

# Лекция 2: Методы Монте-Карло, временных различий, бандиты

Панов А.И.  
[panov.ai@mipt.ru](mailto:panov.ai@mipt.ru)

Центр когнитивного моделирования МФТИ  
Инситут проблем искусственного интеллекта ФИЦ ИУ РАН

2020 – Машинное обучение с подкреплением  
Курс для Сбербанка



# План лекции

- 1 Методы Монте-Карло
- 2 Методы временных различий
- 3 Безмодельное управление
- 4 Многорукие бандиты
- 5 Оптимизм в условиях неопределенности
- 6 Контекстные бандиты

# Монте-Карло подход к обучению с подкреплением

- Методы Монте-Карло (Monte-Carlo, MK) работают напрямую по эпизодам взаимодействия со средой
- MK – безмодельный подход (model-free): модель переходов МППР и функция вознаграждения не известны
- MK обучается по полным эпизодам (нет бутстрэп-выборок)
- MK использует максимально простую идею: полезность равна средней отдаче
- MK может быть применен только для эпизодических МППР

# Монте-Карло оценка стратегии

- Цель: построить  $V^\pi$  по эпизодам взаимодействия по стратегии  $\pi$ :

$$s_1, a_1, r_1, \dots s_k \sim \pi$$

- Отдача (return) – это суммарное вознаграждение:

$$R_t = r_{t+1} + \gamma r_{t+2} + \dots + \gamma^{T-1} r_T$$

- Функция полезности – это матожидание отдачи:

$$V^\pi(s) = \mathbb{E}_\pi[R_t | s_t = s]$$

- Монте-Карло оценка стратегии используется *эмпирическое среднее отдачи* вместо матожидания

# Монте-Карло оценка стратегии с первым посещением

Оцениваем состояние  $s$ :

- для **первого** по времени посещения состояния  $s$  в эпизоде:
  - $N(s) \leftarrow N(s) + 1$ ,
  - $S(s) \leftarrow S(s) + R_t$

Полезность оцениваем как среднюю отдачу:  $V(s) = S(s)/N(s)$

По закону больших чисел:

$$V(s) \xrightarrow{N(s) \rightarrow \infty} V^\pi(s)$$

Несмешенная, состоятельная оценка с высокой дисперсией

# Монте-Карло оценка стратегии с каждым посещением

Оцениваем состояние  $s$ :

- для каждого по времени посещения состояния  $s$  в эпизоде:
- $N(s) \leftarrow N(s) + 1$ ,
- $S(s) \leftarrow S(s) + R_t$

Полезность оцениваем как среднюю отдачу:  $V(s) = S(s)/N(s)$

По закону больших чисел:

$$V(s) \xrightarrow{N(s) \rightarrow \infty} V^\pi(s)$$

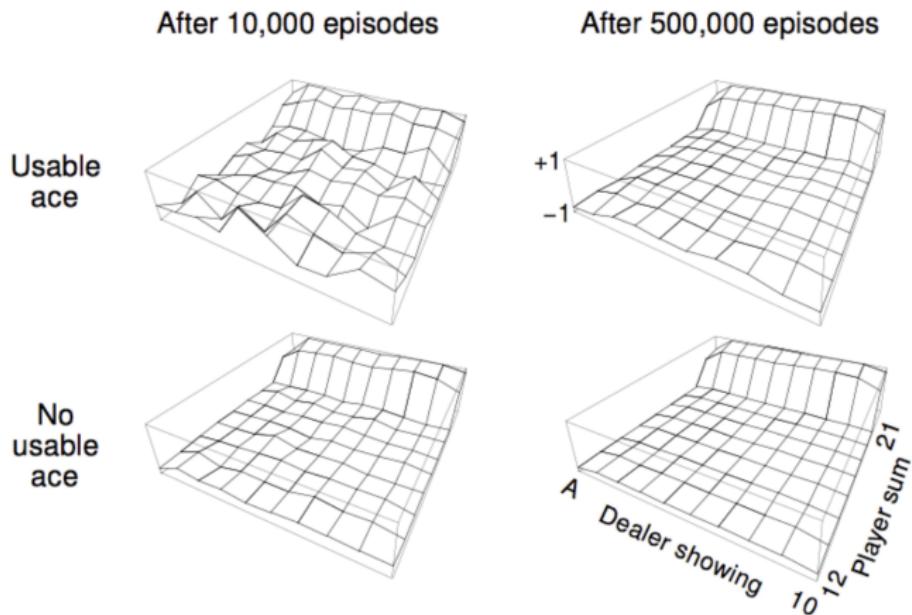
Смешенная, состоятельная оценка с более низкой дисперсией.

## Пример: блек-джек

- 200 состояний:
  - ▶ текущая сумма (12-21),
  - ▶ дилер показывает карту (максимально – 10 очков),
  - ▶ есть ли особая комбинация (да-нет)
- Действие stick: не получать карты (завершить игру)
- Действие twist: взять новую карту (без замены)
- Вознаграждения за stick:
  - ▶ +1, если сумма карт больше, чем у дилера,
  - ▶ 0, если сумма карт та же, чем у дилера,
  - ▶ -1, если сумма карт меньше, чем у дилера
- Вознаграждения за twist:
  - ▶ -1, если сумма карт больше 21,
  - ▶ 0 иначе
- Переходы: автоматически twist, если сумма карт меньше 12



# Функция полезности после обучение MC



Стратегия: stick, если сумма больше 20, иначе – twist

# Среднее приращений

Средние последовательности могут быть вычислены последовательно (incremental mean):

$$\begin{aligned}
 \mu_k &= \frac{1}{k} \sum_{j=1}^k x_j = \\
 &= \frac{1}{k} (x_k + \sum_{j=1}^{k-1} x_j) = \\
 &= \frac{1}{k} (x_k + (k-1)\mu_{k-1}) = \\
 &= \mu_{k-1} + \frac{1}{k} (x_k - \mu_{k-1})
 \end{aligned}$$

# Монте-Карло для приращений

- Обновим  $V(s)$  с приращением (incrementally) для эпизода  $s_1, a_1, r_1, \dots, s_T$
- Для каждого состояния  $s_t$  с отдачей  $R_t$ :

$$N(s_t) \leftarrow N(s_t) + 1,$$

$$V(s_t) \leftarrow V(s_t) + \frac{1}{N(s_t)}(R_t - V(s_t))$$

- В нестационарных задачах может быть полезно отслеживать текущее среднее:

$$V(s_t) \leftarrow V(s_t) + \alpha(R_t - V(s_t))$$

# Обучение на основе временных различий

- Методы временных различий (temporal-difference, ВР) обучаются напрямую по эпизодам взаимодействия со средой
- ВР – безмодельный подход (model-free): модель переходов МППР и функция вознаграждения не известны
- ВР обучается по неполным эпизодам, используя бутстрэп (bootstrapping, раскрутка)
- ВР приближает значение на основе предыдущего приближения

# Монте-Карло и временные различия

- Цель: обновлять  $V^\pi$  в реальном времени (online) по эпизодам взаимодействия по стратегии  $\pi$
- Монте-Карло с каждым посещением для приращений: обновляем  $V(s_t)$  на основе текущей отдачи  $R_t$

$$V(s_t) \leftarrow V(s_t) + \alpha(R_t - V(s_t)).$$

- Самый простой подход временных различий: TD(0):
  - обновляем на основе *ожидаемой* отдачи  $r_{t+1} + \gamma V(s_{t+1})$

$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)),$$

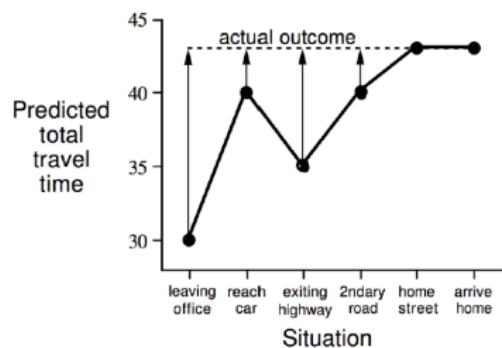
- $r_t + \gamma V(s_{t+1})$  называется *TD показателем*,
- $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$  называется *TD ошибкой*

## Пример: поездка домой

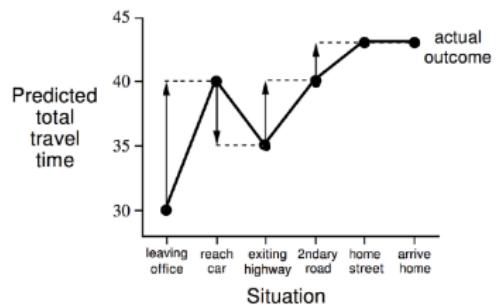
| Состояние             | Истекшее время (мин) | Ожидаемое оставшееся время (мин) | Предсказываемое общее время (мин) |
|-----------------------|----------------------|----------------------------------|-----------------------------------|
| Выйти из офиса        | 0                    | 30                               | 30                                |
| Начать поездку, дождь | 5                    | 35                               | 40                                |
| Съехать с шоссе       | 20                   | 15                               | 35                                |
| Ехать за грузовиком   | 30                   | 10                               | 40                                |
| Домашняя улица        | 40                   | 3                                | 43                                |
| Зайти домой           | 43                   | 0                                | 43                                |

# Пример: Монте-Карло и временные различия

Обновления по методу  
Монте-Карло ( $\alpha = 1$ ):



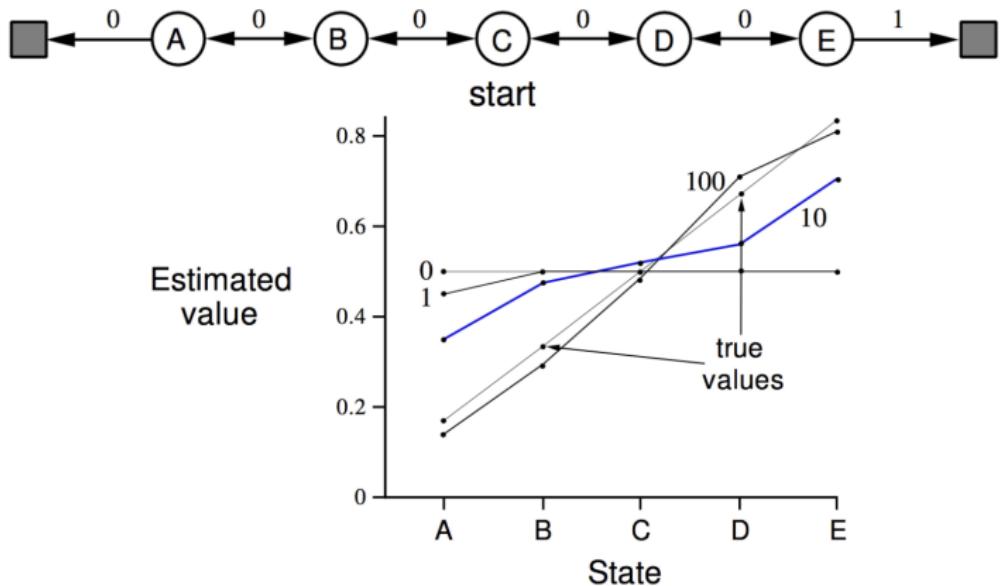
Обновления по методу временных  
различий ( $\alpha = 1$ ):



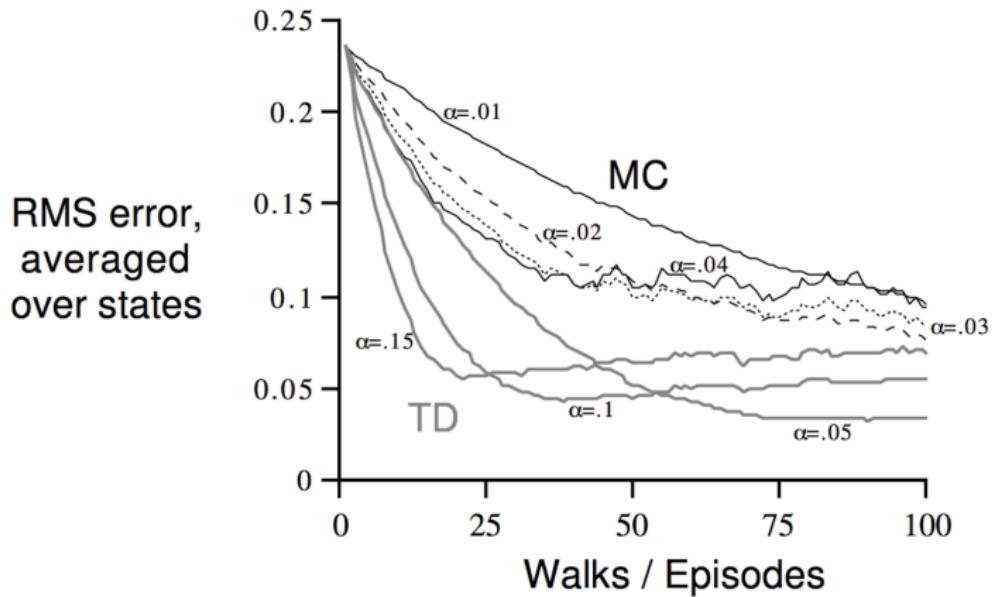
# Преимущества и недостатки МК и ВР

- ВР может обучаться до того, как стала известна итоговая отдача:
  - ▶ ВР может обучаться интерактивно на каждом шаге,
  - ▶ МК должен дожидаться окончания эпизода, когда становится известной отдача.
- ВР может обучаться без информации об итоговой отдаче:
  - ▶ ВР может обучаться на неполных эпизодах,
  - ▶ МК может обучаться только на полных эпизодах,
  - ▶ ВР работает и для бесконечных (без терминального состояния) окружений
  - ▶ МК работает только в эпизодических окружениях (с терминальными состояниями)

## Пример: случайные блуждания



# Пример: MK vs. BP



# Пакетные МК и ВР

- МС и TD сходятся к  $V(s) \rightarrow V^\pi(s)$ , если опыт  $\rightarrow \infty$
- А если мы применим пакетный (batch) подход для конечного опыта?

$$s_1^1, a_1^1, r_1^1, \dots, s_{T_1}^1$$

⋮

$$s_1^K, a_1^K, r_1^K, \dots, s_{T_K}^K$$

- Например, многократно выбирать эпизод  $k \in [1, K]$
- Применять МС или TD(0) к эпизоду  $k$

## Пример: АВ

Два состояния А, В; нет дисконтирования, 8 эпизодов опыта:

А, 0, В, 0

В, 1

В, 1

В, 1

В, 1

В, 1

В, 1

В, 0

Какие значения  $V(A)$  и  $V(B)$ ?

## Пример: АВ

Два состояния А, В; нет дисконтирования, 8 эпизодов опыта:

А, 0, В, 0

В, 1

В, 1

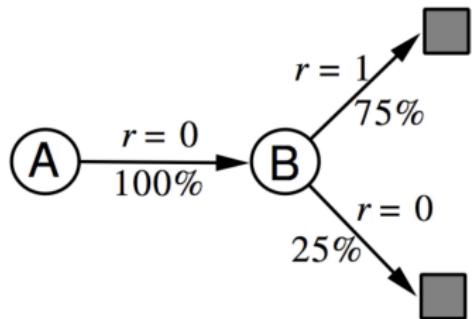
В, 1

В, 1

В, 1

В, 1

В, 0



Какие значения  $V(A)$  и  $V(B)$ ?

## Эквивалентность

- МК сходится к решению с минимальной среднеквадратичной ошибкой
  - ▶ Наилучшее приближение к наблюдаемой отдаче:

$$\sum_{k=1}^K \sum_{t=1}^{T_k} (R_t^k - V(s_t^k))^2$$

- ▶ Для АВ примера  $V(A) = 0$
- TD(0) сходится к решению максимально правдоподобной марковской модели
  - ▶ Решение для МППР  $\langle S, A, \hat{\mathcal{P}}, \hat{\mathcal{R}}, \gamma \rangle$ , который лучше всего удовлетворяет данным

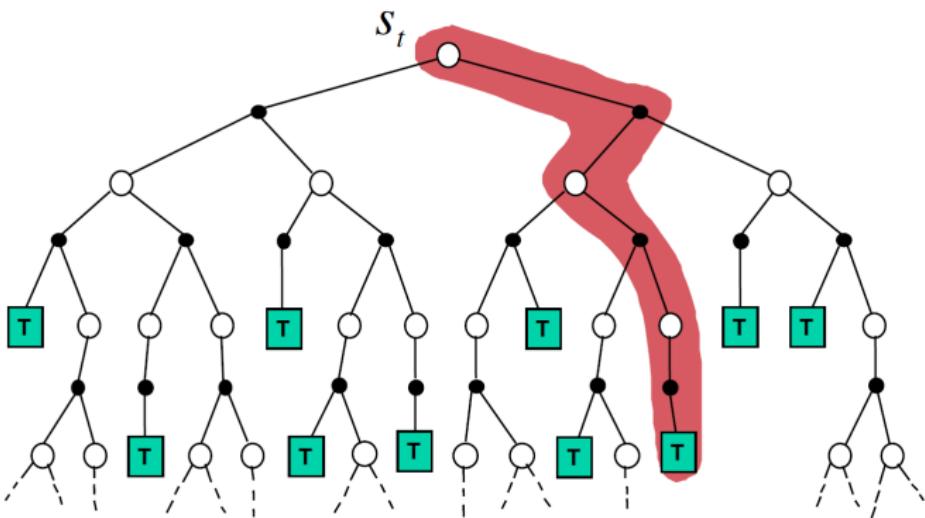
$$\hat{\mathcal{P}}_{ss'}^a = \frac{1}{N(s, a)} \sum_{k=1}^K \sum_{t=1}^{T_k} \mathbf{1}(s_t^k, a_t^k, s_{t+1}^k = s, a, s'),$$

$$\hat{\mathcal{R}}_s^a = \frac{1}{N(s, a)} \sum_{k=1}^K \sum_{t=1}^{T_k} \mathbf{1}(s_t^k, a_t^k = s, a) r_t^k$$

- ▶ Для АВ примера  $V(A) = 0.75$

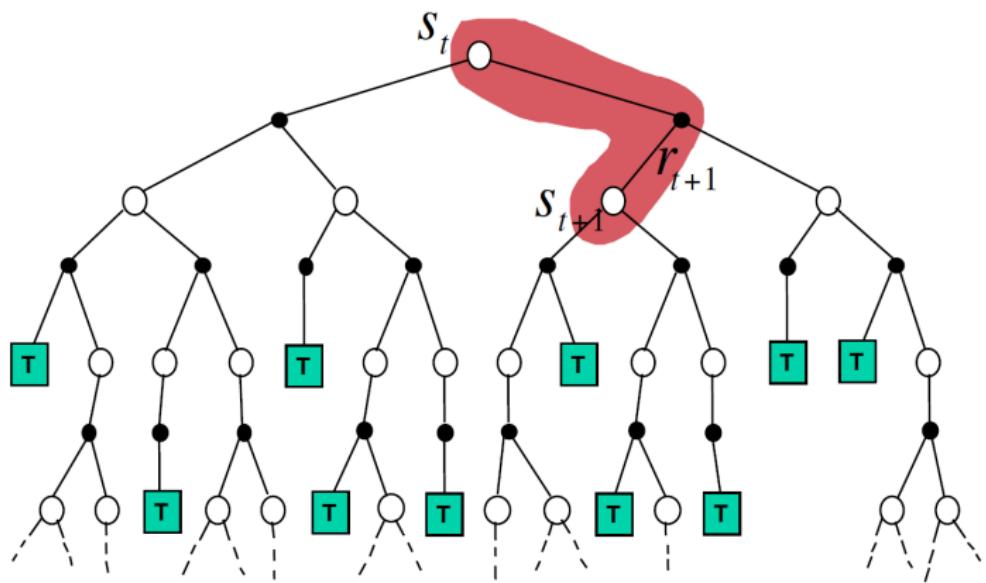
# Монте-Карло обновление

$$V(s_t) \leftarrow V(s_t) + \alpha(R_t - V(s_t))$$



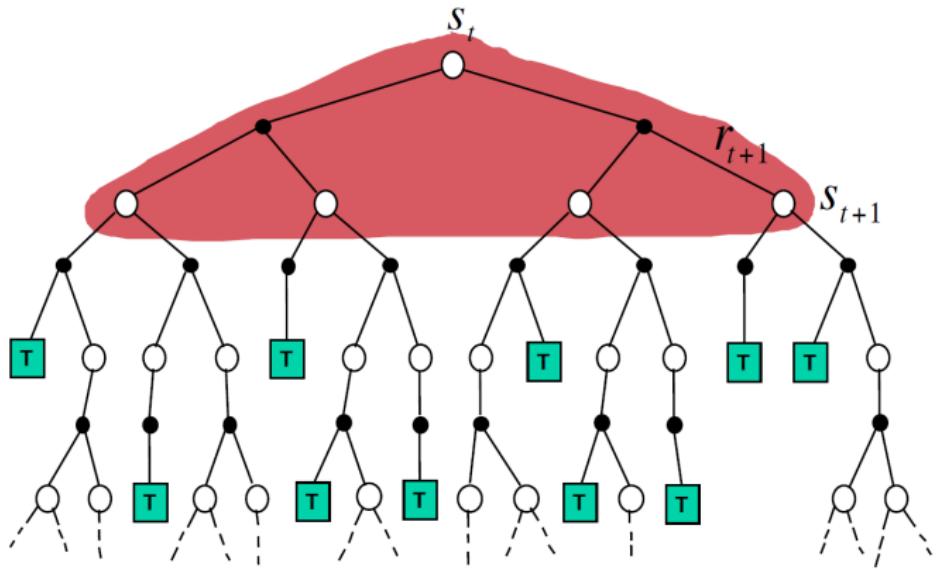
# Обновление временных различий

$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$

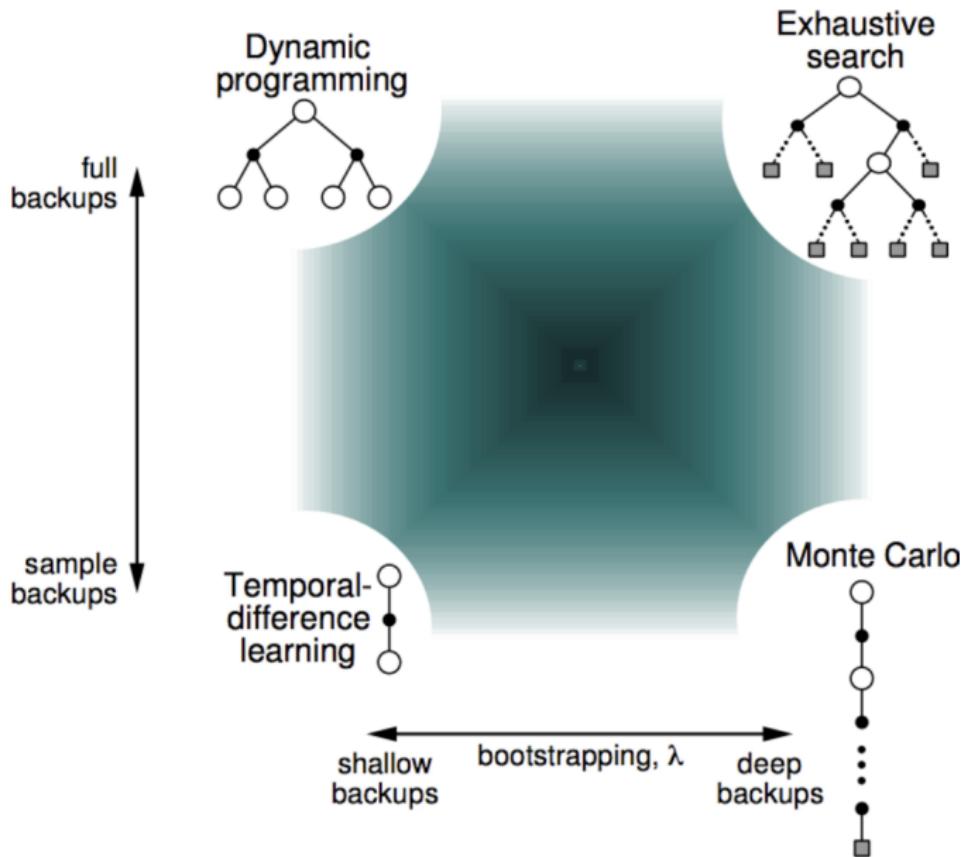


# Обновление динамического программирования

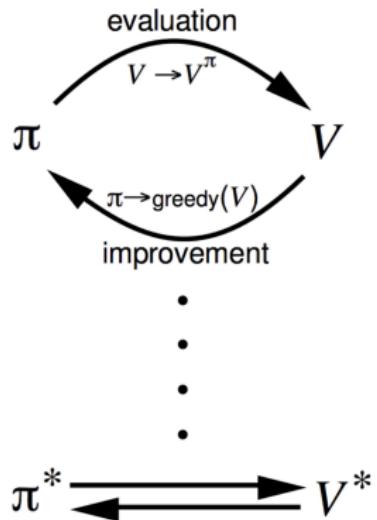
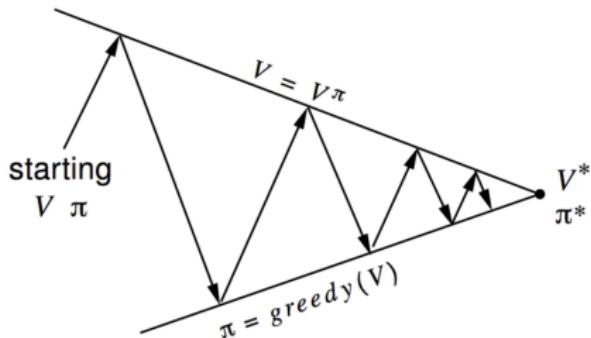
$$V(s_t) \leftarrow \mathbb{E}_\pi[r_{t+1} + \gamma V(s_{t+1})]$$



# Обобщенный подход к обучению с подкреплением



# Обобщенные итерации по стратегиям



**Оценка стратегии** – вычисление  $V^\pi$

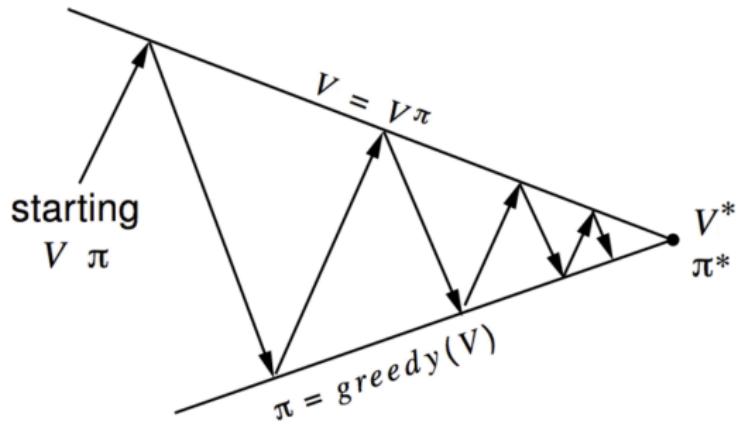
Итеративная оценка стратегии

**Улучшение стратегии** – генерация

$\pi' \geq \pi$

Жадное обновление стратегии

# Обобщенные итерации по стратегиям с Монте-Карло оценкой



## Оценка стратегии

Монте-Карло оценка стратегии по  $V = V^\pi$ ?

## Улучшение стратегии

Жадное обновление стратегии?

# Безмодельная итерация по стратегиям с функцией полезности действий

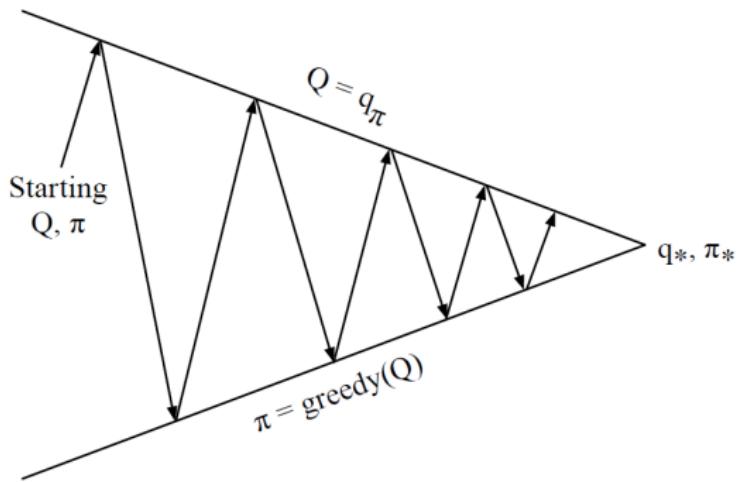
- Жадное улучшение стратегии по  $V(s)$  требует знания модели МППР:

$$\pi'(s) = \arg \max_{a \in A} (\mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s'))$$

- Жадное обновление стратегии по  $Q(s, a)$  не требует знания модели:

$$\pi'(s) = \arg \max_{a \in A} Q(s, a)$$

# Обобщенные итерации по стратегиям с функцией полезности действий



## Оценка стратегии

Монте-Карло оценка  $Q = Q^\pi$

## Улучшение стратегии

Жадное обновление стратегии?

# Пример жадного выбора действий



"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

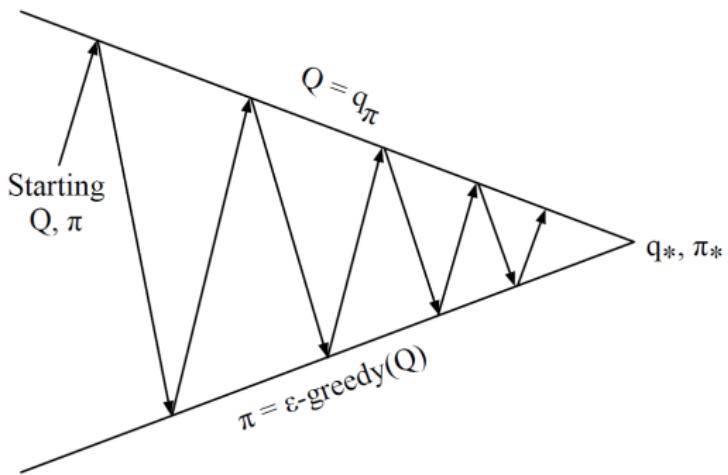
- Перед вами две двери
- Вы открываете левую дверь и получаете вознаграждение 0  $V(left) = 0$
- Вы открываете правую дверь и получаете вознаграждение +1  $V(right) = 1$
- Вы открываете правую дверь и получаете вознаграждение +2  $V(right) = 2$
- Вы открываете правую дверь и получаете вознаграждение +2  $V(right) = 2$
- :
- Вы уверены, что выбирали лучшую дверь?

## $\epsilon$ -жадное исследование

- Простейшая идея, обеспечивающая постоянное исследование среды
- Все действия выбираются с ненулевой вероятностью
- С вероятностью  $\epsilon - 1$  выбираем действие жадно
- С вероятностью  $\epsilon$  выбираем действие случайно

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon, & \text{если } a^* = \arg \max_{a \in A} Q(s, a), \\ \epsilon/m, & \text{иначе} \end{cases}$$

# Монте-Карло итерация итерация по стратегиям



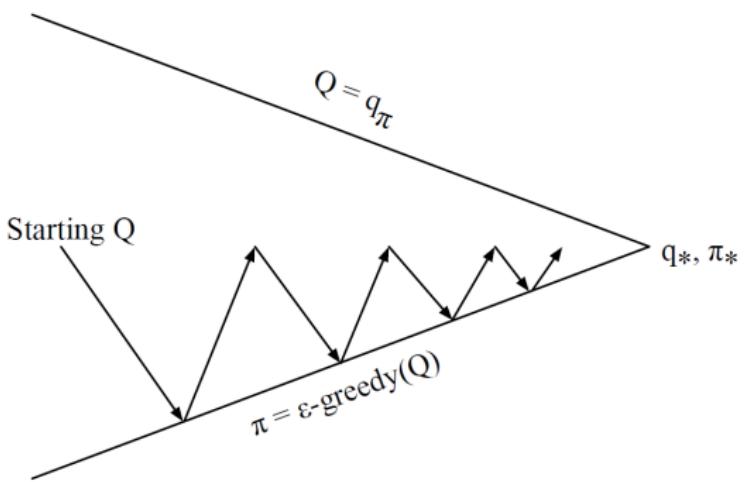
## Оценка стратегии

Монте-Карло оценка стратегии  $Q = Q^\pi$

## Улучшение стратегии

$\epsilon$ -жадное улучшение стратегии

# Монте-Карло управление



Для каждого эпизода:

Оценка стратегии

Монте-Карло оценка стратегии  $Q \approx Q^\pi$ .

Улучшение стратегии

$\epsilon$ -жадное обновление стратегии

# Алгоритм Монте-Карло управления

- Выбираем  $k$ -ый эпизод  $(s_1, a_1, r_2, \dots, s_T) \sim \pi$
- Для каждой пары состояния и действия в эпизоде выполняем

$$N(s_t, a_t) \leftarrow N(s_t, a_t) + 1,$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{1}{N(s_t, a_t)}(R_t - Q(s_t, a_t))$$

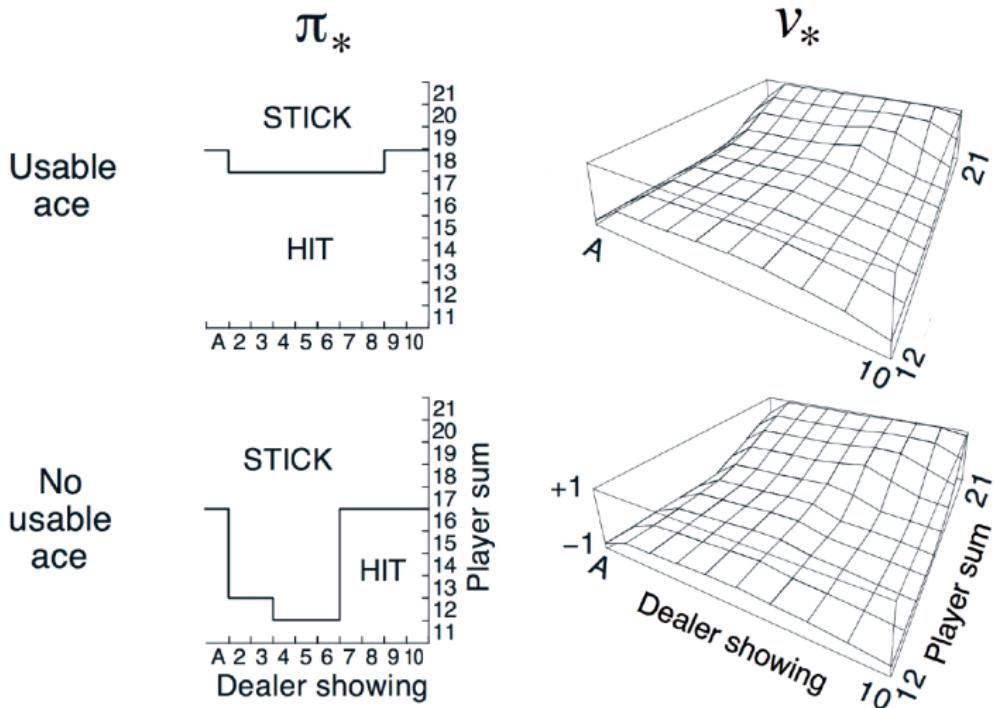
- Улучшаем стратегию на основе новых значений полезности:

$$\epsilon \leftarrow 1/k,$$

$$\pi \leftarrow \epsilon\text{-жадная}(Q)$$

- Этот алгоритм сходится к оптимальному решению  
 $Q(s, a) \rightarrow Q^*(s, a)$

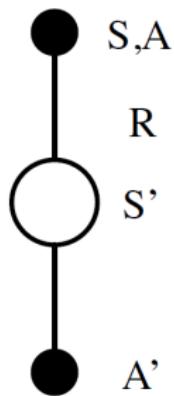
# Пример: МК и блек-джек



# МК vs. ВР управление

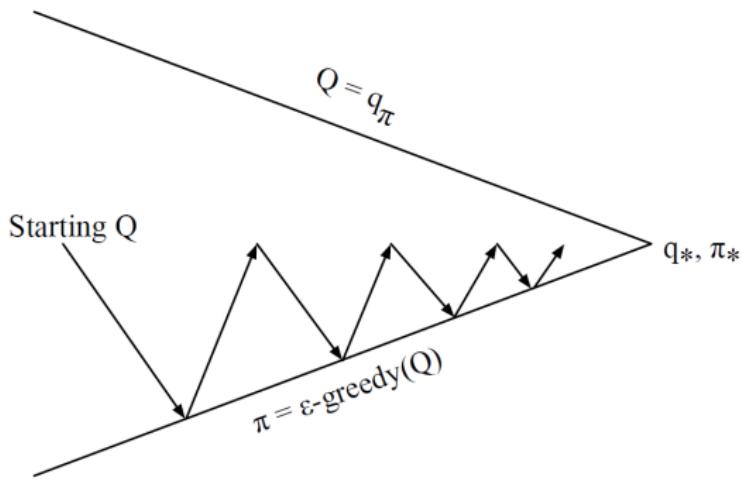
- ВР имеет несколько преимуществ относительно МК:
  - ▶ меньше дисперсия,
  - ▶ интерактивное,
  - ▶ неполные последовательности
- Естественная идея: использовать ВР вместо МК в цикле управления:
  - ▶ применить TD к  $Q(s, a)$ ,
  - ▶ использовать  $\epsilon$ -жадное улучшение,
  - ▶ обновлять на каждом шаге

# Обновление функции полезности по SARSA



$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma Q(s', a') - Q(s, a))$$

# Управление по собственному опыту с SARSA



**Для каждого временного шага:**

**Оценка стратегии**

SARSA  $Q \approx Q^\pi$

**Улучшение стратегии**

$\epsilon$ -жадное обновление стратегии

# SARSA алгоритма для управления

---

## Algorithm 1 SARSA с $\epsilon$ -жадной стратегией

---

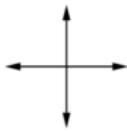
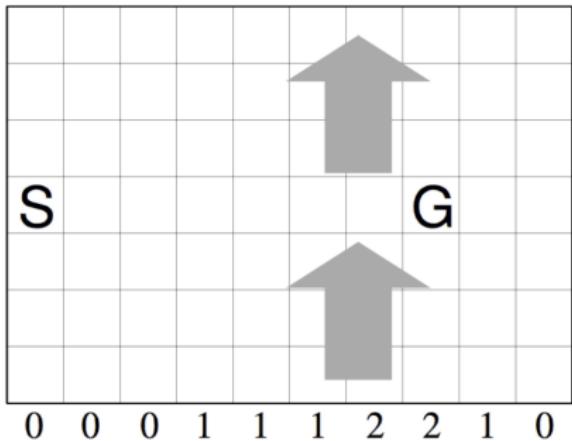
```

1:  $MDP = \langle S, A, \gamma \rangle$                                 ▷ задача МППР
2:  $Q \leftarrow [0, \dots]$ ;                                     ▷ таблица значений полезности действий
3: for all эпизодов do
4:    $s$  - начальное состояние;
5:    $a \leftarrow \epsilon\text{-жадная } \pi(s)$ 
6:   for all шагов эпизода do
7:      $r, s' \leftarrow$  применение  $a$ ;
8:      $a' \leftarrow \epsilon\text{-жадная } \pi(s');$ 
9:      $Q[s, a] \leftarrow Q(s, a) + \alpha(r + \gamma Q[s', a'] - Q[s, a]);$ 
10:     $s \leftarrow s', a \leftarrow a';$ 

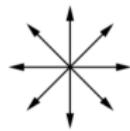
```

---

## Пример: ветреный клеточный мир



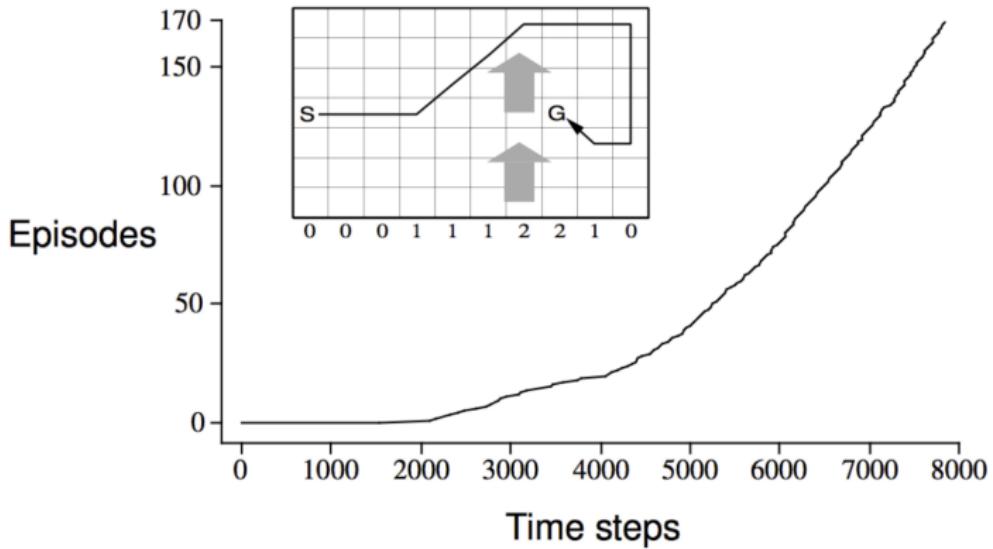
standard  
moves



king's  
moves

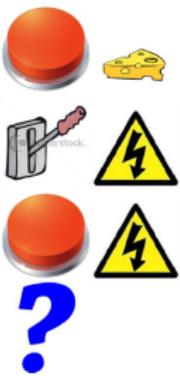
Вознаграждение -1 за каждый шаг.  
Без дисконтирования.

# SARSA в ветреном клеточном мире



# Проблема исследования и использования

- Фундаментальная проблема в теории последовательного принятия решений
  - выбор между:
    - ▶ **исследованием** среды (eXploration) – сбор дополнительной информации,
    - ▶ **использование** полученных знаний (eXploitation) – применение наилучшего действия в текущей ситуации
- Наилучшая долгосрочная стратегия может приводить к краткосрочным потерям
- Сбор дополнительной информации для принятия наилучших решений в дальнейшем

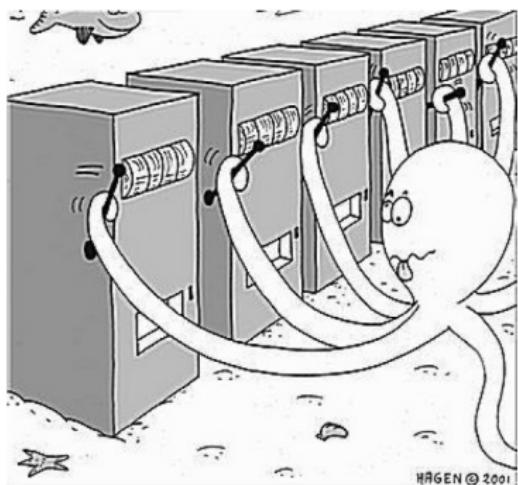


# Способы решения XX дилеммы

- **Наивный подход** – добавление шума к жадной стратегии ( $\epsilon$ -жадная стратегия)
- **Оптимистичная инициализация** – предполагаем лучшее, пока не убедимся в обратном
- **Оптимизм в условиях неопределенности** (Optimism in the Face of Uncertainty) – предпочитаем действия с неопределенной полезностью
- **Вероятностное соответствие** (Probability Matching) – выбор действий в соответствии с оцениваемым распределением
- **Поиск в информационном пространстве** – поиск с учетом полезности информации

# Многорукий бандит

- Многорукий бандит (multi-armed bandits) – это двойка  $\langle A, \mathcal{R} \rangle$ , где
  - ▶  $A$  – множество из  $m$  действий (ручек игорного автомата),
  - ▶  $\mathcal{R}^a = \mathbb{P}[r|a]$  – неизвестное распределение по вознаграждениям
- В каждый момент времени агент выбирает действие  $a_t \in A$
- Окружение (автомат) генерирует вознаграждение  $r_t \sim \mathcal{R}^{a_t}$
- Цель агента – максимизировать суммарную отдачу по всем эпизодам
$$\sum_t r_t$$



# Потери

- Полезность действия – среднее вознаграждение для действия  $a$ :

$$Q(a) = \mathbb{E}[r|a]$$

- Оптимальная полезность  $V^*$ :

$$V^* = Q(a^*) = \max_{a \in A} Q(a)$$

- Потери (regret) – возможность упустить вознаграждение на одном шаге:

$$l_t = \mathbb{E}[V^* - Q(a_t)]$$

- Полные потери – полная упущенная выгода:

$$L_t = \mathbb{E} \left[ \sum_t V^* - Q(a_t) \right]$$

- Максимизация накопленного вознаграждения = минимизация полных потерь

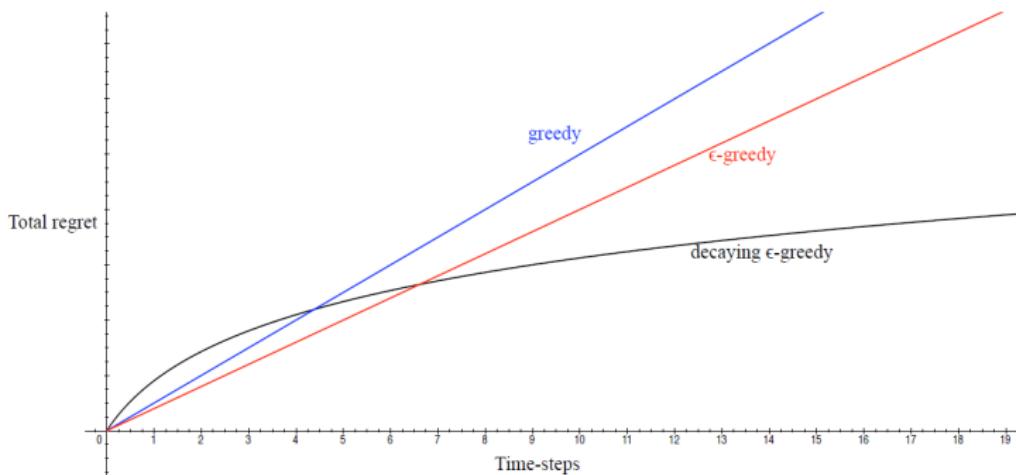
## Подсчет потерь

- Пусть  $N_t(a)$  – ожидаемое количество случаев, когда было выбрано действие  $a$
- Расхождение (gap)  $\Delta_a$  – разница в полезностях между выбранным действием и оптимальным действием  $\Delta_a = V^* - Q(a)$
- Потери зависят от расхождения и частоты действия

$$\begin{aligned} \mathbb{L}_t &= \mathbb{E} \left[ \sum_t V^* - Q(a_t) \right] = \\ &= \sum_{a \in A} \mathbb{E}[N_t(a)](V^* - Q(a)) = \\ &= \sum_{a \in A} \mathbb{E}[N_t(a)]\Delta_a \end{aligned}$$

- Хороший алгоритм должен обеспечить небольшое количество больших расхождений
- Проблема: расхождения не известны!

# Линейные и сублинейные потери



- Если стратегия всегда исследует среду – полные потери будут линейно расти со временем
- Если стратегия никогда не исследует среду – полные потери будут линейно расти со временем
- Возможна ли стратегия с сублинейным ростом полных потерь?

# Жадный алгоритм

- Пусть наш алгоритм оценивает полезность  $\hat{Q}_t(a) \approx Q(a)$
- Например, оценка полезности по Монте-Карло:

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_t r_t 1(a_t = a)$$

- Жадный алгоритм выбирает действие с максимальной полезностью:

$$a_t^* = \arg \max_{a \in A} \hat{Q}_t(a)$$

- Жадность может привести к постоянному выбору субоптимального действия
- $\Rightarrow$  у жадного алгоритма линейно растут потери

## ε-жадный алгоритм

- ε-жадный алгоритм продолжает постоянно исследовать среду:
  - ▶ с вероятностью  $1 - \epsilon$  выбирается  $a = \arg \max_{a \in A} \hat{Q}(a)$ ,
  - ▶ с вероятностью  $\epsilon$  – случайное действие
- Константа  $\epsilon$  только обеспечивает уменьшение потерь:

$$l_t \geq \frac{\epsilon}{|A|} \sum_{a \in A} \Delta_a$$

- ⇒ у ε-жадного алгоритма линейно растут потери

# Оптимистичная инициализация

- Простая идея – инициализируем  $Q(a)$  наивысшими значениями
- Будем обновлять полезность действия итерационной оценкой Монте-Карло, начиная с  $N(a) > 0$ :

$$\hat{Q}_t(a) = \hat{Q}_{t-1} + \frac{1}{N_t(a_t)}(r_t - \hat{Q}_{t-1})$$

- Это поощряет систематическое исследование на ранних стадиях
- Однако также возможно застревание на субоптимальном действии
- $\Rightarrow$  у жадного алгоритма с оптимистичной инициализацией линейно растут потери
- $\Rightarrow$  у  $\epsilon$ -жадного алгоритма с оптимистичной инициализацией линейно растут потери

## Затухающий $\epsilon_t$ -жадный алгоритм

- Выберем некоторое расписания для изменения параметра  $\epsilon$
- Например, такое:

$$c > 0$$

$$d = \min_{a|\Delta_a > 0} \Delta_i$$

$$\epsilon_t = \min \left\{ 1, \frac{c|A|}{d^2 t} \right\}$$

- У затухающего  $\epsilon_t$ -жадного алгоритма потери растут **логарифмически!**
- Однако, в данном расписании мы использовали априорные знания о расхождениях
- Цель: найти сублинейный по потерям алгоритм для любого многорукого бандита (без знания функции вознаграждения  $\mathcal{R}$ )

## Нижняя граница

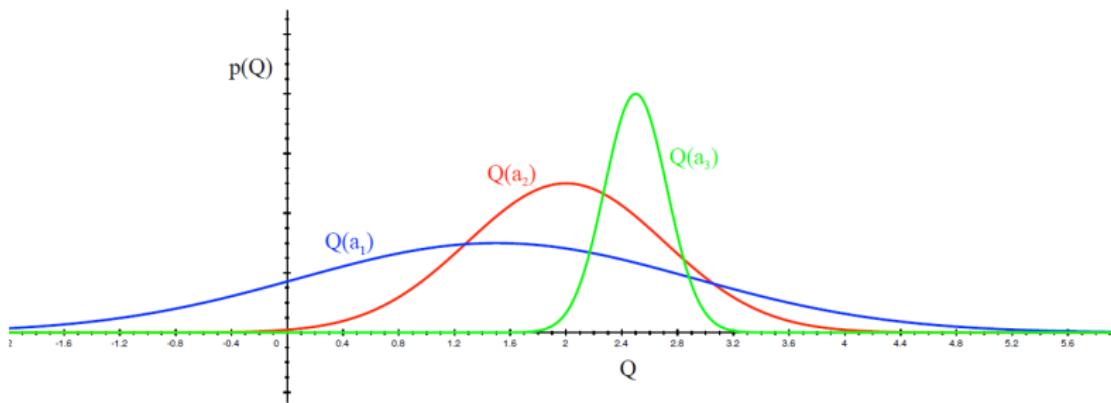
- Качество работы алгоритма определяется тем, насколько похожи ручки автомата
- Самый трудный случай, когда у похожих ручек разные средние значения вознаграждения
- Формально это описывает расхождением  $\Delta_a$  и похожестью распределений  $KL(\mathcal{R}^a \parallel \mathcal{R}^{a^*})$

### Theorem (Лай и Роббинса)

Асимптотически полные потери как минимум логарифмичны по количеству шагов:

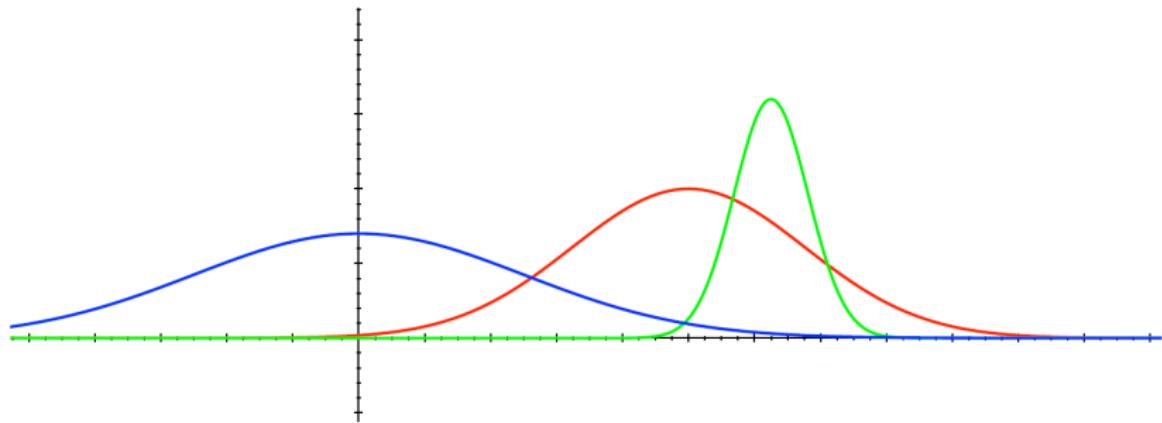
$$\lim_{t \rightarrow \infty} \geq \log t \sum_{a | \delta_a > 0} \frac{\Delta_1}{KL(\mathcal{R}^a \parallel \mathcal{R}^{a^*})}$$

# Оптимизм в условиях неопределенности



- Какое действие выбрать?
- Чем более мы не уверены в полезности действия, тем более важным является исследование его результатов
- Оно может оказаться лучшим действием

# Оптимизм в условиях неопределенности



- После выбора **синего** действия
- Наша определенность в его полезности увеличилась
- Теперь мы выберем другое действие с большей вероятностью

## Верхняя доверительная граница

- Оценим верхнюю границу  $\hat{U}_t(a)$  для полезности каждого действия
- Необходимо, чтобы  $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$  выполнялось с высокой вероятностью
- Это зависит от того сколько раз было выбрано действие
  - ▶ чем меньше  $N_t(a)$ , тем выше граница  $\hat{U}_t(a)$  (оценка полезности неопределена)
  - ▶ чем больше  $N_t(a)$ , тем ниже граница  $\hat{U}_t(a)$  (оценка полезности точна)
- Действие выбираем по максимуму верхней доверительной границы (Upper Confidence Bound, UCB):

$$a_t = \arg \max_{a \in A} \hat{Q}_t(a) + \hat{U}_t(a)$$

# Неравенство Хёфдинга

## Theorem (Неравенство Хёфдинга)

Пусть  $x_1, \dots, x_t$  – независимые одинаково распределенные случайные величины в интервале и  $\bar{x}_t = \frac{1}{t} \sum_{\tau=1}^t x_\tau$  – выборочное среднее, тогда

$$\mathbb{P}[\mathbb{E}[X] > \bar{x}_t + u] \geq e^{-2tu^2}$$

- Применим это неравенство к задаче многорукого бандита:

$$\mathbb{P}[Q(a) > \hat{Q}_t(a) + U_t(a)] \geq e^{-2N_t(a)U_t(a)^2}$$

# Вычисление верхней доверительной границы

- Выберем некоторую вероятность  $p$ , с которой истинное значение превышает
- Найдем значение границы  $U_t(a)$ :

$$e^{-2N_t(a)U_t(a)^2} = p$$

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

- Вероятность должна уменьшаться с наблюдением все новых вознаграждений, например  $p = t^{-4}$ , чтобы убедиться, что в асимптотике мы выберем оптимальное действие

$$U_t(a) = \sqrt{\frac{2 \log t}{N_t(a)}}$$

# UCB1

Алгоритм UCB1 выбора действий:

$$a_t = \arg \max_{a \in A} Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

## Theorem

UCB алгоритм достигает логарифмических асимптотических полных потерь:

$$\lim_{t \rightarrow \infty} L_t \leq 8 \log t \sum_{a | \delta_a > 0} \Delta_a$$

# Пример: 10-рукий бандит

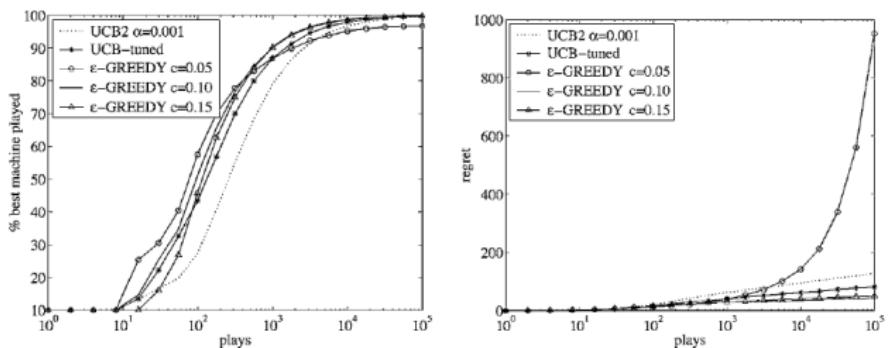


Figure 9. Comparison on distribution 11 (10 machines with parameters 0.9, 0.6, . . . , 0.6).

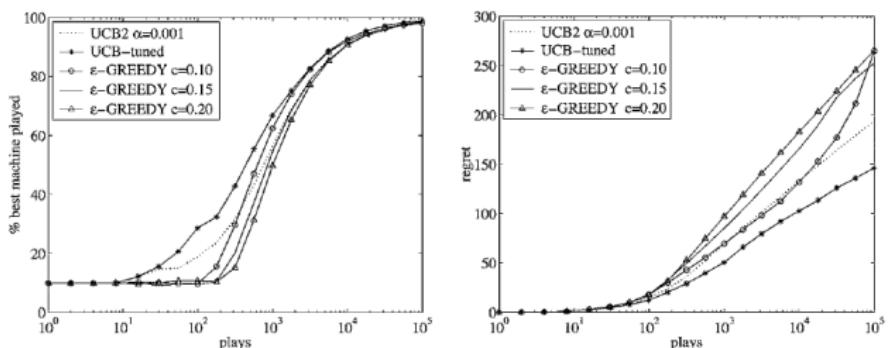


Figure 10. Comparison on distribution 12 (10 machines with parameters 0.9, 0.8, 0.8, 0.8, 0.7, 0.7, 0.7, 0.7, 0.6, 0.6, 0.6).

# Контекстный бандит

- Контекстный бандит (contextual bandit) – это тройка  $\langle A, S, \mathcal{R} \rangle$ , где
    - ▶  $A$  – множество из  $t$  действий (ручек игрового автомата),
    - ▶  $S = \mathbb{P}[s]$  – неизвестное распределение на множестве состояний (контекстов),
    - ▶  $\mathcal{R}_s^a = \mathbb{P}[r|s, a]$  – неизвестное распределение по вознаграждениям
  - В каждый момент времени среда генерирует состояние  $s_t \sim S$
  - Агент выбирает действие  $a_t \in A$
  - Окружение (автомат) генерирует вознаграждение  $r_t \sim \mathcal{R}_{s_t}^{a_t}$
  - Цель агента – максимизировать суммарную отдачу по всем эпизодам
- $$\sum_t r_t$$



# Пример: новости в Yahoo!



- Пользователь открывает домашнюю страницу Yahoo! (с историей прошлых визитов, IP адресом, данным аккаунта)
- Алгоритм выбирает информацию, которую разместить на странице (реклама, новости)
- Пользователь реагирует на предложенную информацию (кликает, возвращается и т.д.)

# Верхняя доверительная граница для контекстного бандита

- Параметризация функции полезности  $Q_\theta(s, a)\phi(s, a)^T\theta \approx Q(s, a)$
- Оцениваем среднее значение полезности методом наименьших квадратов
- Одновременно получаем оценку дисперсии  $\sigma_\theta^2(s, a)$
- Верхняя граница пропорциональна дисперсии:  $U_\theta(s, a) = c\sigma$
- Аналогично обычному бандиту выбираем действие с максимизацией верхней границы:

$$a_t = \arg \max_{a \in A} Q_\theta(s, a) + c\sigma_\theta$$

# Заключение

- Основные варианты решения проблемы соотношения исследования среды и использования полученных знаний:
  - ▶ наивные методы –  $\epsilon$ -жадный алгоритм,
  - ▶ оптимистичная инициализация,
  - ▶ верхняя доверительная граница
- Эти методы разрабатываются для многоруких бандитов, но применимы и для обычных МППР