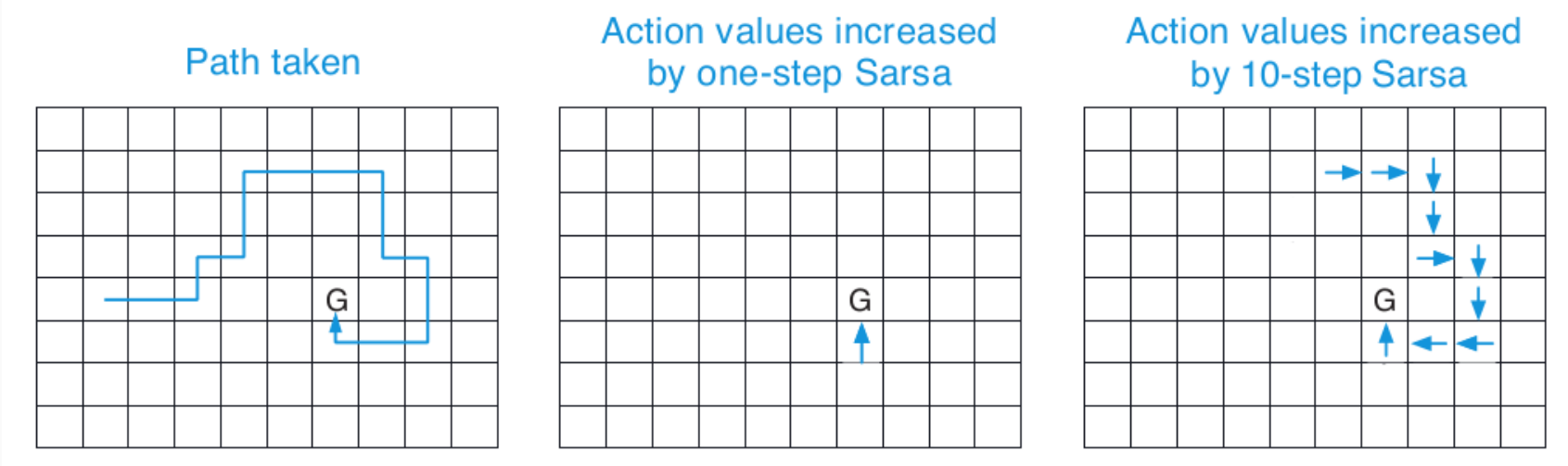
инициализировать случайно $Q(s, a)$; повторять до сходимости:

- инициализировать S_0, выбрать A_0 по стратегии, полученной из Q (например, по ϵ-жадной стратегии); $T := \infty$;
- для каждого шага в эпизоде $t = 0, \dots, T$:
  - если $t < T$, то:
    - сделать действие A_t, получить R_{t+1}, S_{t+1};
    - если это терминальное состояние, то $T := t + 1$, а если нет, выбрать A_{t+1} в состоянии S_{t+1} по стратегии π;
  - $\tau := t - n + 1$ (мы будем обновлять оценку на шаге τ)
  - если $\tau \geq 0$, то обновляем:
    - $G := \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
    - если $\tau + n < T$, то $G := G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$;
    - $Q(S_\tau, A_\tau) := Q(S_\tau, A_\tau) + \alpha \left(G - Q(S_\tau, A_\tau)\right)$
    - если обучаем π, то поменять π на ϵ-жадную по Q

- Та же Sarsa, но дальше заглядывает.

n-шаговые алгоритмы

- Смысл здесь в том, что n-step алгоритмы обновляют сразу много значений по одной и той же награде, даже если всё остальное пока что нулевое:



- Всё то же самое можно переписать в терминах просто изменения TD-ошибки:

$$G_{t:t+n} = Q_{t-1}(S_t, A_t) + \sum_{k=t}^{\min(t+n,T)-1} \gamma^{k-t} \left(R_{k+1} + \gamma Q_k(S_{k+1}, A_{k+1}) - Q_{k-1}(S_k, A_k) \right)$$

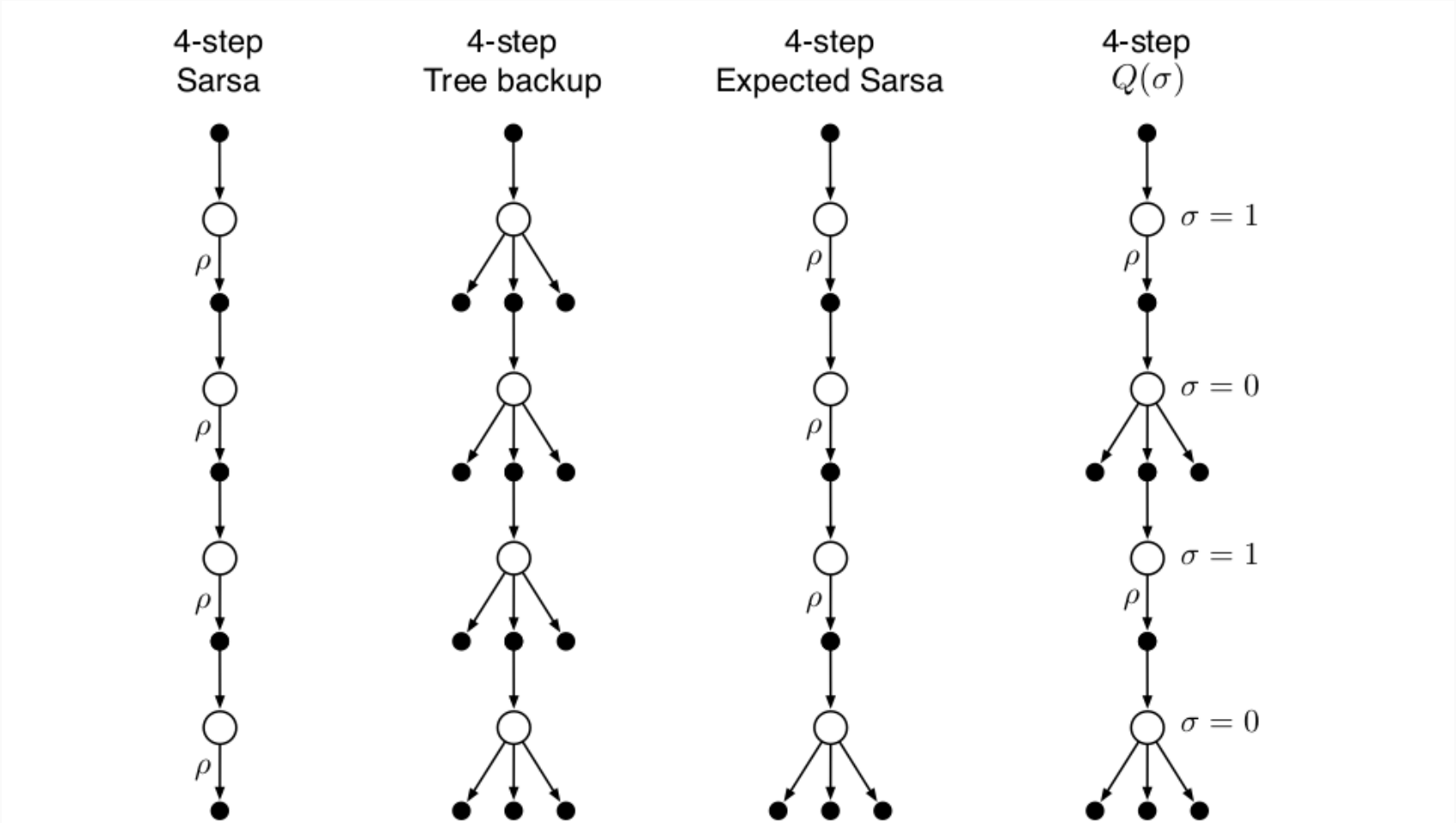
n-шаговые алгоритмы

- Если хочется сделать off-policy, то можно сделать, конечно, с importance sampling, как мы обсуждали.
- А можно сделать поиск по дереву: идём обратно по дереву от (S_t, A_t) к концу эпизода; в каждый момент у нас есть одно действие, которое реально произошло, а для других берём bootstrap-оценки:

$$
\begin{aligned}
G_{t:t+1} &= R_{t+1} &&+\gamma \textstyle\sum_a \pi\left(a|S_{t+1}\right) Q_t(S_{t+1}, a), \\
G_{t:t+2} &= R_{t+1} &&+\gamma \textstyle\sum_{a \neq A_{t+1}} \pi\left(a|S_{t+1}\right) Q_{t+1}(S_{t+1}, a) \\
&&&+\gamma\pi\left(A_{t+1}|S_{t+1}\right)\left(R_{t+2} + \gamma \textstyle\sum_a \pi\left(a|S_{t+2}\right) Q_{t+1}(S_{t+2}, a)\right), \\
\ldots &\quad \ldots \\
G_{t:t+n} &= R_{t+1} &&+\gamma \textstyle\sum_{a \neq A_{t+1}} \pi\left(a|S_{t+1}\right) Q_{t+1}(S_{t+1}, a) \\
&&&+\gamma\pi\left(A_{t+1}|S_{t+1}\right) G_{t+1:t+n}.
\end{aligned}
$$

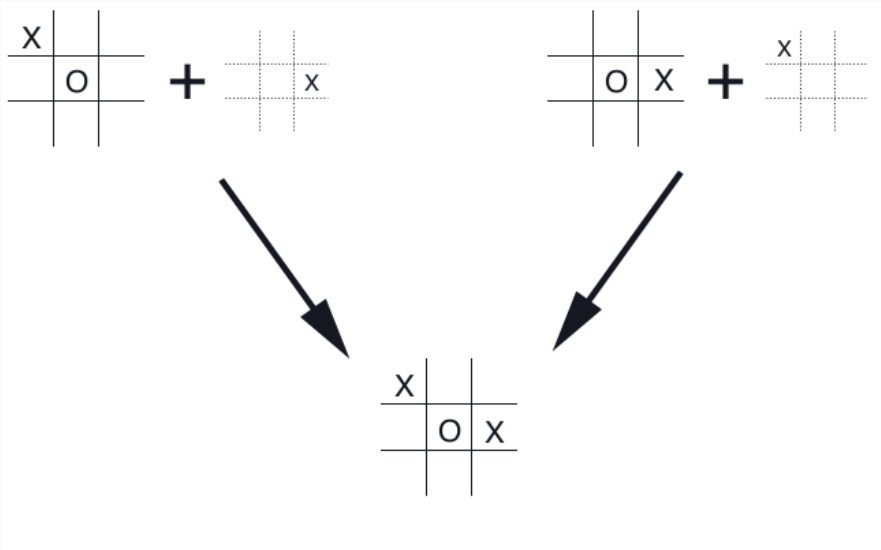
n-шаговые алгоритмы

- А можно пытаться нечто среднее сделать, выбирая то сэмплы, как в Sarsa, то апдейт по дереву, как выше:



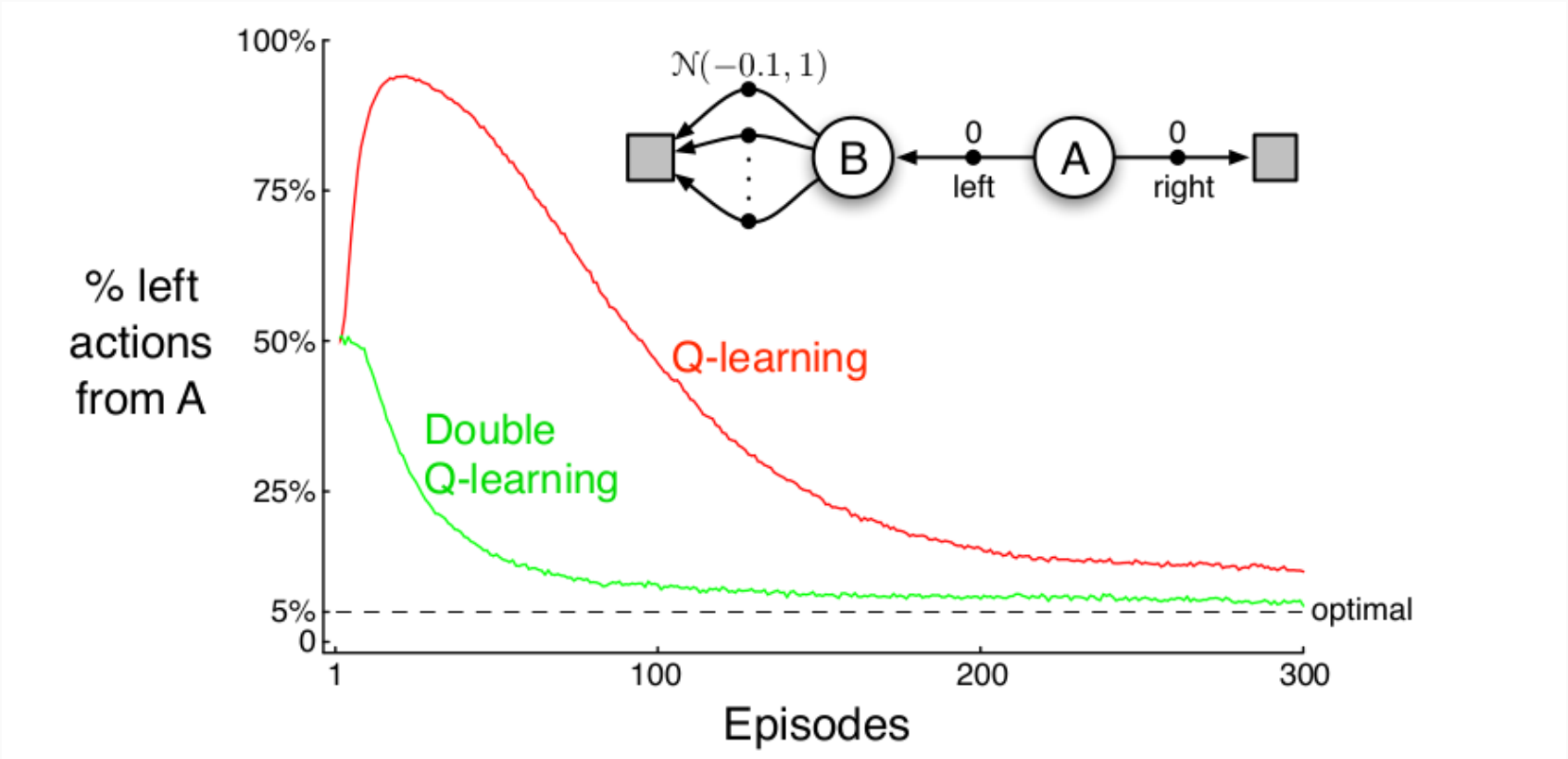
Afterstates

- Часто обучают не по состояниям $V(s)$ и не по парам $Q(s, a)$, а по т.н. *afterstates* – состояниям после действия.
- Это хорошо, когда действия имеют немедленный эффект, а случайный процесс происходит уже потом.
- Крестики-нолики – многие пары приводят к одной и той же позиции.



Двойное обучение

- В TD-обучении есть проблема: наша целевая стратегия – это жадная стратегия по текущей Q, т.е. мы используем максимум по оценкам, чтобы оценить максимальное значение, а это вносит смещение в оценки.



Двойное обучение

- Чтобы это поправить, можно использовать двойное обучение:
  - поддерживаем две стратегии Q_1 и Q_2;
  - каждый раз бросаем монетку, выбираем, какую брать стратегию для обновления, а какую для цели:

$$Q_1(S, A) := Q_1(S, A) + \alpha \left(R + \gamma \max_a Q_2(S', a) - Q_1(S, A) \right)$$

или

$$Q_2(S, A) := Q_2(S, A) + \alpha \left(R + \gamma \max_a Q_1(S', a) - Q_2(S, A) \right).$$

Double Q-learning

- (van Hasselt, 2010): у Q-обучения основное уравнение апдейта

$$Q(S_t, A_t) := Q(S_t, A_t) + \alpha_t \left(R_{t+1} + \gamma \max_a Q_t(S_{t+1}, a) - Q(S_t, A_t) \right)$$

- И по умолчанию Q-обучение использует максимум из имеющихся оценок $Q_t(S_{t+1}, a)$.
- Т.е. получается, что мы оцениваем максимум из нескольких ожиданий, $\max_i \mathbb{E}[X_i]$, через максимум из оценок каждого их них.
- А на самом деле это будет несмещённая оценка для ожидания максимума $\mathbb{E}[\max_i X_i]$!

Double Q-learning

- Поэтому ван Хасселт предложил использовать Double Q-learning: давайте возьмём два разных независимых набора сэмплов для X_i, и будем использовать один из них в тот момент, когда делаем апдейт Q-обучения для другого.
- Тогда (при независимых оценках) всё будет сходиться.

Algorithm 1 Double Q-learning

Initialize Q^A,Q^B,s
repeat
  Choose a, based on $Q^A(s,\cdot)$ and $Q^B(s,\cdot)$, observe r, s'
  Choose (e.g. random) either UPDATE(A) or UPDATE(B)
  if UPDATE(A) **then**
    Define $a^* = \arg\max_a Q^A(s',a)$
    $Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)\left(r + \gamma Q^B(s',a^*) - Q^A(s,a)\right)$
  else if UPDATE(B) **then**
    Define $b^* = \arg\max_a Q^B(s',a)$
    $Q^B(s,a) \leftarrow Q^B(s,a) + \alpha(s,a)(r + \gamma Q^A(s',b^*) - Q^B(s,a))$
  end if
  $s \leftarrow s'$
until end

Спасибо!

Спасибо за внимание!