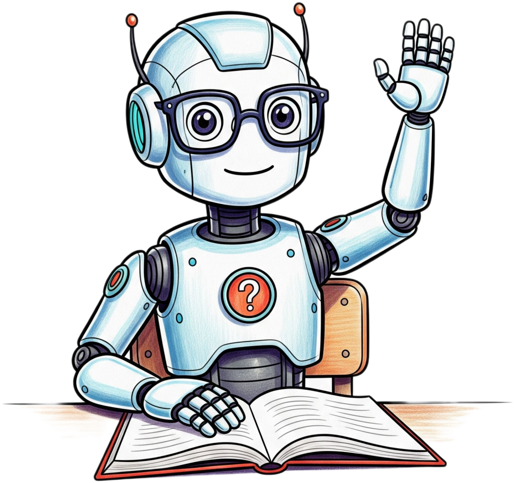


Policy-based

30 сентября 2025

О чем сегодня поговорим?

- ➔ Откроем новый тип алгоритмов RL

- ➔ Докажем теорему
- ➔ И ещё одну

- ➔ Соответственно получим +2 новых алгоритма

Recap. Value-based

Что мы обучали в Q-learning и DQN?

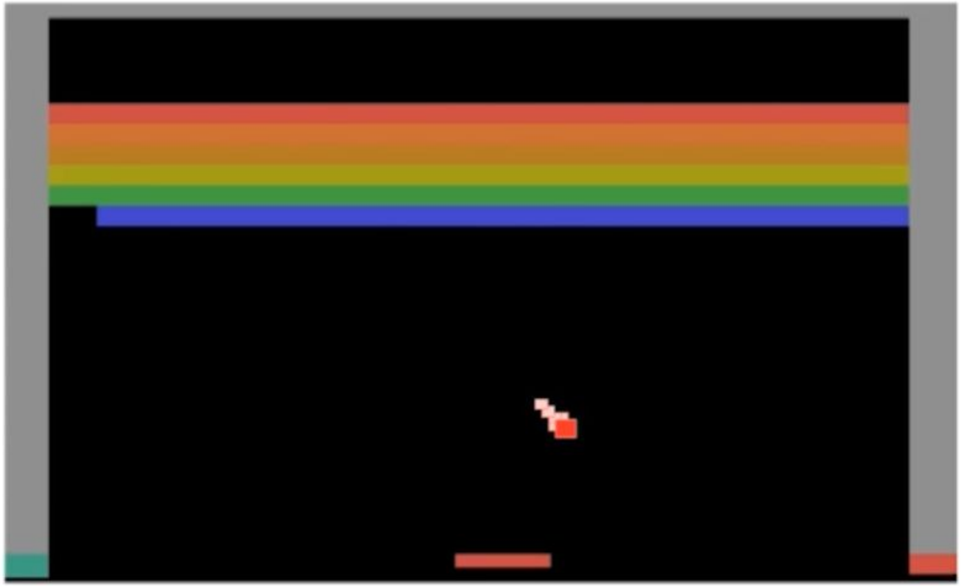
Ответ: Q

Social Experiment 1

left or **right**?

Social Experiment 2

Too hard
(harder than the task actually)



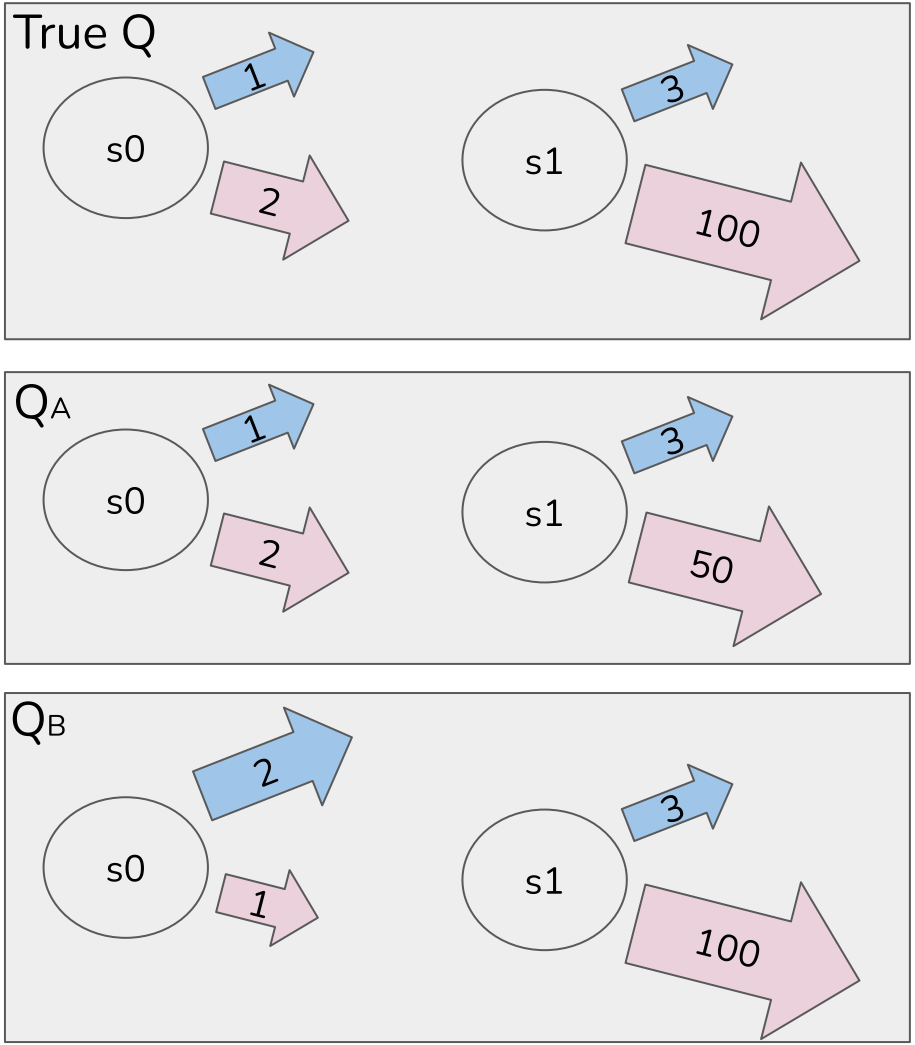
What's **Q(s,right)** under gamma=0.99?

Минусы Q-learning?

Simple 2-state world

	True	(A)	(B)
Q(s0,a0)	*1*	*1*	*2*
Q(s0,a1)	*2*	*2*	*1*
Q(s1,a0)	*3*	*3*	*3*
Q(s1,a1)	*100*	*50*	*100*

$$L \approx E[Q(s,a) - (R_{t+1} + \gamma \max_{a'} Q(s',a'))]^2$$



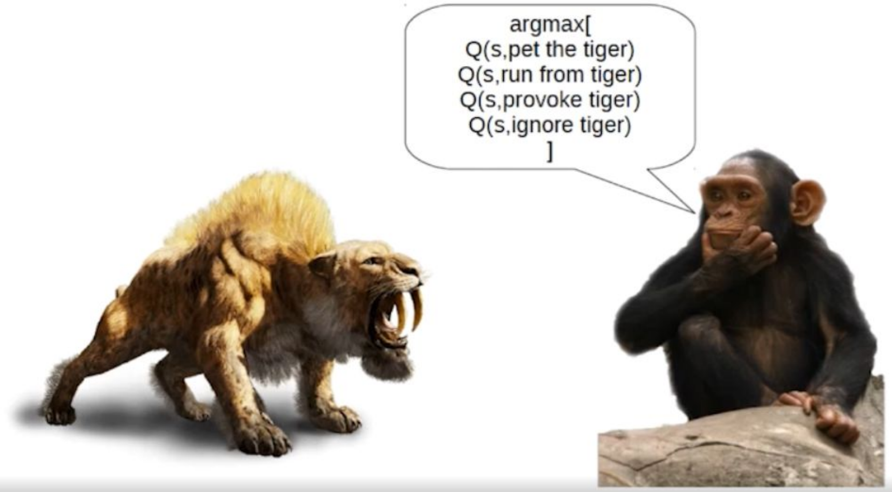
RIGHT POLICY

LOW MSE

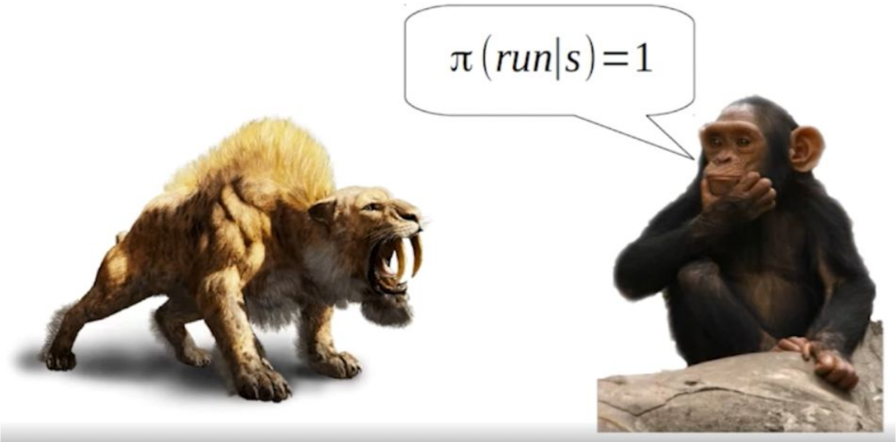
Итоги

- Вычислять Q сложнее, чем выбирать лучшие действия
- Для того, чтобы сразу учиться выбирать лучшие действия, будем учить политику напрямую

NOT how humans survived



how humans survived



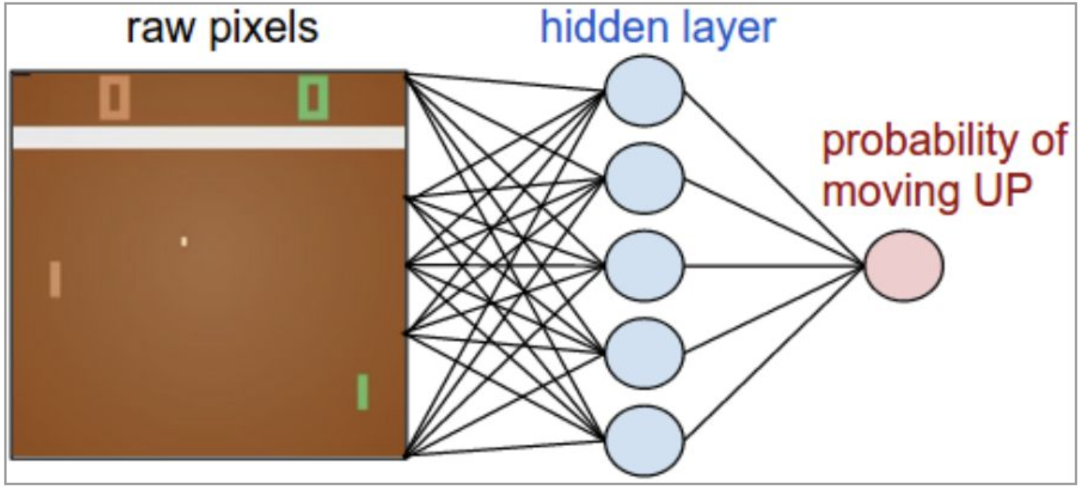
Стохастические и детерминированные политики

$a \sim \pi(a \mid s)$

$a = \pi(s)$

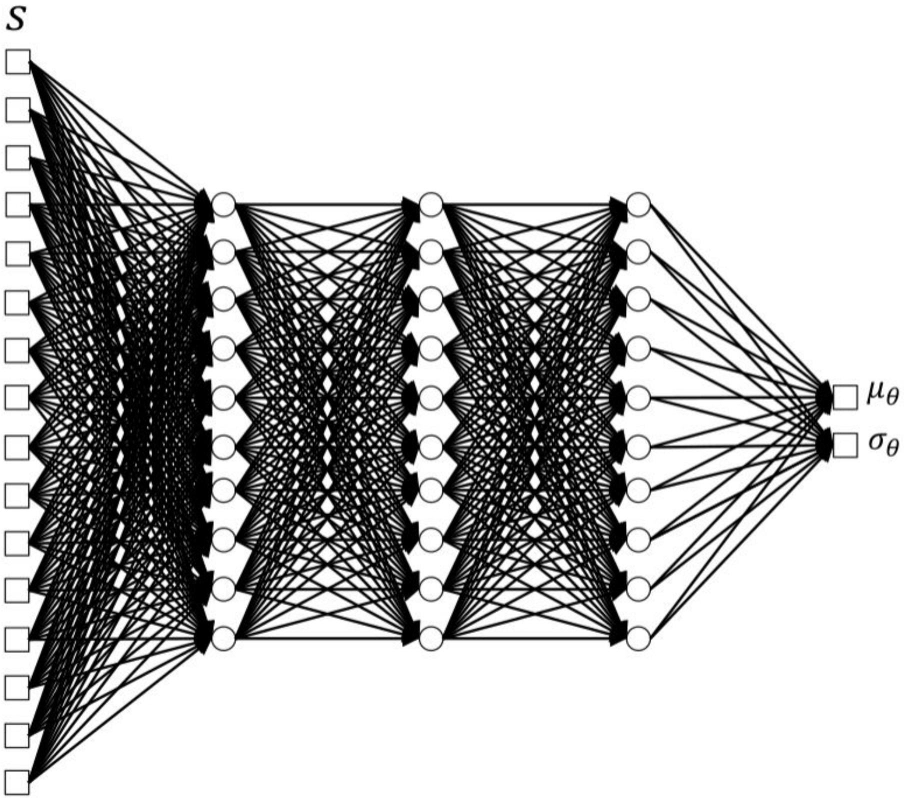
- Детерминированные частный случай стохастических.
- Где нужны стохастические?

Как работать с дискретными действиями?



Our policy network is a 2-layer fully-connected net.

Как работать с непрерывными действиями?



$\mu, \sigma \leftarrow \pi_\Theta(\text{state})$

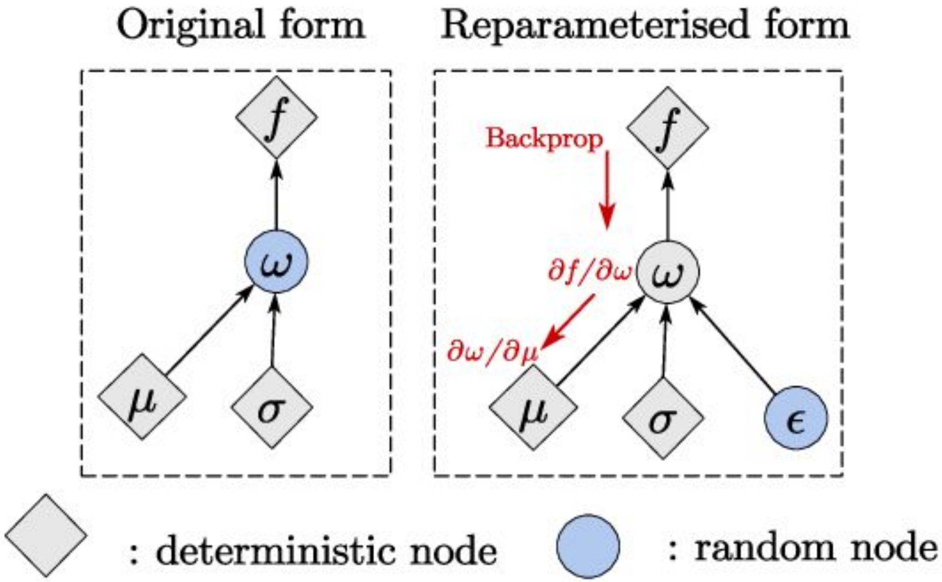
$\text{action} \sim N(\cdot \mid \mu, \sigma)$

wait, but how I backprop then?

$\text{action} \leftarrow \mu + \sigma \cdot \epsilon$, where $\epsilon \sim N(\cdot \mid 1, 0)$

https://lilianweng.github.io/posts/2018-08-12-vae/

Reparametrization trick



$\mu, \sigma \leftarrow \pi_\Theta(\text{state})$

$\text{action} \sim N(\cdot \mid \mu, \sigma)$

wait, but how I backprop then?

$\text{action} \leftarrow \mu + \sigma \cdot \epsilon$, where $\epsilon \sim N(\cdot \mid 1, 0)$

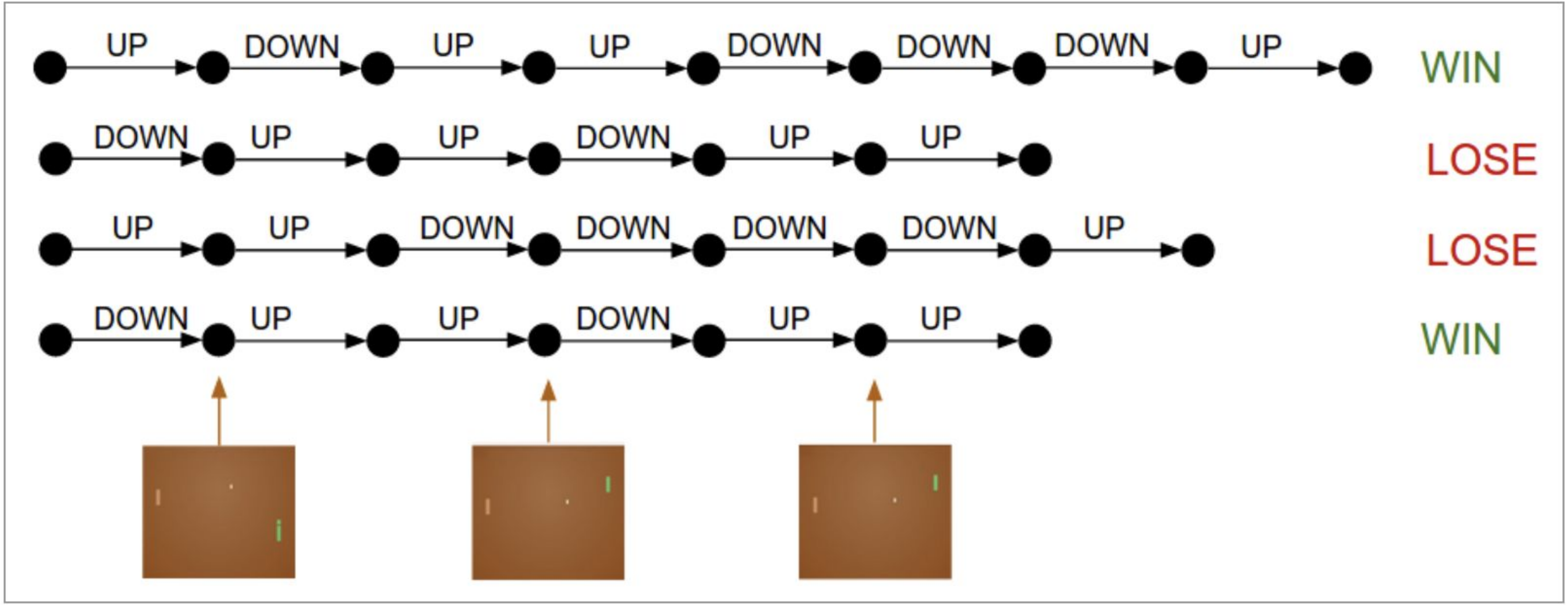
Что мы хотим?

We want max G over π

G - ожидаемая кумулятивная дисконтированная награда

https://karpathy.github.io/2016/05/31/rl/

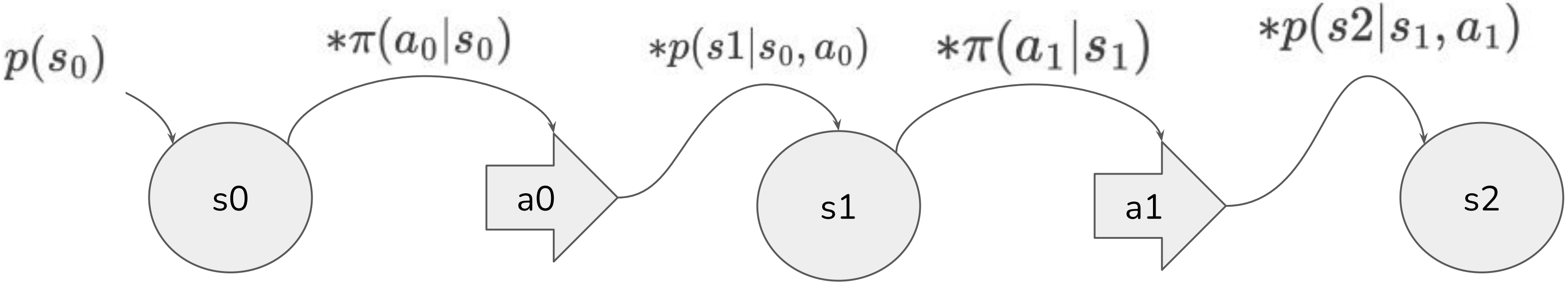
Что мы хотим?



Теперь более аккуратно

1. Траектория (эпизод): $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T)$
2. Функция награды за траекторию: $G(\tau) = \sum_{t=0}^{T} \gamma^t \, r_t$
3. Распределение траекторий под политикой π_θ:

$$p(\tau;\theta) = p(s_0) \prod_{t=0}^{T} \pi_\theta(a_t|s_t)\, p(s_{t+1}|s_t, a_t)$$



Целевая функция (Objective)

Ожидаемое суммарное вознаграждение: $J(\theta) = \mathbb{E}_{\tau \sim p(\tau;\theta)}[G(\tau)]$

или более развёрнуто: $J(\theta) = \int p(\tau;\theta)\, G(\tau)\, d\tau$

А нам нужен градиент (чтобы максимизировать)

$$\nabla_\theta J(\theta) = \nabla_\theta \int p(\tau;\theta)\, G(\tau)\, d\tau$$

Вносим градиент внутрь: $\nabla_\theta J(\theta) = \int \nabla_\theta p(\tau;\theta)\, G(\tau)\, d\tau$

Offtop

Кто помнит, чему равно $d\ln(x)$?

Варианты ответа:

1. dx
2. $\frac{1}{x}\,dx$
3. $x\,dx$
4. $\ln(x)\,dx$

Log-derivative trick OR трюк с логарифмом 🪄

$$d\ln(x) = \frac{1}{x}\,dx$$

$$x\,d\ln(x) = dx$$

$$dx = x\,d\ln(x)$$

Тогда:

$$\nabla_\theta J(\theta) = \int \nabla_\theta\, p(\tau;\theta)\; G(\tau)\; d\tau$$

У нас тут есть $\nabla_\theta\, p(\tau;\theta)$

Используем трюк с логарифмом: $\nabla_\theta\, p(\tau;\theta) = p(\tau;\theta)\; \nabla_\theta \log p(\tau;\theta)$

Подставляем: $\nabla_\theta J(\theta) = \int p(\tau;\theta)\; \nabla_\theta \log p(\tau;\theta)\; G(\tau)\; d\tau$

Итого: $\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau\sim p(\tau;\theta)}\left[\nabla_\theta \log p(\tau;\theta)\; G(\tau)\right]$

А зачем нам $\log p(\tau;\theta)$?

$$a^{\log_a b} = b$$

Свойства логарифмов:

1. $\log_a 1 = 0$
2. $\log_a a = 1$
3. $\boxed{\log_a bc = \log_a b + \log_a c}$
4. $\log_a \frac{b}{c} = \log_a b - \log_a c$
5. $\log_a b^n = n \cdot \log_a b$
6. $\log_{a^k} b = \frac{1}{k} \cdot \log_a b$
7. $\log_{a^k} b^n = \frac{n}{k} \cdot \log_a b$
8. $\log_{a^n} b^n = \log_a b$
10. $\log_a b = \frac{\log_d b}{\log_d a} = \frac{1}{\log_b a}$
11. $\log_a b \cdot \log_b a = 1$
12. $a^{\log_b c} = c^{\log_b a}$

$a > 0$

$a \neq 1$

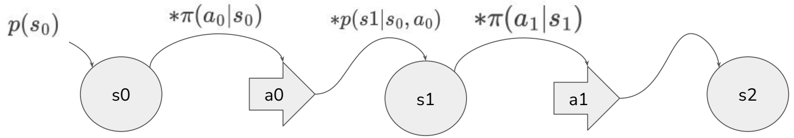
$b > 0$

$c > 0$

$d > 0$

$d \neq 1$

А зачем нам $\log p(\tau;\theta)$?



Напомним: $p(\tau;\theta) = p(s_0)\prod_{t=0}^{T}\pi_\theta(a_t|s_t)\,p(s_{t+1}|s_t,a_t)$

Берём логарифм: $\log p(\tau;\theta) = \log p(s_0) + \sum_{t=0}^{T}\log\pi_\theta(a_t|s_t) + \sum_{t=0}^{T}\log p(s_{t+1}|s_t,a_t)$

Зависимость от θ только в π_θ. Поэтому: $\nabla_\theta \log p(\tau;\theta) = \sum_{t=0}^{T}\nabla_\theta\log\pi_\theta(a_t|s_t)$

Итого: перешли

от вероятности траектории (много шагов политики + шаги среды)

к политике (один шаг политики)

Получили REINFORCE

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau \sim p(\tau;\theta)} \left[\left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) G(\tau) \right]$$

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, G_t \right]$$

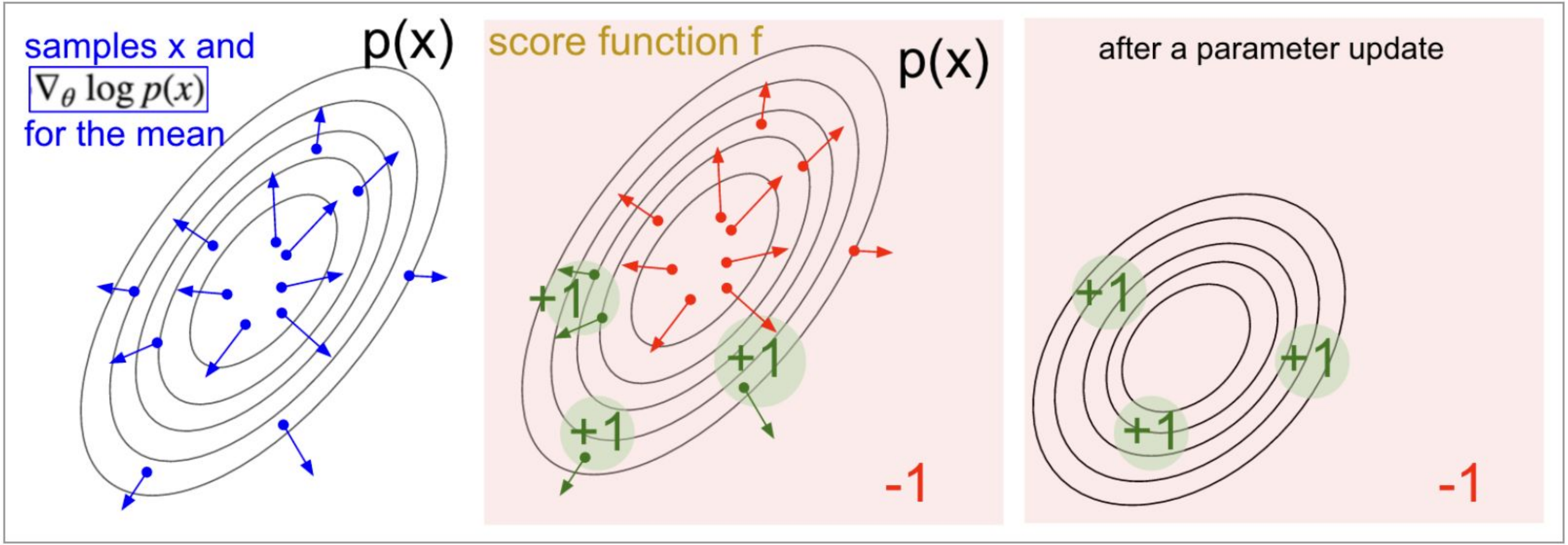
Почему $\nabla_\theta J(\theta) = \mathbb{E}\Big[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, G_t\Big]$ — это круто?

«Если действие привело к большой награде, увеличим вероятность его выбора». Всё!

1. **Не нужна модель среды**
    - Учимся **на опыте**, на основе реальных траекторий.
    - Подходит для любых наград и неизвестных динамик среды.
    - То есть для любой награды, на основе опыта сможем учиться улучшать действия, которые максимизируют её
2. **Гибкость действий**
    - Работает для **дискретных** и **непрерывных** действий.
    - Любая параметризованная политика: нейросеть, линейная модель и т.д.
3. **Теоретическая гарантия сходимости**
    - Градиент действительно направлен в сторону увеличения ожидаемой награды.
    - Если шаги обучения малы и эпизоды разнообразны, мы постепенно улучшаем политику.

https://karpathy.github.io/2016/05/31/rl/

Иллюстрация для policy gradient



А теперь минусы REINFORCE (по сравнению с DQN)

1. Высокая дисперсия градиента

REINFORCE:

- Градиент, вычисленный по одной траектории, может сильно отличаться от «истинного».
- **Последствия:**
  - Медленная сходимость → нужно много эпизодов.
  - Нестабильное обучение → большие колебания в производительности.

Value-Based (DQN):

- Буфер опыта и целевая сеть стабилизируют градиент → меньше дисперсия.

А теперь минусы REINFORCE (по сравнению с DQN)

2. Неэффективное использование данных (On-policy)

REINFORCE:

- Использует только данные текущей политики → старые траектории не переиспользуются.
- Требуется собирать много новых данных на каждом шаге.

Value-Based (DQN):

- Off-policy → переиспользует данные из буфера опыта многократно.
- Более эффективное и стабильное обучение.

A теперь минусы REINFORCE (по сравнению с DQN)

3. Применимость только к эпизодическим задачам

REINFORCE:

- Требует завершения эпизода для вычисления полного возврата G_t.
- Непригоден для непрерывных задач без естественного завершения.

Value-Based (DQN):

- Обновляет Q-функцию на основе одношаговых переходов.
- Может использоваться в непрерывных задачах.

Пример из реального мира:

- Робот, который должен ходить вечно
- Система охлаждения сервера 24/7
- Торговый агент на фондовом рынке

Компромиссы и применимость

Использовать REINFORCE / Policy-Based:

- Пространство действий непрерывное (роботы, физические симуляции)
- Оптимальная политика стохастическая
- Нужна теоретическая гарантия сходимости

Использовать Value-Based / DQN:

- Пространство действий дискретное и небольшое
- Требуется высокая эффективность данных
- Задача непрерывная (неэпизодическая)

Можно ли улучшить ситуацию?

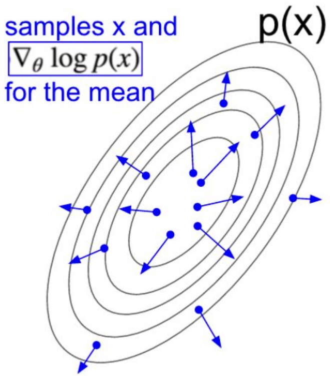
Переход от G_t к оценке G_t

Чтобы снизить дисперсию и учесть вклад конкретного действия:

1. **Вспомним о Q-функции**: $Q^{\pi}(s_t, a_t) = \mathbb{E}[G_t \mid s_t, a_t]$
2. **Градиент через** Q: REINFORCE можно переписать с использованием Q:
   - Было: $\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, G_t\right]$
   - Стало: $\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, Q^{\pi}(s_t, a_t)\right]$

Но вообще оценка всё равно так себе :(

Expected Grad-Log-Prob (EGLP) Lemma



Если $P_\theta(x)$ — это параметризованное распределение вероятностей над случайной величиной x, то:

$$\underset{x \sim P_\theta}{\mathbb{E}} [\nabla_\theta \log P_\theta(x)] = 0.$$

Почему это важно?

- Эта лемма показывает, что градиент логарифма вероятности, усредненный по всем возможным значениям x, равен нулю.
- Это свойство позволяет нам манипулировать выражениями для градиентов, не меняя их математического ожидания.

Докажем $\underset{x\sim P_\theta}{\mathbb{E}}[\nabla_\theta \log P_\theta(x)] = 0.$ (EGLP)

1. Начнем с того, что любое распределение вероятностей нормализовано:

$$\int_x P_\theta(x)dx = 1.$$

2. Выводим:

$$\mathbb{E}_{x\sim P_\theta}[\nabla_\theta \log P_\theta(x)] = \int_x P_\theta(x)\nabla_\theta \log P_\theta(x)dx =$$

$$\left[\text{log derivative trick:} \nabla_\theta P_\theta(x) = P_\theta(x)\nabla_\theta \log P_\theta(x)\right]$$

$$= \int_x \nabla_\theta P_\theta(x)dx = \nabla_\theta \int_x P_\theta(x)dx = \nabla_\theta 1 = 0$$

Expected Grad-Log-Prob (EGLP) Lemma

$$\underset{x\sim P_\theta}{\mathbb{E}}\left[\nabla_\theta \log P_\theta(x)\right] = 0.$$

Теперь можно “упростить” формулу

Из EGLP-леммы следует, что мы можем добавить или вычесть любую функцию $b(s_t)$, зависящую только от состояния, без изменения математического ожидания градиента:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\left(G_t - b(s_t)\right)\right].$$

Что такое базовая функция?

- Базовая функция $b(s_t)$ — это функция, которая помогает уменьшить дисперсию градиента.
- Наиболее распространенный выбор: $b(s_t) = V^\pi(s_t)$, где $V^\pi(s_t)$ — это значение состояния (ожидаемая награда, если агент начинает из состояния s_t).

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\left(Q(s_t,a_t) - V(s_t)\right)\right] = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t|s_t)\left(A(s_t,a_t)\right)\right]$$

Advantage

$$A(s,a) = Q(s,a) - V(s)$$

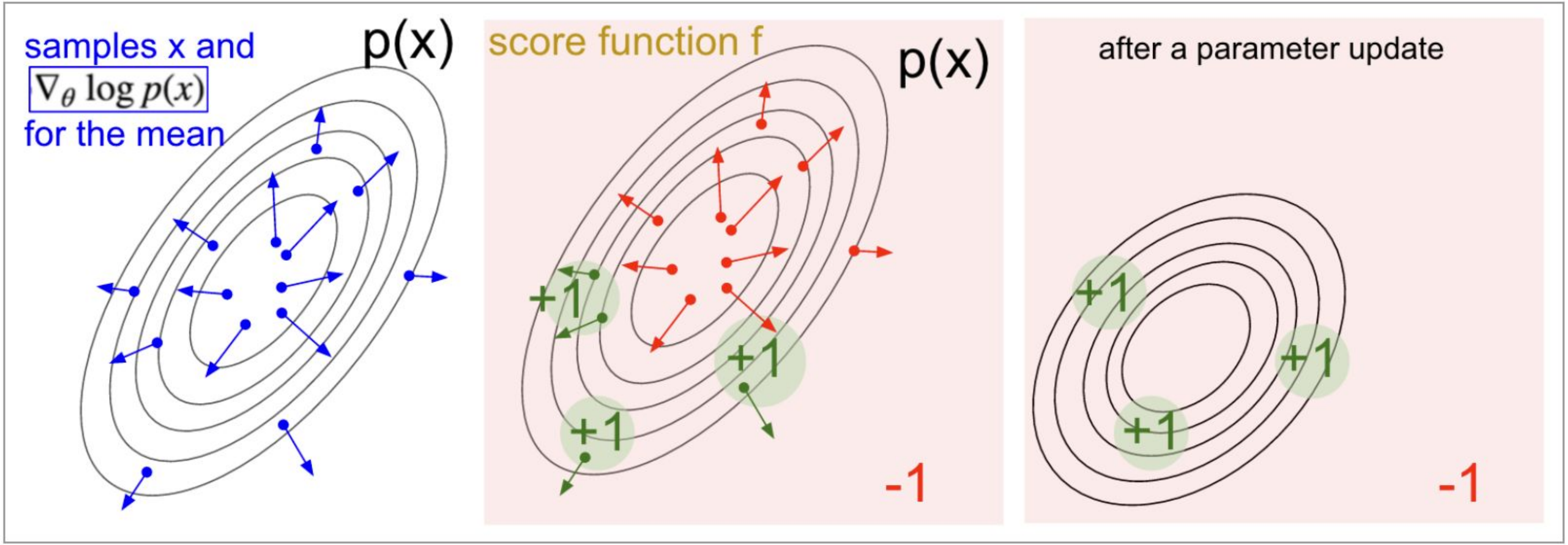
$Q(s_t, a_t)$: Ожидаемая награда за выполнение действия a_t в состоянии s_t.

$V(s_t)$: Ожидаемая награда за нахождение в состоянии s_t (независимо от действия).

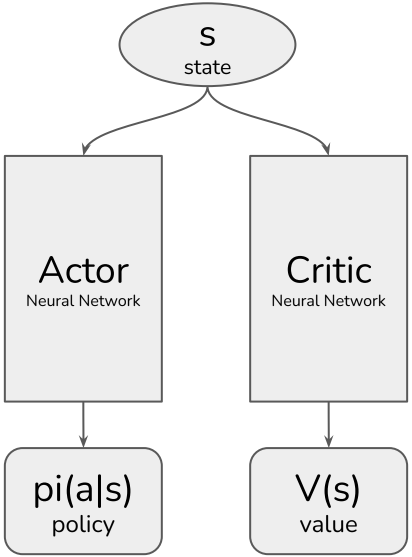
- **Интуиция**:
  - $Q(s_t, a_t)$ показывает, насколько хорошо конкретное действие a_t в состоянии s_t.
  - $V(s_t)$ показывает, насколько хорошее состояние s_t в среднем (для всех возможных действий).
  - Разница $A(s_t, a_t)$ говорит, насколько действие a_t лучше или хуже среднего.

https://karpathy.github.io/2016/05/31/rl/

Иллюстрация для policy gradient



Actor-Critic



$$A(s,a) = Q(s,a) - V(s)$$
$$Q(s,a) = r + \gamma V(s')$$
$$A(s,a) = r + \gamma V(s') - V(s)$$

Actor-Critic — это гибридный метод, который сочетает:

- **Actor**: Политика $\pi_\theta(a_t|s_t)$, которая обучается выбирать действия.
- **Critic**: Оценка состояния $V^\pi(s_t)$, которая помогает оценивать политику.

Как работает A2C?

1. **Actor**:
   - Обновляет политику, используя advantage function:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t,a_t)\right].$$

2. **Critic**:
   - Обучается минимизировать ошибку между предсказанными значениями состояний $V(s_t)$ и реальными наградами:

$$L = \mathbb{E}\left[(G_t - V(s_t))^2\right].$$

3. **Объединение**:
   - Critic предоставляет более точную оценку advantage, что делает обновления Actor более стабильными и менее шумными.

Итоги

➔ Попробовали учить политику напрямую

➔ Для этого вывели REINFORCE

$$\nabla_\theta J(\theta) = \mathbb{E}\Big[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, G_t\Big]$$

➔ А потом улучшили его и получили Actor-Critic!

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\,(Q(s_t,a_t) - V(s_t))\right] = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\,(A(s_t,a_t))\right]$$

Вопросы?