Методы машинного обучения

Лекция 10

Обучение с подкреплением

Обучение с подкреплением

Обучение с подкреплением *(reinforcement learning)* - способ машинного обучения, при котором испытуемая система (*агент*) обучается, взаимодействуя с некоторой средой. Роль объектов играют пары «ситуация, принятое решение», ответами являются значения функционала качества, характеризующего правильность принятых решений (реакцию среды). При обучении с подкреплением, в отличии от обучения с учителем, не предоставляются верные пары "входные данные - ответ", а принятие субоптимальных решений (дающих локальный экстремум) не ограничивается явно.

Обучение с подкреплением пытается найти компромисс между исследованием неизученных областей и применением имеющихся знаний (**<u>*exploration vs exploitation*</u>**).

Примеры прикладных задач: формирование инвестиционных стратегий, автоматическое управление технологическими процессами, самообучение роботов, и т.д.

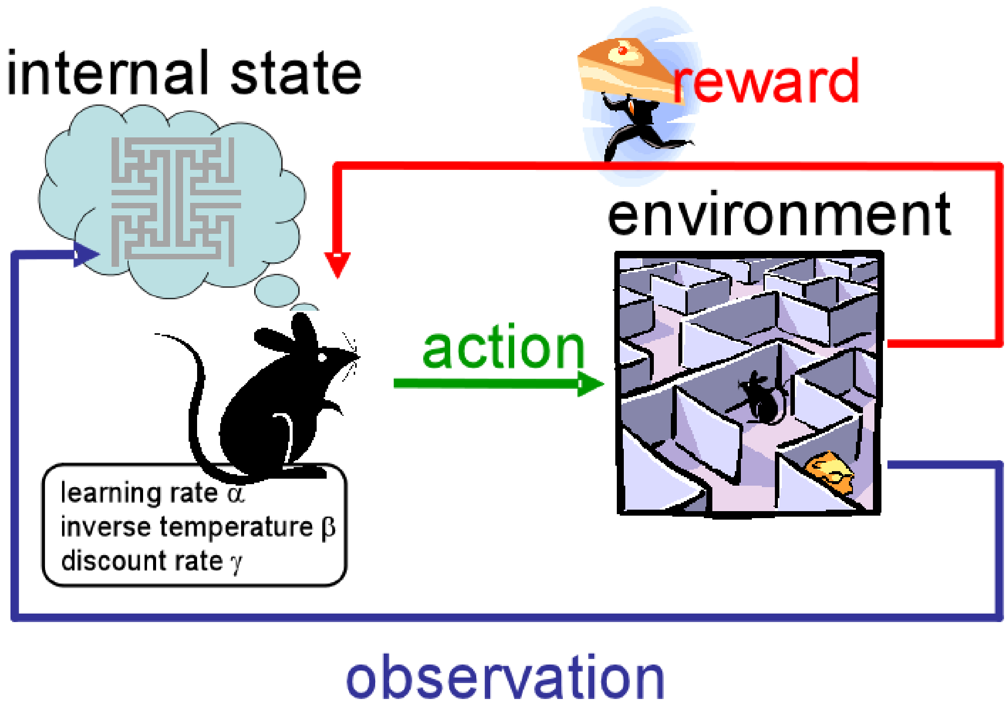
Осуществляется поиск субоптимальных решений или стратегий того, как агент должен действовать в окружении, чтобы максимизировать некоторый долговременный выигрыш.

> Маленький мальчик заходит в парикмахерскую. Парикмахер сразу же его узнаёт и говорит своим клиентам: «Смотрите, это самый глупый мальчик среди всех на свете! Сейчас я вам докажу!». В одной руке парикмахер держит доллар, в другой 25 центов. Зовёт мальчика, тот подходит и выбирает 25 центов. Все смеются, мальчик уходит. По дороге обратно, мальчика догоняет один из смеявшихся и спрашивает:
> — А почему всё-таки ты выбрал 25 центов, а не 1 доллар?
> — Потому что в тот день, когда я выберу 1 доллар, игра будет окончена.

Среда и агент

В обучении с подкреплением существует агент (***agent***), который взаимодействует с окружающей средой (***environment***), предпринимая действия (***actions***). Окружающая среда дает награду (***reward***) за эти действия, а агент продолжает их предпринимать (обучение методом проб и ошибок).

В искусственном интеллекте под термином **интеллектуальный агент** понимаются сущности, получающие информацию через систему сенсоров о состоянии управляемых ими процессов и осуществляющие влияние на них через систему актуаторов, при этом их реакция рациональна в том смысле, что процессы выполняемые ими содействуют достижению определённых параметров.

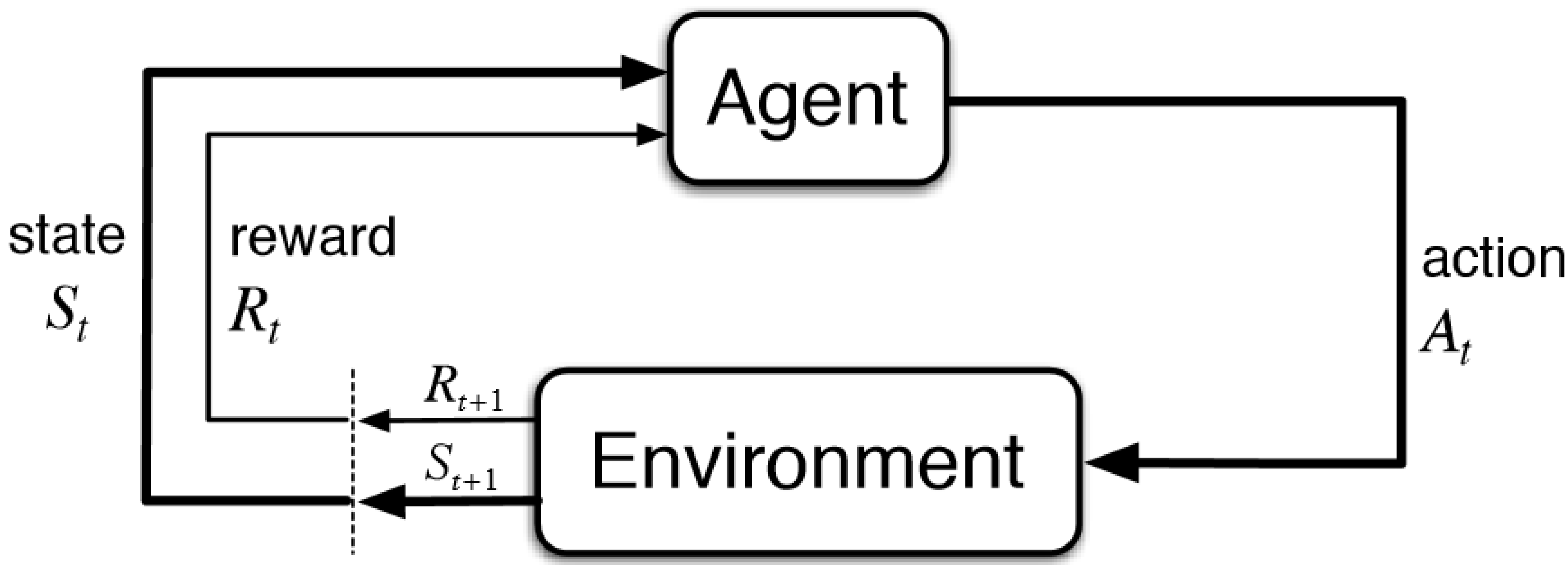


Необходимые термины в Reinforcement Learning

- **Агент (agent):** Наша система, которая выполняет действия в среде, чтобы получить некоторую награду.
- **Среда (environment, e):** сценарий/окружение, с которым должен взаимодействовать агент.
- **Награда (reward, R):** немедленный возврат, который предоставляется агенту, после выполнения определенного действия или задачи. Является положительной или отрицательной, как было упомянуто выше.
- **Состояние (state, s):** Состояние относится к текущему положению, возвращаемому средой.
- **Политика (policy, π):** стратегия, которая применяется агентом для принятия решения о следующем действии на основе текущего состояния.
- **Стоимость (value, V):** награда, которая ожидается в долгосрочной перспективе. По сравнению с краткосрочным вознаграждением, принимаем во внимание скидку (discount).
- **Функция полезности состояния (value function):** определяет размер переменной, которой является общая сумма награды.
- **Модель среды (Model of the environment):** имитатор поведения окружающей среды (демо-версия модели). Помогает определить, как будет вести себя среда.
- **Значение Q или значение полезности действия (Q):** значение Q очень похоже на value (V). Но главное различие между ними в том, что он принимает дополнительный параметр в качестве текущего действия.

Простейшая постановка задачи

- 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.



Пример взаимодействия среды и агента

- Взаимодействие:
  – Среда: Агент, ты в состоянии №1. Есть 5 возможных действий.
  – Агент: Выбираю действие 2.
  – Среда: Вознаграждение 2 единицы. Новое состояние № 5. Есть 2 возможных действия.
  – Агент: Выбираю действие 1.
  – Среда: Вознаграждение -5 единиц. Новое состояние № 1. Есть 5 возможных действий.
  – Агент: Выбираю действие 4.
  – Среда: Вознаграждение 14 единиц. Новое состояние № 3. Есть 3 возможных действия.

- Результат:

  Агент успел вернуться в состояние 1 и исследовать ранее незнакомую опцию 4, получив за это существенную награду.

Exploration vs Exploitation

- Каждый алгоритм должен и изучать окружающую среду, и пользоваться своими знаниями, чтобы максимизировать прибыль.
- Та или иная стратегия может быть хороша, но вдруг она не оптимальная? Как достичь оптимального соотношения между исследованием нового и использованием имеющихся знаний?
- Обучение с подкреплением, как раз пытается найти компромисс между исследованием неизученных областей и применением имеющихся знаний, т.е. ***exploration vs exploitation*.** Эта проблематика всегда присутствует в обучении с подкреплением.

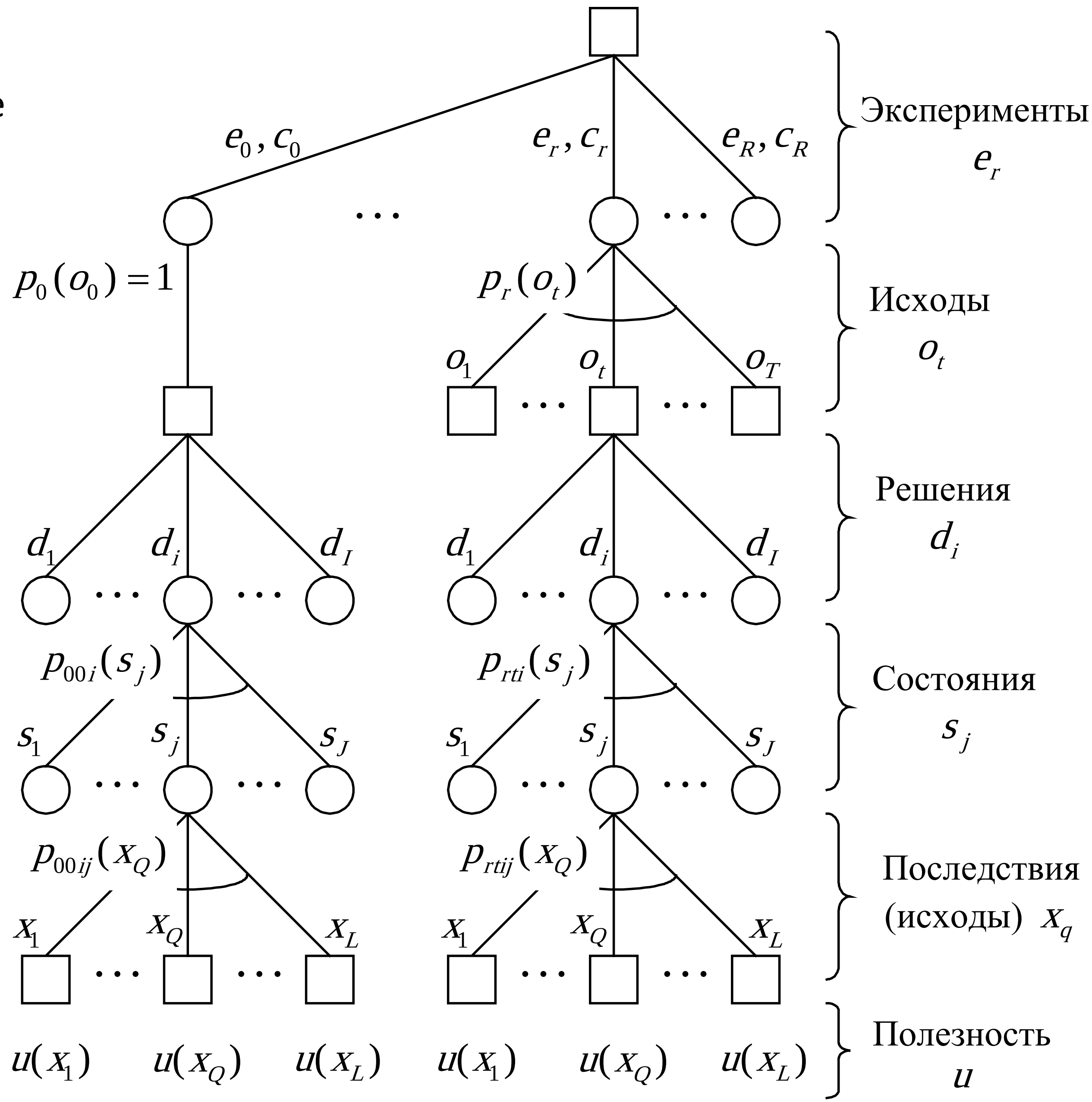
Дерево принятия решений

Представление задачи принятия решений в виде дерева решений.

Два типа узлов:

- узлы решений, обозначенные квадратами;
- узлы возможностей, обозначенные окружностями.

Анализ дерева решений осуществляется снизу вверх, используя принцип максимизации ожидаемой полезности.



Анализ дерева решений

В узлах возможностей с помощью полученного для данного узла распределения вероятностей вычисляется ожидаемая полезность. Для любого узла решений лицо, принимающее решение (ЛПР), выбирает альтернативу, которая приводит к наибольшей ожидаемой полезности, и приписывает полученную полезность узлу решений.

Так обозначим через $\bar{u}_{rtij}$ ожидаемую полезность проведенного эксперимента e_r при наблюдаемом исходе o_t, выбранном решении d_i и внешних условиях s_j, а $\bar{u}_{rtijl}$ является функцией полезности последствий $u(x_l)$, где индекс *l* обозначает соответствующее последствие.

Для дискретных задач:

$$\bar{u}_{rtij} = \sum_x u(x_l) p_{rtij}(x_l).$$

Аналогично ожидаемая полезность выбранного эксперимента e_r, наблюдаемого исхода o_t и выбранного решения d_i равна:

$$\bar{u}_{rti} = \sum_s \bar{u}_{rtij} p_{rti}(s_j).$$

В узле решений выбирается решение d_i, приводящее к максимальной ожидаемой полезности. Следовательно:

$$\bar{u}_{rt} = \max_{d_i}(\bar{u}_{rti}).$$

Выражение для ожидаемой полезности эксперимента e_r:

$$\bar{u}_r = \sum_o \bar{u}_{rt} p_r(o_t).$$

Наилучшим является эксперимент e_{r*}, который позволяет получить максимальное значение ожидаемой полезности из соотношения:

$$\bar{u}_{r*} = \max_{e_r}(\bar{u}_r).$$

Пусть выбран эксперимент $r*$ и реализовался исход o_t; тогда оптимальное решение d_{i*} определяется с помощью выражения:

$$\bar{u}_{r*ti*} = \max_{d_i}(\bar{u}_{r*ti}).$$

Любую задачу принятия решений можно представить последовательностью узлов решения и узлов возможностей.

Марковские процессы

Марковский процесс — случайный процесс, эволюция которого после любого заданного значения временного параметра t *не зависит* от эволюции, предшествовавшей t, при условии, что значение процесса в этот момент фиксировано («будущее» процесса зависит от «прошлого» лишь через «настоящее»).

Марковское свойство — в теории вероятностей и статистике термин, который относится к памяти случайного процесса.

Стохастический процесс обладает марковским свойством, если условное распределение вероятностей будущих состояний процесса зависит только от нынешнего состояния, а не от последовательности событий, которые предшествовали этому. Процесс, обладающий этим свойством, называется марковским процессом.

Для марковских цепей с дискретным временем. В случае, если *S является* дискретным множеством состояний и время дискретно, то марковское свойство может быть сформулировано следующим образом:

$$\mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \dots, X_0 = x_0) = \mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1})$$

Марковский процесс принятия решений (МППР) *Markov decision process (MDP)*

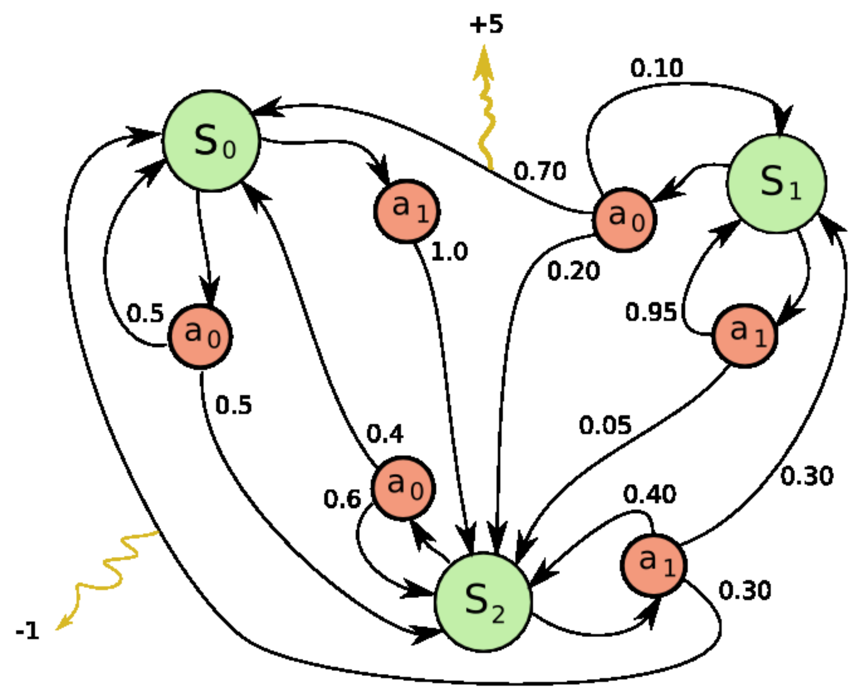
Спецификация задачи последовательного принятия решений для полностью наблюдаемой среды с марковской моделью перехода и дополнительными вознаграждениями. Слово марковский в названии отражает выполнение марковского свойства для таких процессов. Такой процесс служит математической основой для моделирования последовательного принятия решений в ситуациях, где результаты частично случайны и частично под контролем лица, принимающего решения.

Эта спецификация используется во множестве областей, включая робототехнику, автоматизированное управление, экономику, производство, а также в качестве основы обучения с подкреплениями.

МППР - Определение

Чтобы определить марковский процесс принятия решений, нужно задать 4-кортеж $(S, A, P_{\cdot}(\cdot,\cdot), R_{\cdot}(\cdot,\cdot))$, где:

- S - конечное множество состояний;
- A - конечное множество действий (часто представляется в виде множеств A_s, действий доступных из состояния s);
- $P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ - вероятность, что действие a в состоянии s во время t приведет в состояние s'ко времени $t+1$;
- $R_a(s, s')$ - вознаграждение, получаемое после перехода в состояние s'из состояния s, при совершении действия a.



Стратегия $\boldsymbol{\pi}$ — функция (в общем случае распределение вероятностей), сопоставляющая состоянию действие. Такой Марковский процесс принятия решений можно рассматривать, как Марковскую цепь.

МППР - Цель оптимизации

Решить марковский процесс принятия решений означает найти стратегию, максимизирующую "вознаграждение" (функцию ценности) - оптимальную стратегию. Самая простая функция ценности это математическое ожидание формального ряда:

$$E\left[\sum_{t=0}^{\infty} R_{a_t}(s_t, s_{t+1})\right]$$

где $a_t = \pi(s_t)$, а математическое ожидание берётся в соответствии с $s_{t+1} \sim P_{a_t}(s_t, \cdot)$.

Такую функцию можно использовать если гарантируется, что ряд сходится, а значит наличие терминального состояния, где $P_a(s, s) = 1$ и $R_a(s, s) = 0$.

Если же сходимость ряда не гарантируется, то обычно делают одно из двух:

- Рассматривают только конечное число слагаемых:

$$E\left[\sum_{t=0}^{N} R_{a_t}(s_t, s_{t+1})\right]$$

- Вводят коэффициент обесценивания (дисконтирования) $\gamma \in [0,1]$:

$$E\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

На практике второй вариант более гибкий, так как учитывает более долгосрочную перспективу и чаще используется именно он.

МППР - Функции полезности

Для максимизации ряда $E\left[\sum_{t=0}^{\infty} R_{a_t}(s_t, s_{t+1})\right]$ вводят две функции полезности:

- Функция полезности состояния:

$$V_\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) | s_0 = s, a_t = \pi(s_t)\right]$$

- Функция полезности действия:

$$Q_\pi(s, a) = E\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) | s_0 = s, a_0 = a, a_t = \pi(s_t) \forall t \geq 1\right]$$

где математическое ожидание берётся в соответствии с $s_{t+1} \sim P_{a_t}(s_t, \cdot)$.

А также их максимумы по всем стратегиям:

$$V_*(s) = \max_\pi V_\pi(s) \text{ и } Q_*(s, a) = \max_\pi Q_\pi(s, a)$$

Можно доказать, что эти функции также являются функциями полезности состояния и полезности действия соответственно, а также, что они достигаются на детерминированной стратегии. Заметим, что по функции Q_* можно восстановить её стратегию, которая будет оптимальной.

Для определения оптимальной стратегии используется отношение порядка на множестве стратегий.

$$\pi_1 \preccurlyeq \pi_2 \Leftrightarrow \forall V_{\pi_1}(s) \leq V_{\pi_2}(s), s \in S$$

Наибольшая стратегия называется оптимальной.

Постановка задачи обучения с подкреплением

Составные части:

- множество состояний среды (*states*) *S*;
- множество действий (*actions*) *A*;
- множество вещественнозначных скалярных "выигрышей" (*rewards*) *R*.

Игра агента со средой:

- инициализация стратегии $\pi_1(a|s)$ и состояния среды s_1
- для всех $t = 1 \ldots T$:
  - агент выбирает действие $a_t \sim \pi_1(a|s_t)$
  - среда генерирует награду $r_{t+1} \sim p(r|a_t, s_t)$ и новое состояние $s_{t+1} \sim p(s|a_t, s_t)$
  - агент корректирует стратегию $\pi_{t+1}(a|s)$

Таким образом в произвольный момент времени t агент характеризуется состоянием $s_t \in S$ и множеством возможных действий $A(s_t)$. Выбирая действие $a \in A(s_t)$, он переходит в состояние s_{t+1} и получает выигрыш r_{t+1}. Основываясь на таком взаимодействии с окружающей средой, агент, обучающийся с подкреплением, должен выработать стратегию $\pi: S \to A$, которая максимизирует величину:

- $R = r_0 + r_1 + \cdots + r_n$ в случае марковского процесса принятия решений (МППР), имеющего терминальное состояние;
- $R = \sum_t \gamma^t r_t$ для МППР без терминальных состояний (где $0 \leq \gamma \leq 1$ — дисконтирующий множитель для "предстоящего выигрыша").

Марковское свойство (МППР):

$$P(s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots, s_1, a_1) = \mathrm{P}(s_{t+1} = s', r_{t+1} = r | s_t, a_t)$$

МППР называется финитным, если $|A|<\infty$, $|S|<\infty$.

Таким образом, обучение с подкреплением особенно хорошо подходит для решения задач, связанных с выбором между долгосрочной и краткосрочной выгодой.

Подход к решению

Наивный подход к решению этой задачи подразумевает следующие шаги:

- опробовать все возможные стратегии;
- выбрать стратегию с наибольшим ожидаемым выигрышем.

Проблемы:

- количество доступных стратегий может быть велико или бесконечно;
- выигрыши стохастические — чтобы точно оценить выигрыш от каждой стратегии потребуется многократно применить каждую из них.

Решения:

- оценка функций полезности;
- прямая оптимизация стратегий.

Подход с использованием функции полезности использует множество оценок ожидаемого выигрыша только для одной стратегии π (либо текущей, либо оптимальной). При этом пытаются оценить либо ожидаемый выигрыш, начиная с состояния s, при дальнейшем следовании стратегии π,

$$V(s) = E[R|s, \pi],$$

либо ожидаемый выигрыш, при принятии решения a в состоянии s и дальнейшем соблюдении π,

$$Q(s, a) = E[R|s, \pi, a].$$

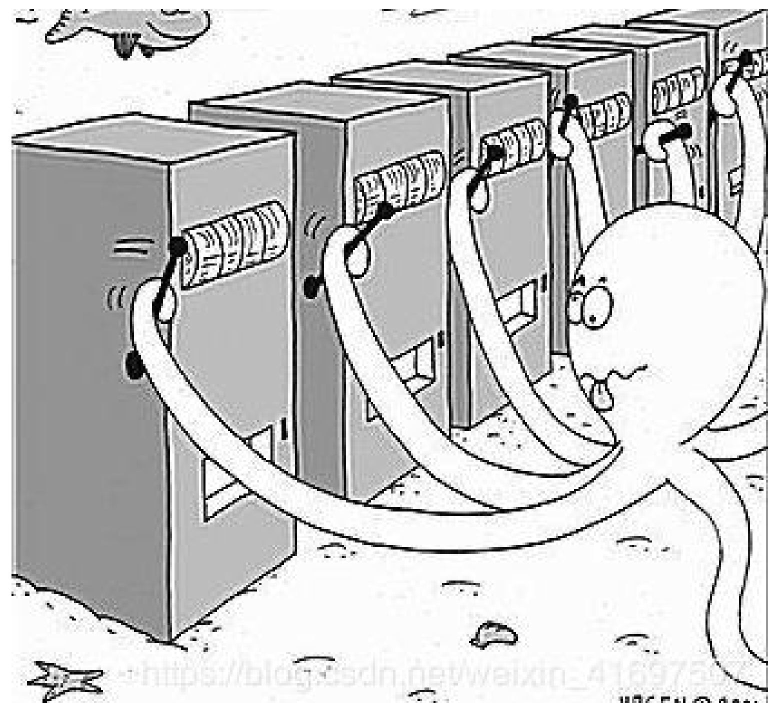
Задача о многоруком бандите
(The multi-armed bandit problem)

- Агенты с одним состоянием, т.е. состояние агента не меняется. У него фиксированный набор действий и возможность выбора из этого набора действий.
- Модель: агент в комнате с несколькими игровыми автоматами. У каждого автомата своё ожидание выигрыша.
- Нужно заработать побольше:

  Exploration vs. Exploitation

  (разведка против эксплуатации).
- Жадные и ϵ-жадные стратегии

  (greedy & ϵ-greedy)



Задача о многоруком бандите Формулировка

A — множество возможных *действий* (ручек автомата),
$p_a(r)$ — неизвестное распределение *награды* $r \in R \ \forall a \in A$,
$\pi_t(a)$ — *стратегия* агента в момент t $\forall a \in A$.

Игра агента со средой:

- инициализация стратегии $\pi_1(a)$;
- для всех $t = 1 \dots T$:
  - агент выбирает действие (ручку) $a_t \sim \pi_t(a)$;
  - среда генерирует награду $r_t \sim p_{a_t}(r)$;
  - агент корректирует стратегию $\pi_{t+1}(a)$.

Средняя награда в t играх: $Q_t(a) = \frac{\sum_{i=1}^{t} r_i [a_i = a]}{\sum_{i=1}^{t} [a_i = a]} \to max$,

Ценность действия a: $Q^*(a) = \lim\limits_{t \to \infty} Q_t(a) \to max$

N-рукий бандит - на каждом шаге выбираем за какую из *N* ручек автомата дернуть. Полагаем, что каждому действию соответствует некоторое распределение, которое не меняется со временем. Если распределения известны, то стратегия заключается в том, чтобы подсчитать математическое ожидание для каждого из распределений, выбрать действие с максимальным математическим ожиданием и далее совершать это действие на каждом шаге. Проблема, что распределения неизвестны, однако можно оценить математическое ожидание случайной величины $\boldsymbol{\xi}$ с неизвестным распределением.

$$E(\xi) = \frac{1}{K} \sum_{k=1}^{K} \xi_k$$

Жадная (greedy) стратегия

Начальные значения:

- $P_a = 0$ для $\forall a \in \{1 \dots N\}$ — сколько раз было выбрано действие a,
- $Q_a = 0$ для $\forall a \in \{1 \dots N\}$ — текущая оценка математического ожидания награды для действия a

На каждом шаге t:

- Выбираем действие с максимальной оценкой математического ожидания:

$$a_t = \operatorname*{argmax}_{a \in A} Q_a$$

- Выполняем действие a_t и получаем награду $R(a_t)$;
- Обновляем оценку математического ожидания для действия a_t:

$$P_{a_t} = P_{a_t} + 1$$

$$Q_{a_t} = Q_{a_t} + \frac{1}{P_{a_t}}\left(R(a_t) - Q_{a_t}\right)$$

В чем проблема?

Пусть у нас есть "двурукий" бандит. Первая ручка всегда выдаёт награду равную 1, вторая всегда выдаёт 2. Действуя согласно жадной стратегии мы дёрнем в начале первую ручку, так как в начале оценки математических ожиданий равны нулю, увеличим её оценку до $Q_1 = 1$. В дальнейшем всегда будем выбирать первую ручку, а значит на каждом шаге будем получать на 1 меньше, чем могли бы.

ϵ-жадная (ϵ-greedy) стратегия

Введем параметр $\epsilon \in (0,1)$.

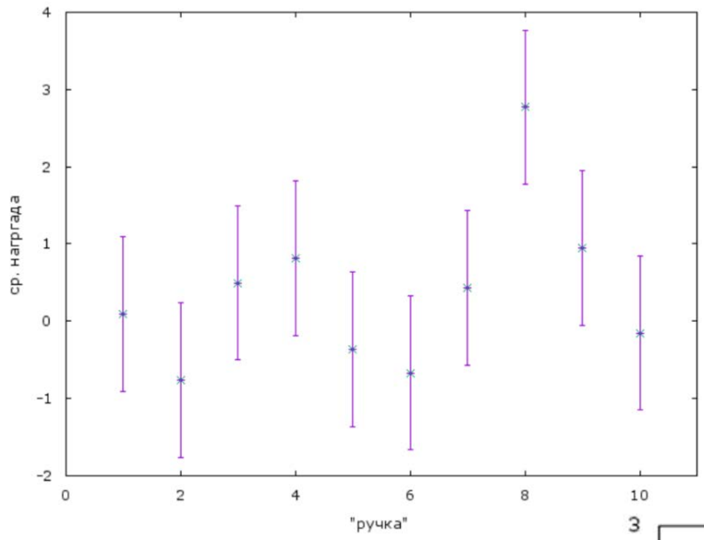
На каждом шаге *t:*

- Получим значение a - случайной величины равномерно распределенной на отрезке (0,1);
- Если $a \in (0, \epsilon)$, то выберем действие $a_t \in A$ случайно и равновероятно, иначе как в жадной стратегии выберем действие с максимальной оценкой математического ожидания;
- Обновляем оценки так же как в жадной стратегии.

Если $\epsilon = 0$, то это обычная жадная стратегия. Однако если ϵ>0, то в отличии от жадной стратегии на каждом шаге с вероятностью ϵ происходит "исследование" случайных действий.

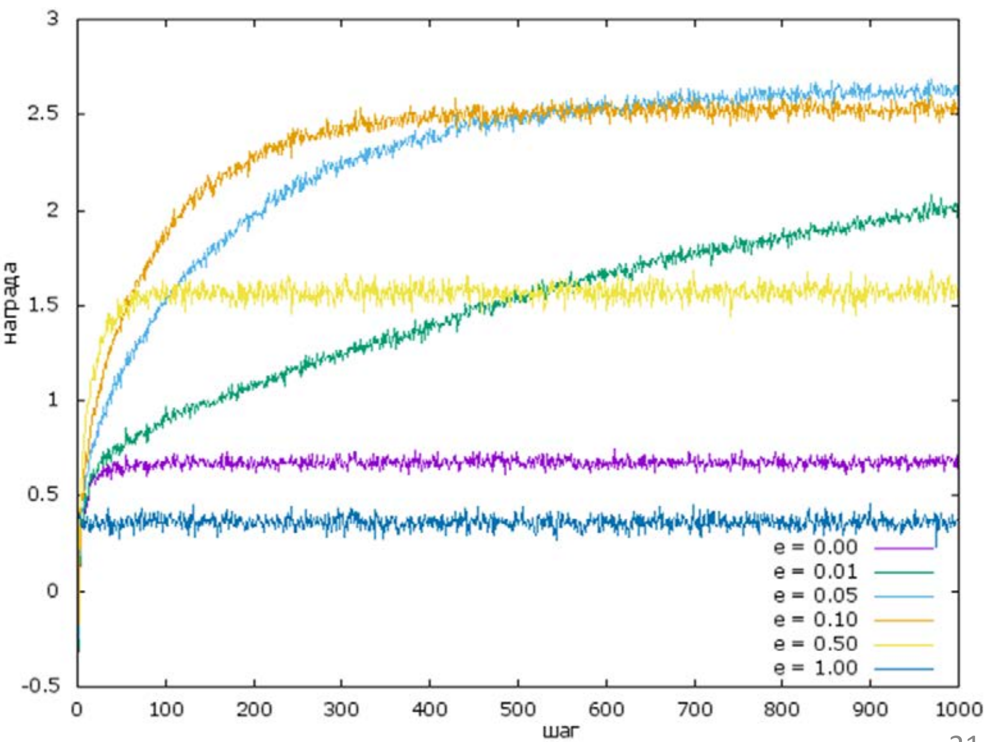
Пример: ϵ-жадная стратегия

Ручка	Ср. награда
1	0.089668
2	-0.757752
3	0.497168
4	0.811979
5	-0.367975
6	-0.666402
7	0.432601
8	2.769230
9	0.943522
10	-0.151123



Рассмотрим 10-рукого бандита. Выберем 10 случайных нормально распределенных чисел с центром в нуле и единичной дисперсией: $E = \{E_a | a = 1, \dots, 10\}$. Каждой ручке поставим соответствие нормальное распределение с математическим ожиданием из E и дисперсией 1.

- ϵ = 1 - худший результат, т.е. ручка выбирается случайно и равновероятно.
- ϵ = 0 - “жадная” стратегия находится на предпоследнем месте.
- ϵ = 0.5 - лучше предыдущих, но тратить половину ходов, выбирая ручку случайно; начиная с некоторого момента полученная награда стабилизируется не в максимальном значении.
- ϵ = 0.01 - растет слишком медленно на начальном этапе (слишком мало исследований), вероятно догонит варианты с ϵ=0.05 и ϵ=0.1, но для этого ей надо больше времени.
- ϵ = 0.05 и ϵ = 0.1 - работают вполне не плохо. Правда достигнуть среднего 2.76923 (т.е. собственно того, которое будет, если дергать только 8-ю ручку) ни та, ни другая не смогли.



Стратегия Softmax

Основная идея алгоритма *softmax* — уменьшение потерь при исследовании за счёт более редкого выбора действий, которые получали небольшую награду в прошлом. Чтобы этого добиться для каждого действия вычисляется весовой коэффициент на базе которого происходит выбор действия. Чем больше $Q_t(a)$, тем больше вероятность выбора a:

$$\pi_{t+1}(a) = \frac{\exp(Q_t(a)/\tau)}{\sum_{b \in A} \exp(Q_t(b)/\tau)}$$

$\tau \in (0, \infty)$ — параметр, с помощью которого можно настраивать поведение алгоритма.

При $\tau \to \infty$ стратегия стремится к равномерной, то есть softmax будет меньше зависеть от значения выигрыша и выбирать действия более равномерно (exploration).

При $\tau \to 0$ стратегия стремится к жадной, то есть алгоритм будет больше ориентироваться на известный средний выигрыш действий (exploitation).

Экспонента используется для того, чтобы данный вес был ненулевым даже у действий, награда от которых пока нулевая. Параметр τ имеет смысл уменьшать со временем.

Алгоритм верхнего доверительного интервала (*upper confidence bound* или UCB)

UCB — семейство алгоритмов, которые при выборе действия используют данные не только о среднем выигрыше, но и о том, насколько можно доверять значениям выигрыша.

Также как *softmax* в UCB при выборе действия используется весовой коэффициент, который представляет собой верхнюю границу доверительного интервала (upper confidence bound) значения выигрыша:

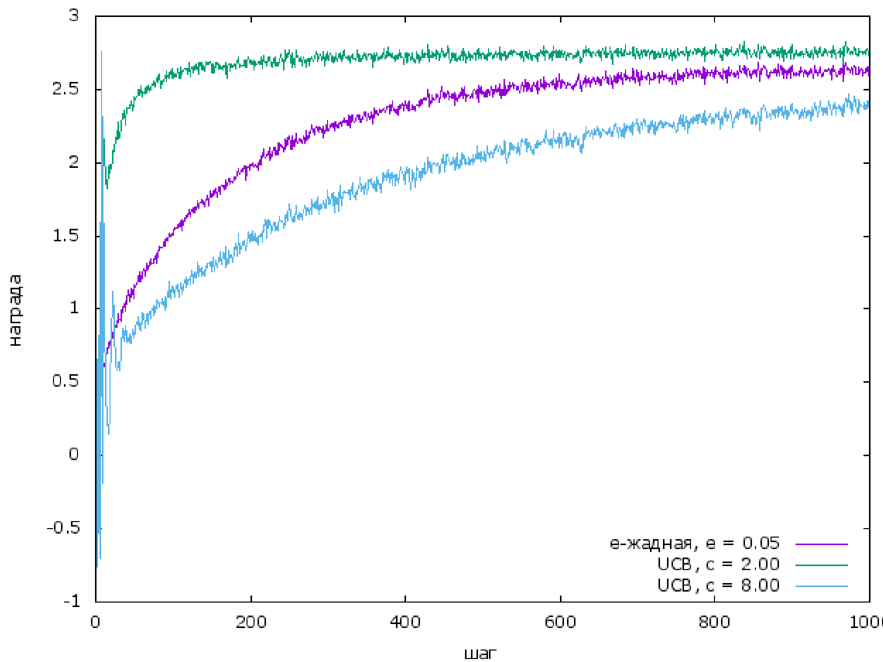
$$a_{t+1} = \underset{a=1,\dots,N}{\operatorname{argmax}} \{ Q_t(a) + b_a \}$$

где b_a — бонусное значение, которое показывает, насколько недоисследовано действие по сравнению с остальными:

$$b_a = c \cdot \sqrt{\frac{\ln(t)}{P_a}} \quad \text{или} \quad b_a = \sqrt{\frac{2 \cdot ln \sum_a P_a}{P_a}}$$

P_a - сколько раз было выбрано действие a;
$Q_t(a)$ - текущая оценка математического ожидания награды для действия a;
c – коэффициент настройки.

В начале работы алгоритма каждое из действий выбирается по одному разу (для того чтобы можно было вычислить размер бонуса для всех действий). После этого в каждый момент времени выбирается действие с максимальным значением весового коэффициента.



Q-обучение (Q-learning)

На основе получаемого от среды вознаграждения агент формирует функцию полезности Q, что впоследствии дает ему возможность уже не случайно выбирать стратегию поведения, а учитывать опыт предыдущего взаимодействия со средой. Преимущество Q-learning — способен сравнить ожидаемую полезность доступных действий, не формируя модели окружающей среды. Применяется для ситуаций, которые можно представить в виде МППР.

Таким образом, алгоритм это функция качества от состояния и действия:

$$Q : S \times A \to \mathbb{R}$$

Перед обучением Q инициализируется случайными значениями. После этого в каждый момент времени t агент выбирает действие a_t, получает награду r_t, переходит в новое состояние s_{t+1}, которое может зависеть от предыдущего состояния s_t и выбранного действия, и обновляет функцию Q. Обновление функции использует взвешенное среднее между старым и новым значениями:

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

где r_t- это награда, полученная при переходе из состояния s_t в состояние s_{t+1}, и α - скорость обучения $(0 < \alpha \leq 1)$.

Алгоритм заканчивается, когда агент переходит в терминальное состояние s_{t+1}.

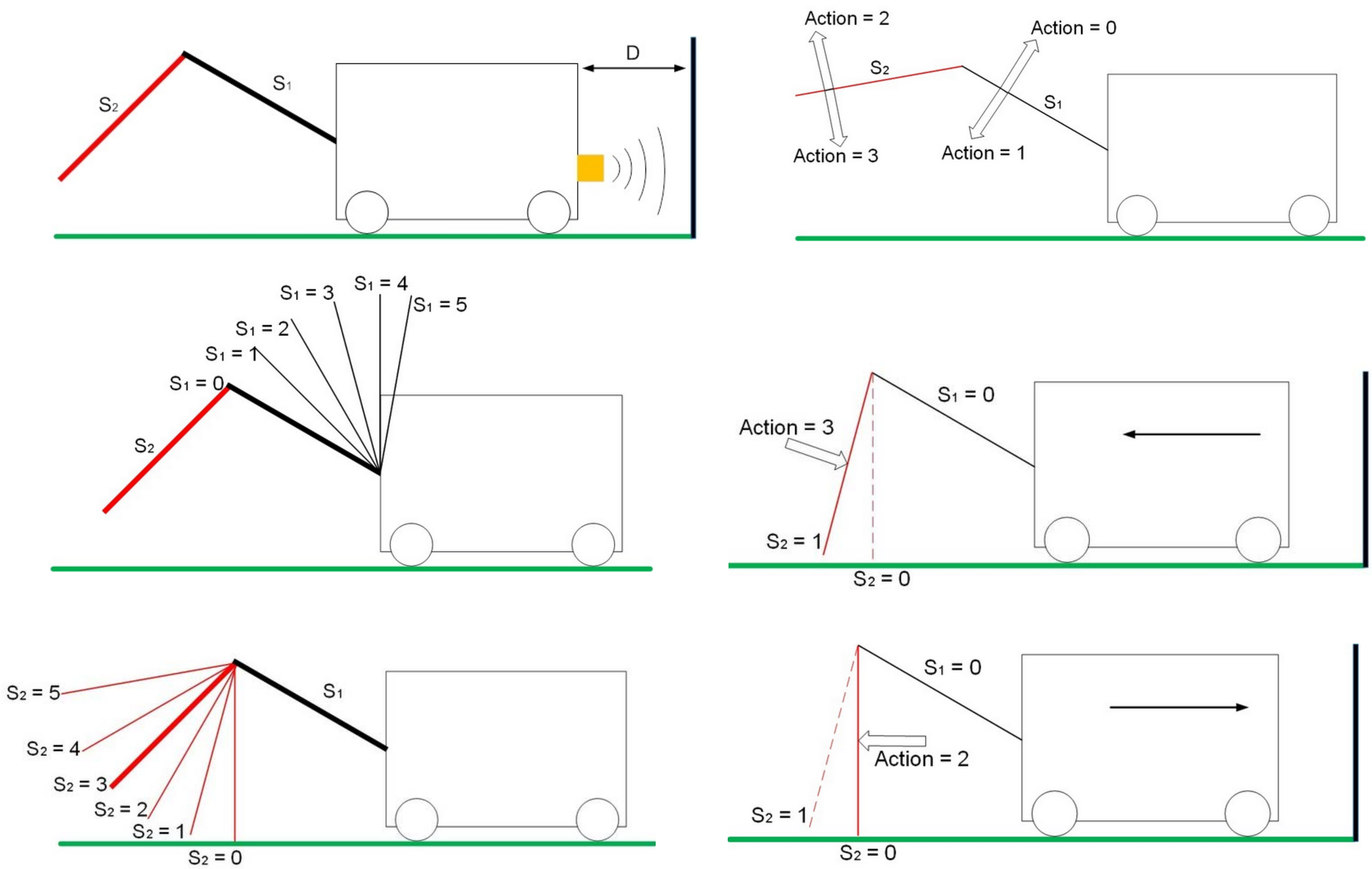
Алгоритм Q-learning

Обозначения:

- S — множество состояний; A — множество действий;
- $R = S \times A \to \mathrm{R}$ — функция награды; $T = S \times A \to S$ — функция перехода;
- $\alpha \in [0,1]$ — learning rate (обычно 0.1), чем он выше, тем сильнее агент доверяет новой информации;
- $\gamma \in [0,1]$ — discounting factor, чем он меньше, тем меньше агент задумывается о выгоде от будущих своих действий.

```
fun Q-learning(S, A, R, T, α, γ):
    for s ∈ S:
        for a ∈ A:
            Q(s, a) = rand()
    while Q is not converged:
        s = ∀s ∈ S
        while s is not terminal:
            π(s) = argmax_a Q(s, a)
            a = π(s)
            r = R(s, a)
            s' = T(s, a)
            Q(s', a) = (1 − α)Q(s', a) + α(r + γ max_{a'} Q(s', a'))
            s = s'
    return Q
```

Пример Q-learning – Тележка 1



Пример Q-learning – Тележка 2

Index	S1	S2	A0	A1	A2	A3
0	0	0	10	10	10	10
1	0	1	10	10	10	10
2	0	2	10	10	10	10
3	0	3	10	10	10	10
4	0	4	10	10	10	10
5	0	5	10	10	10	10
6	1	0	10	10	10	10
7	1	1	10	10	10	10
8	1	2	10	10	10	10
9	1	3	10	10	10	10
10	1	4	10	10	10	10
11	1	5	10	10	10	10

Index	S1	S2	A0	A1	A2	A3
0	0	0	0.617	10	0.332	10
1	0	1	1.16	10	1.06	1.46
2	0	2	1.13	10	1.13	1.14
3	0	3	1.09	10	1.08	1.09
4	0	4	1.06	10	1.05	1.06
5	0	5	1.03	10	10	1.04
6	1	0	0.796	0.704	0.823	10
7	1	1	0.971	1.1	1.05	1.09
8	1	2	1.07	1.07	1.07	1.07
9	1	3	1.08	1.07	1.06	1.06
10	1	4	1.04	1.05	1.05	1.06
11	1	5	1.05	1.03	10	1.04

Index	S1	S2	A0	A1	A2	A3
0	0	0	$\boxed{0.617}$	10	0.332	10
6	1	0	0.796	0.704	$\boxed{0.823}$	10
7	1	1	0.971	$\boxed{1.1}$	1.05	1.09
1	0	1	1.16	10	1.06	$\boxed{1.46}$

Index	S1	S2	A0	A1	A2	A3
0	0	0	$\boxed{0.617}$	10	0.332	10
6	1	0	0.796	0.704	$\boxed{0.823}$	10
7	1	1	0.971	$\boxed{1.1}$	1.05	1.09
1	0	1	1.16	10	1.06	$\boxed{1.46}$

Спасибо за внимание