Введение в обучение с подкреплением, Многорукие бандиты, Марковский процесс принятия решений

Алексей Скрынник

ПЛАН

01 Введение. Определения и примеры

02 Многорукие бандиты

03 Марковский процесс принятия решений

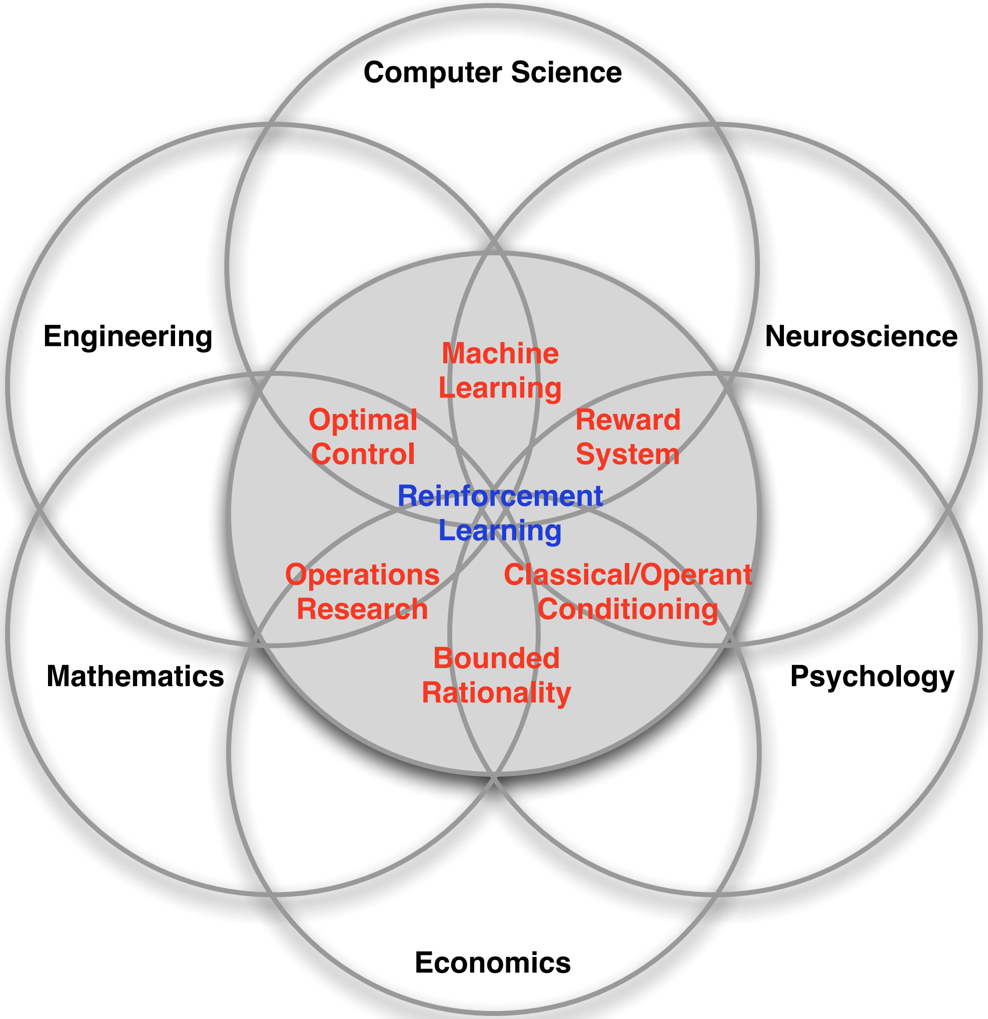
01

Введение. Определения и примеры

Обучение с подкреплением

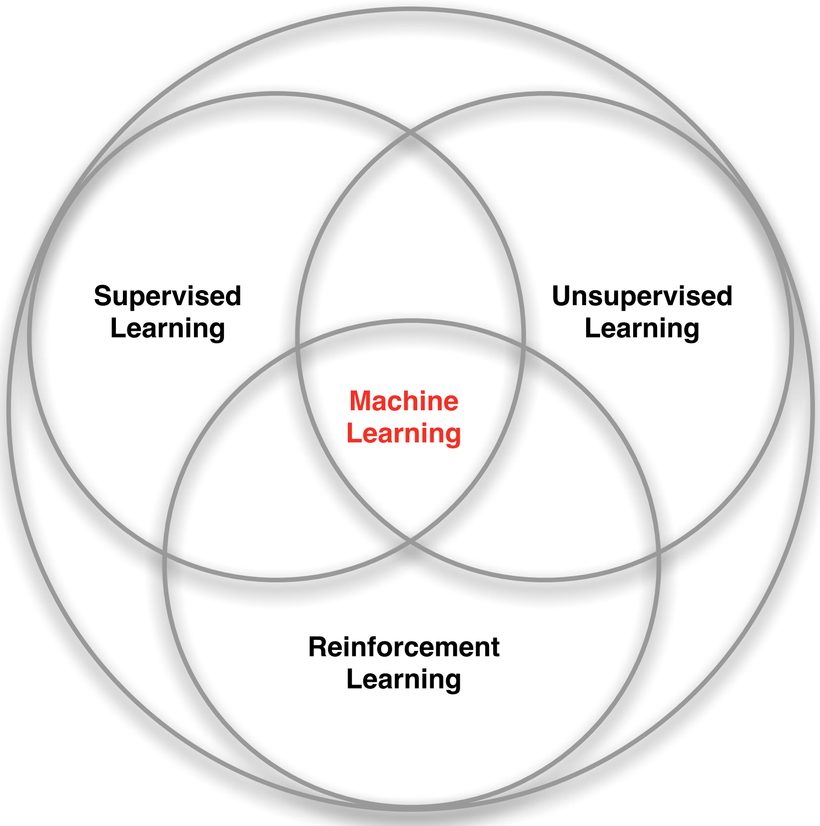
Обучение интеллектуального агента принимать последовательные решения в среде с неполной информацией

Обучение с подкреплением и другие науки



Особый вид машинного обучения

- Отделение среды (environmnet) и агента (agent) – источник и акцепторв данных в явном виде присутствуют в постановке задачи
- Нет учителя, т.е. ошибка не задается явно, а косвенно передается через вознаграждение (reward)
- Обратная связь от среды может поступать с задержкой
- Параметр времени имеет особое значение – последовательные данные
- Действия агента влияют на поступающие в дальнейшем данные



Примеры обучения с подкреплением

- AlphaGo, AlphaZero – игра в Go лучше человека
- Игры: Atari, StarCraft, MineCraft, Dota – иногда лучше человека
- Управление энергетической станцией
- Управление инвестиционным пакетом
- Автоматизация “Холодных звонков”
- Реализация сложных движений робота

02

Многорукие бандиты

Проблема исследования и использования

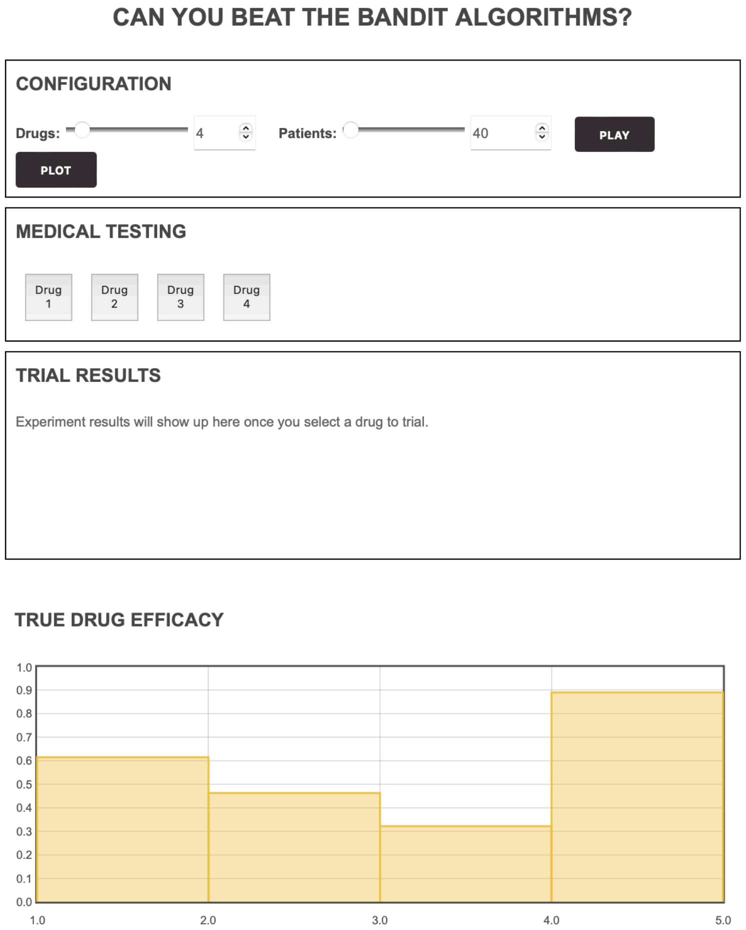
- Фундаментальная проблема в теории последовательного принятия решений - выбор между:
  - **исследованием** среды (eXploration) – сбор дополнительной информации,
  - **использование** полученных знаний (eXploitation) – применение наилучшего действия в текущей ситуации
- Наилучшая долгосрочная стратегия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

Примеры

- Выбор ресторана:
  - использование – пойти в любимый ресторан,
  - исследование – попробовать новый ресторан
- Баннерная реклама в сети:
  - использование – показать наиболее успешный баннер,
  - исследование – показать другой баннер
- Настольные игры:
  - использование – выбрать наилучший по текущему представлению ход,
  - исследование – попробовать рискованный ход

Проблема исследования и использования

- Тестирование нескольких лекарств от болезней на пациентах
- Действие – выбор конкретного лекарства
- Вознаграждения: – если пациент жив, 0 – иначе



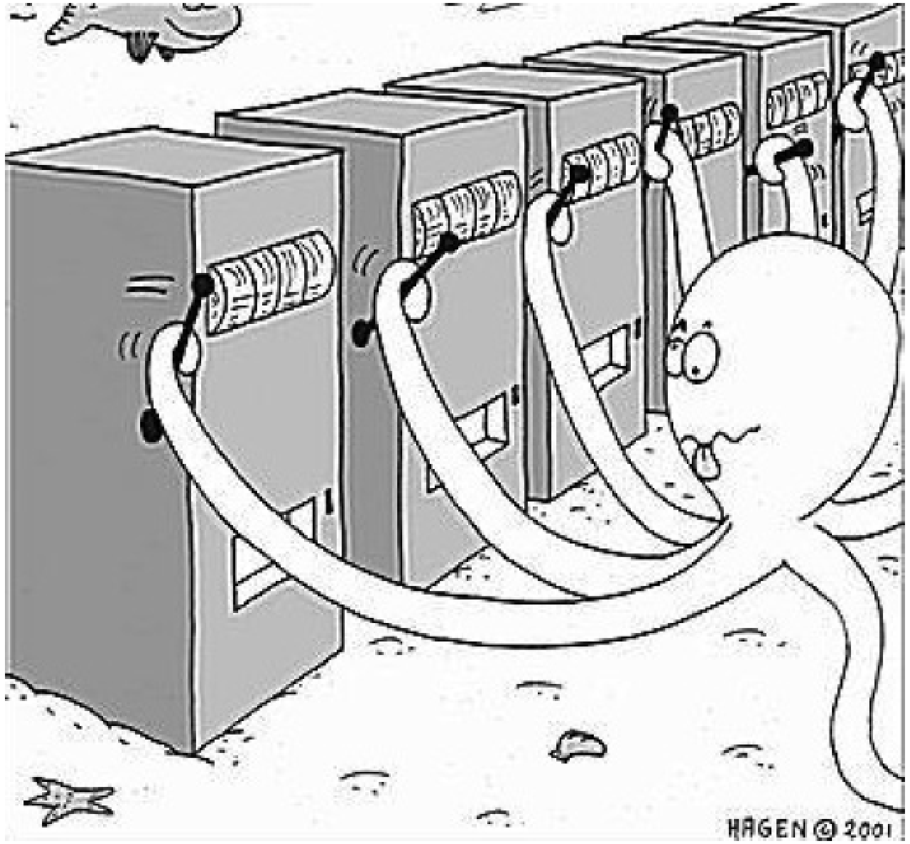
Интерактивный пример – `http://iosband.github.io/2015/07/28/Beat-the-bandit.html`

Способы решения ХХ дилеммы

- → **Наивный подход** – добавление шума к жадной стратегии (ϵ-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
- → **Сопоставление вероятностей** (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

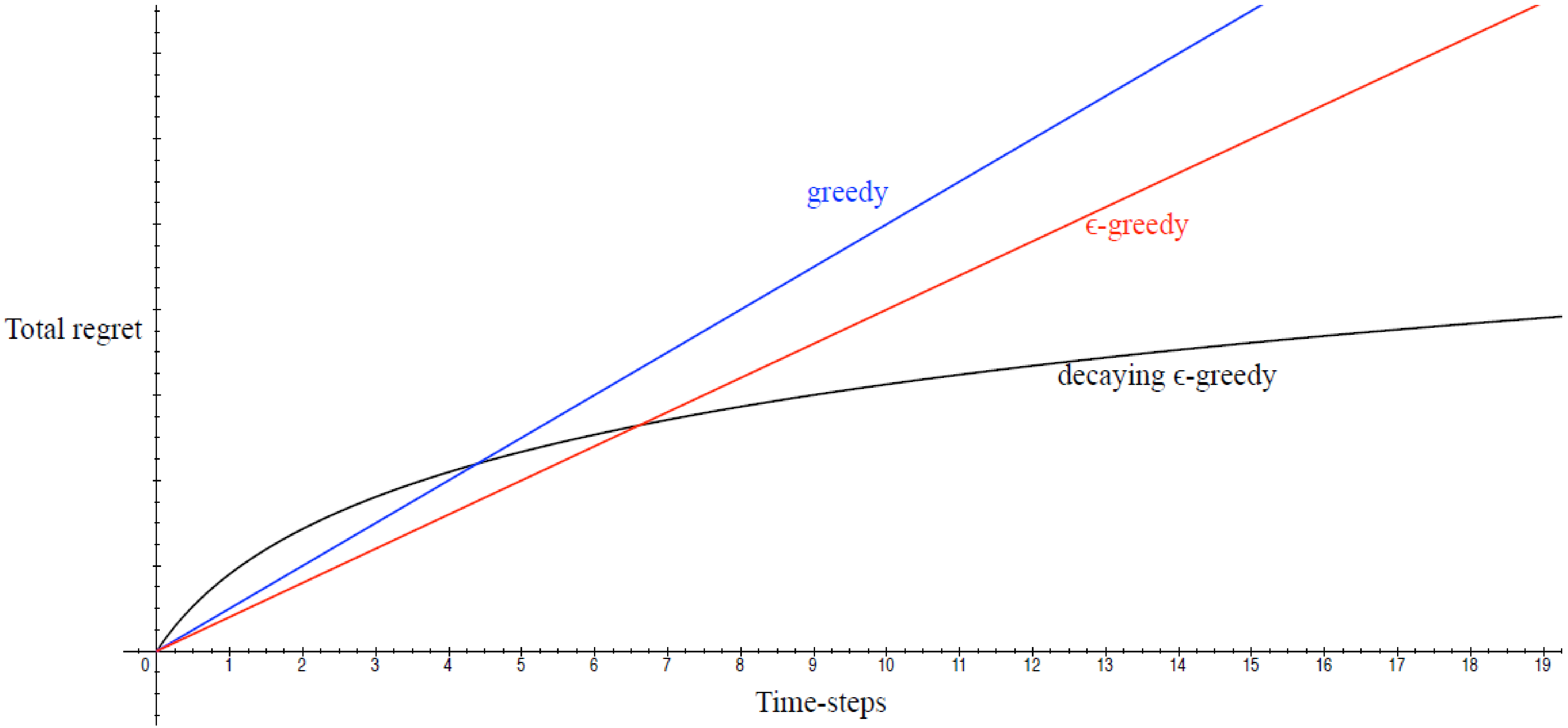
Подсчет потерь

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

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

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

Линейные и сублинейные потери



Жадный алгоритм

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

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

- Жадный алгоритм выбирает действие с максимальной полезность:

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

- 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 ϵ – случайное действие
- Константа ϵ только обеспечивает уменьшение потерь:

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

- $\Rightarrow$ 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 ϵ-жадного алгоритма с оптимистичной инициализацией линейно растут потери

Затухающий ϵ_t-жадный алгоритм

- Выберем некоторое расписания для изменения параметра ϵ
- 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 ϵ_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 $\mathcal{R}$)

Пример UCB1 – логарифмические полные потери

Алгоритм UCB1 выбора действий:

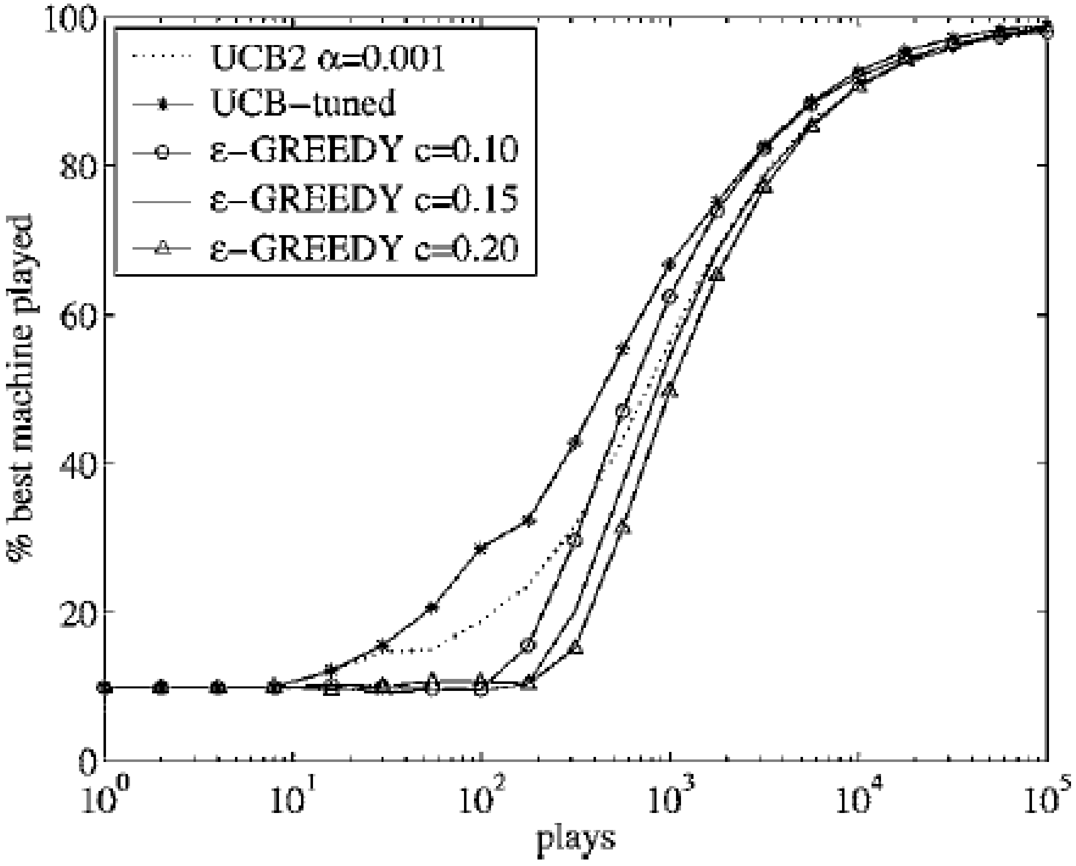
$$a_t = \arg\max_{a \in A} Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

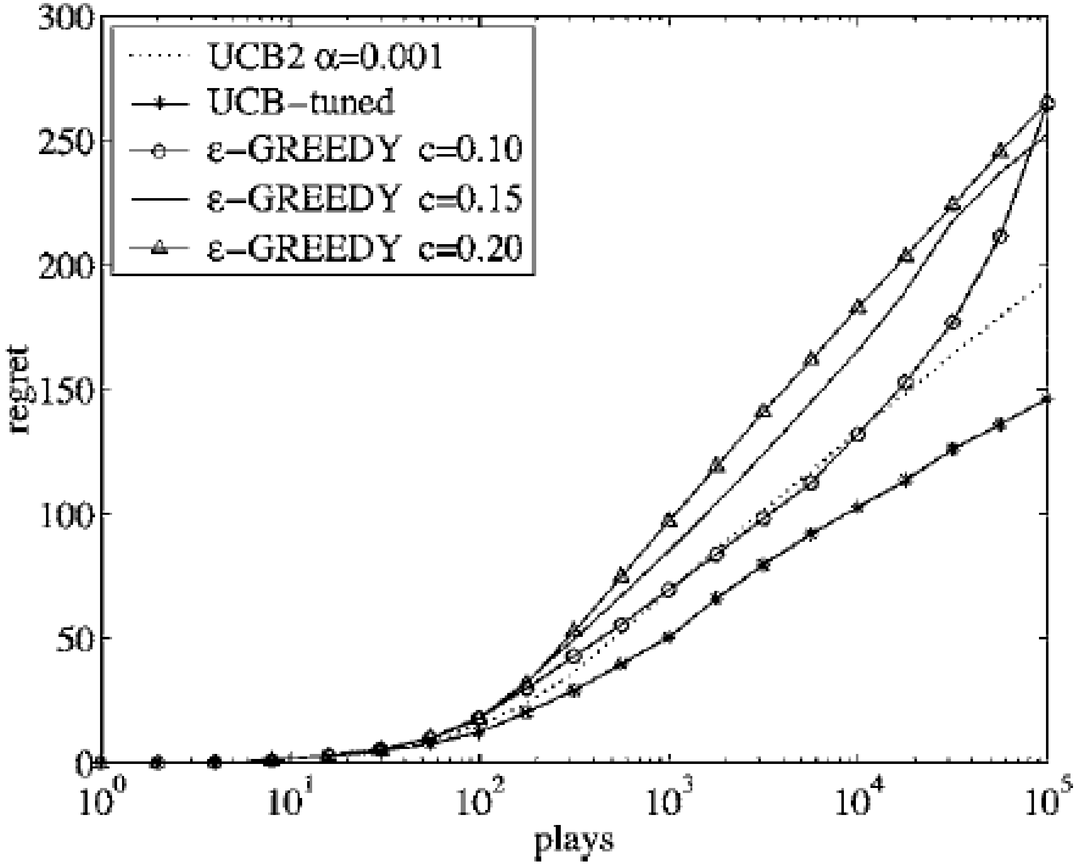
Theorem

UCB алгорим достигает логарифмических асимптотических полных потерь:

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

Пример: 10-рукий бандит





Ограничения многоруких бандитов

- Состояние среды не меняется
- Действия не влияют на будущее
- Учитывается только текущая награда

03

Марковский процесс принятия решений

Постановка задачи: вознаграждение, состояние, наблюдаемость

Вознаграждение

- Вознаграждение r_t – скалярный сигнал обратной связи
- Показывает насколько агент был успешен на шаге t
- Задача агента - максимизировать суммарное вознаграждение

В обучении с подкреплением принята следующая гипотеза:

Definition

Все цели могут быть описаны через максимизацию суммарного вознаграждения.

Примеры:

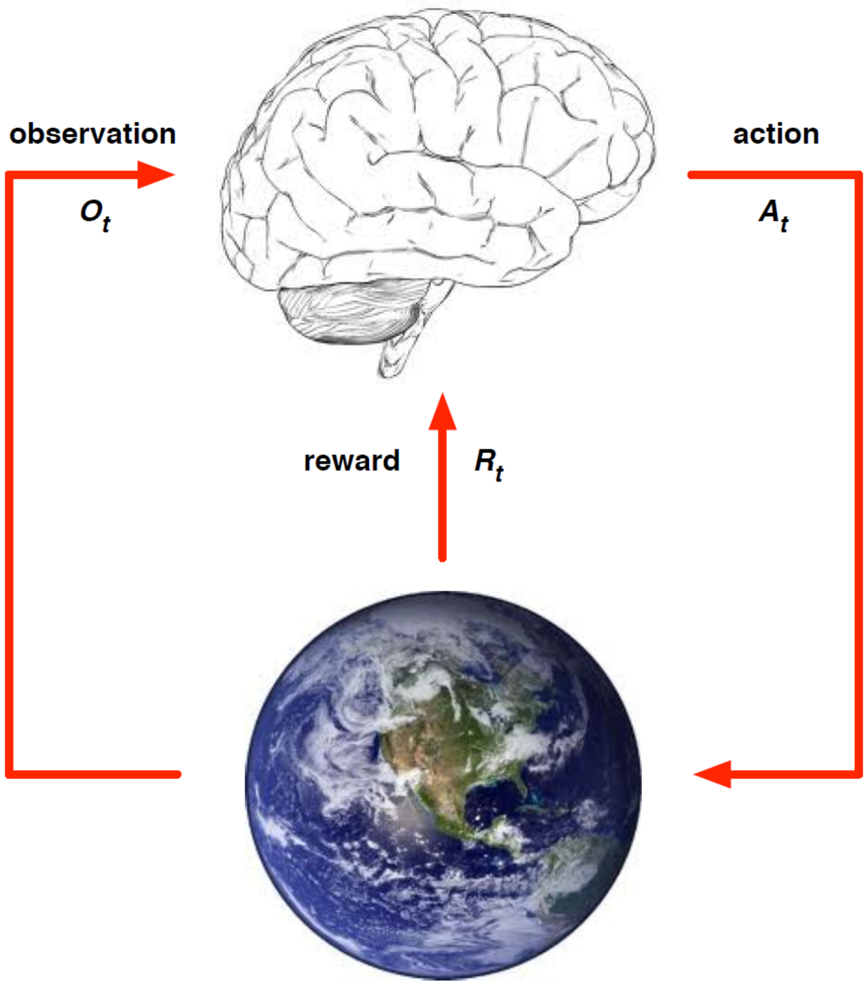
- *AlphaGo*: $+r$ за победу в партии, $-r$ за проигрыш
- *Atari*: $+r$ за увеличение счета, $-r$ за уменьшение счета
- *Энергетическая станция*: $+r$ за генерацию энергии, $-r$ за превышение ограничений безопасности
- *Инвестиционный пакет*: $+r$ за каждый заработанный рубль, $-r$ за каждый потерянный рубль

Последовательное принятие решений

- Цель - выбор действий, максимизирующих суммарное будущее вознаграждение
- Действия могут иметь долговременные (long term) последствия
- Вознаграждения могут быть отложенными
- Иногда выгоднее пренебречь немедленным вознаграждением для получения большего долгосрочного вознаграждения
- Примеры:
  - инвестирование - последствия решений могут проявиться через несколько месяцев,
  - заправка вертолета может предотвратить его падение в следующие несколько часов,
  - блокирование хода противника может помочь выиграть спустя много ходов

Взаимодействие агенты и среды

- В каждый момент времени t агент:
  - выполняет действие a_t,
  - воспринимает наблюдение o_t,
  - получает скалярное вознаграждение r_t
- В каждый момент времени t среда:
  - реагирует на действие a_t,
  - генерирует следующее наблюдение o_{t+1},
  - генерирует скалярное вознаграждение r_{t+1}
- Переход к шагу $t + 1$



История и состояние

- История (взаимодействия) это последовательность наблюдений, действий и вознаграждений:
$$H_t = \langle o_1, r_1, a_1, \ldots, a_{t-1}, o_t, r_t \rangle$$
Например: все наблюдаемые переменные к моменту времени t или сенсомоторный поток робота.
- То, что случится далее зависит от истории:
  - агент выберет действие,
  - среда сгенерирует наблюдение и вознаграждение.
- *Состояние* – это информация используемая для определения того, что произойдет далее:
$$s_t = f(H_t)$$

Марковское состояние

Марковское (или *информационное*) состояние содержит всю необходимую информацию, которая может быть иметься в истории.

Definition

Состояние s_t называется марковским, если и только если

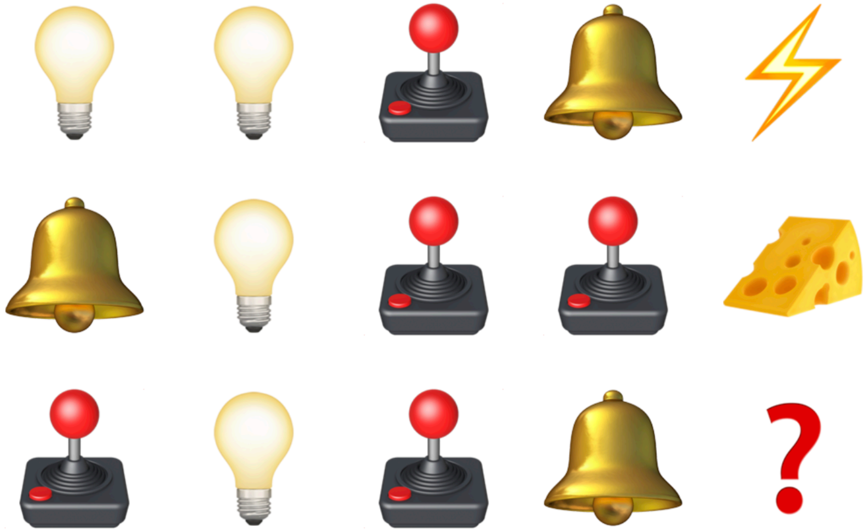
$$\mathbb{P}[s_{t+1}|s_t] = \mathbb{P}[s_{t+1}|s_1, \ldots, s_t]$$

- Иными словами, будущее не зависит от прошлого и определяется только настоящим:

$$H_{1:t} \to s_t \to H_{t+1:\infty}$$

- Если известно текущее состояние, история может быть отброшена
- Текущее состояние содержит всю необходимую статистику о будущем
- Предполагаем, что состояние s_t является марковским

Мышь и сыр



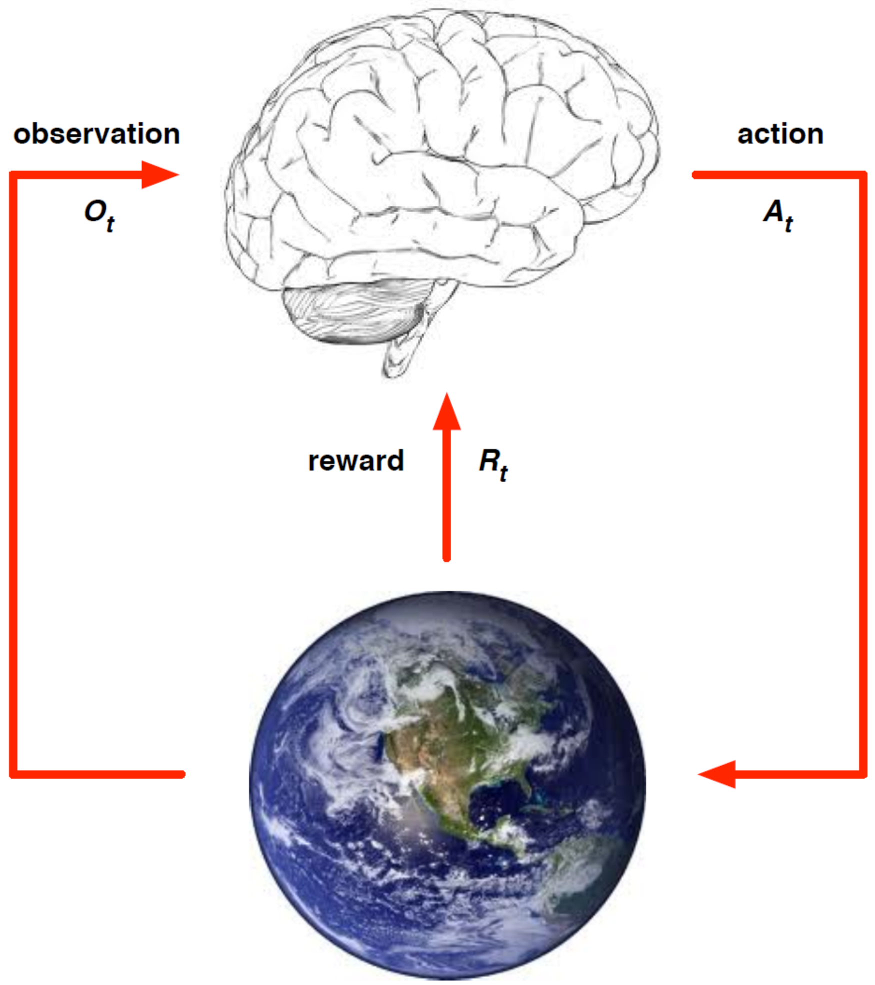
- s_t^a = последние три события в последовательности,
- s_t^a = количество появлений света, звонка и рычага,
- s_t^a = вся последовательность

Полностью наблюдаемые среды

- Полная наблюдаемость: агенту напрямую доступно состояние среды:

$$o_t = s_t$$

- Формально такой случай взаимодействия называется *марковским процессом принятия решений* (МППР, Markov decision process, MDP)



Частично наблюдаемые среды

- Частичная наблюдаемость, агент наблюдает среду опосредованно:
  - робот через видеокамеру не может определить своего точного местоположения,
  - торговый агент может наблюдать только текущие цены,
  - игроку в покер доступны только открытые всем карты
- В этом случае наблюдение агента не соответствует состоянию среды
- Формально – это *частично наблюдаемый марковский процесс принятия решений* (partially observable Markov decision process, POMDP)
- Агент должен сформировать свое собственное представление s_t^a:
  - полная история: $s_t^a = H_t$,
  - *представление* о состоянии среды: $s_t^a = (P[s_t^e = s^1], \dots P[s_t^e = s^n])$,
  - рекуррентная нейронная сеть: $s_t^a = \sigma(s_{t-1}^a w_s + o_t w_o)$

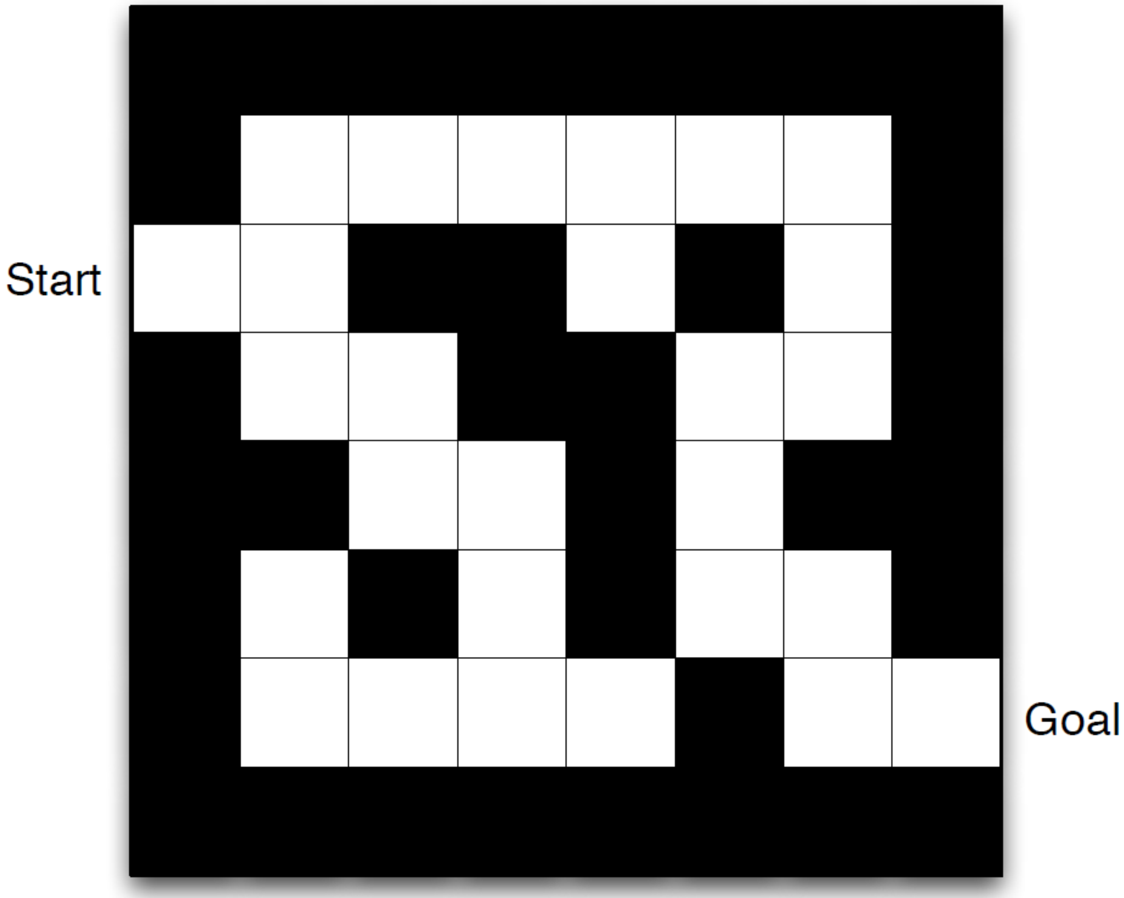
Схема RL-агента: стратегия, полезность, модель

Строение RL агента

Определение любого RL-агент обычно включает одну или несколько следующих составляющих:

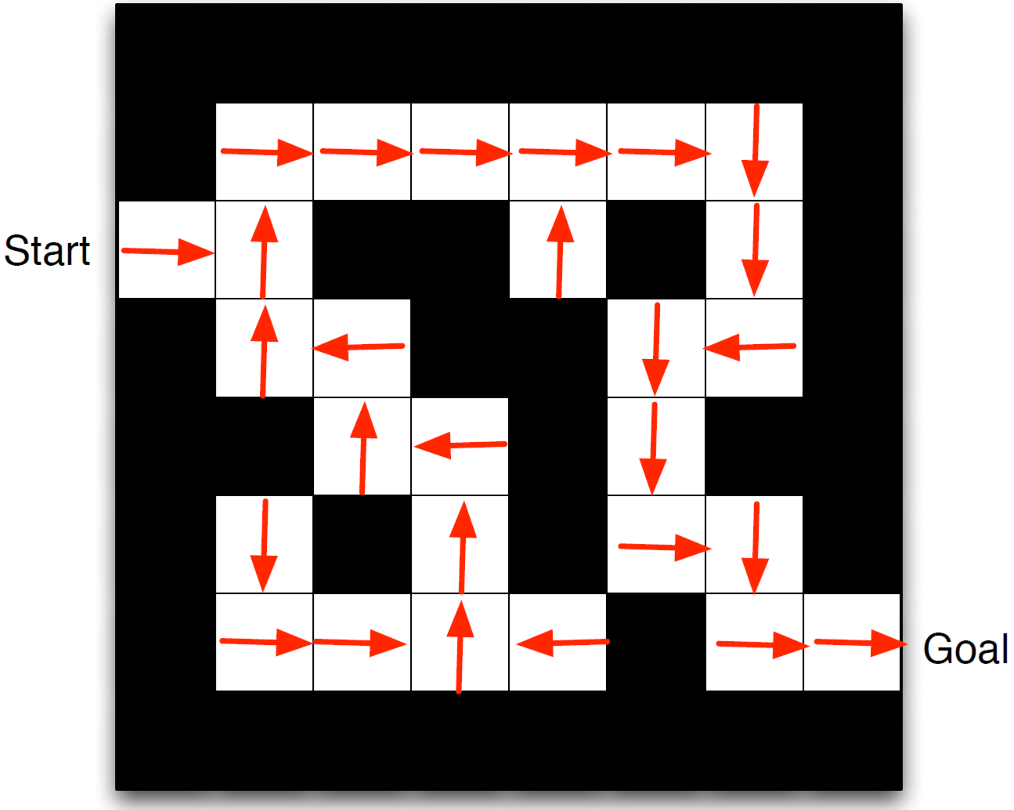
- **стратегия** (policy): функция поведения агента, обычно это отображение из множества состояний в множество действий (детерминированная стратегия $a = \pi(s)$, стохастическая стратегия $\pi(a|s) = \mathbb{P}[a_t = a|s_t = s]$),
- **функция полезности** (value function): оценка того, насколько полезно состояние и/или действие, это предсказание будущего вознаграждения: $V^\pi = \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots|s_t = s]$,
- **модель** (model): представление агента о среде (модель переходов $\mathcal{P}^a_{ss'} = \mathbb{P}[s_{t+1} = s'|s_t = s, a_t = a]$, модель вознаграждений $\mathcal{R}^a_s = \mathbb{E}[r_t|s_t = s, a_t = a]$)

Лабиринт



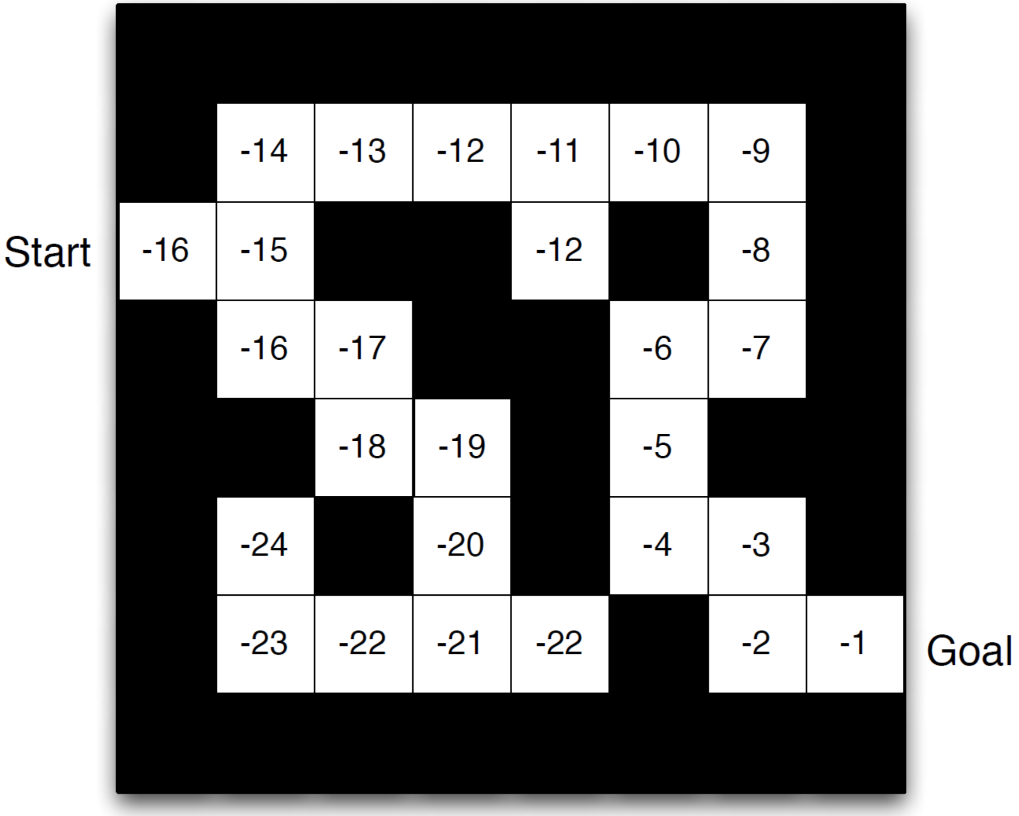
- Вознаграждения: -1 за каждый шаг
- Действия: $\uparrow, \leftarrow, \rightarrow, \downarrow$
- Состояния: местоположение агента

Лабиринт: стратегия



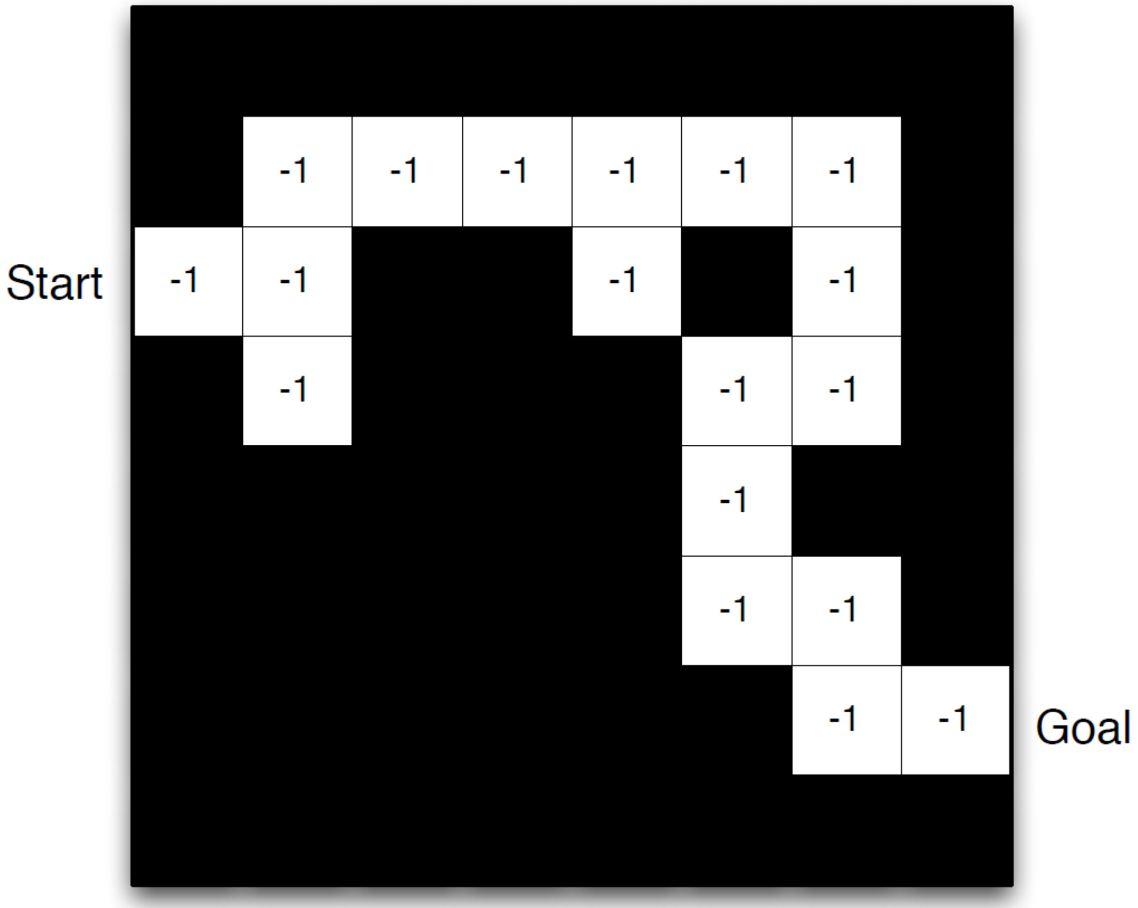
Стрелки отображают стратегию $\pi(s)$ для каждого состояния s

Лабиринт: функция полезности



Числа отображают полезность $V^{\pi}(s)$ для каждого состояния s

Лабиринт: модель



- Агент может обладать внутренней моделью среды
- Динамика: как действия меняют среду
- Вознаграждения: какое вознаграждение может быть получено в каждом состоянии
- Модель может быть неточной

- Клетки отображают модель переходов $\mathcal{P}^a_{ss'}$

Подзадачи в обучении с подкреплением

Планирование и обучение

В теории последовательного принятия решений существует две основных постановки задачи

- Обучение с подкреплением:
  - среда изначально неизвестна,
  - агент взаимодействует со средой,
  - агент оптимизирует свою стратегию
- Планирование поведения:
  - модель среды известна,
  - агент производит вычисления, используя свою модель без взаимодействия со средой,
  - агент оптимизирует стратегию,
  - является по сути вариантом поиска в соответствующем пространстве

Исследование и применение

- *Исследование (exploration)* – это процесс поиска новой информации о среде
- *Применение (explotation)* – это процесс использования найденной информации для максимизации вознаграждения
- Обычно важно как исследовать (среду), так и применять (знания)

Примеры:

- выбор ресторана: исследование – попробовать новый ресторан, применение – пойти в любимый ресторан,
- баннерная онлайн-реклама: исследование – показать новое объявление, применение – показывать наиболее привлекательное объявление,
- добыча нефти: исследование – пробурить новую скважину, применение – бурить в наилучшем известном месте,
- игры: исследование – сыграть экспериментальный ход, применение – сыграть ход, который кажется наилучшим

Марковское свойство и марковские процессы

Марковский процесс принятия решений

- *Марковский процесс принятия решений* (МППР, MDP) моделирует взаимодействие агенты и среды
- Предполагаем, что среда *полностью наблюдаема* (fully observable)
- Текущее состояние полностью характеризует весь процесс взаимодействия
- Почти все задачи обучения с подкреплением (RL) могут быть сведены к задаче с MDP:
  - оптимальное управление – MDP непрерывным множеством состояний и действий,
  - игровые автоматы – пример MDP с одним состоянием,
  - частично наблюдаемые среды (partially observable) могут быть сведены к MDP

Матрица переходов

Вероятность перехода для марковского состояния s в следующее состояние s' определяе
как

$$\mathcal{P}_{ss'} = \mathbb{P}[s_{t+1} = s' | s_t = s]$$

Матрица переходов $\mathcal{P}$ определяет вероятности переходов между всеми возможными состояниями:

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix}$$

Каждая строчка матрицы в сумме дает 1

Марковский процесс

Марковский процесс (Markov process, Markov chain) – это случайный процесс без памяти, т.е. последовательность случайных состояний $s_1, s_2, \ldots$ с марковским свойством

Definition

Марковский процесс (или марковская цепь) – это двойка $\langle S, \mathcal{P} \rangle$, где

- S – (конечное) множество состояний,
- $\mathcal{P}$ – матрица переходов

$$\mathcal{P}_{ss'} = \mathbb{P}[s_{t+1} = s' | s_t = s]$$

Суммарное вознаграждение

Definition

Суммарное вознаграждение (*отдача*, return) – сумма дисконтированных вознаграждений с момента времени t:

$$R_t = r_{t+1} + \gamma r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- Дисконтирующий множитель $\gamma \in [0, 1]$ – оценка значений будущих вознаграждений
- Значение получаемого вознаграждения после $k + 1$ шагов – $\gamma^k r$
- Моментальное вознаграждение важнее отложенных будущих:
  - $\gamma \sim 0$ – близорукий агент,
  - $\gamma \sim 1$ – дальнозоркий агент

Функция полезности

Функция полезности $V(s)$ дает долгосрочную оценку отдачи, начиная с состояния s

Definition

Функция полезности $V(s)$ марковского процесса вознаграждения – это ожидаемая отдача, получаемая начиная с состояния s:

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

Уравнение Беллмана для полезности

Уравнение Беллмана

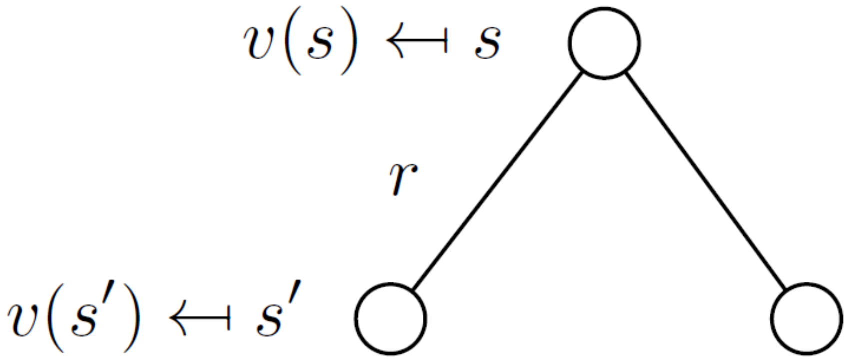
Функцию полезности можно представить в виде двух слагаемых:

- немедленное вознаграждение r_{t+1},
- дисконтированное значение следующего состояния $\gamma V(s_{t+1})$:

$$
\begin{aligned}
V(s) = & \mathbb{E}[R_t | s_t = s] = \\
& \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | s_t = s] = \\
& \mathbb{E}[r_{t+1} + \gamma (r_{t+2} + \gamma r_{t+3} + \ldots) | s_t = s] = \\
& \mathbb{E}[r_{t+1} + \gamma R_{t+1} | s_t = s] = \\
& \mathbb{E}[r_{t+1} + \gamma V(s_{t+1}) | s_t = s]
\end{aligned}
$$

Уравнение Беллмана

$$V(s) = \mathbb{E}[r_{t+1} + \gamma V(s_{t+1}) | s_t = s]$$



$$V(s) = \mathcal{R}_s + \gamma \sum_{s' \in S} \mathcal{P}_{ss'} V(s')$$

Уравнение Беллмана в матричной форме

Мы можем записать уравнение Беллмана в матричной форме:

$$V = \mathcal{R} + \gamma \mathcal{P} V,$$

где V – это вектор-колонка с одной компонентной на состояние:

$$\begin{bmatrix} V(1) \\ \vdots \\ V(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}(1) \\ \vdots \\ \mathcal{R}(n) \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} V(1) \\ \vdots \\ V(n) \end{bmatrix}$$

Решение уравнения Беллмана

- Уравнение Беллмана – это линейное уравнение (система линейных уравнений).
- Аналитическое решение:

$$
\begin{aligned}
V &= \mathcal{R} + \gamma \mathcal{P} V \\
(I - \gamma \mathcal{P}) V &= \mathcal{R} \\
V &= (I - \gamma \mathcal{P})^{-1} \mathcal{R}
\end{aligned}
$$

- Вычислительная сложность $O(n^3)$, где n – число состояний.
- Аналитическое решение возможно только для небольших задач MDP.
- Но есть много итерационных приближенных методов для больших задач:
  - динамическое программирование,
  - Монте-Карло подход,
  - метод временных различий.

Марковский процесс принятия решений

Марковский процесс принятия решений – это марковский процесс вознаграждения для действий. Он описывает взаимодействие со средой с марковскими состояниями.

Definition

Марковский процесс принятия решений (МППР) – это кортеж $\langle S, A, \mathcal{P}, \mathcal{R}, \gamma \rangle$, где

- S – конечное множество состояний,
- A – конечное множество действий,
- $\mathcal{P}$ – матрица вероятностей переходов:

$$\mathcal{P}^a_{ss'} = \mathbb{P}[s_{t+1} = s' | s_t = s, a_t = a],$$

- $\mathcal{R}$ – функция вознаграждения $\mathcal{R}^a_s = \mathbb{E}[r_{t+1} | s_t = s, a_t = a]$,
- $\gamma \in [0, 1]$ – дисконтирующий множитель.

Стратегии

Definition

Стратегией π будем называть вероятностное распределение на множестве действий при текущем состоянии s:

$$\pi(a|s) = \mathbb{P}[a_t = a|s_t = s]$$

- Стратегия полностью определяет поведение агента.
- МППР стратегии зависят от текущего состояния.
- Стратегии стационарны (не зависят от времени):

$$a_t \sim \pi(\cdot|s_t), \forall t > 0.$$

Вопросы?