Машинное обучение (Machine Learning)

Обучение с подкреплением (Reinforcement Learning)

Уткин Л.В.

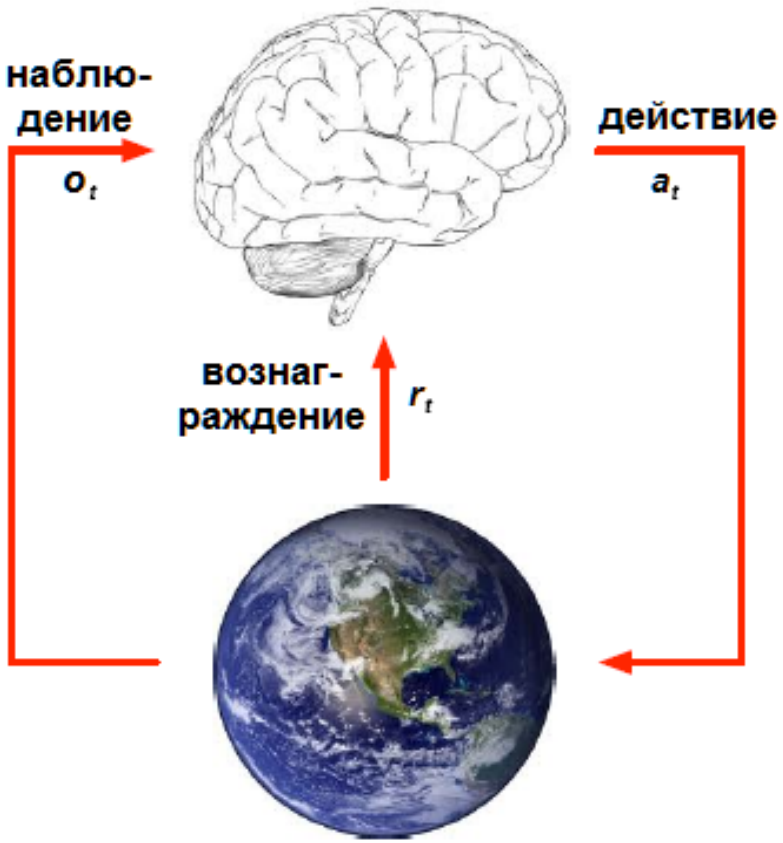
Санкт-Петербургский политехнический университет Петра Великого

Содержание

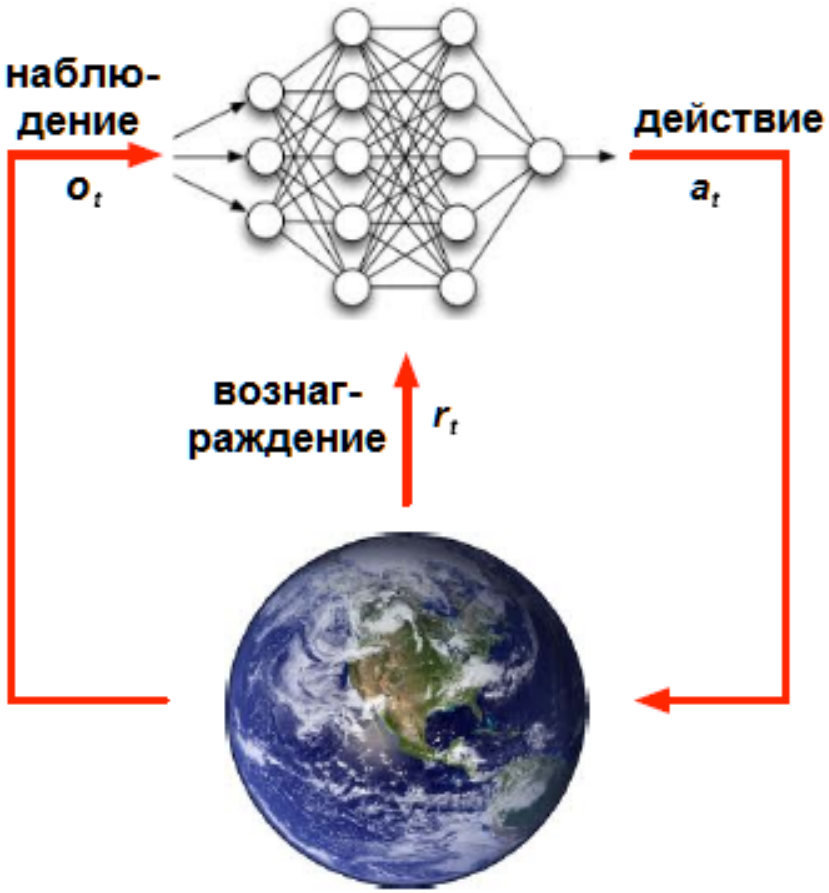
Постановка задачи

- В отличие от стандартных задач обучения с учителем и без учителя, появляется новый “субъект” - агент.
- Параметры задачи обучения могут изменяться в процессе решения при получении информации, насколько то или иное действие (решение, параметр) правильное или нет.
- Для этого агент взаимодействует с окружающей средой, предпринимая действия; окружающая среда его поощряет за эти действия или, наоборот, наказывает, а агент продолжает предпринимать эти же или другие действия.

Постановка задачи



Что мы хотим?



Постановка задачи

- На каждом шаге t агент:
  - Выполняет действие a_t
  - Получает наблюдение o_t
  - Получает вознаграждение r_t
- Среда:
  - Получает действие a_t
  - Выдает наблюдение o_{t+1}
  - Выдает вознаграждение r_{t+1}

Постановка задачи

- Опыт - последовательность наблюдений, действий, вознаграждений

$$o_1, r_1, a_1, o_2, r_2, a_2, \ldots, o_t, r_t$$

- **Состояние** - функция опыта

$$s_t = f(o_1, r_1, a_1, o_2, r_2, a_2, \ldots, o_t, r_t)$$

- В полностью наблюдаемой среде

$$s_t = f(o_t)$$

Состояние системы

- Состояние системы - параметр или множество параметров, используемых для описания системы.
- Например, географические координаты робота - **состояние**.
- Если состояние системы изменяется во времени - динамическая система (двигающийся робот)
- Очередь в магазин: **состояние** - число людей - динамическая система
- Переход из состояния в состояние обычно случайный
- Время перехода $= 1$

Действие

- Движение робота управляется. Предположим робот может двигаться дискретными шагами на север, юг, восток и запад.
- Эти четыре направления - **действия**.
- Для очереди в кассу: когда число покупателей преышает заданное число (пусть 10), оставшиеся покупатели перенаправляются в другую кассу. Два **действия**: 1 - перенапрявлять, 2 - не перенаправлять.

Вознаграждение и стратегия

- Вознаграждение: Положительное или отрицательное при переходе из одного состояния в другое $r(i, a, j)$ или $r_{i,j}(a)$
- Стратегия π: определяет действие, которое должно быть выбрано в каждом состоянии (отображение из состояния в действие).
  - детерминированная стратения: $a = f(s)$
  - вероятностная стратения: $\pi(a|s) = \Pr\{a|s\}$

Постановка задачи

- Без обратной связи, указывающей, какое действие является хорошим, а какое плохим, агент не сможет принимать решения, что делать дальше.
- Агент должен знать, что его выигрыш - это благоприятный исход, а проигрыш - неблагоприятный.
- Обратная связь называется **подкреплением**.
- Получаем **обучение с подкреплением** (reinforcement learning).

Постановка задачи

Задача обучения с подкреплением состоит в том, чтобы обеспечить использование наблюдаемых вознаграждений для определения в процессе обучения оптимальной (или почти оптимальной) стратегии для данной среды.

Более формально

- На каждом шаге агент может находиться в состоянии $s \in S$.
- На каждом шаге агент выбирает из имеющегося набора действий некоторое действие $a \in A$.
- Окружающая среда сообщает агенту, какую награду r он за это получил и в каком состоянии s' после этого оказался.

Отличие от обучения с учителем (1)

- В обучении с учителем: имеется внешний “учитель”, который имеет знания среды и который ими делится с агентом.
- Но имеются некоторые задачи, в которых существует так много комбинаций подзадач, что только агент может их выполнять для достижения цели. “Учитель” практически нецелесообразен. Например, в шахматной игре есть десятки тысяч ходов. Поэтому создание базы знаний всех ходов - утомительная задача.

Отличие от обучения с учителем (2)

- **Более целесообразно учиться на собственном опыте и получать от него знания**. Это основное различие RL от обучения с учителем.
- В обоих способах имеется функция между входом и выходом. Но в RL есть функция вознаграждения, которая действует как обратная связь с агентом в отличии от обучения с учителем.

Пассивное обучение

- Пассивный агент просто наблюдает за происходящим миром и пытается узнать полезность нахождения в различных состояниях
- Или: пассивный агент имеет **фиксированную стратегию** π
- Служит в качестве компонента активных алгоритмов обучения

Активное обучение

- Агент изменяет свою стратегию в процессе обучения
- Агент пытается найти **оптимальную (или, по крайней мере, хорошую) стратегию**
- Аналогично решению, лежащему в основе процесса принятия решений Маркова

Мышь в лабиринте

- Мышь ищет самое большое вознаграждение (+1000) - сыр в конце лабиринта
- Или меньшее вознаграждение - воду (+10)
- Между тем, мышь хочет избежать местоположений, которые могут привести к поражению эл. током (-100)

Мышь в лабиринте

- После небольшого исследования мышь может найти “оазис” с 3 источниками воды возле входа и оставаться там, постоянно получая небольшие вознагражд. этих источников и никогда не отправляясь дальше в лабиринт, чтобы искать более крупный приз.
- Но мышь не достигнет лучший “оазис” или сыр!
- Компромисс между разведкой и эксплуатацией (дилемма разведка-эксплуатация).

Мышь в лабиринте

- Простая стратегия - мышь принимает самое известное действие большую часть времени (скажем, 80% времени), но иногда изучает новое, случайно выбранное направление, даже если оно уходит от известного вознаграждения.
- Эта стратегия - ε-жадная, где ε - это процент времени, когда агент принимает **случайное** действие.
- Со временем ε может уменьшаться (разведываем все меньше, так как уже что-то знаем)

Мышь в лабиринте - марковский процесс решений (MDP)

- Мышь, блуждающая по лабиринту, может быть формализована как процесс принятия решений Маркова (для которого указаны вероятности перехода из состояния в состояние)
- MDP включает: конечное множество состояний, набор действий в состоянии, переходы между состояниями, вознаграждения, коэффициент дисконтирования, отсутствие памяти

MDP

- Конечное **множество состояний** (возможные позиции мыши в лабиринте)
- Набор **действий** в состоянии: { вперед, назад} в коридоре и { вперед, назад, влево, вправо} на перекрестке.
- **Переходы** между состояниями (переходные вероятности)
- **Вознаграждение**, связанное с переходом (у мыши большая часть вознаграждений равна 0, часть полож., если достигли воду или сыр, и отриц., если достигли поражение током)

MDP

- **Коэффициент дисконтирования** $\gamma \in [0, 1]$. Определяет разницу в важности между текущим и будущими вознаграждениями. Например, если $\gamma = 0.9$ и есть вознаграждение в размере 5 после 3 шагов, настоящая стоимость этого вознаграждения $0.9^3 \cdot 5$.
- **Отсутствие памяти**. Как только текущее состояние известно, история перемещения мыши через лабиринт может быть стерта, потому что “будущее не зависит от прошлого при данном настоящем”.

Value function (ценность действия)

- Для каждого состояния введ̧м функцию $V(a, s)$ (value function,ценность действия a):
  - Какое вознаграждение я получу от действия a в состоянии s?
- Q-value function - ожидаемое полное вознаграждение
  - из состояния s и действия a
  - при стратегии π
  - с коэффициентом дисконтирования γ (discounted reward)

$$Q^{\pi}(s, a) = E\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a\right]$$

- Value function декомпозируется в уравнение Беллмана

$$Q^{\pi}(s, a) = E_{s', a'}\left[r + \gamma Q^{\pi}(s', a') \mid s, a\right]$$

Оптимальная value function

- Оптимальная ценность действия - ее максимум

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) = Q^{\pi*}(s,a)$$

- Если получена Q^*, то оптимальное действие

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

- Оптимальное значение максимизируется по всем решениям (неформально):

$$\begin{aligned} Q^*(s,a) &= r_{t+1} + \gamma \max_{a_{t+1}} r_{t+2} + \gamma^2 \max_{a_{t+2}} r_{t+3} + \dots \\ &= r_{t+1} + \gamma \left[\max_{a_{t+1}} r_{t+2} + \gamma \max_{a_{t+2}} r_{t+3} + \dots \right] \\ &= r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \end{aligned}$$

Оптимальная value function

- Оптимальное значение максимизируется по всем решениям (неформально):

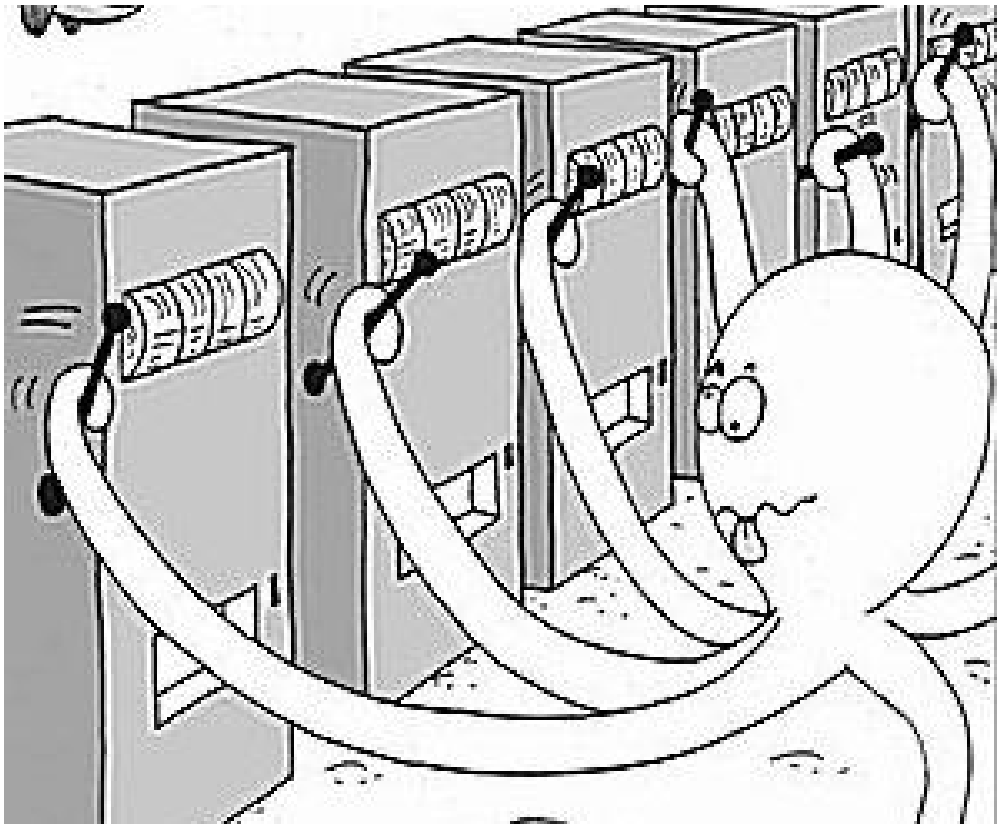
$$\begin{aligned} Q^*(s,a) &= r_{t+1} + \gamma \max_{a_{t+1}} r_{t+2} + \gamma^2 \max_{a_{t+2}} r_{t+3} + \dots \\ &= r_{t+1} + \gamma \left[\max_{a_{t+1}} r_{t+2} + \gamma \max_{a_{t+2}} r_{t+3} + \dots \right] \\ &= r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \end{aligned}$$

- Формально - в уравнение Беллмана

$$Q^*(s,a) = E_{s'} \left[r + \gamma \max_{a'} Q^*(s', a') \mid s, a \right]$$

Многорукий бандит

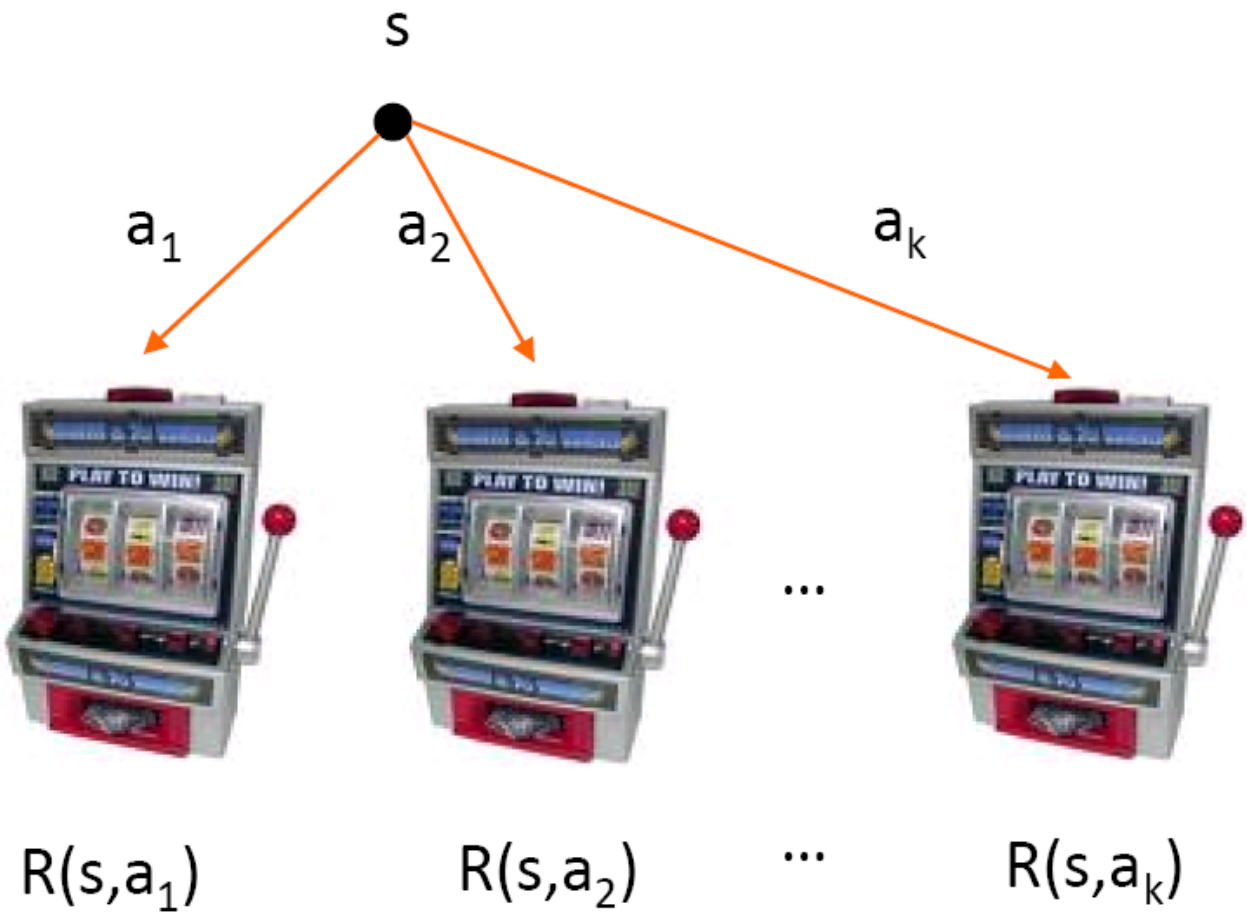
Многорукий бандит



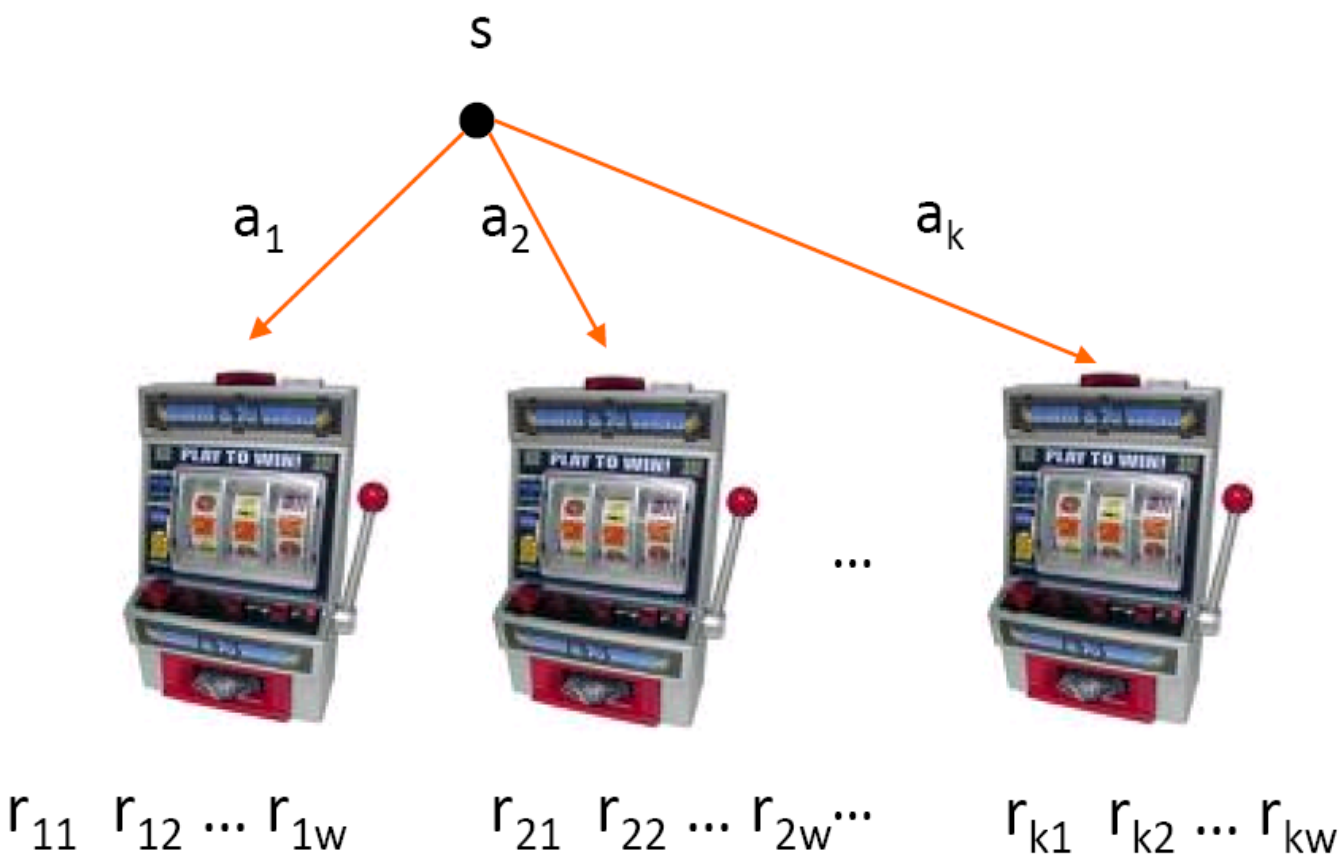
Многорукий бандит

- Агент многократно выбирает одно из n действий (n-рукий бандит - игральные автоматы)
- У каждого автомата есть свой ожидаемый выигрыш
- После каждого выбора подкрепление определяется из стационарного распределения, зависящего от выбора
- Задача: максимизировать ожидаемое суммарное вознаграждение за большое число (N) попыток

Многорукий бандит



Многорукий бандит



Многорукий бандит - более формально

- Многорукий бандит - кортеж $\langle A, R \rangle$
- A - известное множество m действий (или "рук")
- $p^a(r) = \Pr[r|a]$ - неизвестное распределение вероятностей вознаграждений за $a \in A$
- На каждом шаге t агент выбирает действие $a_t \in A$ с учетом стратегии $a_t \sim \pi_t(a)$
- Среда генерирует вознаграждение $r_t \sim p^{a_t}(r)$
- Цель - максимизировать суммарное вознаграждение $\sum_{i=1}^{t} r_i$

Многорукий бандит - жадный алгоритм

- Ожидаемое (среднее) значение вознаграждения $Q(a)$
- Агент вычисляет оценки $Q_t(a)$
- Стратегия, максимизирующая текущую оценку ценности $A_t \subseteq A$: $A_t = \arg\max_{a \in A} Q_t(a)$
- **Жадная стратегия** выбирать любое действие из A_t:

$$\pi_{t+1}(a) = \frac{1}{|A_t|}[a \in A_t]$$

- Или более просто

$$Q_t(a) = \frac{r_1(a) + r_2(a) + \ldots + r_N(a)}{N}$$

- $r_N(a)$ - число, когда выбирается действие a за первые t шагов, $r_i(a)$ - полученные в результате подкрепления.

Многорукий бандит - жадный алгоритм

$$
\begin{aligned}
Q_{t+1}(a) &= \frac{1}{t+1} \sum_{i=1}^{t+1} r_i \\
&= \frac{1}{t+1} \left(r_{t+1} + \sum_{i=1}^{t} r_i \right) \\
&= \frac{1}{t+1} \left(r_{t+1} + tQ_t(a) + Q_t(a) - Q_t(a) \right) \\
&= \frac{1}{t+1} \left(r_{t+1} + (t+1)Q_t(a) - Q_t(a) \right) \\
&= Q_t(a) + \frac{1}{t+1} \left(r_{t+1} - Q_t(a) \right)
\end{aligned}
$$

Многорукий бандит - жадный алгоритм

$$Q_{t+1}(a) = Q_t(a) + \frac{1}{t+1}\left(r_{t+1} - Q_t(a)\right)$$

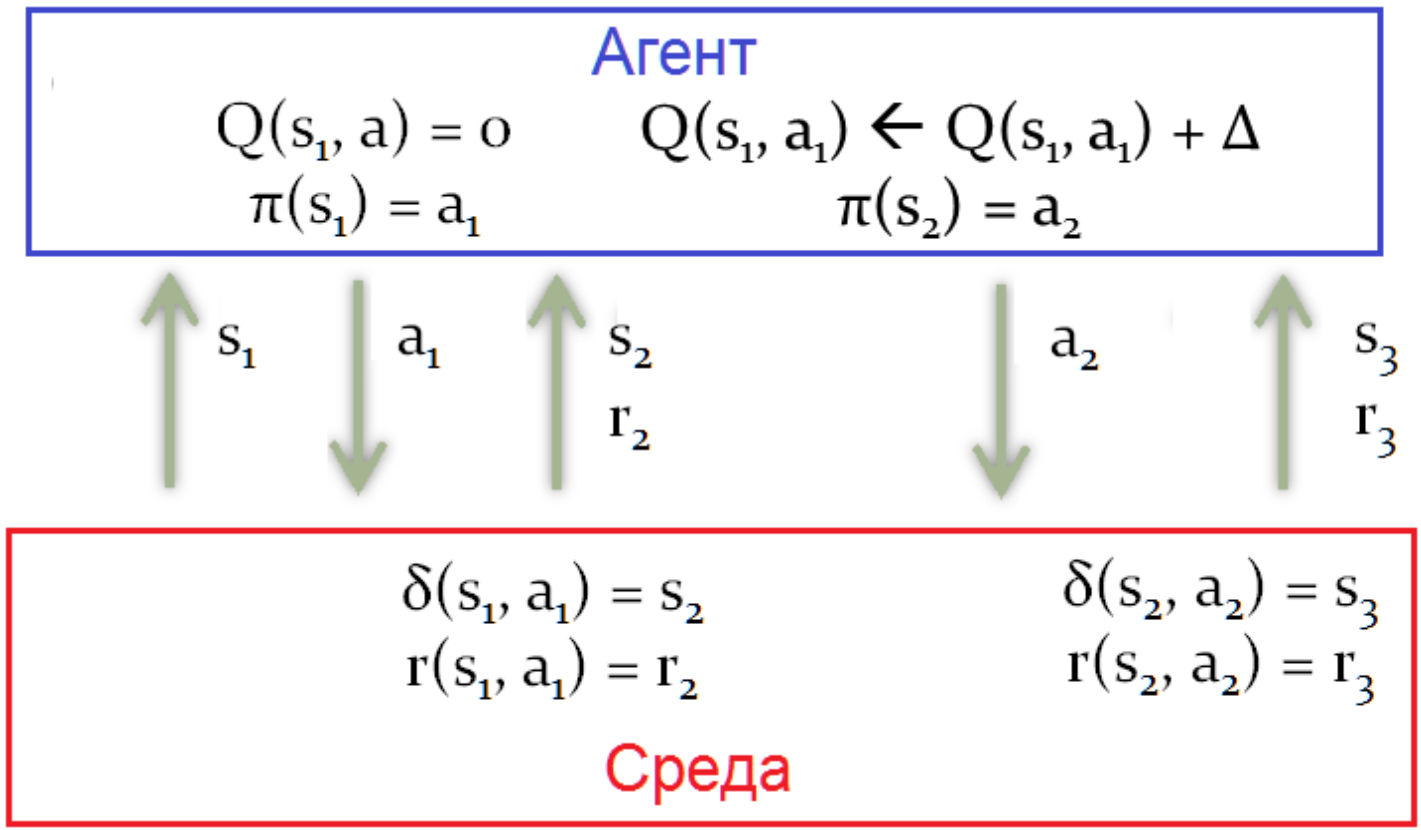
- $Q_{t+1}(a)$ - среднее вознаграждение за $t+1$ шагов
- $\frac{1}{t+1}$ - скорость обучения или шаг
- r_{t+1} - вознаграждение на шаге $t+1$
- Шаг $\frac{1}{t+1}$ не изменятеся, но если изменять, то можно улучшить процедуру обучения

Вычисление оценок

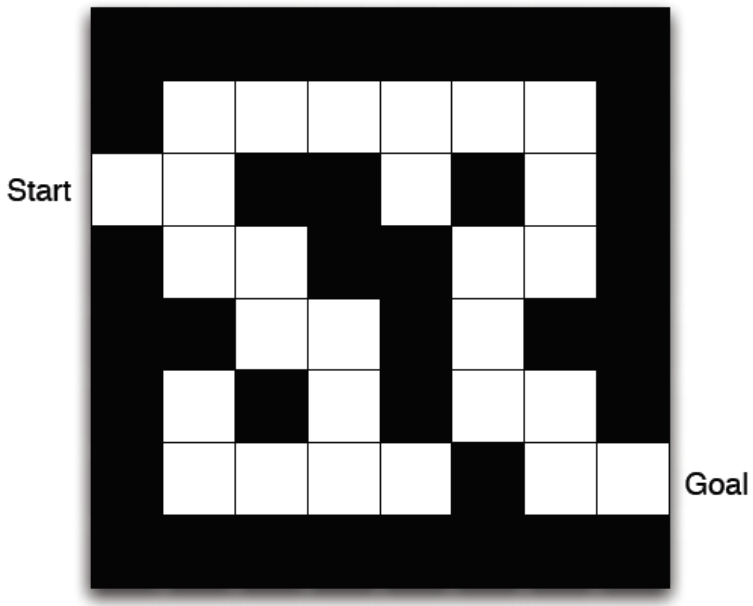
*Новая оценка=Старая оценка + шаг * (Цель - Старая оценка)*

(Цель - Старая оценка) - ошибка

Вычисление оценок

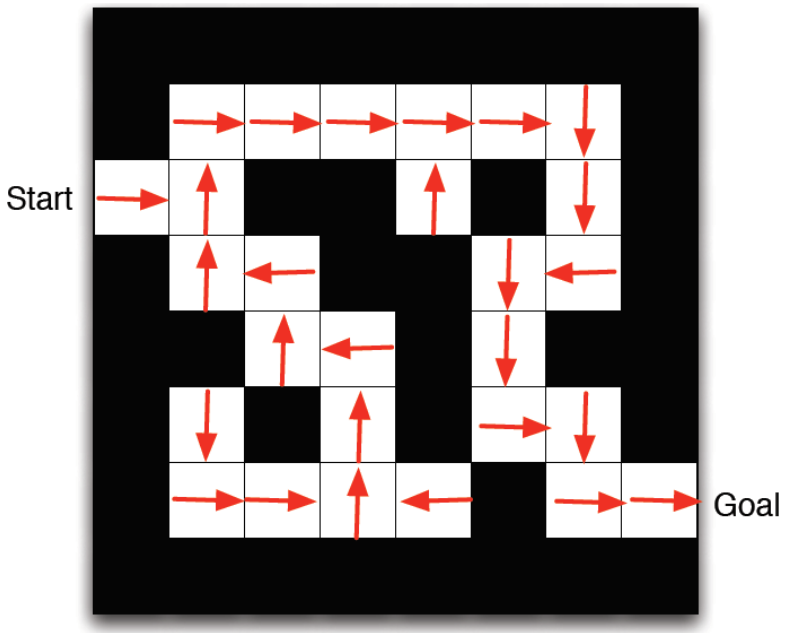


Пример



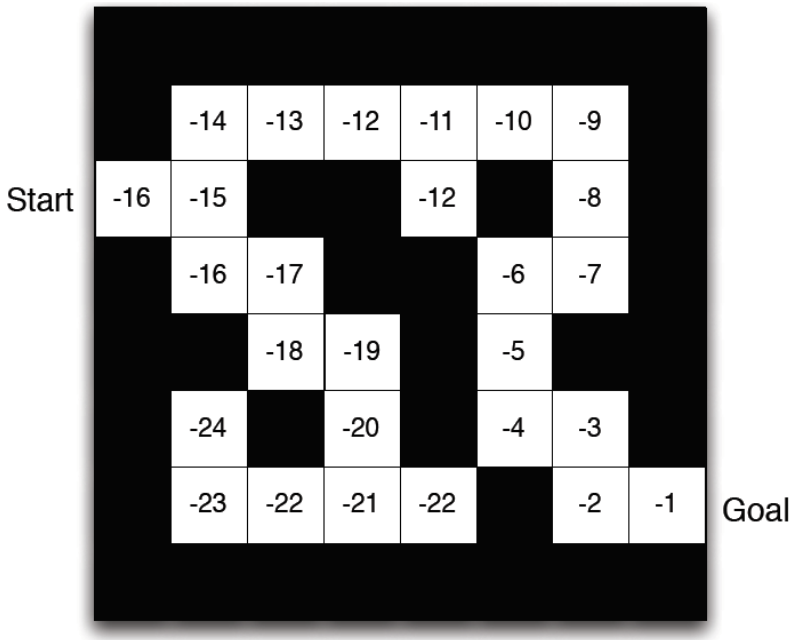
- Вознаграждение: -1 за каждый шаг
- Действия: N,E,S,W
- Состояния: положение агента

Пример



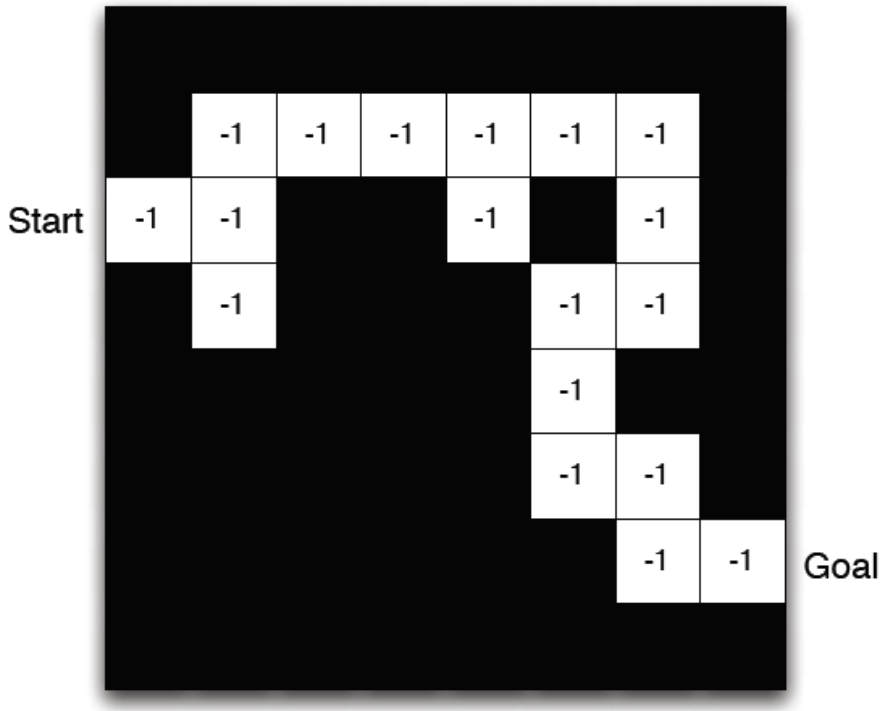
- Стрелки представляют собой стратегию $\pi(s)$ для каждого состояния s

Пример



- Числа - значение $V_\pi(s)$ для каждого состояния

Пример



- Макет сетки - модели переходов
- Числа - немедленное вознаграждение для каждого состояния

Сожаления

- Функция $Q(a)$ - среднее вознаграждение для действия a: $Q(a) = E[r|a]$
- Оптимальное значение V^*:

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

- Сожаление - возможные потерь за один шаг

$$l_t = E[V^* - Q(a_t)]$$

- Суммарные сожаления -

$$L_t = E\left[\sum_{i=1}^{t} V^* - Q(a_i)\right]$$

- Максимум суммарного вознаграждения = минимум суммарного сожаления

Подсчет сожалений

- $N_t(a)$ - среднее число выбора действия a
- Зазор Δ_a - разность между значений действия a и оптимального действия a^*: $\Delta_a = V^* - Q(a)$
- Сожаление - функция зазора и $N_t(a)$:

$$L_t = E\left[\sum_{i=1}^{t} V^* - Q(a_i)\right]$$
$$= \sum_{a \in A} E\left[N_t(a)\right](V^* - Q(a)) = \sum_{a \in A} E\left[N_t(a)\right]\Delta_a$$

- Хороший алгоритм обеспечивает малое значение $N_t(a)$ для больших зазоров
- Проблема: зазоры не известны!

Проблемы и изменение условий

- Агент может иметь внутреннюю модель среды
- Динамика: как действия изменяют состояние
- Вознаграждение: какое вознаграждение в каждом состоянии
- Модель может быть неправильной

Epsilon - жадный алгоритм или случайные стратегии

- с вероятностью $1 - \varepsilon$ выбираем жадное действие

$$Q_t(a^*) = \max_a Q_t(a)$$

- с вероятностью ε выбираем не жадное действие

Мягкий максимум или случайные стратегии

- Мягкий вариант дилеммы **Разведка-Эксплуатация**
- Вероятность выбора действия пропорциональна его текущей оценке $Q_t(a)$:

$$\pi_t(a) = \frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}}$$

- Увеличивая температуру τ, уменьшаем разницу между вероятностями выбора
- Уменьшая температуру τ, приближаем выбор к жадному
- Обычно температура понижается со временем

Метод преследования жадной стратегии

- Вместо жадной стратегии с распределением

$$\pi_{t+1}(a) = \frac{1}{|A_t|}[a \in A_t]$$

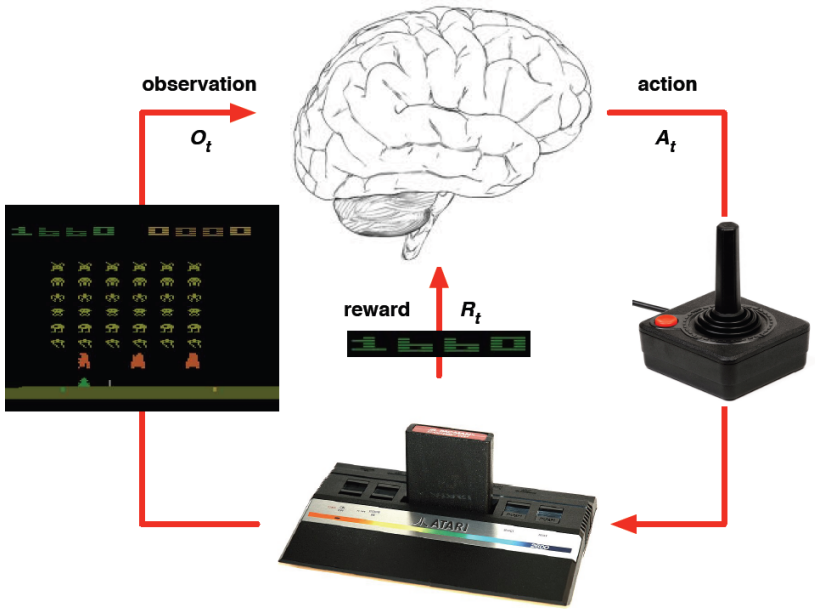
- используем преследование (сглаживание) жадной стратегии:

$$\pi_{t+1}(a) = \pi_t(a) + \beta\left(\frac{[a \in A_t]}{|A_t|} - \pi_t(a)\right)$$

Обучение и планирование

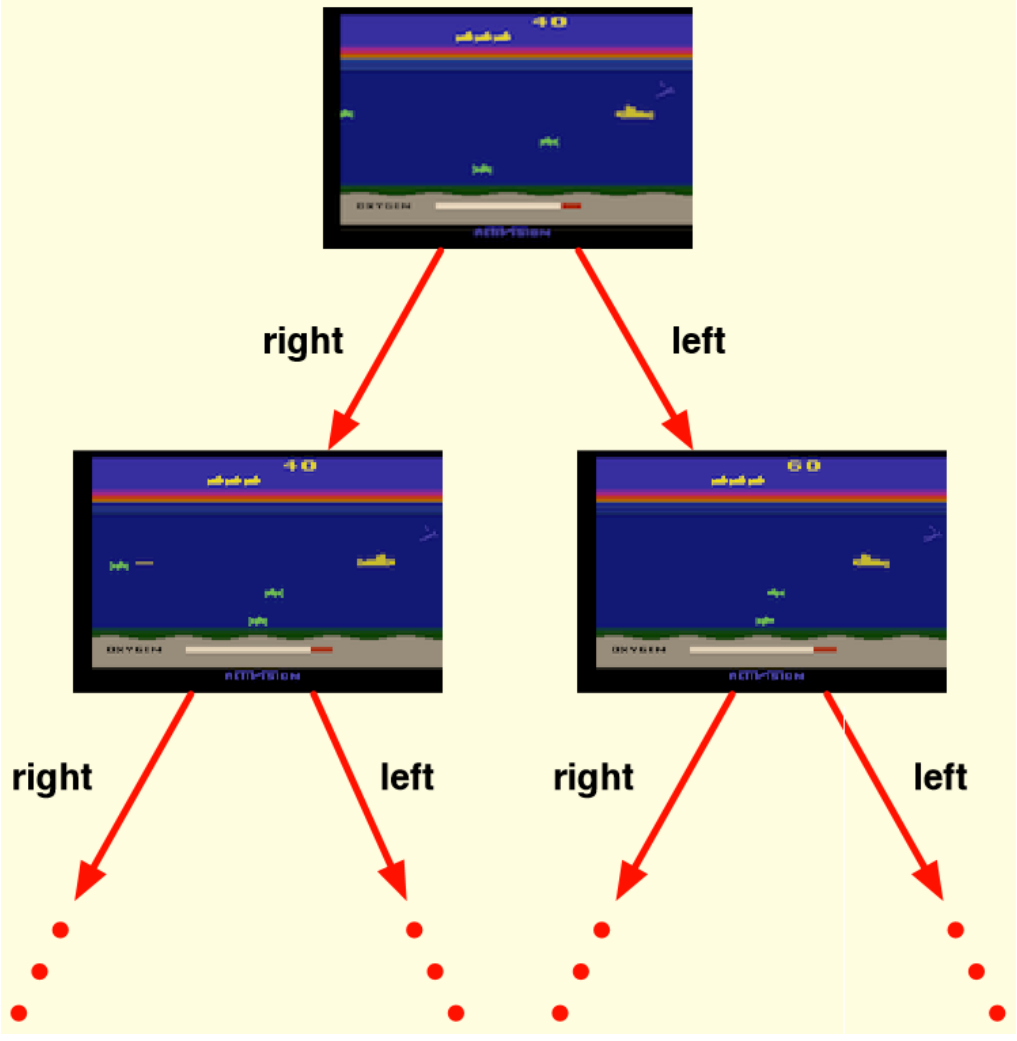
- Две фундаментальные проблемы в последовательном принятии решений
- Обучение с подкреплением:
  - Среда априори неизвестна
  - Агент взаимодействует со средой
  - Агент улучшает свою стратегию
- Планирование:
  - Модель среды известна
  - Агент выполняет вычисления с его моделью (без какого-либо внешнего взаимодействия)
  - Агент улучшает свою стратегию (обсуждение, поиск)

Обучение с подкреплением - Atari



- Правила игры неизвестны
- Обучение непосредственно из взаимодействия игра-игрок
- Действия выбираются джойстиком

Планирование - Atari (1)



Планирование - Atari (2)

- Правила игры известны
- Можно запросить эмулятор идеальной модели “внутри мозга” оператора
- Если я возьму действие a из состояния s:
  - Каким будет следующее состояние?
  - Каков был бы счет?
- Планируйте заранее, чтобы найти оптимальную политику, например, поиск дерева

Обучение - главные отличия от планирования

- Имеется доступ к “реальной системе”, но не модели
- Генерируется опыт $s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_t, a_t, r_t$. Как в жизни!
- Два класса методов:
  1. Косвенные методы
  2. Прямые методы

Косвенные методы

- Используются опытные данные для оценки модели

$$\widehat{P}(j|i,a) = \frac{\#j \leftarrow i,a}{\#j \leftarrow i,\cdot}$$

- Вычисляется оптимальная стратегия в соответствии с оцененной моделью
- Дилемма **Разведка-Эксплуатация**

Прямые методы: Q-learning

- $s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_t, a_t, r_t, \ldots$
- Единичный опыт $\langle s_t, a_t, r_t, s_{t+1} \rangle$
- Модифицируем

$$Q_{\text{new}}(s_k, a_k) = r_k + \gamma \max_b Q_{\text{old}}(s_{k+1}, b)$$

или

$$Q_{\text{new}}(s_k, a_k) = (1-\alpha) Q_{\text{old}}(s_k, a_k) + \alpha \left[r_k + \gamma \max_b Q_{\text{old}}(s_{k+1}, b) \right]$$

Дилемма Разведка-Эксплуатация (1)

- Повсеместно распространено в жизни:
  - покупать или продолжать искать
  - использовать код как есть или ...
  - использовать экспериментальную установку как есть или ...
  - использовать модель как есть или ...

Дилемма Разведка-Эксплуатация (2)

- Выбор ресторана
  - *Эксплуатация*: Иди в твой предпочтительный ресторан
  - *Разведка*: Try a new restaurant
- Интернет-банеры
  - *Эксплуатация*: Показать наиболее успешное объявление
  - *Разведка*: Показать различные объявления

Дилемма Разведка-Эксплуатация (3)

- Бурение нефтяных скважин
  - *Эксплуатация*: Бурение в лучших известных местах
  - *Разведка*: Бурение в лучшем месте
- Игра
  - *Эксплуатация*: Делай ход, в который веришь сам больше всего
  - *Разведка*: Делай экспериментальный ход

Дилемма Разведка-Эксплуатация (4)

- Онлайн принятие решений имеет фундаментальный выбор:
  - *Эксплуатация*: принятие наилучшего решения при текущей информации
  - *Разведка*: сбор большей информации
- Лучшая долгосрочная стратегия может включать краткосрочные “жертвы”
- Сбор достаточной информации для принятия наилучших решений

Вопросы

?