Оптимизации градиента стратегии

Алексей Скрынник

ПЛАН

01 Прямой поиск стратегии

02 Теорема о градиенте стратегии

03 Монте-Карло градиент стратегии

04 Базовый уровень полезности

05 Оценка стратегии с помощью критика

06 Алгоритм A3C

01

Прямой поиск стратегии

Прямой поиск стратегии в RL

- Ранее мы использовали аппроксимацию функции полезности состояния или действия, параметризованную с помощью $\mathbf{w}$:

$$\hat{V}(s, \mathbf{w}) \approx V^{\pi}(s),$$

$$\hat{Q}(s, a, \mathbf{w}) \approx Q^{\pi}(s, a)$$

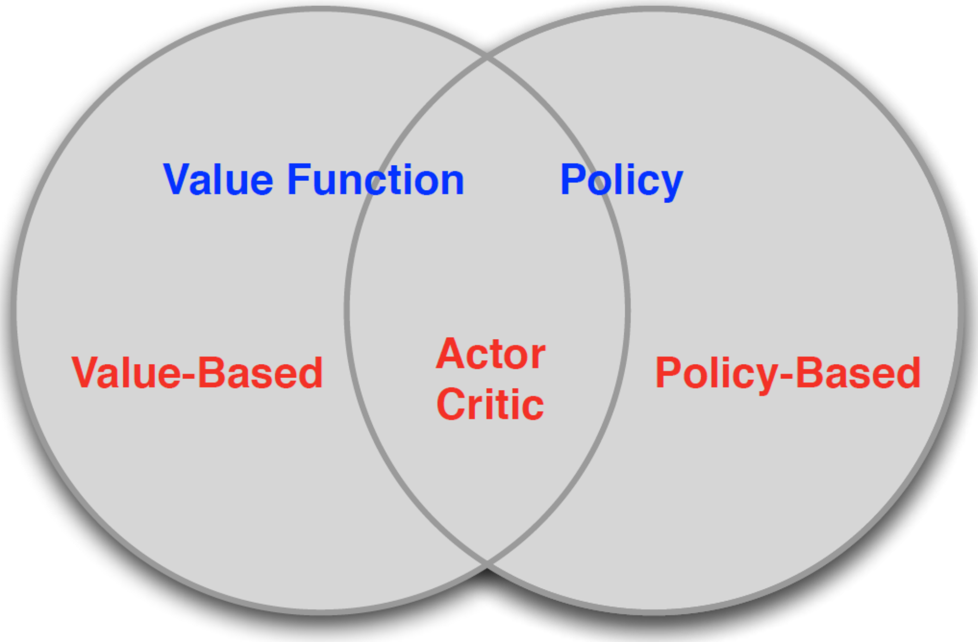
- Стратегия генерировалась напрямую по функции полезности (например, ϵ-жадно)
- Однако, можно напрямую параметризовать стратегию:

$$\hat{\pi}(s, \theta) = \mathbb{P}[a|s, \theta]$$

- Вапник (1998) – не нужно решать общую задачу через промежуточные шаги

Типология методов RL

- Основанные на полезности (value-based):
  - легко интерпретируемы,
  - могут концентрироваться на не самых важных признаках
- Поиск стратегии (policy-based):
  - цель поиска – сама стратегия,
  - игнорируются другие полезные данные
- Актор-критик (actor-critic)
  - строят как функцию полезности,
  - так и стратегию



Преимущества методов, основанных на стратегии

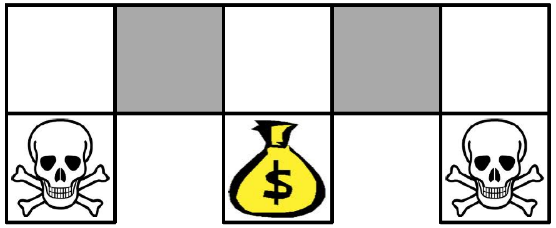
Плюсы:

- Лучшие свойства сходимости процесса обучения
- Эффективны в задачах большой размерности и непрерывных пространствах действий
- Могут обучаться стохастическим стратегиям
- Иногда обучить стратегию проще, чем функцию полезности

Минусы:

- Обычно сходятся к локальному оптимуму, а не к глобальному (особенно с нелинейными аппроксиматорами)
- Полученная модель обычно специфична для конкретной задачи и плохо обобщаема
- Обычно процесс вычисления стратегии неэффективен и обладает высокой дисперсией

Стохастическая стратегия: затемненный клеточный мир



- Агент не различает серые клетки
- Рассмотрим признаки следующего вида, для всех направлений движения вверх, вниз, влево, вправо. Пример, стена сверху и снизу, идем на вправо:

$$\phi(s, a) = (1\ 1\ 0\ 0 | \textbf{вправо})$$

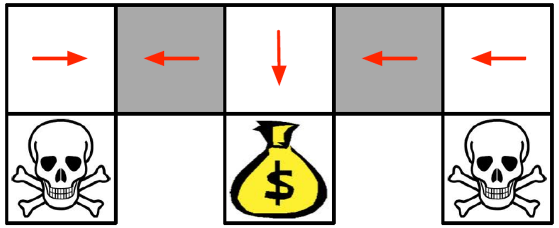
- Сравним метод, основанный на полезности с аппроксимацией функции:

$$\hat{Q}(s, a, \mathbf{w}) = f(\phi(s, a), \mathbf{w}),$$

- с методом, основанным на стратегии с параметризацией:

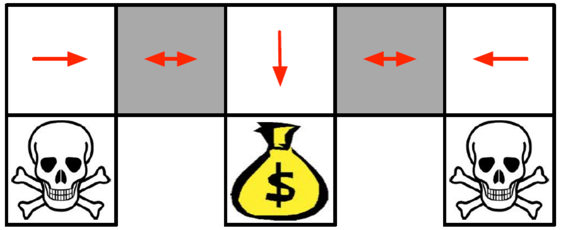
$$\hat{\pi}(s, a, \mathbf{w}) = g(\phi(s, a), \mathbf{w})$$

Пример: альтернативный клеточный мир



- Оптимальная **детерминированная** стратегия будет следующей:
  - двигаться **влево** в обоих серых состояниях (красные стрелки)
- В любом случае, агент может застрять и никогда не отыскать целевого состояния
- Основанные на полезности методы находят близкую к детерминированной стратегию (ϵ-жадную)
- В этом случае агент может блуждать по коридору длительное время

Пример: альтернативный клеточный мир



- Оптимальная стохастическая стратегия состоит в том, чтобы двигаться случайно **влево** или **вправо** в серых клетках:

$$\hat{\pi}(\text{стена } \textbf{снизу}, \text{двигаться } \textbf{вправо}, \mathbf{w}) = 0.5,$$

$$\hat{\pi}(\text{стена } \textbf{снизу}, \text{двигаться } \textbf{влево}, \mathbf{w}) = 0.5$$

- Это позволит достичь целевого состояния за небольшой количество шагов с высокой вероятностью
- Основанные на стратегии методы позволяют обучиться оптимальной стохастической стратегии

Функция полезности стратегии

- Цель: по данной стратегии $\hat{\pi}(s, a, \mathbf{w})$ с параметрами $\mathbf{w}$, найти наилучшее значение $\mathbf{w}$
- Как оценить качество стратегии $\hat{\pi}(\mathbf{w})$?
- В эпизодических средах мы можем использовать **начальную полезность**:

$$J_1(\mathbf{w}) = V^{\hat{\pi}(\mathbf{w})}(s_1) = \mathbb{E}_{\hat{\pi}(\mathbf{w})}[R_1]$$

- В непрерывных средах мы можем использовать **среднюю полезность**:

$$J_{avV}(\mathbf{w}) = \sum_s d^{\hat{\pi}(\mathbf{w})}(s) V^{\hat{\pi}(\mathbf{w})}(s)$$

  $d^{\hat{\pi}(\mathbf{w})}(s)$ – стационарное распределение марковской цепи для $\hat{\pi}(\mathbf{w})$

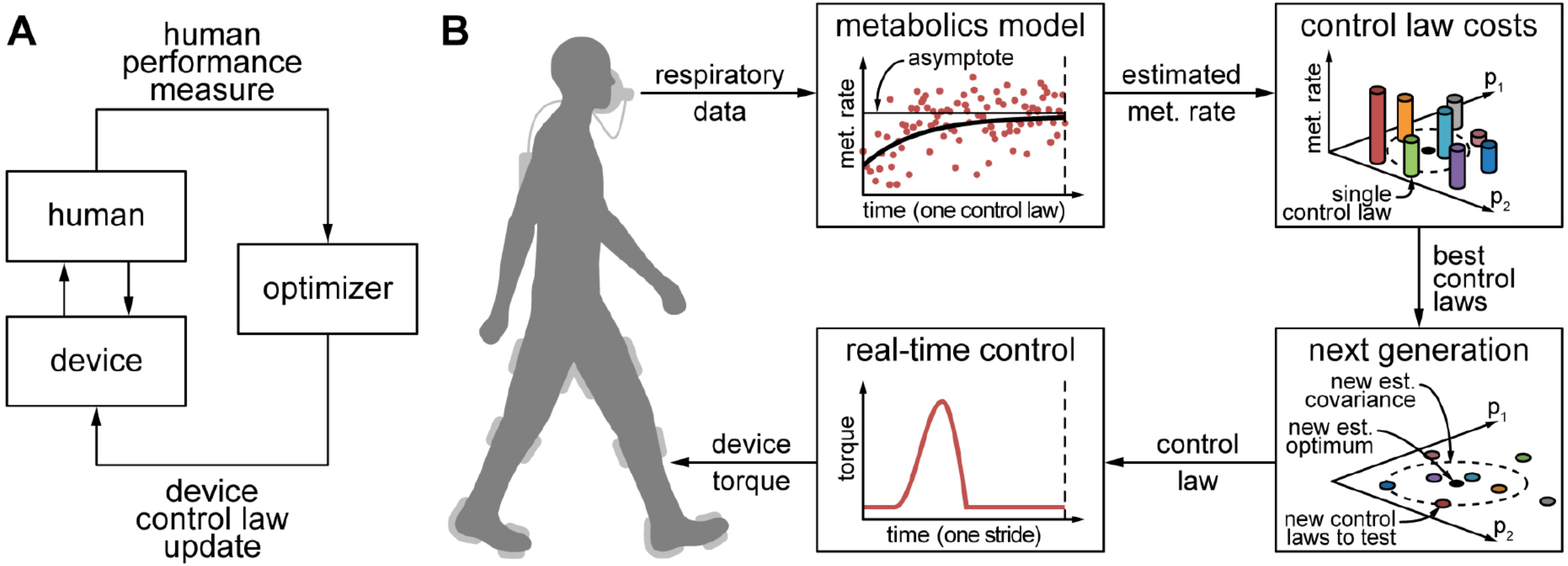
- Еще один вариант – среднее вознаграждение за шаг:

$$J_{avR}(\mathbf{w}) = \sum_s d^{\hat{\pi}(\mathbf{w})}(s) \sum_a \hat{\pi}(s, a, \mathbf{w}) \mathcal{R}_{sa}$$

Оптимизация стратегии

- Методы поиска стратегии – это алгоритмы оптимизации
- Необходимо найти значение $\mathbf{w}$, которое максимизирует функционал $J(\mathbf{w})$ или $V^{\hat{\pi}(\mathbf{w})}$
- Есть ряд методов, не использующих градиентный спуск:
  - восхождение по выпуклой поверхности (hill climbing),
  - симплекс метод (simplex) или метод Нелдера-Мида,
  - генетические алгоритмы
- Иногда могут демонстрировать отличную производительность (например, эволюционные алгоритмы как альтернатива классическому RL)

Неградиентеные методы: экзоскелет



Оптимизация выполнялась с помощью метода CMA-ES - варианта адаптации матрицы ковариаций (Zhang et al. Science 2017)

Оптимизация стратегии

- Однако большей эффективности можно добиться с помощью градиентных методов:
  - градиентный спуск,
  - метод сопряженный градиентов (conjugate gradient),
  - квази-ньютоновские методы (quasi-newton)
- Для нас наибольший интерес представляет градиентный спуск с большим количеством вариаций
- Будем иметь в виду эпизодический марковский процесс принятия решений

02

Теорема о градиенте стратегии

Траектория

- Посчитаем градиент стратегии **аналитически**
- Предположим, что $\hat{\pi}(\mathbf{w})$ дифференцируема, если она отлична от 0
- И мы можем вычислить градиент $\nabla_{\mathbf{w}}\hat{\pi}(s, a, \mathbf{w})$
- Будем называть траекторией следующую цепочку состояний-действий:

$$\tau = (s_0, a_0, r_0, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$$

- Пусть $r(\tau) = \sum_{t=0}^{T} r(s_t, a_t)$ - сумма вознаграждений по траектории τ

Стратегия и отношение правдоподобия

- Полезность стратегии будет определяться как

$$J(\mathbf{w}) = \mathbb{E}_{\pi(\mathbf{w})}\left[\sum_{t=0}^{T} r(s_t, a_t)\right] = \sum_{\tau} p(\tau, \mathbf{w}) r(\tau).$$

- Здесь $p(\tau, \mathbf{w})$ - распределение вероятностей по траекториям τ при стратегии $\pi(\mathbf{w})$
- Оптимизационная задача запишется следующим образом:

$$\arg\max_{\mathbf{w}} J(\mathbf{w}) = \arg\max_{\mathbf{w}} \sum_{\tau} p(\tau, \mathbf{w}) r(\tau)$$

- Наша задача - найти параметры $\mathbf{w}$ стратегии, распишем градиент:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \nabla_{\mathbf{w}} \sum_{\tau} p(\tau, \mathbf{w}) r(\tau) =$$

$$= \sum_{\tau} \nabla_{\mathbf{w}} p(\tau, \mathbf{w}) r(\tau) = \sum_{\tau} \frac{p(\tau, \mathbf{w})}{p(\tau, \mathbf{w})} \nabla_{\mathbf{w}} p(\tau, \mathbf{w}) r(\tau) =$$

$$= \sum_{\tau} p(\tau, \mathbf{w}) r(\tau) \frac{\nabla_{\mathbf{w}} p(\tau, \mathbf{w})}{p(\tau, \mathbf{w})} = \sum_{\tau} p(\tau, \mathbf{w}) r(\tau) \nabla_{\mathbf{w}} \log p(\tau, \mathbf{w})$$

Стратегия и отношение правдоподобия

- Оптимизационная задача запишется следующим образом:

$$\arg\max_{\mathbf{w}} J(\mathbf{w}) = \arg\max_{\mathbf{w}} \sum_{\tau} p(\tau, \mathbf{w}) r(\tau)$$

- Градиент по $\mathbf{w}$:

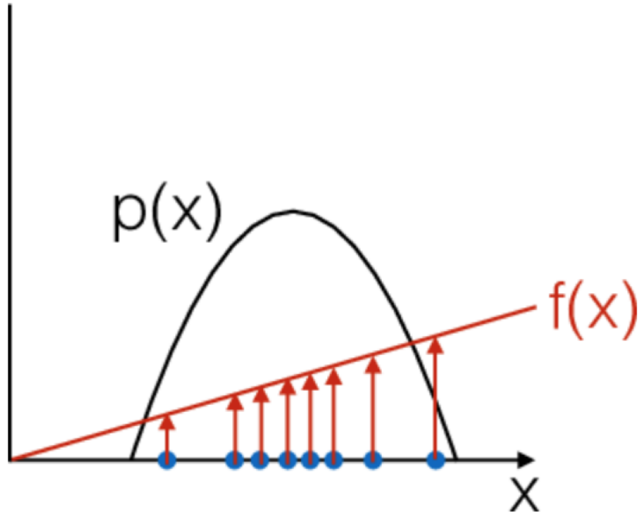
$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \sum_{\tau} p(\tau, \mathbf{w}) r(\tau) \nabla_{\mathbf{w}} \log p(\tau, \mathbf{w})$$

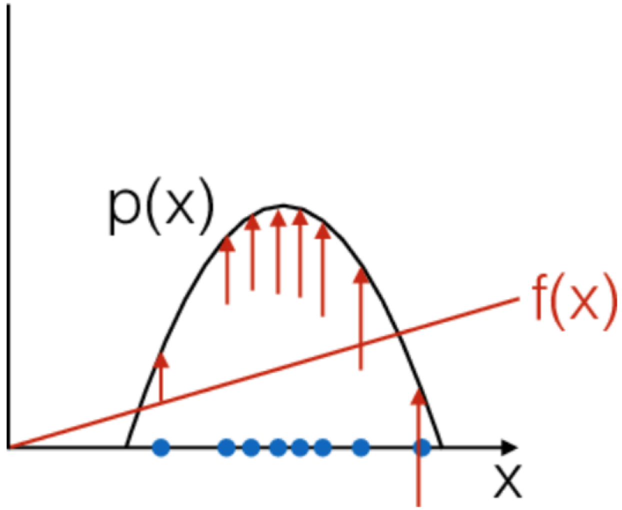
- В качестве приближения - эмпирическая оценка по выборке размера m:

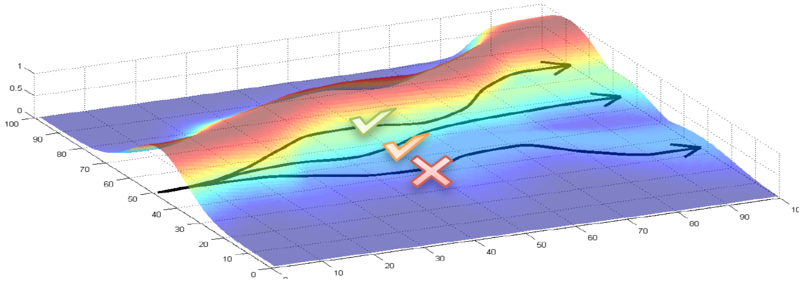
$$\nabla_{\mathbf{w}} J(\mathbf{w}) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} r(\tau^{(i)}) \nabla_{\mathbf{w}} \log p(\tau^{(i)}, \mathbf{w})$$

Результирующая функция

- Общий вид для $r(\tau^{(i)})\nabla_{\mathbf{w}} \log p(\tau^{(i)}, \mathbf{w})$: $\hat{g}_i = f(x_i)\nabla_{\mathbf{w}} \log p(x_i, \mathbf{w})$
  $f(x)$ измеряет насколько полезен пример x
- Сдвигаясь в направлении $\hat{g}_i$, увеличиваем $\log p$ примера пропорционально его полезности
- Это справедливо и для неизвестной функции $f(x)$ и дискретного множества примеров







Результирующая функция

- Эмпирическая оценка полезности по выборке размера m

$$\nabla_{\mathbf{w}} J(\mathbf{w}) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} r(\tau) \nabla_{\mathbf{w}} \log p(\tau^{(i)}, \mathbf{w})$$

- Пусть $\mu(s_0)$ - распределение начальных состояний, тогда

$$\nabla_{\mathbf{w}} \log p(\tau^{(i)}, \mathbf{w}) = \nabla_{\mathbf{w}} \log \left[\mu(s_0) \prod_{t=0}^{T-1} \hat{\pi}(a_t, s_t, \mathbf{w}) P(s_{t+1} | s_t, a_t) \right] =$$

$$= \nabla_{\mathbf{w}} \left[\log \mu(s_0) + \sum_{t=0}^{T-1} \log \hat{\pi}(a_t, s_t, \mathbf{w}) + \log P(s_{t+1} | s_t, a_t) \right] =$$

$$= \sum_{t=0}^{T-1} \log \hat{\pi}(a_t, s_t, \mathbf{w})$$

- Результирующая функция (score function) – это $\nabla_{\mathbf{w}} \log \hat{\pi}_{\mathbf{w}}(s, a, \mathbf{w})$

Градиент стратегии для эпизодической среды

- Оптимизационная задача запишется следующим образом:

$$\arg\max_{\mathbf{w}} J(\mathbf{w}) = \arg\max_{\mathbf{w}} \sum_{\tau} p(\tau, \mathbf{w}) r(\tau)$$

- Эмпирическая оценка полезности по выборке размера m для стратегии $\pi(\mathbf{w})$ по результирующей функции:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} r(\tau) \nabla_{\mathbf{w}} \log p(\tau^{(i)}, \mathbf{w}) =$$

$$= \frac{1}{m} \sum_{i=1}^{m} r(\tau^{(i)}) \sum_{t=0}^{T-1} \nabla_{\mathbf{w}} \log \pi(a_t^{(i)}, s_t^{(i)}, \mathbf{w})$$

- Нам не нужна модель динамики!
- Несмещенная, но очень зашумленная оценка

Теорема о градиенте стратегии

- Теорема о градиенте стратегии обобщает подход коэффициентов правдоподобия
- Заменяем текущее значение отдачи на долговременное значение оценки полезности $Q^{\hat{\pi}(\mathbf{w})}(s,a)$
- Теорема о градиенте стратегии применяется к полной отдаче, среднему вознаграждению и средней полезности

Теорема

Для любой дифференцируемой стратегии $\hat{\pi}(s,a,\mathbf{w})$, для любой функции полезности стратегии $J = J_1$, J_{avR} или $\frac{1}{1-\gamma}J_{avV}$ градиент стратегии равен:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \mathbb{E}_{\hat{\pi}(\mathbf{w})}[\nabla_{\mathbf{w}} \log \hat{\pi}(s,a,\mathbf{w}) Q^{\hat{\pi}(\mathbf{w})}(s,a)]$$

03

Монте-Карло градиент стратегии

Монте-Карло градиент стратегии

- Будем использовать отдачу R_t в качестве несмещенной оценки $Q^{\hat{\pi}(\mathbf{w})}(s, a)$:

$$\nabla_{\mathbf{w}} \mathbb{E}[r(\tau)] \approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T-1} \nabla_{\mathbf{w}} \log \hat{\pi}(s, a, \mathbf{w}) R_t$$

- Обновляем параметры с помощью стохастического градиентного спуска.

Algorithm 1 REINFORCE

function REINFORCE
    инициализируем $\mathbf{w}$;
    for каждого эпизода $\{s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T\} \sim \hat{\pi}(\mathbf{w})$ **do**
        for $t = 1$ и до $T - 1$ **do**
            $\mathbf{w} \leftarrow \mathbf{w} + \alpha \nabla_{\mathbf{w}} \log \hat{\pi}(s_t, a_t, \mathbf{w}) R_t$
        end for
    end forreturn $\mathbf{w}$
end function

Пример параметризации: логистическая стратегия

- Будем использовать логистическую стратегию в качестве примера
- Взвесим действия, используя линейную комбинацию признаков $\phi(s,a)^T\mathbf{w}$
- Вероятность действия пропорциональна экспоненциальным весам:

$$\hat{\pi}(s,a,\mathbf{w}) = \frac{e^{\phi(s,a)^T\mathbf{w}}}{\sum_a e^{\phi(s,a)^T\mathbf{w}}}$$

- Результирующая функция:

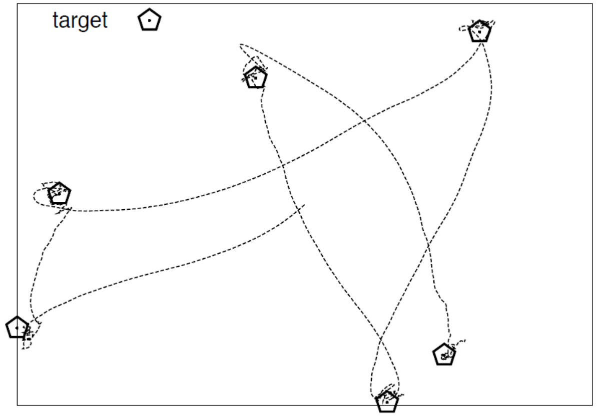
$$\nabla_{\mathbf{w}} \log \hat{\pi}(s,a,\mathbf{w}) = \phi(s,a) - \mathbb{E}_{\hat{\pi}(\mathbf{w})}[\phi(s,\cdot)]$$

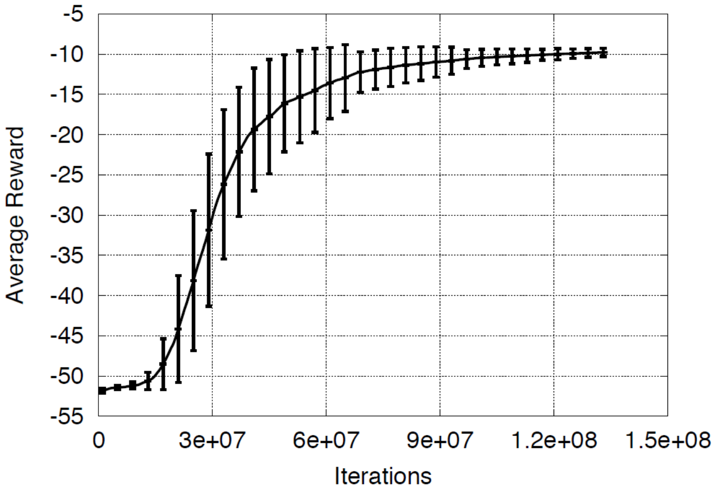
Гауссова стратегия

- В непрерывном пространстве действий подойдет гауссова стратегия
- Среднее – это линейная комбинация признаков состояния $\mu(s) = \phi(s)^T \mathbf{w}$
- Дисперсия может быть фиксированной σ^2, или тоже может быть параметризована
- Гауссова стратегия задается как $\approx \mathcal{N}(\mu(s), \sigma^2)$
- Результирующая функция:

$$\nabla_{\mathbf{w}} \log \hat{\pi}(s, a, \mathbf{w}) = \frac{(a - \mu(s))\phi(s)}{\sigma^2}$$

Пример с шайбой





- Непрерывное пространство действий – оказание небольшого воздействия на шайбу
- Мы получаем вознаграждение, когда шайба оказалась возле цели
- Положение цели меняется каждые 30 секунд
- Стратегия была построена с использованием одного из вариантов Монте-Карло градента стратегии

04

Базовый уровень полезности

Требования к градиенту стратегии

- Цель - максимально быстро сойтись к локальному минимуму
  - Вычисляя вознаграждения при выполнении стратегии, хотим минимизировать количество итераций до достижения нужной стратегии
- Во время поиска стратегии поочередно оцениваем стратегию и обновляем ее по аналогии с итерациями по стратегиям
- Основная задача - добиться максимального монотонного улучшения стратегии на каждой итерации:
  - получение более точной оценки градиента (улучшение обновления параметров стратегии),
  - изменение способа обновления параметров стратегии на основе полученного градиента

Оценка градиента стратегии

- Оценка градиента по траекториям:

$$\nabla_\theta J(\theta) \approx \hat{g} = \frac{1}{m}\sum_{i=1}^{m} r(\tau^{(i)}) \sum_{i=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}).$$

- Оценка несмещенная, но с высокой дисперсией
- Способы борьбы с дисперсией:
  - использование временной структуры MDP,
  - **учет базового уровня**,
  - использование других оценок вместо Монте-карло подхода

Базовый уровень для градиента стратегии

- Уменьшение дисперсии введением базового уровня $B(s_t)$:

$$\nabla_\theta \mathbb{E}_\tau[R(\tau)] = \mathbb{E}_\tau \left[\sum_{i=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}) \left(\sum_{t'=t}^{T-1} r_{t'} - B(s_t) \right) \right]$$

- При любом выборе $B(S_t)$ оценка градиента останется несмещенной
- Квази-оптимальный выбор базового уровня - ожидаемая отдача:

$$B(s_t) \approx [r_t + r_{t+1} + \ldots r_{T-1}]$$

- Интерпретация: увеличение $\log p$ действия a_t пропорционально тому, насколько отдача $\sum r_{t'}$ лучше ожидаемой

Базовый алгоритм градиента стратегии

Algorithm 2 Vanilla PG

function VPG
    Инициализируем параметры стратегии θ и базовый уровень B
    for итераций $i = 1, 2, \dots,$ **do**
        набираем множество траекторий, выполняя текущую стратегию
        for каждого шага t траектории τ^i **do**
            $R_t^i = \sum_{t'=t}^{T-1} r_{t'}$,
            $\hat{A}_t^i = R_t^i - B(s_t)$ - **оценка преимущества**,
        end for
        обновляем базовый уровень, минимизируя $\sum_i \sum_t \|B(s_t) - R_t^i\|^2$,
        $\hat{g} = \nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t$,
        обновляем стратегию, используя оценку градиента $\hat{g}$.
    end for
end function

Выбор базового уровня

- Функция полезности состояния-действия:

$$Q^{\pi,\gamma}(s,a) = \mathbb{E}_\pi[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots | s_0 = s, a_0 = a]$$

- Функция полезности состояния может служить отличным базовым уровнем:

$$V^{\pi,\gamma}(s) = \mathbb{E}_\pi[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots | s_0 = s] = \mathbb{E}_{a\sim\pi}[Q^{\pi,\gamma}(s,a)]$$

- **Функция преимущества** (advantage function) - комбинация функций полезности и базового уровня:

$$A^{\pi,\gamma} = Q^{\pi,\gamma}(s,a) - V^{\pi,\gamma}(s)$$

05

Оценка стратегии с помощью критика

Оценка градиента стратегии

- Оценка градиента по траекториям:

$$\nabla_\theta J(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} R(\tau^{(i)}) \sum_{i=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}).$$

- Оценка несмещенная, но с высокой дисперсией
- Способы борьбы с дисперсией:
  - использование временной структуры MDP,
  - учет базового уровня,
  - **использование других оценок вместо Монте-Карло подхода**

Уменьшение дисперсии с использованием критика

- В методе Монте-Карло градиента стратегии большая дисперсия решения
- Попробуем использовать **критика** (critic) для оценки функции полезности действия:

$$Q_w(s, a) \approx Q^{\pi_\theta}(s, a)$$

- Алгоритмы актор-критик (actor-critic) поддерживают два множества параметров:

  критик обновляет параметры w функции полезности действия,

  актор обновляет параметры θ стратегии с учетом предположений критика

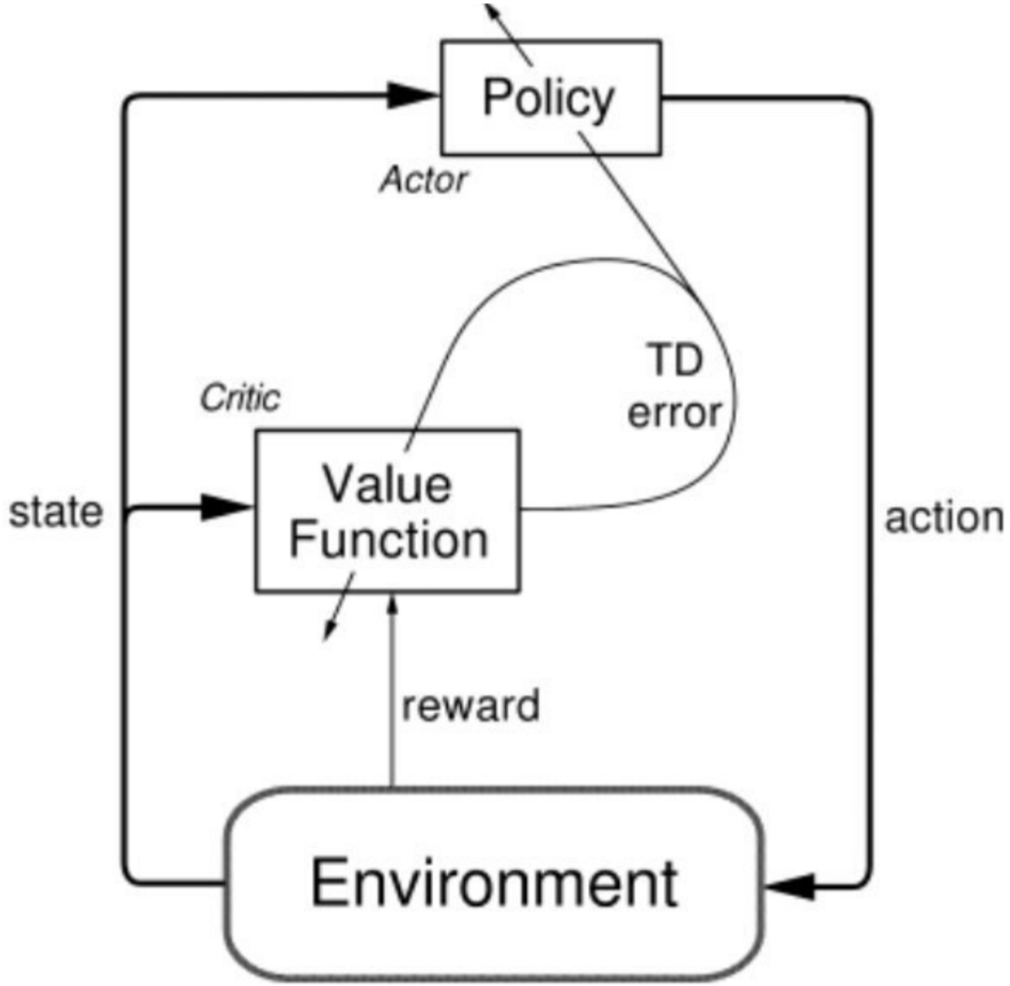
- Алгоритмы актор-критик следуют по градиенту **приближенной** стратегии:

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)],$$

$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$$

Оценка функции полезности действия

- Критик оценивает стратегию
- Насколько хороша стратегия π_θ при текущих параметрах θ
- Мы знаем следующие способы решения этой задачи:
  - Монте-Карло оценка стратегии,
  - обучение на основе временных различий
- Можем также использовать оценку стратегии методом наименьших квадратов



Полезность действия актор-критика

- Рассмотрим простейший алгоритм актор-критика на основе полезности действия критика
- Будем использовать линейную аппроксимационную функцию $Q_w(s,a) = \phi(s,a)^T w$:
  критик обновляет параметры w с помощью TD(0),
  актор обновляет параметры θ, используя градиент стратегии

Algorithm 3 QAC

---

function QAC
  Инициализируем s, θ
  Выбираем $a \sim \pi_\theta$
  for all шагов **do**
    получаем вознаграждение $r = \mathcal{R}_s^a$ и следующее состояние $s' \sim \mathcal{P}_s^a$
    выбираем следующее действие $a' \sim \pi_\theta(s')$
    $\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$
    $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$
    $w \leftarrow w + \beta \delta \phi(s, a)$
    $a \leftarrow a', s \leftarrow s'$
  end for

Смещенность в алгоритмах актор-критик

- Аппроксимация градиента стратегии приводит к смещению оценки
- Смещенный градиент стратегии может не позволить найти правильное решение
- Например, использование признаков в $Q_w(s, a)$ для клеточного мира с неоднозначным определением состояния
- К счастью, у нас есть возможность выбрать такую функцию аппроксимации, которая позволит избежать смещенных оценок
- Можем все-еще использовать точный градиент стратеги

Совместимые функции аппроксимации

Теорема о совместимой функции аппроксимации

Если удовлетворены следующие два условия:

1. аппроксиматор функции полезности **совместим** со стратегией:

$$\nabla_w Q_w(s, a) = \nabla_\theta \log \pi_\theta(s, a),$$

2. параметры функции полезности минимизируют среднеквадратичную ошибку:

$$\epsilon = \mathbb{E}_{\pi_\theta}[(Q^{\pi_\theta}(s, a) - Q_w(s, a))^2],$$

тогда градиент стратегии в точности равен

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)]$$

Оценка функции преимущества

- Функция преимущества (advantage function) может существенно уменьшить дисперсию градиента стратегии
- Критик должен на самом деле оценивать функцию преимущества
- Например, оценивая как $V^{\pi_\theta}(s)$, так и $Q^{\pi_\theta}(s,a)$
- Используем два аппроксиматора и два вектора параметров:

$$V_v(s) \approx V^{\pi_\theta}(s),$$

$$Q_w(s,a) \approx Q^{\pi_\theta}(s,a),$$

$$A(s,a) = Q_w(s,a) - V_v(s)$$

- Обновляем обе функции полезности с помощью, например, TD-обучения

Оценка функции преимущества

- Для истинной функции полезности $V^{\pi_\theta}(s)$, TD-ошибка равна

$$\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$$

- Она является несмещенной оценкой функции преимущества:

$$\begin{aligned}\mathbb{E}_{\pi_\theta}[\delta^{\pi_\theta}|s,a] &= \mathbb{E}_{\pi_\theta}[r + \gamma V^{\pi_\theta}(s')|s,a] - V^{\pi_\theta}(s) \\ &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) \\ &= A^{\pi_\theta}(s,a)\end{aligned}$$

- Таким образом, мы можем использовать TD-ошибку для вычисления градиента стратегии:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\delta^{\pi_\theta}]$$

- На практике, мы используем аппроксимацию TD-ошибки:

$$\delta_v = r + \gamma V_v(s') - V_v(s)$$

- В этом подходе нам достаточно использовать только один вектор параметров v

Семейство алгоритмов градиента стратегии

- Градиент стратегии имеет несколько эквивалентных форм:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) R_t] && REINFORCE \\
&= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^w(s,a)] && Q\ Actor-Critic \\
&= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) A^w(s,a)] && Advantage\ Actor-Critic \\
&= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) \delta] && TD\ Actor-Critic
\end{aligned}$$

- Каждый из может быть использован в стохастическом градиентном спуске
- Критик использует оценку стратегии (MC или TD-обучение) для оценки $Q^\pi(s,a), A^\pi(s,a), V^\pi(s)$

06

Алгоритм АЗС

Выбор оценки полезности траектории

- R_t^i - оценка функции полезности состояния на основе одного прогона (roll out)
- Эта оценка несмещенная, но обладает высокой дисперсией
- Уменьшение дисперсии за счет введения смещения (временные различия и аппроксимация функции)
- Оценка V/Q делается **критиком**
- Методы **актор-критика** поддерживают явное представление и стратегии, и функции полезности
- Пример - метод A3C (Asynchronous Advantage Actor-Critic) `https://arxiv.org/abs/1602.01783` - один из некогда популярных алгоритмов, поддерживает параллельные вычисления

Градиент стратегии с функцией полезности

→ Оценка полезности траектории

$$\nabla_\theta \mathbb{E}_\tau[R(\tau)] = \mathbb{E}_\tau \left[\sum_{i=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}) \left(\sum_{t'=t}^{T-1} r_{t'} - B(s_t) \right) \right]$$

$$\nabla_\theta \mathbb{E}_\tau[R(\tau)] = \mathbb{E}_\tau \left[\sum_{i=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}) \left(Q_w(s_t) - B(s_t) \right) \right]$$

→ Если наш базовый уровень определяется функцией полезности, мы получаем использование функции преимущества:

$$\nabla_\theta \mathbb{E}_\tau[R(\tau)] = \mathbb{E}_\tau \left[\sum_{i=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}) \hat{A}^\pi(s_t, a_t) \right]$$

Оценка полезности траектории: N-шаговые оценки

$$\nabla_\theta V(\theta) \approx (1/m) \sum_{i=1}^{m} \sum_{i=0}^{T-1} R_t^i \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

- Критик может выбрать любую смесь между оценками TD и MC для целевого значения, чтобы заменить истинную функцию полезности состояния-действия:

$$\hat{R}_t^{(1)} = r_t + \gamma V(s_{t+1})$$
$$\hat{R}_t^{(2)} = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2})$$
$$\hat{R}_t^{(\text{inf})} = r_t + \gamma r_{t+1} \gamma^2 r_{t+2} + \dots$$

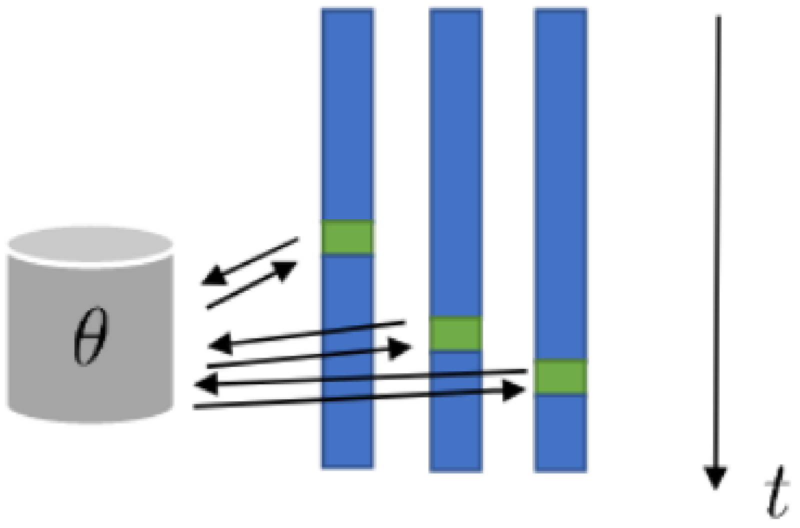
- По аналогии получаем с функцией преимущества:

$$\hat{A}_t^{(1)} = r_t + \gamma V(s_{t+1}) - V(s_t)$$
$$\hat{A}_t^{(\text{inf})} = r_t + \gamma r_{t+1} \gamma^2 - r_{t+2} + \dots - V(s_t)$$

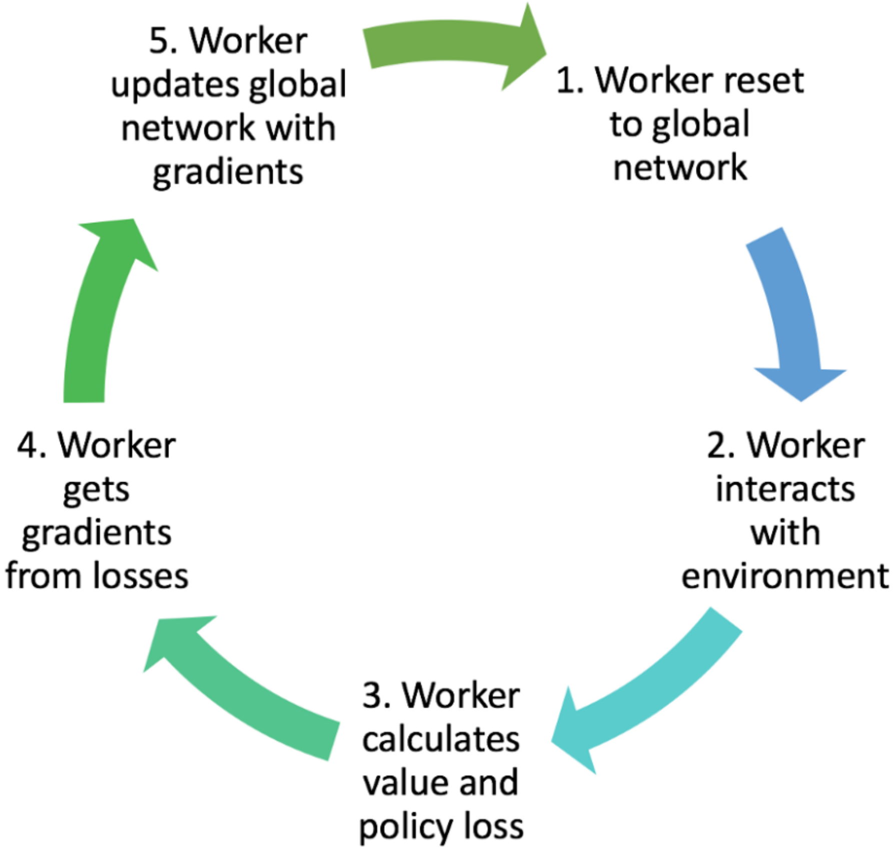
- $A_t^{(1)}$ имеет меньшую дисперсию и большую смещенность, $A_t^{(\text{inf})}$ - наоборот.

Особенности A3C

- Критик обновляет функцию полезности пока множество акторов работают параллельно
- Критики синхронизируются время от времени по глобальным переменным
- Использование n-шаговых оценок полезности



Цикл потоков в A3C

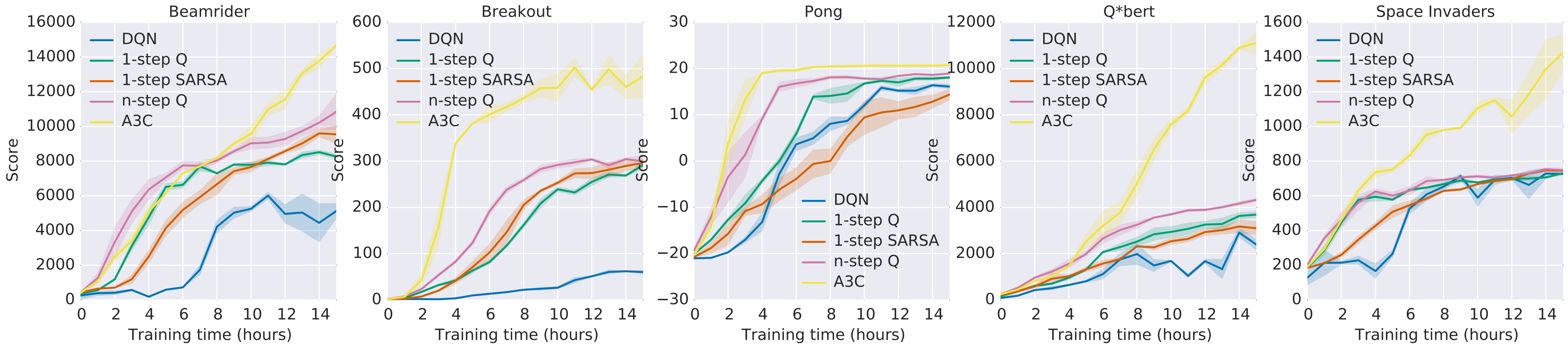


Алгоритм A3C

Algorithm 4 A3C

θ, w – глобальные переменные, θ', w' – для каждого потока, $t = 1$
while $T \leq T_{max}$ **do**
    $\Delta\theta = 0, \Delta w = 0$
    синхронизируем $\theta' = \theta, w' = w$
    $t_{start} = t$, выбираем состояние s_t
    while s_t - нетерминальное, $t - t_{start} < t_{max}$ **do**
        применяем $a_t \sim \pi_{\theta'}(a_t|s_t)$ и получаем r_t, s_{t+1}
        $t \leftarrow t + 1, T \leftarrow T + 1$
    end while
    $R = 0$ или $R = V_{w'}(s_t)$, если s_t - нетерминальное
    for $i = \{t - 1, \dots, t_{start}\}$ **do**
        $r(\tau) \leftarrow \gamma r(\tau) + r_i$
        $\Delta\theta \leftarrow \theta + \nabla_{\theta'} \log \pi_{\theta'}(a_i|s_i)(r(\tau) - V_{w'}(s_i))$
        $\Delta w \leftarrow \Delta w + \nabla_{w'} \left(r(\tau) - V_{w'}(s_i)\right)^2$
    end for
end while

Алгоритм A3C



Вопросы?