Машинное обучение (Machine Learning)

Обучение с подкреплением (Reinforcement Learning)

Уткин Л.В.

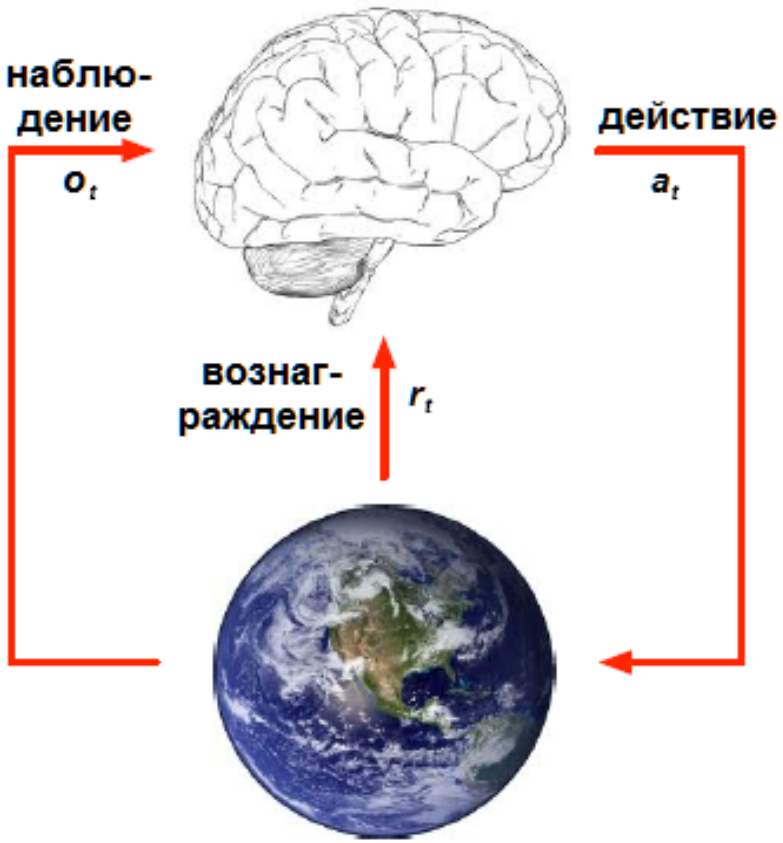
Санкт-Петербургский политехнический университет Петра Великого

Содержание

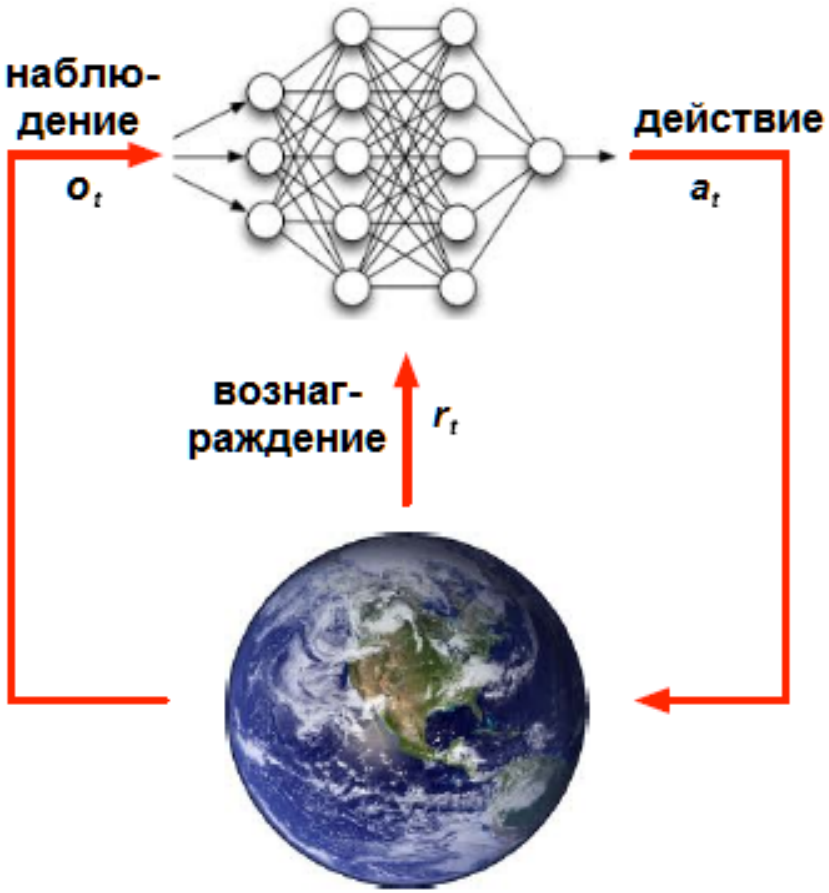
Постановка задачи

- 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 (история) - последовательность наблюдений, действий, вознаграждений

$$H_t = (o_1, r_1, a_1, o_2, r_2, a_2, ..., o_t, r_t)$$

- То, что случится далее, зависит от истории: - агент выберет действие, среда сгенерирует наблюдение и вознаграждение
- **Состояние** - функция опыта для определения, что произойдет далее

$$s_t = f(H_t)$$

- 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

- Вознаграждение: Положительное или отрицательное при переходе из одного состояния в другое $r(i, a, j)$ или $r_{i,j}(a)$ или $r_t(a)$
- Вознаграждение: Показывает, на сколько агент был успешен на шаге t
- Задача агента - максимизировать суммарное вознаграждение! Это главная цель обучения с подкреплением

Примеры вознаграждений

- *AlphaGo*: $+r$ - за победу в партии, $-r$ - за проигрыш
- *Atari*: $+r$ - за увеличение счета, $-r$ - за уменьшение счета
- *Робот*: $+r$ - за прямой шаг, $-r$ - за падение
- *Инвестиции*: $+r$ - за каждый заработанный рубль, $-r$ - за каждый потерянный рубль
- *Вертолет*: $+r$ - за следование намеченной траектории, $-r$ - за отклонение от нее

Строение RL агента

- Стратегия (policy) π: определяет действие, которое должно быть выбрано в каждом состоянии (отображение из состояния в действие), т.е. это функция поведения агента
- Функция полезности (value function) - оценка того, насколько полезно состояние
- Модель (model) - представление агента о среде

Стратегия

- Стратегия (policy) π: определяет действие, которое должно быть выбрано в каждом состоянии (отображение из состояния в действие), т.е. это функция поведения агента
  - детерминированная стратегия: $a = f(s)$
  - вероятностная стратегия: $\pi(a|s) = \Pr\{a_t = a|s_t = s\}$

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

Функция полезности

- Знаем уже: действие, состояние, вознаграждение, стратегия
- Value function, функция полезности $V^{\pi}(s)$ - предсказание будущего вознаграждения, используется для оценки состояния или ожидаемый долгосрочный доход с дисконтом
- По функции полезности происходит отбор действий

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \mid s_t = s\right]$$

Модель

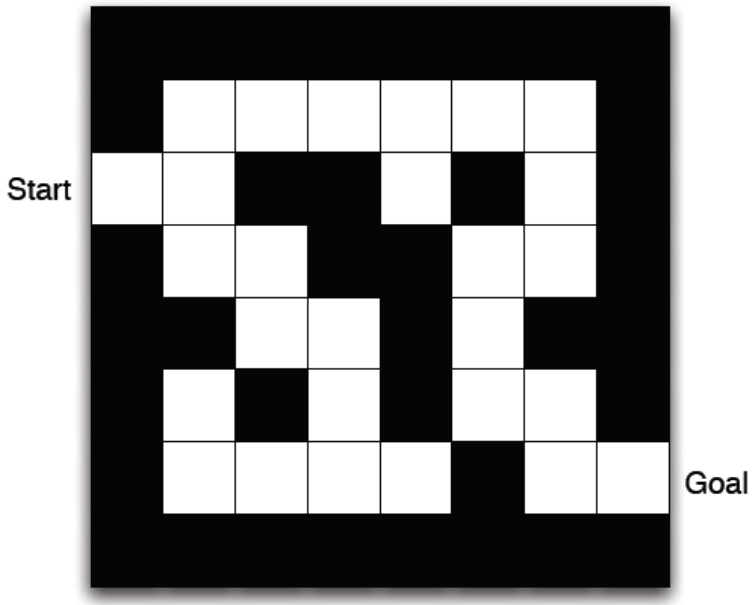
- Модель служит для предсказания того, что произойдет в среде в следующий момент времени
- Например
  - *модель переходов* $\mathcal{P}$ предсказывает следующее состояние

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

  - *модель вознаграждений* $\mathcal{R}$ предсказывает следующее вознаграждение

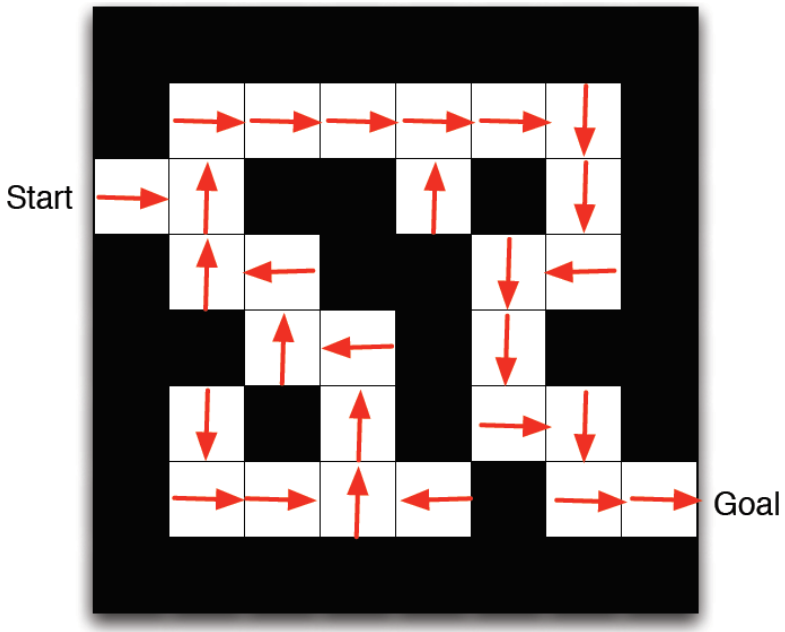
$$\mathcal{R}^a_s = \mathbb{E}\{r_t | s_t = s, a_t = a\}$$

Пример - лабиринт



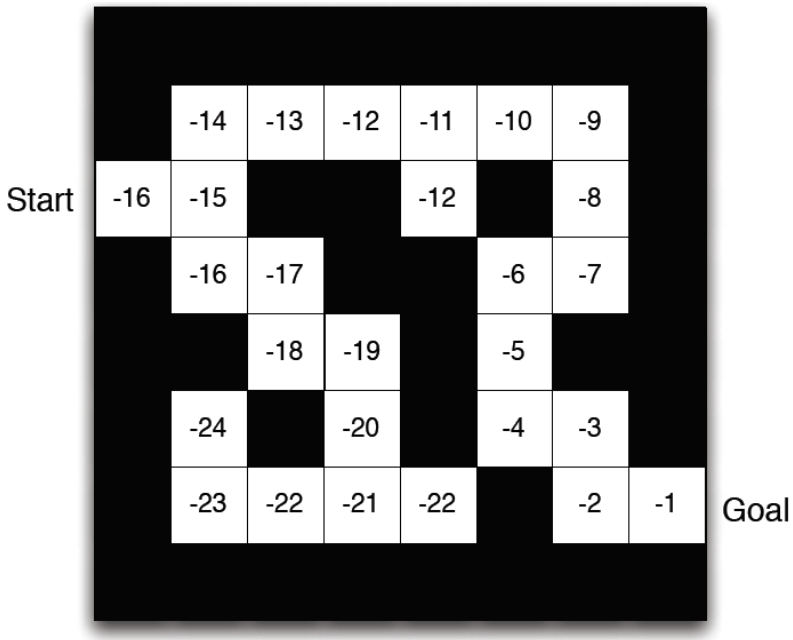
- Вознаграждение: -1 за каждый шаг
- Действия: N $\uparrow$,E $\rightarrow$,S $\downarrow$,W $\leftarrow$
- Состояния: местоположение агента (номер клетки)
- Найти кратчайший путь (чем меньше шагов, тем больше сумма вознаграждений)

Стратегия агента в лабиринте



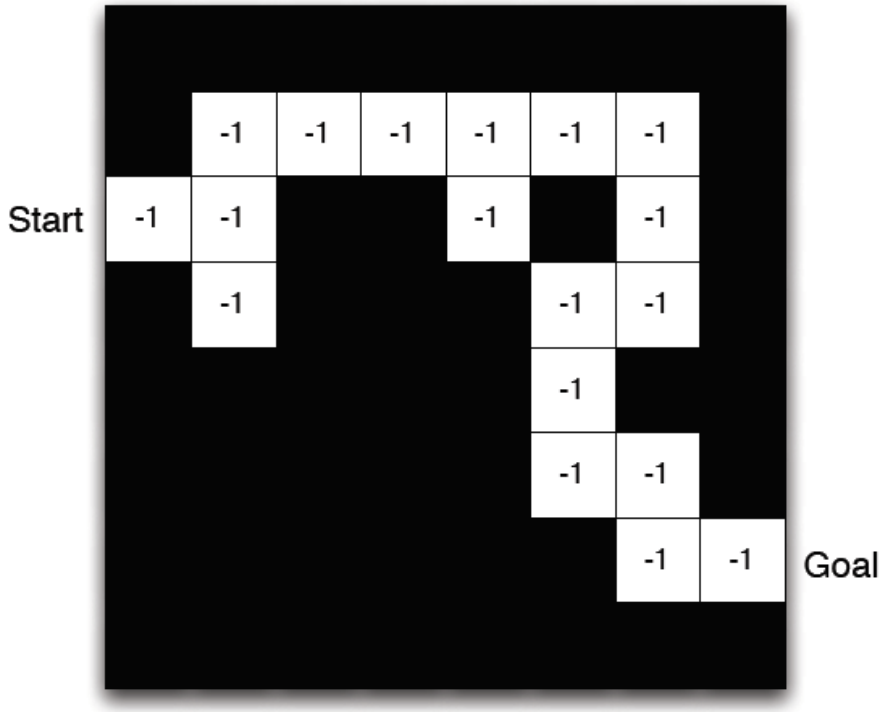
- Стрелки представляют собой стратегию $\pi(s)$ для каждого состояния s

Пример - функция полезности



- Числа - значение $V^{\pi}(s)$ для каждого состояния

Пример



- Макет сетки - модели переходов
- Числа - немедленное вознаграждение для каждого состояния - следующее вознаграждение $\mathcal{R}_s^a$ для каждого состояния s
- Клетки отображают модель переходов $\mathcal{P}_{s,s'}^a$

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” (локальный оптимум) 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

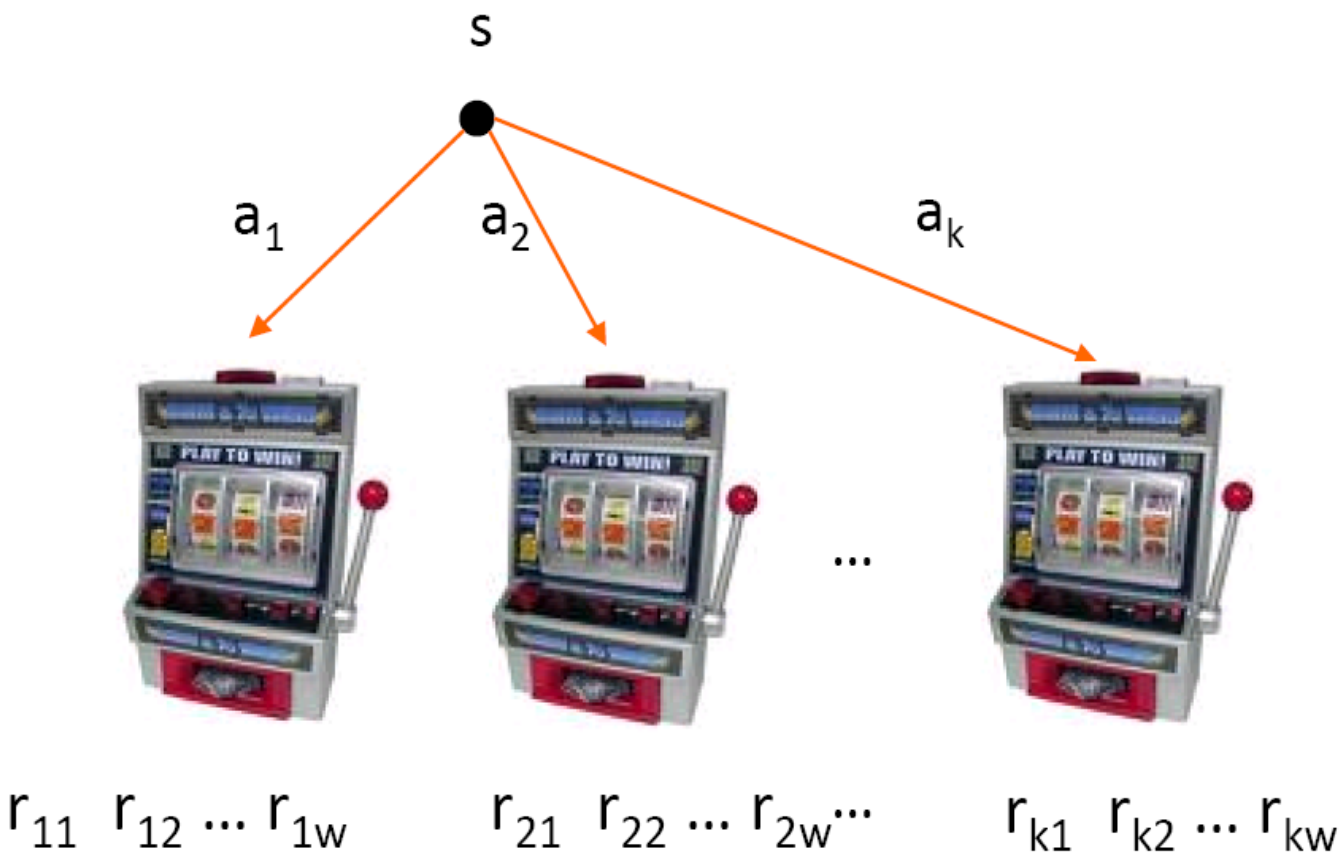
- Простая стратегия - мышь принимает самое известное действие большую часть времени (скажем, 80% времени), но иногда изучает новое, случайно выбранное направление, даже если оно уходит от известного вознаграждения.
- Эта стратегия - ε-жадная, где ε - это процент времени, когда агент принимает **случайное**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

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 - 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 $Q(a)$ (функция полезности действия)
- Агент вычисляет оценки $Q_t(a)$
- Стратегия, максимизирующая текущую оценку ценности $A_t \subseteq A$: $A_t = \arg\max_{a \in A} Q_t(a)$ (дергаем ручку с наибольшим средним)
- **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.

Проблемы

- Если на первом шаге ручка дала 0, то к ней уже не будет обращение

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

- Полезная эвристика - оптимизм при неопределенности
  - ε-жадная стратегия - с вероятностью ε дергаем случайную ручку - можем дернуть за любую ручку
  - доверительный интервал для средних (из распред. Стьюдента)

$$\left(\bar{x} - 1.96\frac{\sigma}{\sqrt{N}}; \bar{x} + 1.96\frac{\sigma}{\sqrt{N}}\right)$$

используем верхнюю границу, интервал сужается с n

Правило обновления

- Как пересчитать Q_t при поступлении новой информации?

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

- Не нужно хранить всю историю, только текущий шаг и число экспериментов

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 $\frac{1}{t+1}$ изменятеся, но им можно управлять, чтобы улучшить процедуру обучения

Вычисление оценок

Сдвигаем оценку так, чтобы уменьшилась ошибка

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

(Цель - Старая оценка) - ошибка

Постоянная скорость обучения

- Пусть

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

- Веса затухают экспоненциально

$$\begin{aligned} Q_t &= Q_{t-1} + \alpha\left(r_t - Q_{t-1}\right) = \alpha r_t + (1-\alpha)Q_{t-1} \\ &= \alpha r_t + (1-\alpha)\alpha r_{t-1} + (1-\alpha)^2 Q_{t-2} \\ &= (1-\alpha)^t Q_0 + \sum_{i=1}^{t} \alpha(1-\alpha)^{t-i} r_i \end{aligned}$$

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
- Многие задачи RL могут быть сведены к задаче MDP

Марковское состояние

- Марковское состояние содержит всю необходимую информацию, которая может быть иметься в истории
- **Состояние s_t называется марковским, если и только если**

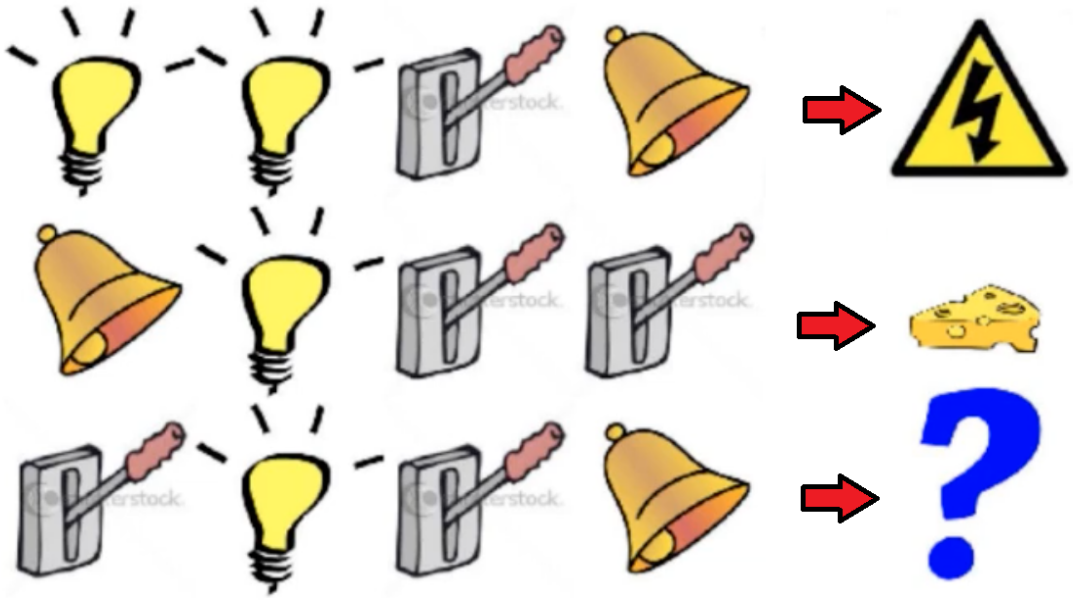
$$P\{s_{t+1}|s_t\} = P\{s_{t+1}|s_1, ..., s_t\}$$

- 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

Марковское состояние



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 каждым переходом (у мыши большая часть вознаграждений равна 0, часть полож., если достигли воду или сыр, и отриц., если достигли поражение током)

Матрица переходов

- Вероятность перехода для марковского состояния s в следующее состояние s' определяется как

$$\mathcal{P}_{s,s'} = \Pr\{s_{t+1} = s' | s_t = s\}$$

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

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

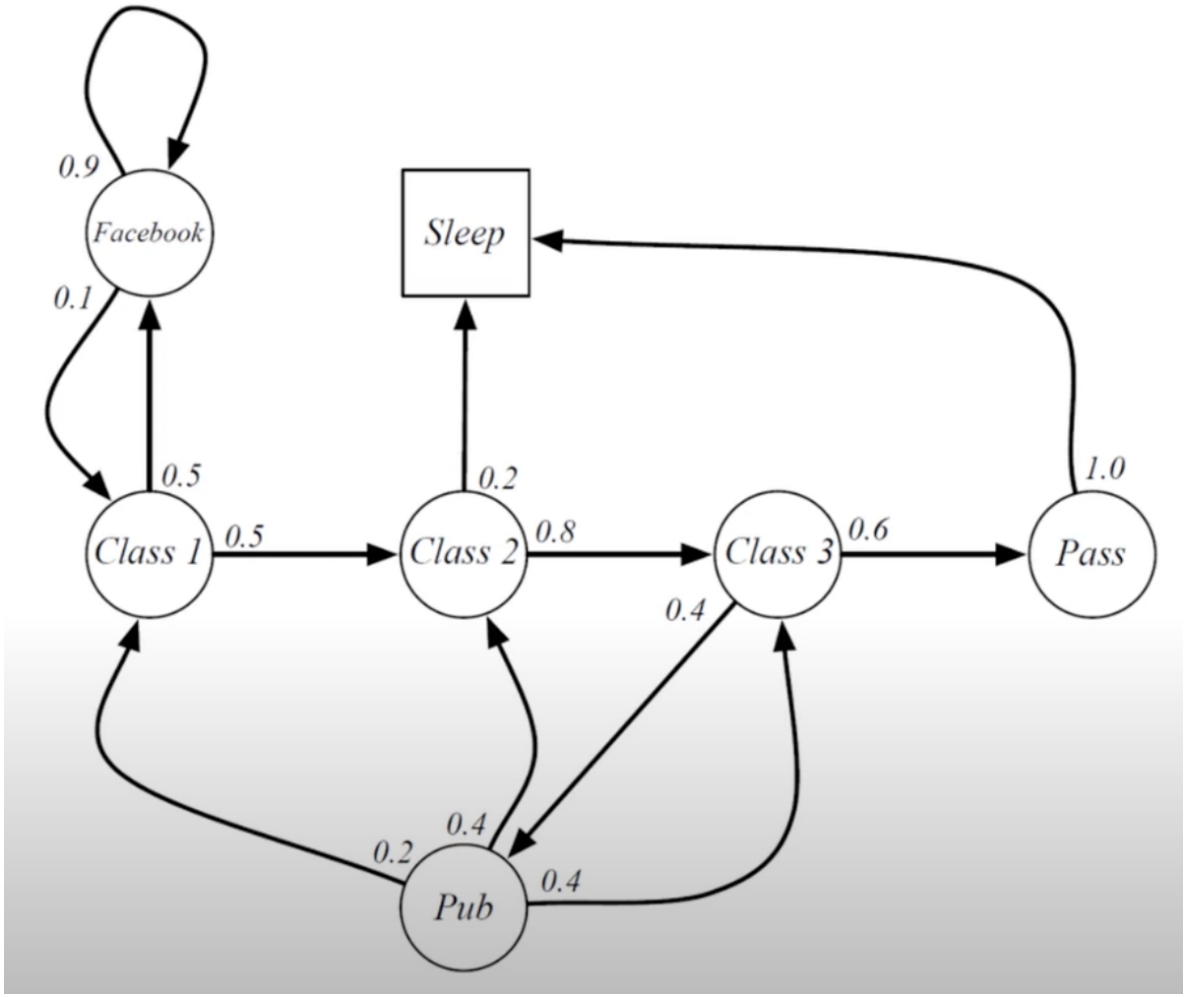
- Каждая строка в сумме равна 1

Марковский процесс

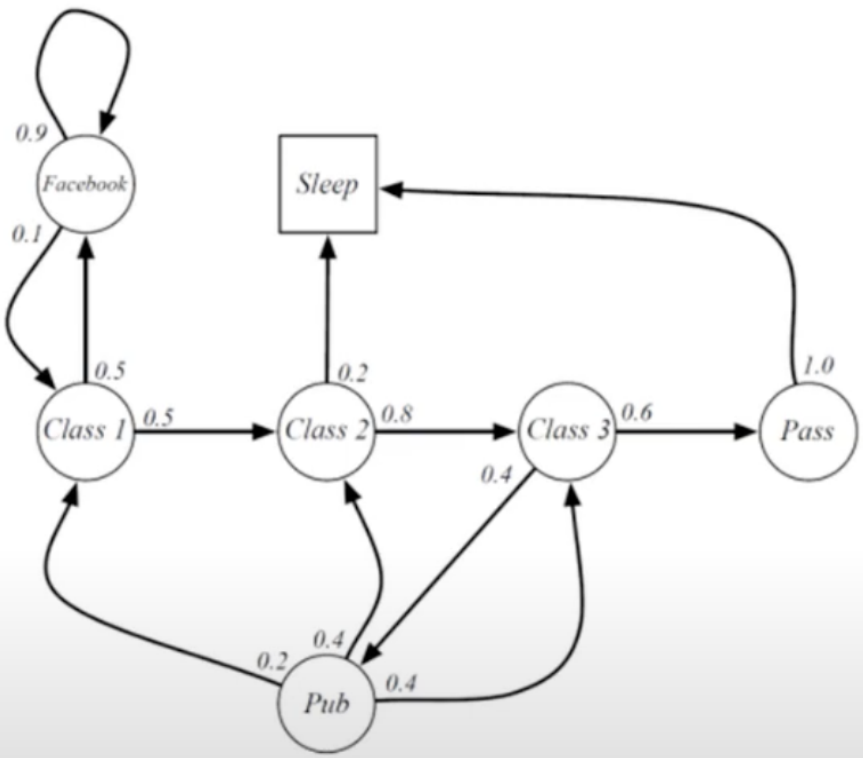
- Марковский процесс - это случайный процесс без памяти, т.е. последовательность случайных состояний $s_1, s_2, \ldots$
- Марковский процесс - это пара $\langle \mathcal{S}, \mathcal{P} \rangle$, где $\mathcal{S}$ - конечное множество состояний, $\mathcal{P}$ - матрица переходов

$$\mathcal{P}_{s,s'} = \Pr\{s_{t+1} = s' | s_t = s\}$$

Пример студенческой марковской цепи



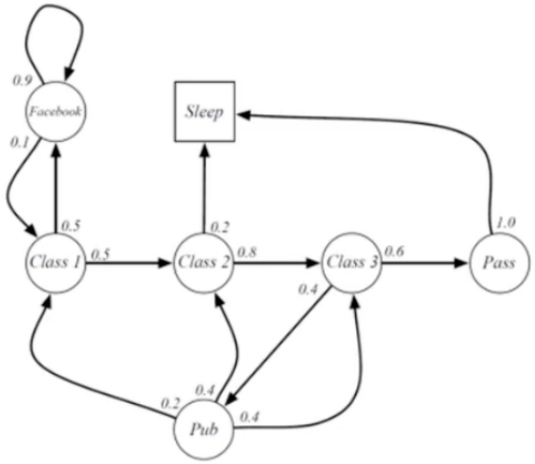
Эпизоды студенческой марковской цепи



Выборка эпизодов для студенческой марковской цепи, начинающихся с $s_1 = C1$:

- C1 C2 C3 Pass Sleep
- C1 FB FB C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 FB FB C1 C2 C3 Pub C1 FB FB FB C1 C2 C3 Pub C2 Sleep

Переходы студенческой марковской цепи

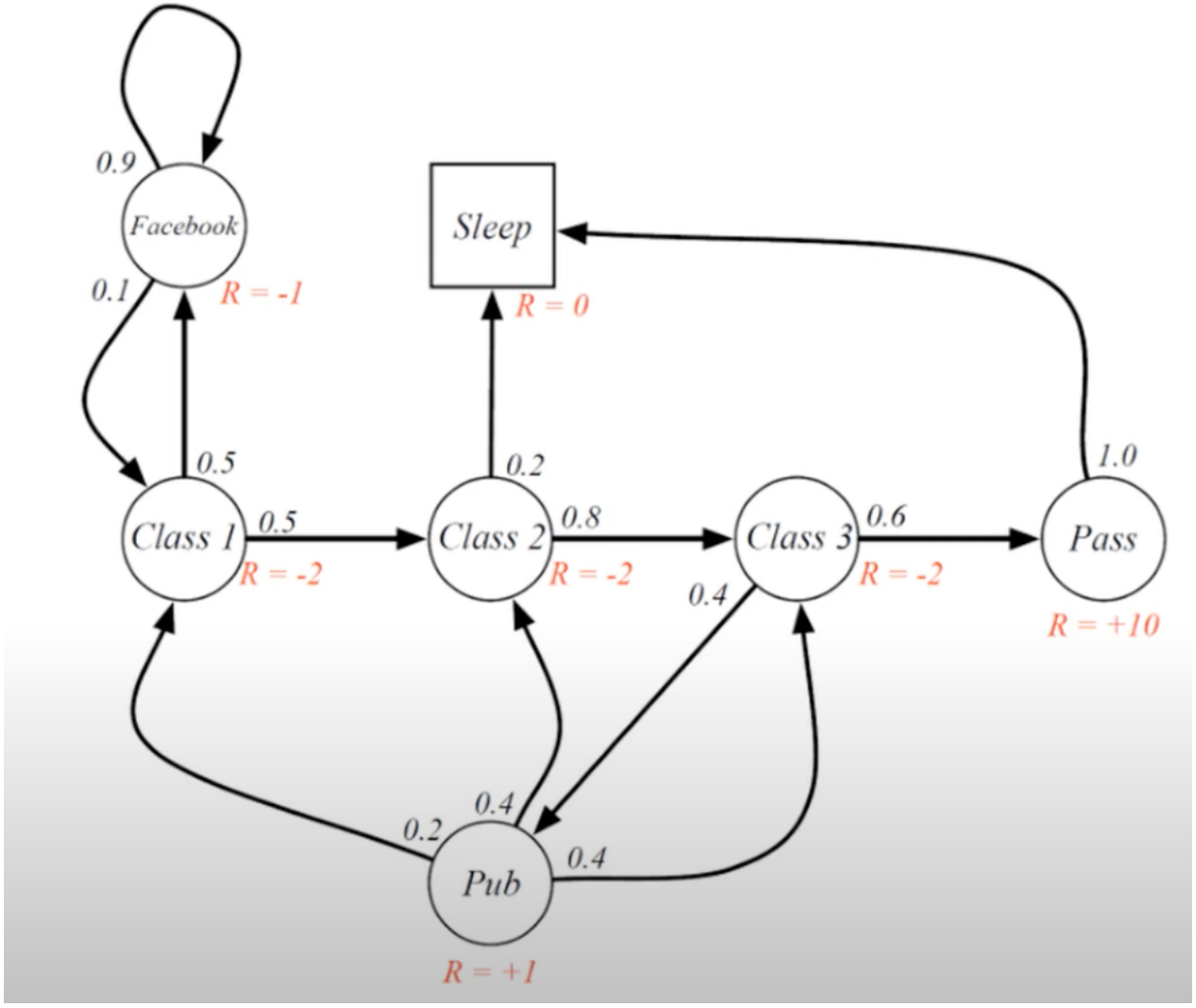


$$\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} \begin{array}{c} \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\ \begin{bmatrix} & 0.5 & & & 0.5 & & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1.0 \end{bmatrix} \end{array}$$

Марковский процесс вознаграждения

- Марковский процесс вознаграждения - марковская цепь с дополнительными значениями вознаграждений
- Марковский процесс вознаграждения (MRP) - это тройка $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, где
  - $\mathcal{S}$ - конечное множество состояний
  - $\mathcal{P}$ - матрица переходов $\mathcal{P}_{s,s'} = \Pr\{s_{t+1} = s' | s_t = s\}$
  - $\mathcal{R}$ - функция вознаграждения $\mathcal{R}_s = \mathbb{E}\{r_{t+1} | s_t = s\}$
  - γ - дисконтирующий множитель

Студенческий MRP



Суммарное вознаграждение

- Суммарное вознаграждение (return) - сумма дисконтированных вознаграждений с момента времени t:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- 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:
  - $\gamma \to 0$ - близорукий агент
  - $\gamma \to 1$ - дальнозоркий агент

Функция полезности

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

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

- В примере про студенческу ю цепь Маркова выборка отдач, начиная с состояния $s_1 = C$ при $\gamma = 1/2$:

$$R_1 = r_2 + \gamma r_3 + ... + \gamma^{T-2} r_T$$

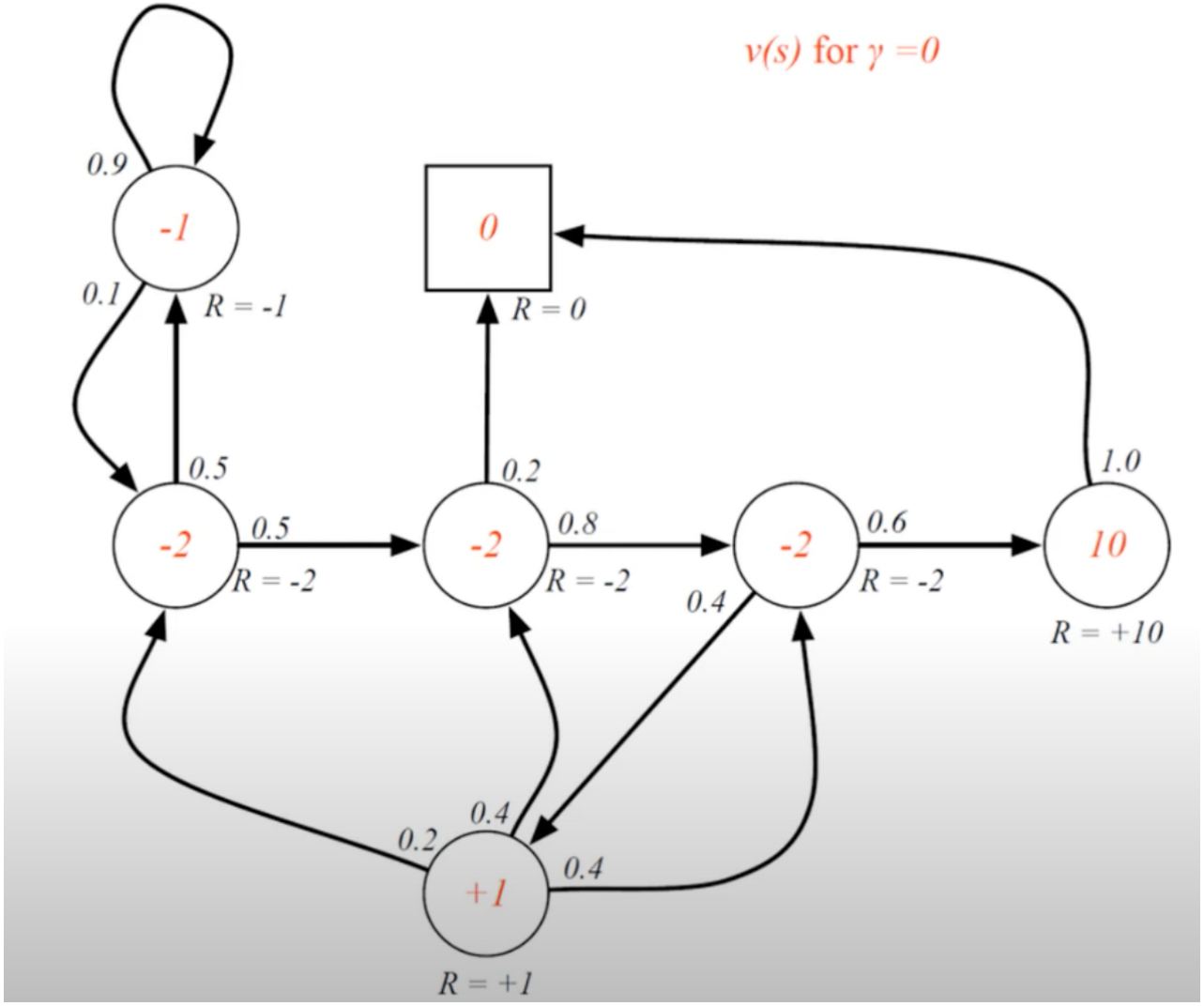
Функция полезности (студенты)

- В примере про студенческу ю цепь Маркова выборка отдач, начиная с состояния $s_1 = C$ при $\gamma = 1/2$:

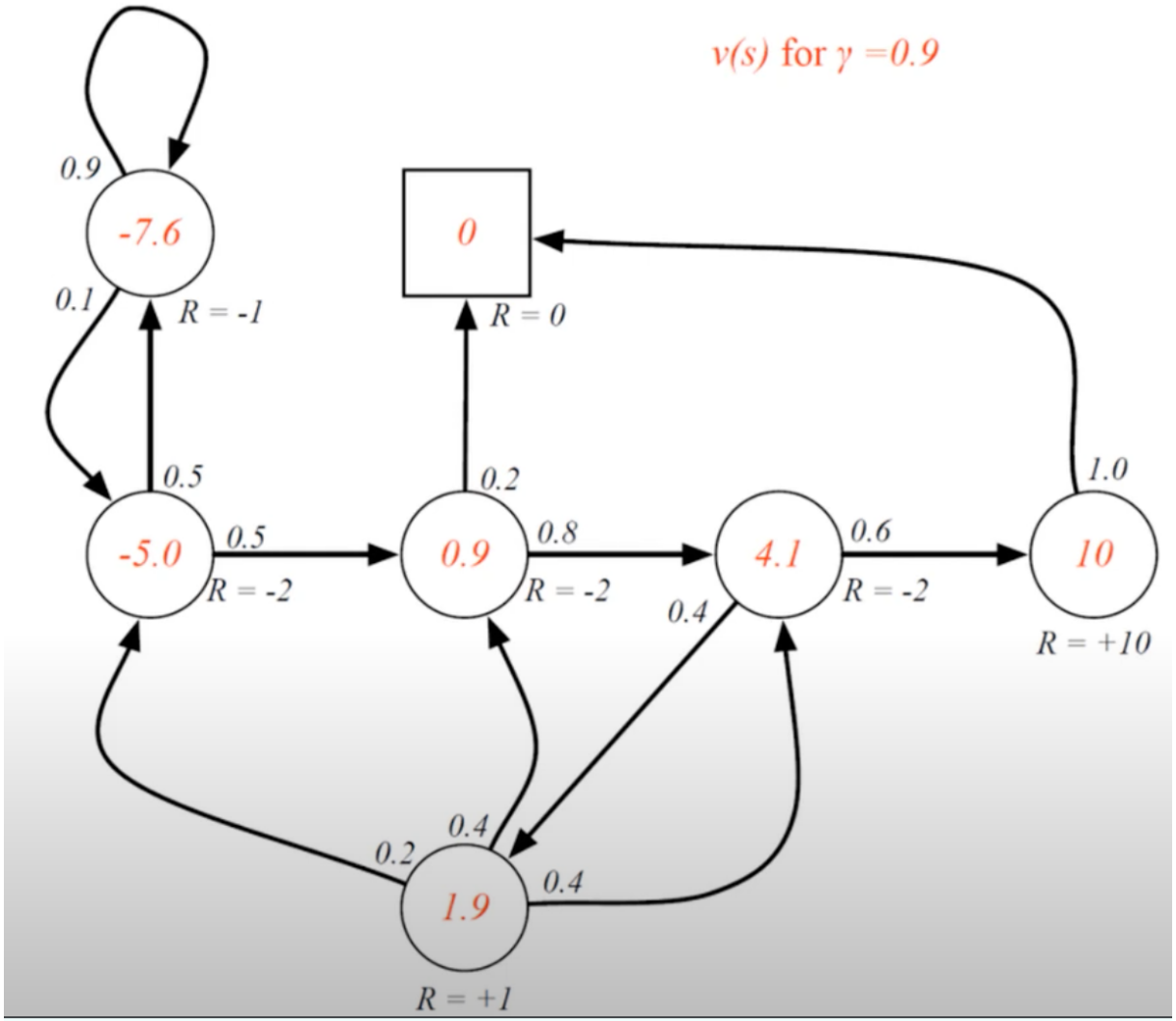
$$R_1 = r_2 + \gamma r_3 + ... + \gamma^{T-2} r_T$$

C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8} = -$
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} = -$
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \ldots = -$
C1 FB FB C1 C2 C3 Pub C1 ...	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \ldots =$
FB FB FB C1 C2 C3 Pub C2 Sleep	$-$

Опять студенты



Опять студенты



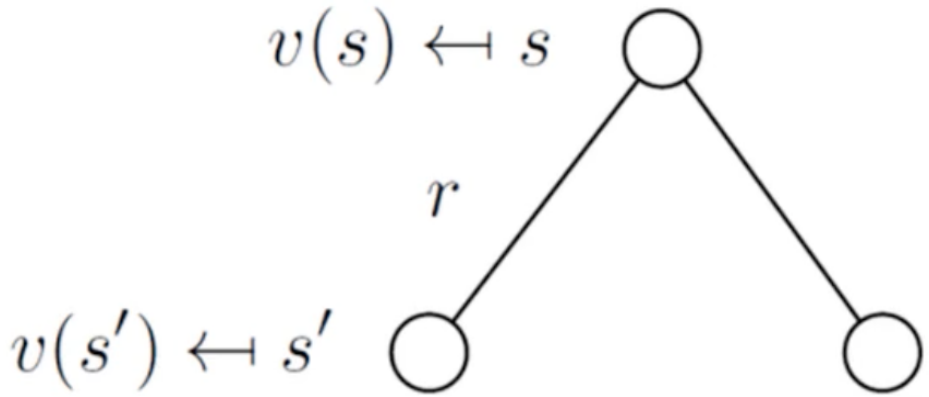
Уравнение Беллмана

- Функцию полезности можно представить в виде двух слагаемых
  - немедленное вознаграждение r_{t+1}
  - дисконтированное значение следующего состояния $\gamma V(s_{t+1})$

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

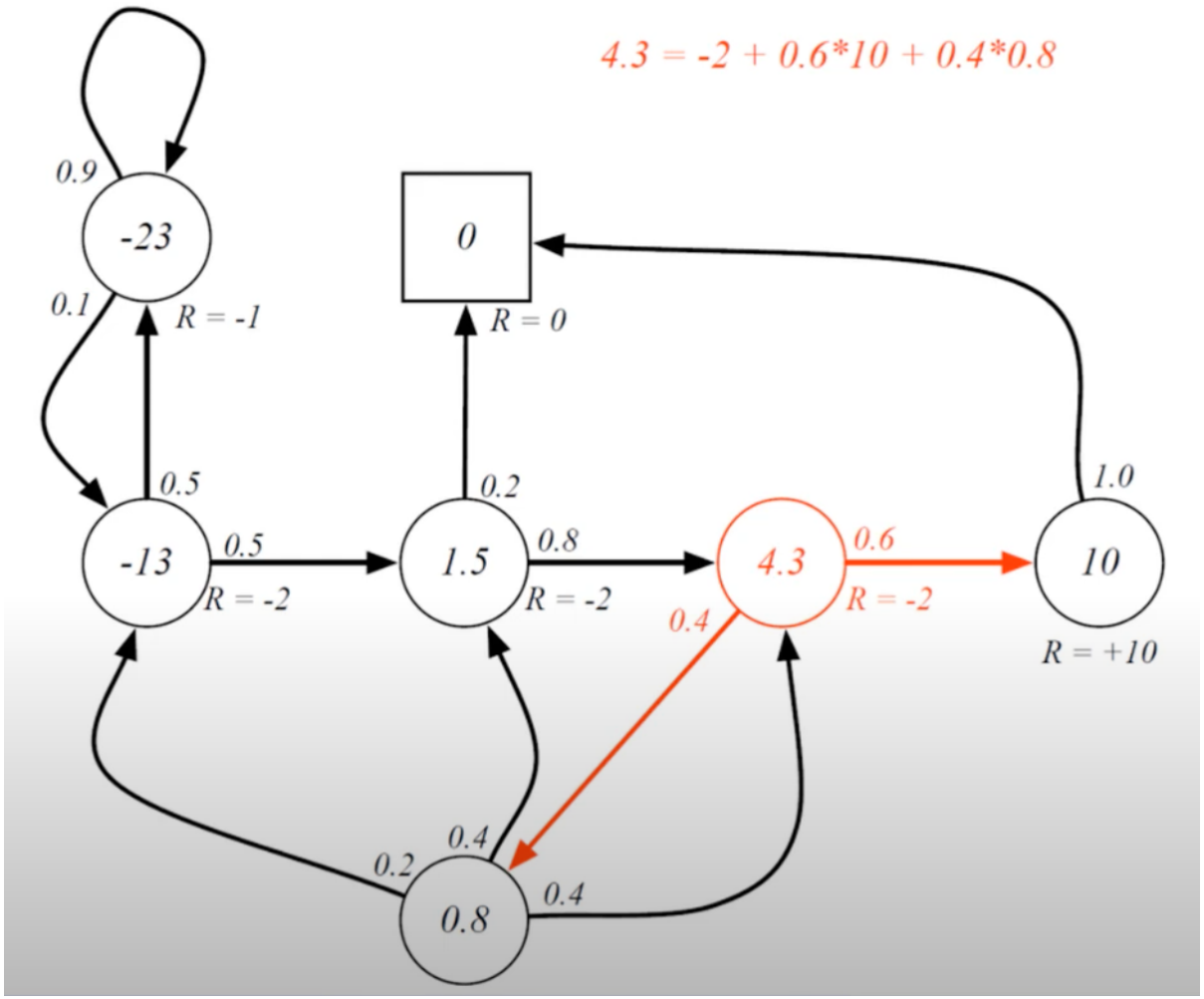
Уравнение Беллмана для MRP

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



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

Уравнение Беллмана для примера



Уравнение Беллмана в матричном виде

Уравнение Беллмана в матричной форме:

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

где V - вектор-колонка с одной компонентой на состояние

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

Решение уравнения Беллмана

- Уравнение Беллмана - это линейное уровнение
- Аналитическое решение

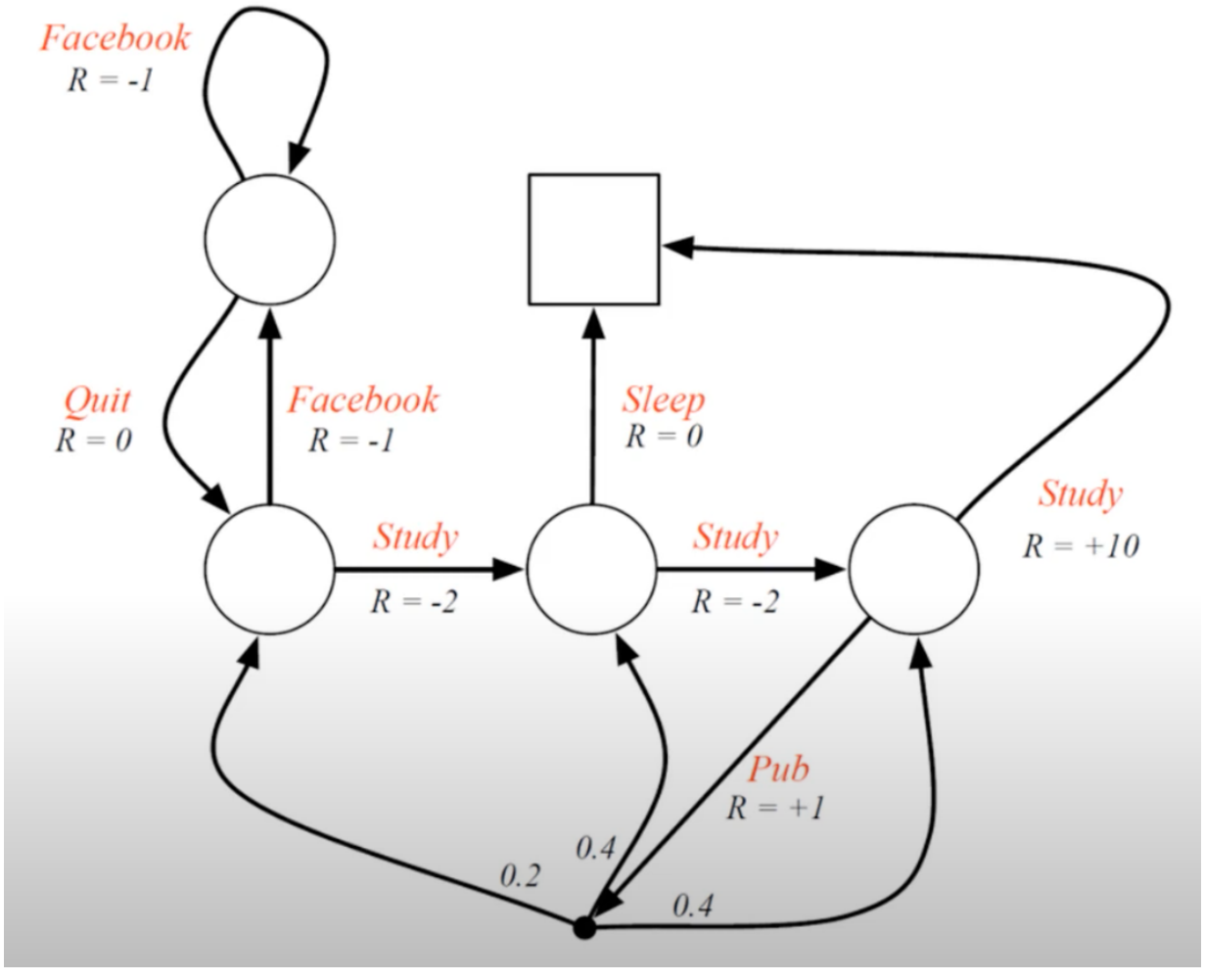
$$
\begin{aligned}
V &= \mathcal{R} + \gamma \mathcal{P} V \\
(I - \gamma \mathcal{P}) V &= \mathcal{R} \\
V &= (I - \gamma \mathcal{P})^{-1} \mathcal{R}
\end{aligned}
$$

- Аналитическое решение только для небольших задач MDP
- Итерационные методы: дин.программирование, Монте-Карло и т.д.

Марковский процесс принятия решений

- Марковский процесс принятия решений - это марковский процесс вознаграждений для действий. Описывает взаимодействие со средой с марковскими состояниями.
- Марковский процесс принятия решений - это кортеж $\langle \mathcal{S}, A, \mathcal{P}, \mathcal{R}, \gamma \rangle$, где
  - $\mathcal{S}$ - конечное множество состояний
  - A - конечное множество действий
  - $\mathcal{P}$ - матрица переходов
    $\mathcal{P}_{s,s'} = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$
  - $\mathcal{R}$ - функция вознаграждения
    $\mathcal{R}_s^a = \mathbb{E}\{r_{t+1} | s_t = s, a_t = a\}$
  - γ - дисконтирующий множитель

Марковский процесс принятия решений - студенты



Стратегии

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

$$\pi(a|s) = \Pr\{a_t = a|s_t = s\}$$

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

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

Стратегии

- Пусть дан MDP $\mathcal{M} = \langle \mathcal{S}, A, \mathcal{P}, \mathcal{R}, \gamma \rangle$ и стратегия π
- Последовательность состояний $s_1, s_2, \ldots$ - марковский процесс $\langle \mathcal{S}, \mathcal{P}^\pi \rangle$
- Последовательность состояний и вознаграждений $s_1, r_1, s_2, \ldots$ - марковский процесс вознаграждений $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$, где

$$\mathcal{P}^\pi_{s,s'} = \sum_{a \in A} \pi(a|s) \mathcal{P}^a_{s,s'}$$

$$\mathcal{R}^\pi_s = \sum_{a \in A} \pi(a|s) \mathcal{R}^a_s$$

Функция полезности

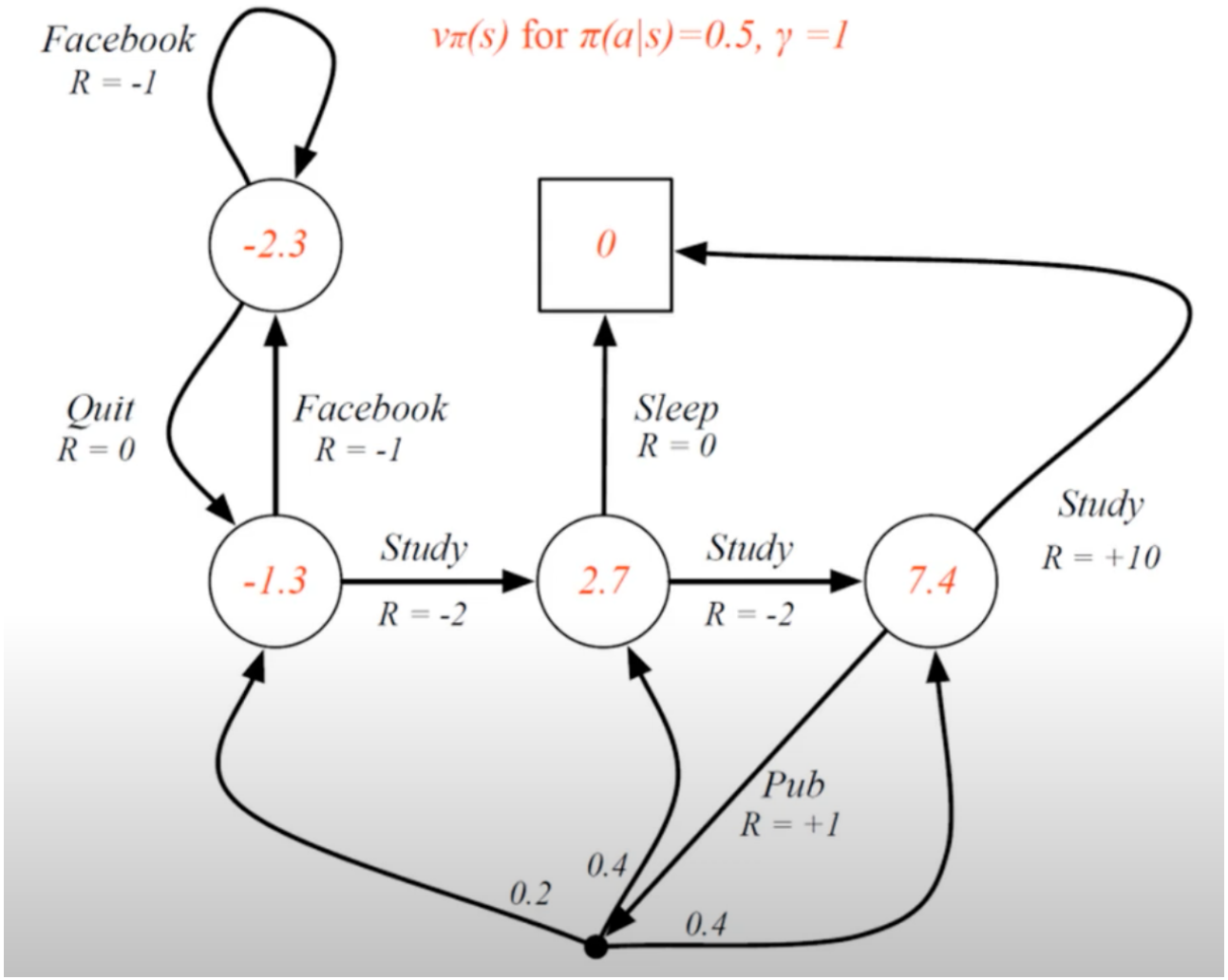
- *Функция полезности состояний* MDP $V^{\pi}(s)$ - мат. ожидание отдачи, полученной, начиная с состояния s, при выполнении стратегии π:

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

- Допустим, агент находится в некотором состоянии и знает его ценность $V^*(s)$, а также знает ценности всех других состояний. Это не дает понимания того, какие действия в какие состояния приведут. Поэтому увеличим кол-во переменных: введцм схожее определение для ценности не состояний, но пар состояние-действие.
- *Функция полезности действий* MDP $Q^{\pi}(s, a)$ - мат. ожидание отдачи, полученной, начиная с состояния s и выбранного действия a, при выполнении стратегии π:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\{R_t | s_t = s, a_t = a\}$$

Функция полезности для студентов



Уравнение Беллмана для MDP

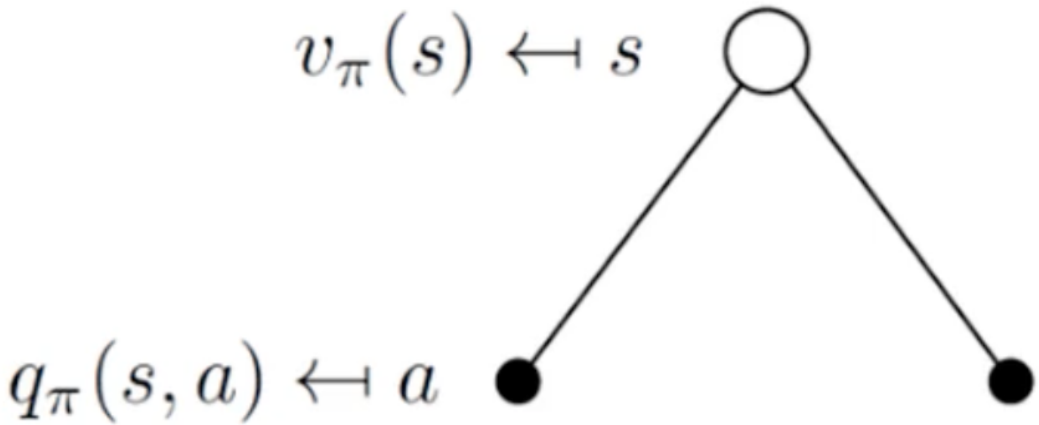
- Функцию полезности можно представить в виде двух слагаемых
  - немедленное вознаграждение r_{t+1}
  - дисконтированное значение следующего состояния

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s\right]$$

- Для функции полезности действий:

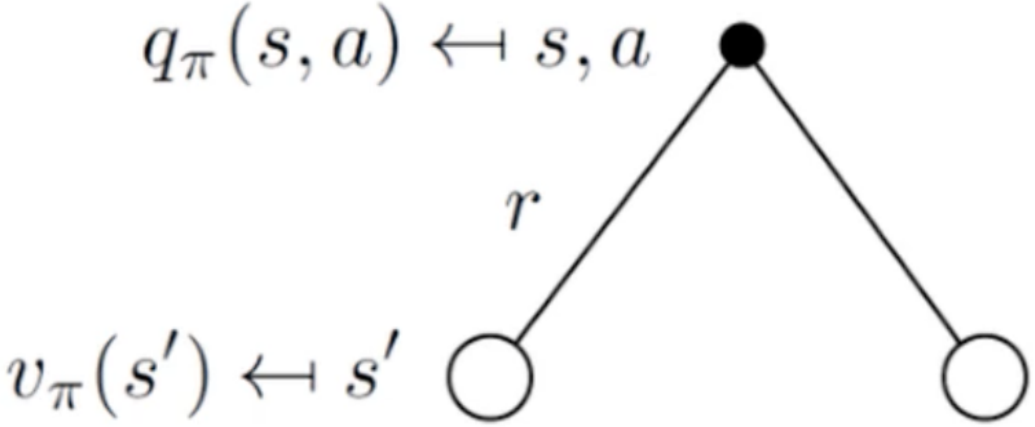
$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a]$$

Уравнение Беллмана для функции полезности



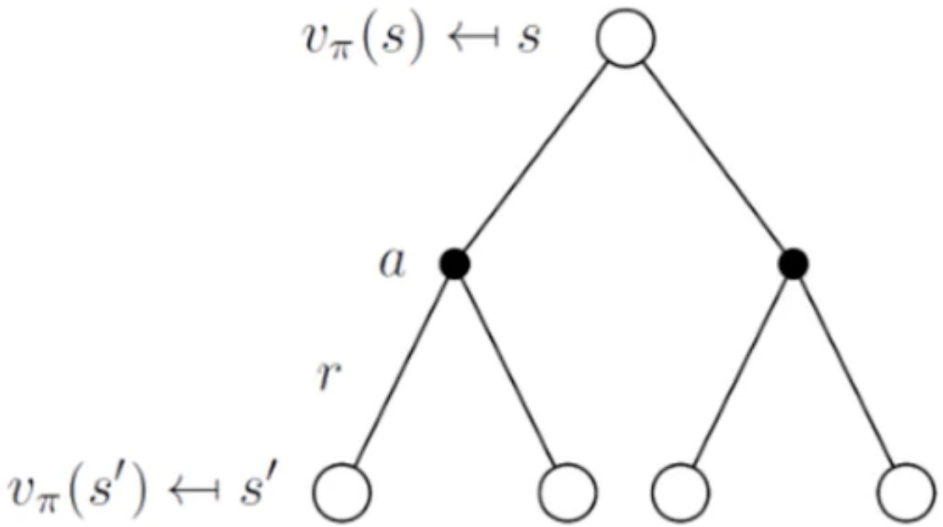
$$V^\pi(s) = \sum_{a \in A} \pi(a|s) Q^\pi(s,a)$$

Уравнение Беллмана для функции полезности действий



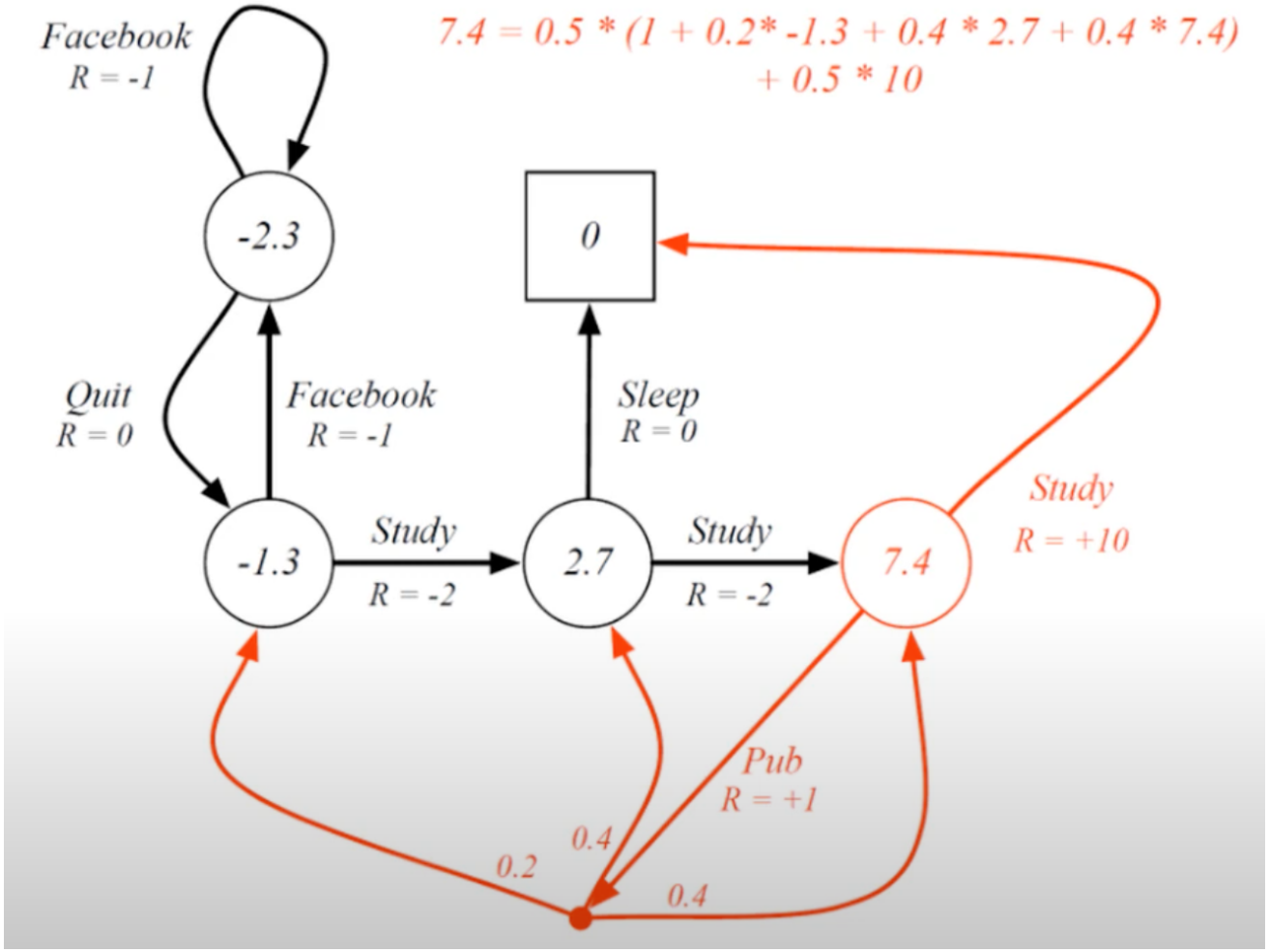
$$Q^\pi(s,a) = \mathcal{R}_s^\pi + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a V(s')$$

Уравнение Беллмана для функции полезности



$$V^\pi(s) = \sum_{a \in A} \pi(a|s) Q^\pi(s,a) = \sum_{a \in A} \pi(a|s) \left(\mathcal{R}_s^\pi + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a V(s') \right)$$

Уравнение Беллмана для студентов



Уравнение Беллмана в матричной форме

В матричном виде

$$V^\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi V^\pi$$

Решение

$$V^\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

Оптимальная функция полезности

- Оптимальная функция полезности $V^*(s)$ - это максимальное значение функции полезности по всем стратегиям

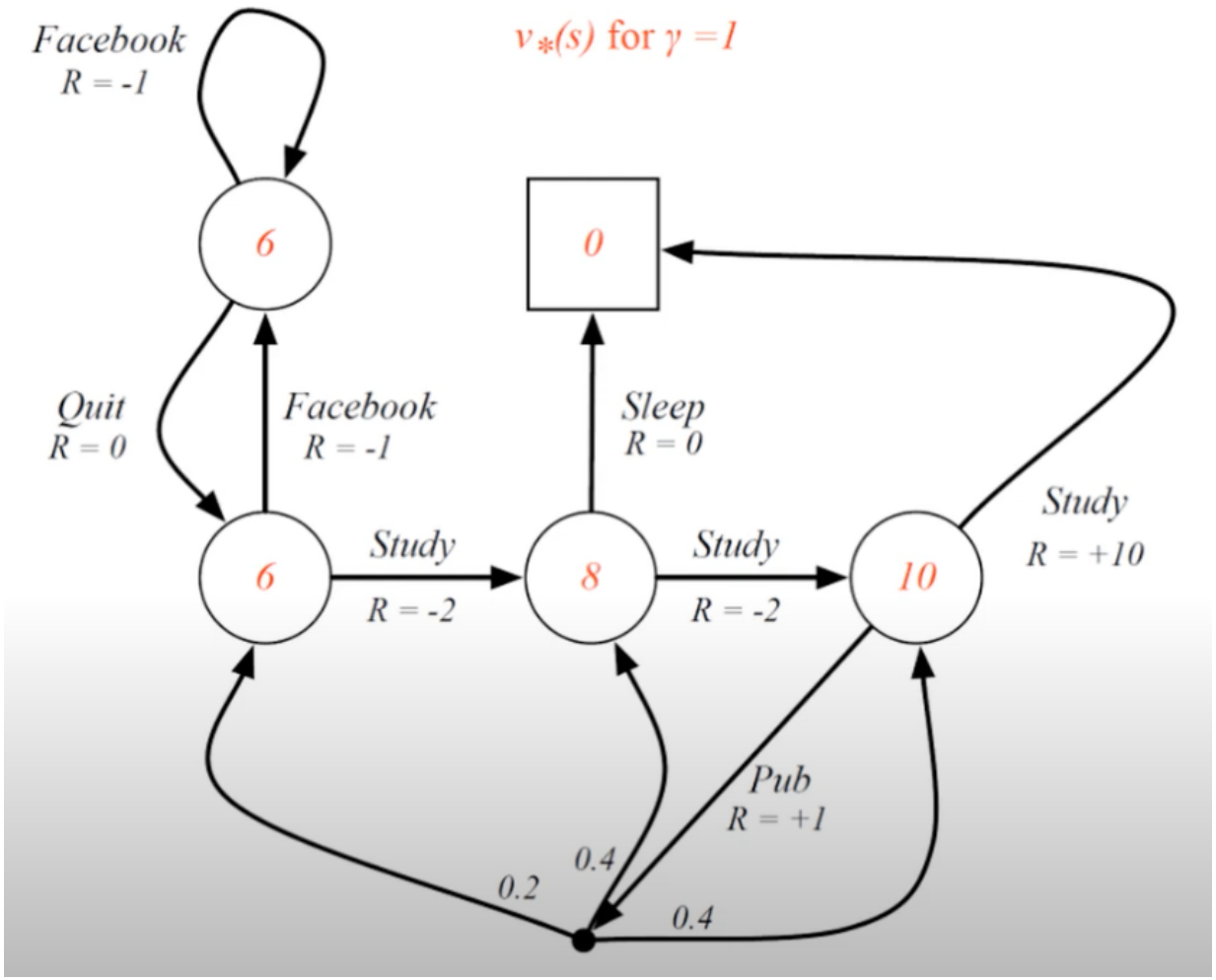
$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

- Оптимальная функция полезности действий $Q^*(s, a)$ - это максимальное значение функции полезности действий по всем стратегиям

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

- Оптимальные стратегии характеризуют лучшее поведение в MDP
- Задача MDP решена, если найдена оптимальная стратегия

Оптимальная функция полезности для студентов



Оптимальная стратегия

Определим частичный порядок на множестве стратегий:

$$\pi \geq \pi', \text{ если } V^{\pi}(s) \geq V^{\pi'}(s)$$

Theorem

Для марковского процесса принятия решений:

- *существует оптимальная стратегия* π^*, *которая лучше или эквивалентна всем другим стратегиям* $\pi^* \geq \pi, \forall \pi$
- *все оптимальные стратегии удовлетворяют оптимальной функции полезности состояний* $V^{\pi^*}(s) = V^*(s)$
- *все оптимальные стратегии удовлетворяют оптимальной функции полезности действий* $Q^{\pi^*}(s, a) = Q^*(s, a)$

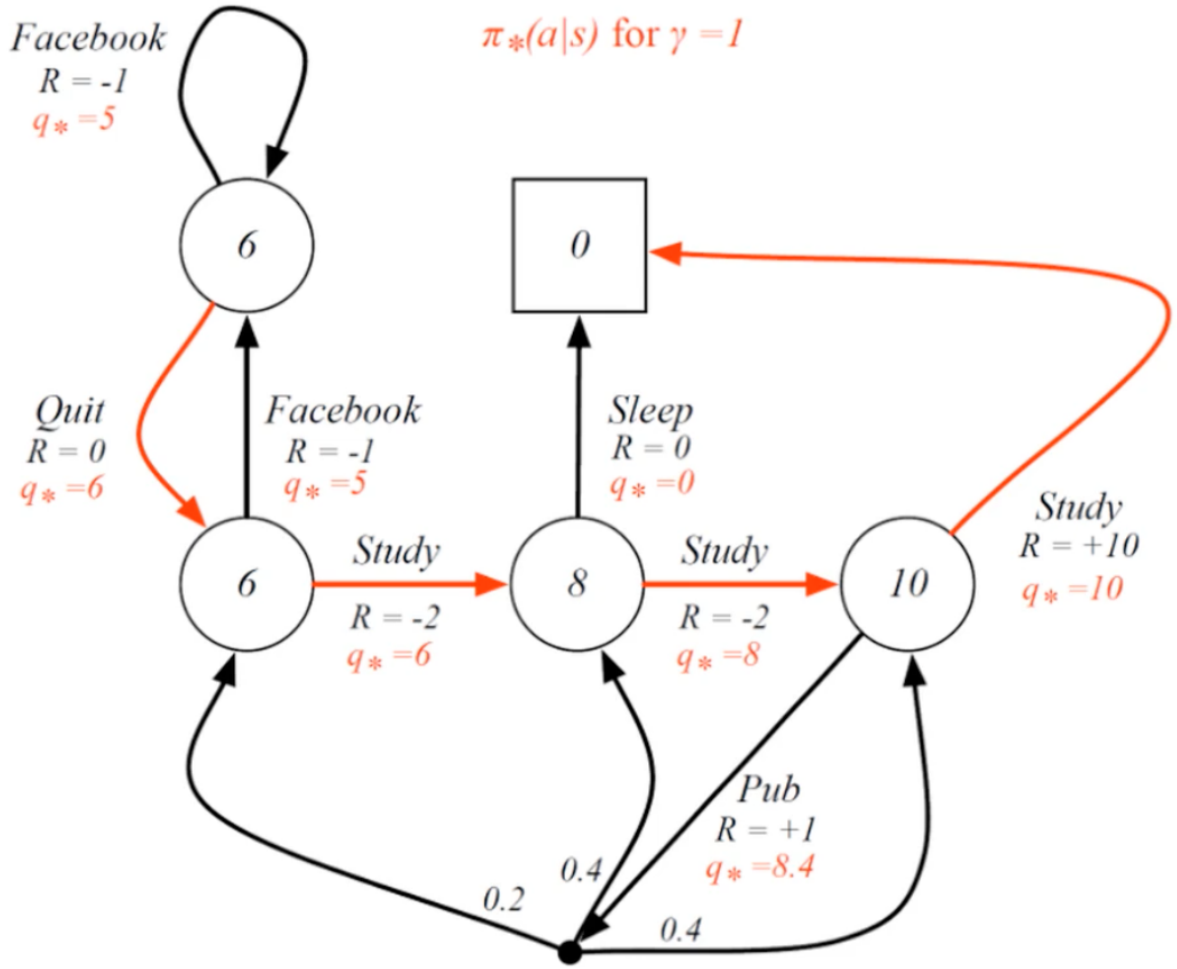
Поиск оптимальной стратегии

- Оптимальная стратегия π^* может быть найдена максимизацией функции полезности действий $Q^{\pi^*}(s, a)$:

$$\pi^*(a|s) = \begin{cases} 1, & a = \arg\max_{a \in A} Q^{\pi^*}(s, a) \\ 0, & \text{иначе} \end{cases}$$

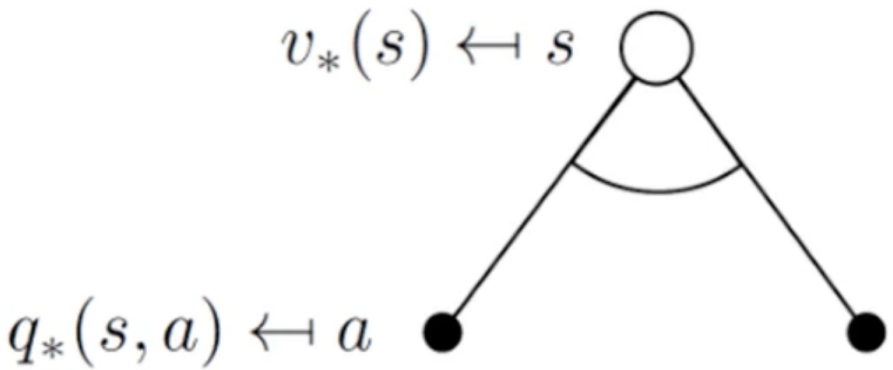
- Для любого MDP существует оптимальная детерминированная стратегия
- Если известна $Q^*(s, a)$, то одновременно получаем оптимальную стратегию

Поиск оптимальной стратегии для студентов



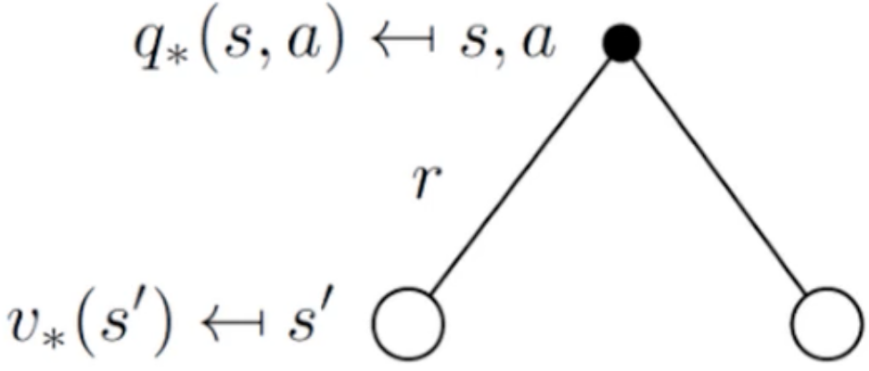
Уравнение Беллмана для оптимальной функции полезности

Оптим. значения функции полезности связаны уравнением оптимальности Беллмана



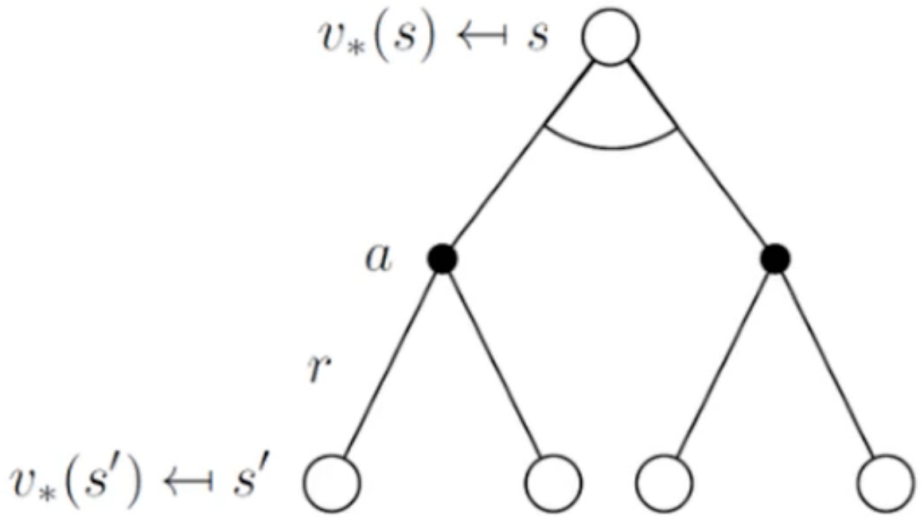
$$V^*(s) = \max_{a \in A} Q^*(s,a)$$

Уравнение Беллмана для оптимальной функции полезности действий



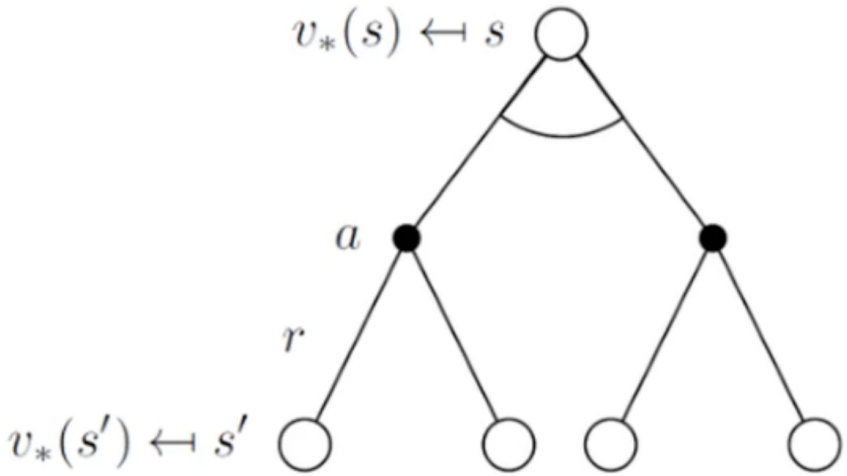
$$Q^*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a V^*(s')$$

Уравнение Беллмана для оптимальной функции полезности действий



$$Q^{\pi}(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a \max_{a' \in A} Q^*(s', a')$$

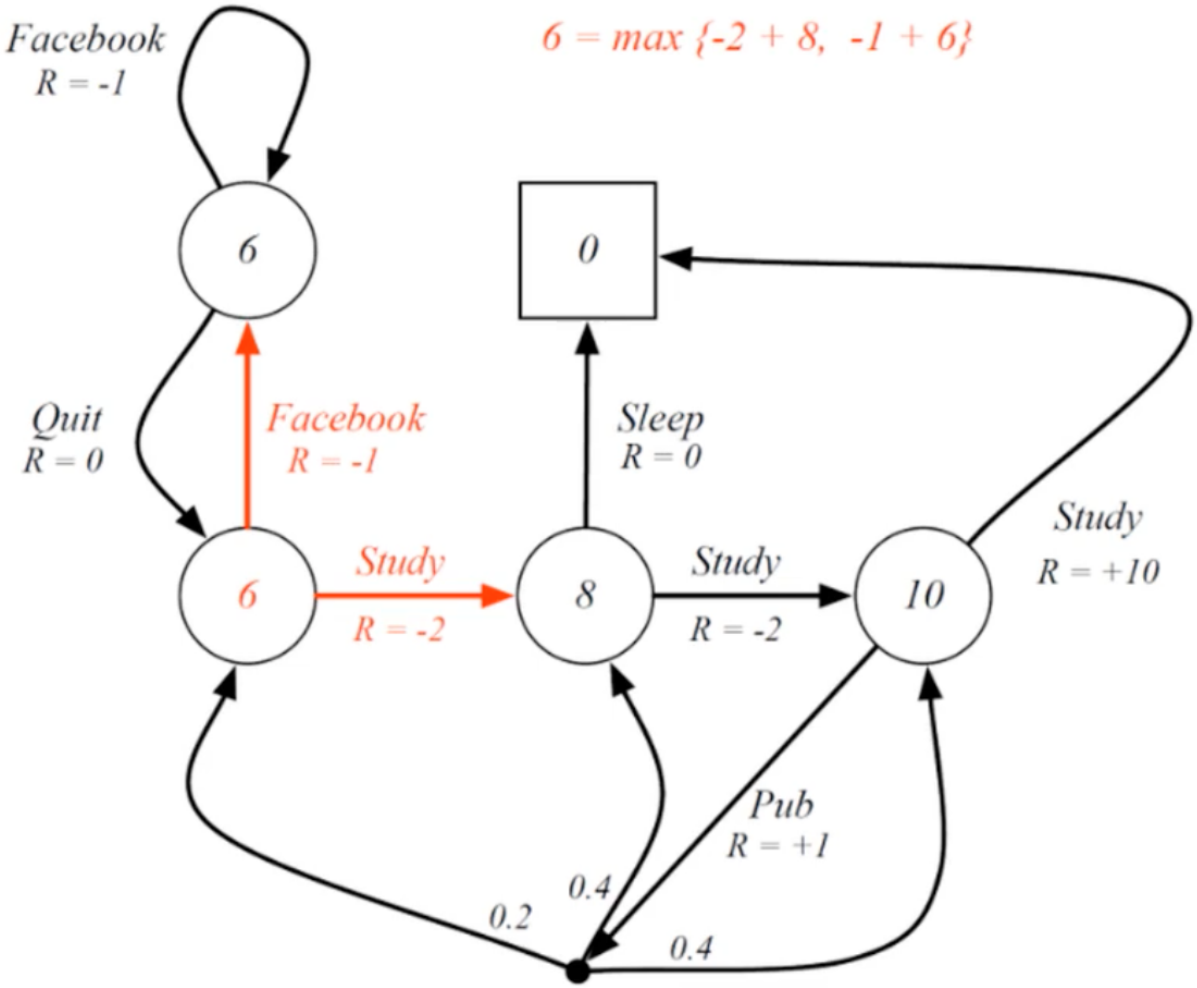
Уравнение Беллмана для оптимальной функции полезности действий



$$Q^*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a V^*(s')$$

$$V^*(s) = \max_{a \in A} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a V^*(s') \right)$$

Оптимальная функция полезности действий для студентов



Решение уравнения оптимальности Беллмана

- Уравнение нелинейно
- Аналитического решения в общем виде нет
- итерации по ценностям, итерации по стратегии, Q-обучение, SARSA

Планирование с помощью DP

- DP предполагает использование всей информации о MDP
- В реальных системах используется на этапе планирования
- Для задачи оценки:

$$\langle \mathcal{S}, A, \mathcal{P}, \mathcal{R}, \gamma \rangle \to V^{\pi}$$

или

$$\langle \mathcal{S}, \mathcal{P}^{\pi}, \mathcal{R}^{\pi}, \gamma \rangle \to V^{\pi}$$

- для задачи управления

$$\langle \mathcal{S}, A, \mathcal{P}, \mathcal{R}, \gamma \rangle \to \langle V^{*}, \pi^{*} \rangle$$

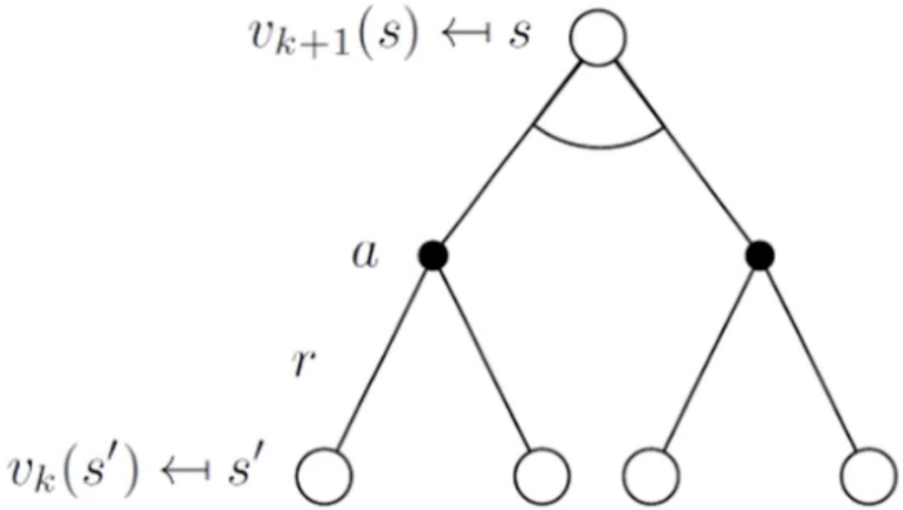
Итерационная оценка стратегии

- **Задача:** оценить текущую стратегию π
- **Решение:** итеративное применение уравнения Беллмана в обратном направлении

$$V^1 \to V^2 \to \ldots \to V^\pi$$

- Использование *синхронных* шагов:
  - для каждой итерации $k+1$:
  - для каждого состояния $s \in \mathcal{S}$:
  - обновить $V^{k+1}(s)$ по $V^k(s')$, где s' - следующее состояние после s
- Можно использовать асинхронные шаги
- Сходится к истинным значениям V^π

Итерационная оценка стратегии



$$V^{k+1}(s) = \sum_{a \in A} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a V^k(s') \right)$$

$$V^{k+1} = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi V^k$$

Пример

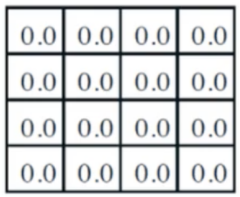
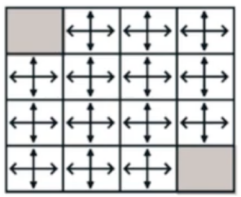
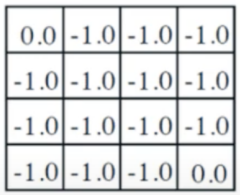
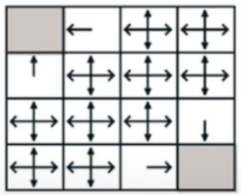
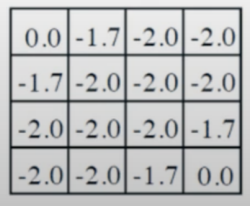
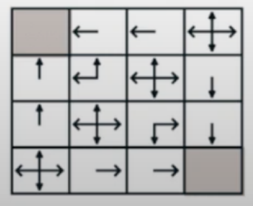
	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$r = -1$
on all transitions

- MDP с $\gamma =$
- Нетерминальные состояния $1, ..., 14$
- Терминальные состояния (серые квадраты)
- Действия, ведущие за пределы карты, не меняют состояния
- Агент следует случайной стратегии

$$\pi(n|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = \pi(e|\cdot) = 0.25$$

Пример

	V^k для случайной стратегии	Жадная стратегия по V^k
$k = 0$		 ← случайная стратегия
$k = 1$		
$k = 2$		

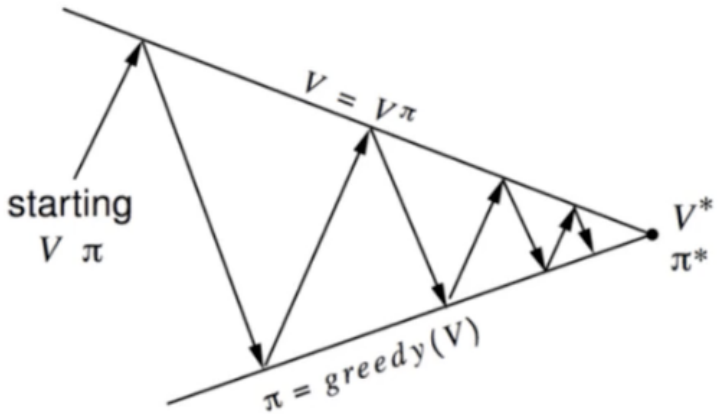
Улучшение стратегии

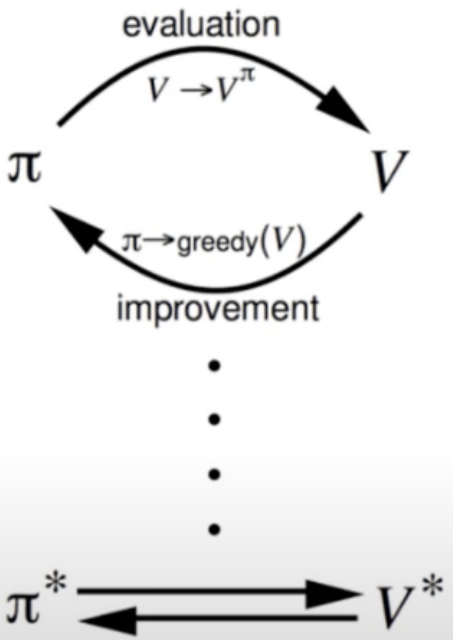
- Дана стратегия π
  - оценить (evaluate) стратегию π

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

  - улучшить (improve) стратегию, выбирая действия жадно (greedy), но в соответствии с V^{π}: $\pi' = greedy(V^{\pi})$
- В примере с клетками улучшенная стратегия оказалась оптимальной: $\pi' = \pi^*$
- В общем случае надо больше итераций
- Этот процесс итераций по стратегии всегда сходится к оптимальной стратегии

Итерации по стратегиям





Оценка стратегии – вычисление V^π

Итеративная оценка стратегии

Улучшение стратегии – генерация $\pi' \geq \pi$

Жадное обновление стратегии

Улучшение стратегии

- Рассмотрим детермин. стратегию $a = \pi(s)$
- Можно улучшить стратегию, действуя жадно

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

- Это улучшает полезность от любого состояния s на один шаг

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

- Таким образом улучшаем функцию полезности $V^{\pi'}(s) \geq V^{\pi}(s)$

Улучшение стратегии

- Когда процесс улучшения останавливается, получаем

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

- Тогда уравнение Беллмана удовлетворено

$$V^{\pi}(s) = \max_{a \in A} Q^{\pi}(s, a)$$

- Это означает, что $V^{\pi}(s) = V^{*}(s)$ и π оптимальна

Принцип оптимальности

- Любая оптимальная стратегия может быть разделена на две части:
  - оптимальный первый шаг a^*
  - следование оптимальной стратегии, начиная со следующего состояния s'

Theorem ((Принцип оптимальности))

Стратегия $\pi(a|s)$ достигает оптимальной оценки состояния s: $V^{\pi}(s) = V^*(s)$, если и только если для любого s', достижимого из s, π достигает оптимальной оценки состояния s': $V^{\pi}(s') = V^*(s')$.

Принцип оптимальности Беллмана: жадный выбор действия в предположении оптимальности дальнейшего поведения оптимален.

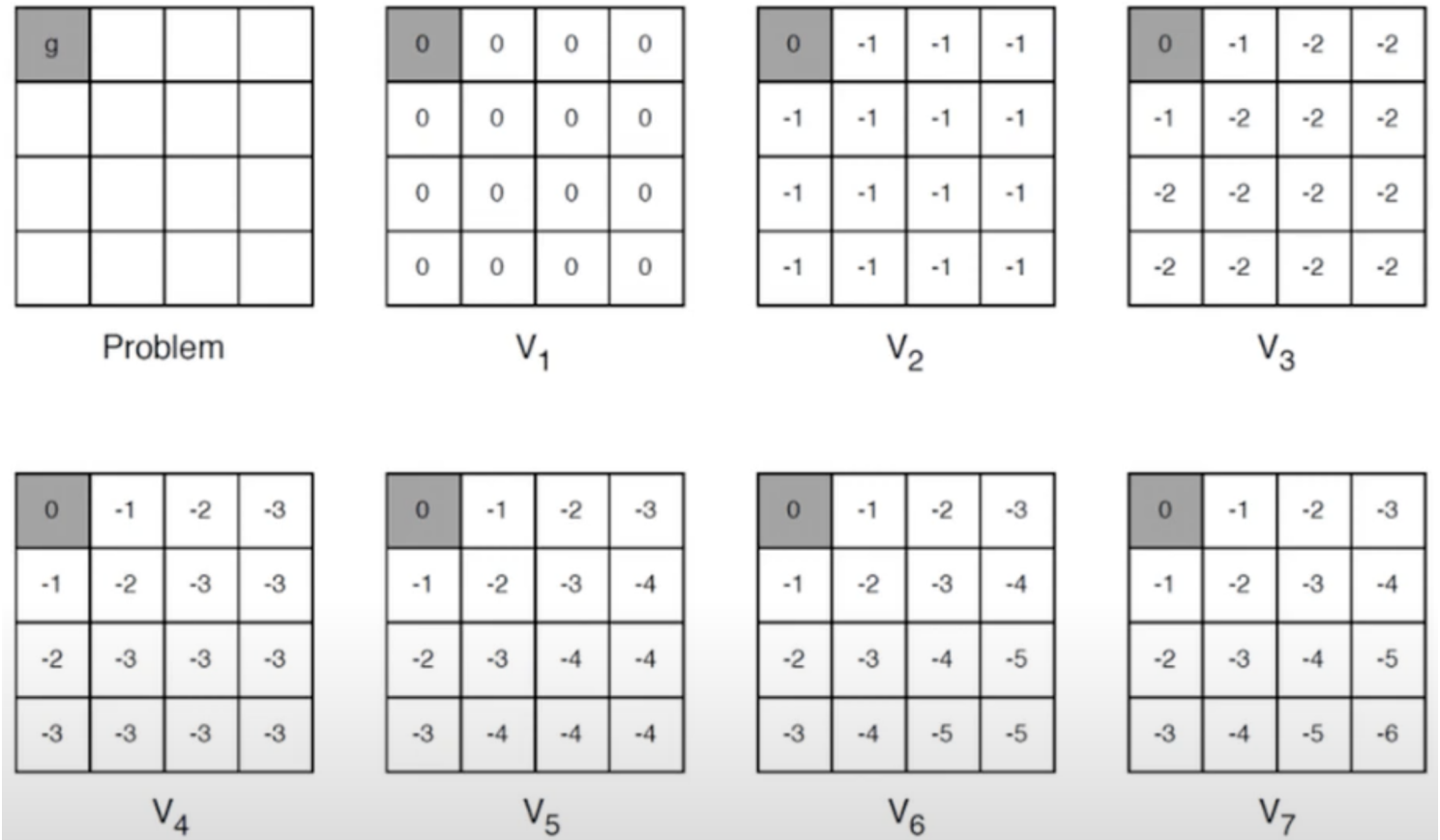
Детерминированные итерации по полезностям

- Пусть знаем решение для подзадачи $V^*(s')$
- Тогда можно найти решение за один шаг

$$V^*(s) \leftarrow \max_{a \in A} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a V^*(s') \right)$$

- Идея итераций по ценностям - применить эти обновления рекурсивно
- Интуиция: начать с конечных вознаграждений и двигаться назад

Пример - кратчайший путь



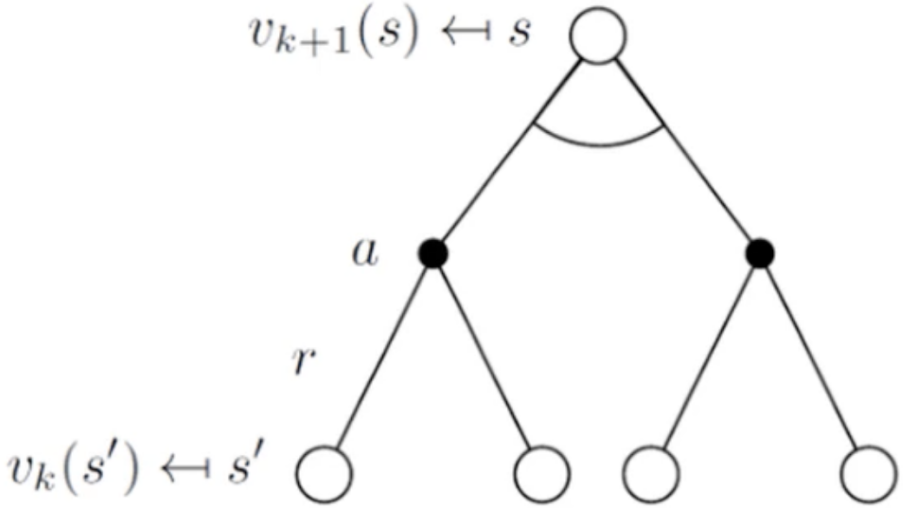
Итерационная оценка по полезности

- **Задача:** оценить текущую стратегию π^*
- **Решение:** итеративное применение уравнения Беллмана в обратном направлении

$$V^1 \to V^2 \to \ldots \to V^\pi$$

- Использование *синхронных* шагов:
  - для каждой итерации $k+1$:
  - для каждого состояния $s \in \mathcal{S}$:
  - обновить $V^{k+1}(s)$ по $V^k(s')$, где s' - следующее состояние после s
- Сходится к истинным значениям V^*
- В отличие от итераций по стратегиям, мы не получаем стратегию в явном виде
- Промежуточные оценки полезности могут не соответствовать ни одной стратегии

Итерационная оценка по полезности



$$V^{k+1}(s) = \max_{a \in A} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s,s'}^a V^k(s') \right)$$

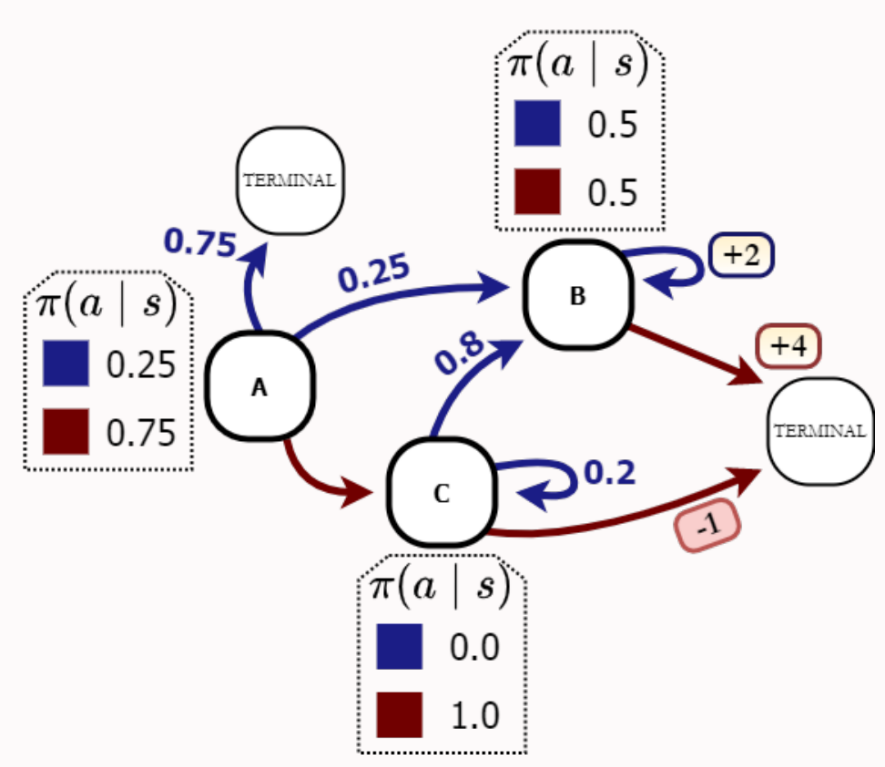
$$V^{k+1} = \max_{a \in A} \mathcal{R}^a + \gamma \mathcal{P}^a V^k$$

Синхронные алгоритмы DP

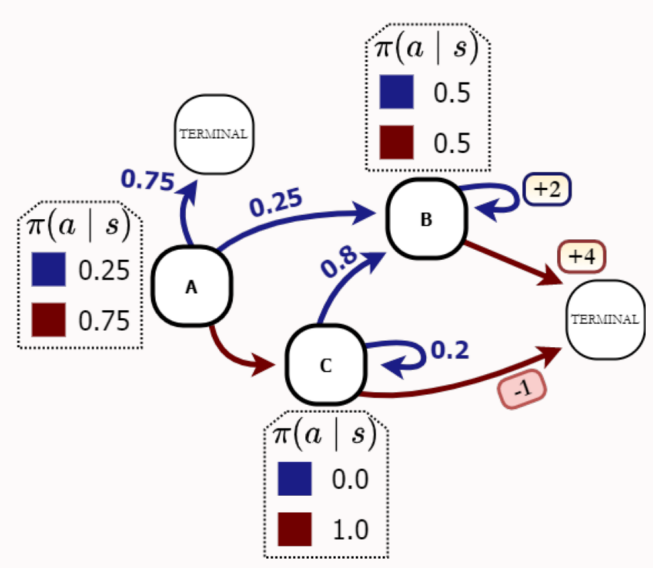
Задача	Уравнение Беллмана	Алгоритм
Оценка	для матожиданий	итеративная оценка стратегии
Управление	для матожиданий + жадное улучшение стратегии	итерация по стратегиям
Управление	оптимальное	итерация по полезностям

Пример 1-1 (V-функция)

Вычислим V-функцию для MDP и стратегии π, $\gamma = 0.8$. Ее часто удобно считать «с конца», начиная с состояний, близких к терминальным, и замечая связи между значениями функции для разных состояний.

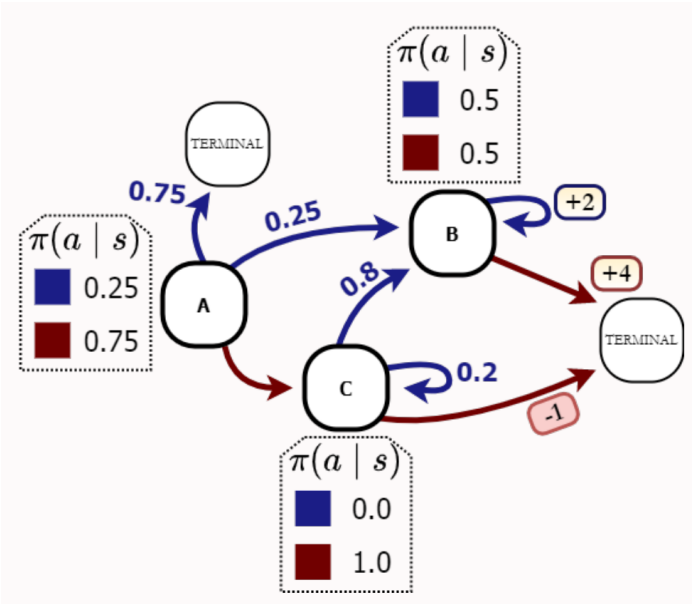


Пример 1-2



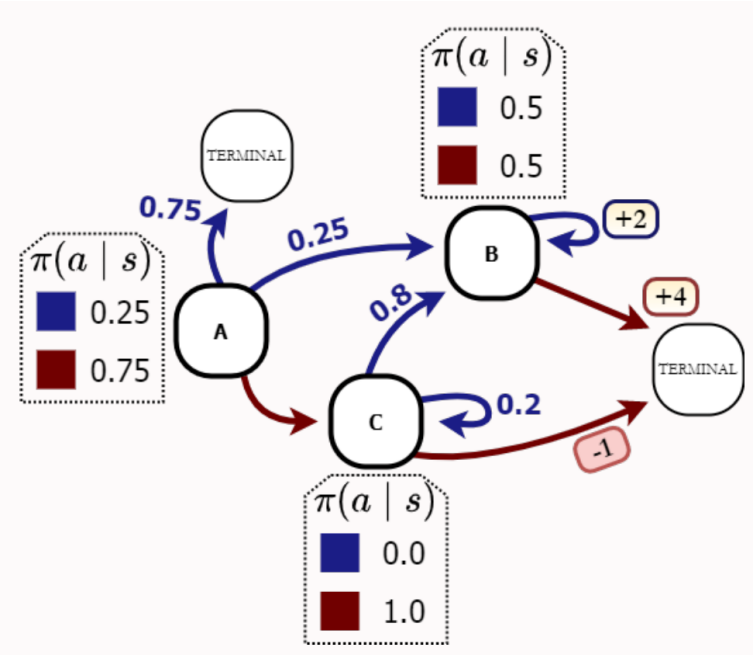
- Состояние C: там агент всегда выбирает действие “к”, получает -1, и эпизод заканчивается: $V^{\pi}(s = C) = -1$
- Состояние B: с вероятностью 0.5 агент выбирает действие “к” и получает +4. Иначе он получает +2 и возвращается снова в состояние B.
- Вся дальнейшая награда будет дисконтирована на $\gamma = 0.8$ и тоже равна $V^{\pi}(s = B)$ по определению.

Пример 1-3



- Итого $V^{\pi}(s = B) = 0.5 \cdot 4 + 0.5\,(2 + \gamma V^{\pi}(s = B))$
- Откуда $V^{\pi}(s = B) = 5$

Пример 1-4



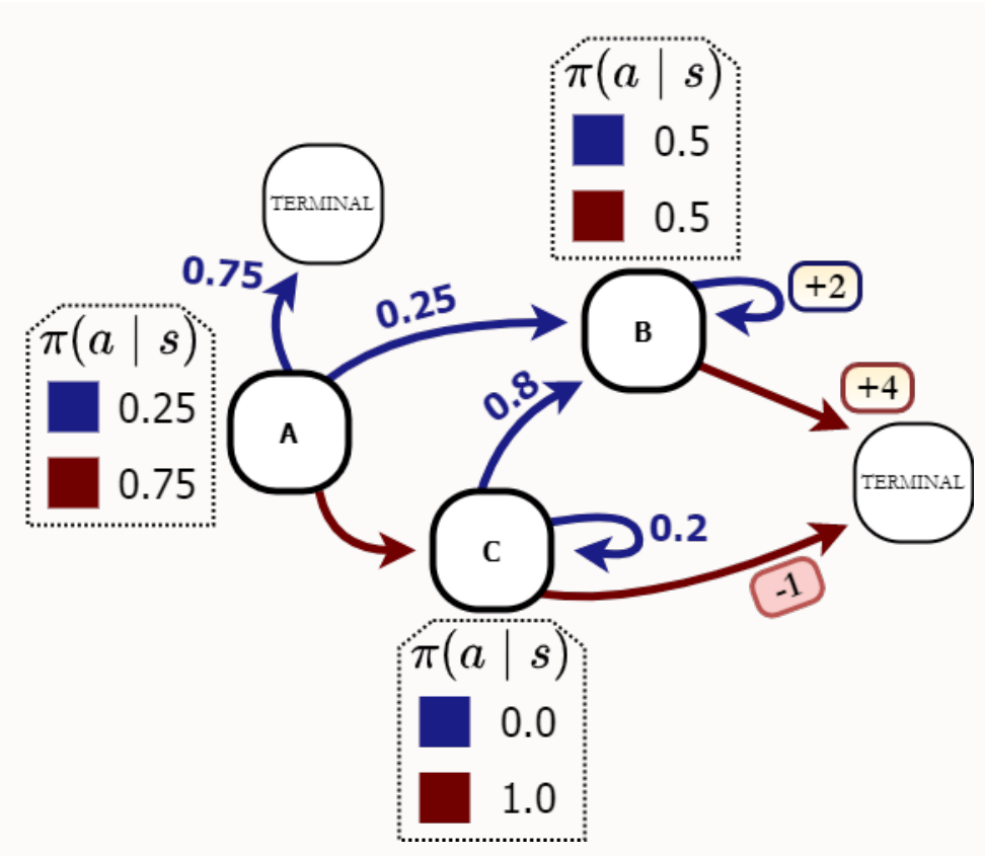
- Для состояния A

$$V^{\pi}(s = A) = 0.25 \cdot (0.75 \cdot 0 + 0.25 \cdot \gamma V^{\pi}(s = B)) + 0.75 \cdot \gamma V^{\pi}(s = C)$$

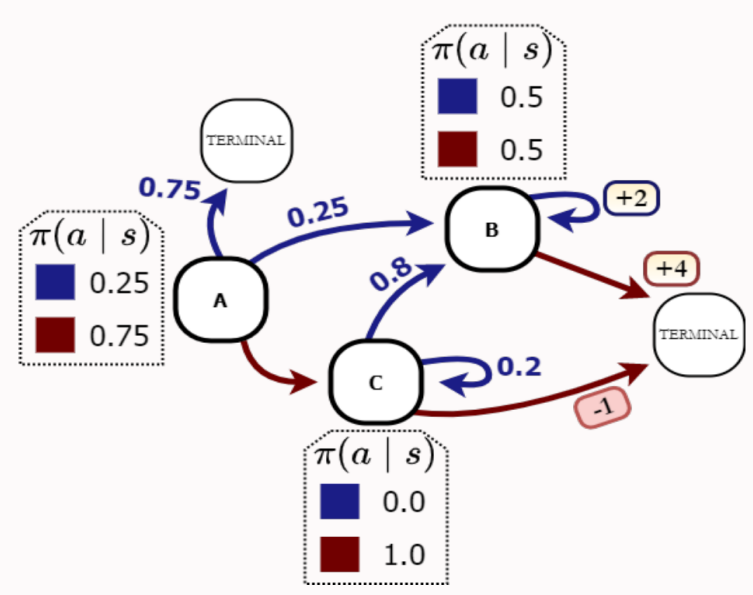
- Откуда $V^{\pi}(s = A) = -0.35$

Пример 2-1 (Уравнения Беллмана)

- Выпишем уравнения Беллмана для MDP и стратегии π из примера 1. Число уравнений совпадает с числом состояний



Пример 2-2

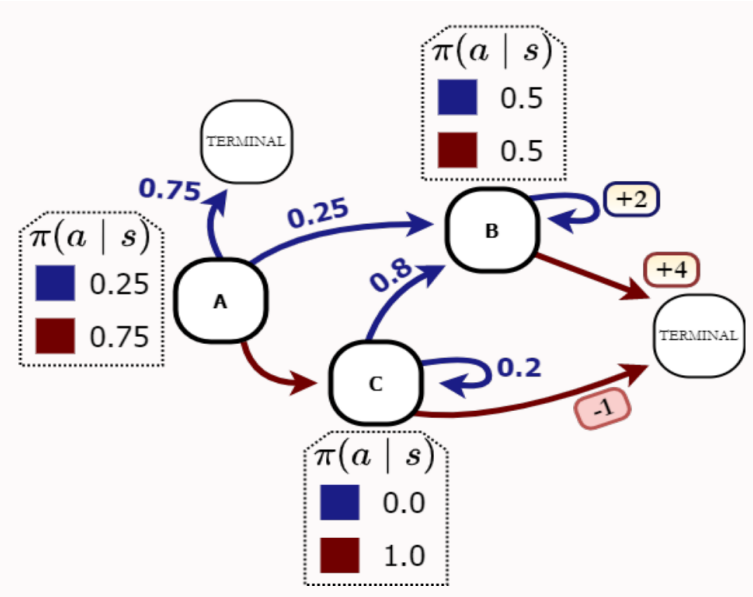


- Уравнение для состояния A:

$$V^{\pi}(A) = 0.25\,(0 + 0.25\gamma V^{\pi}(B)) + 0.75 \cdot (0 + \gamma V^{\pi}(C))$$

- С вероятностью 0.25 выбираем действие “c”, затем дисконтирование γ; с вероят. 0.75 эпизод закончится и будет выдана нулевая награда, с вероятн. 0.25 агент перейдет в состояние B.

Пример 2-3

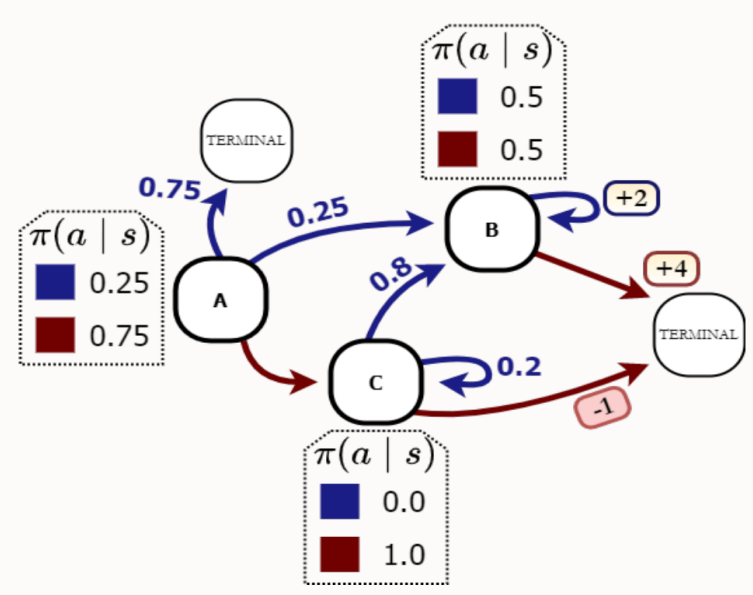


- Уравнение для состояния A (получено выше):

$$V^{\pi}(A) = 0.25\,(0 + 0.25\gamma V^{\pi}(B)) + 0.75 \cdot (0 + \gamma V^{\pi}(C))$$

- Второе слагаемое уравнения будет отвечать выбору действия “к”; агент тогда перейдчт в состояние C и, начиная со следующего шага, получит в будущем $V^{\pi}(C)$.

Пример 2-4



- Аналогично расписываются два оставшихся уравнения.

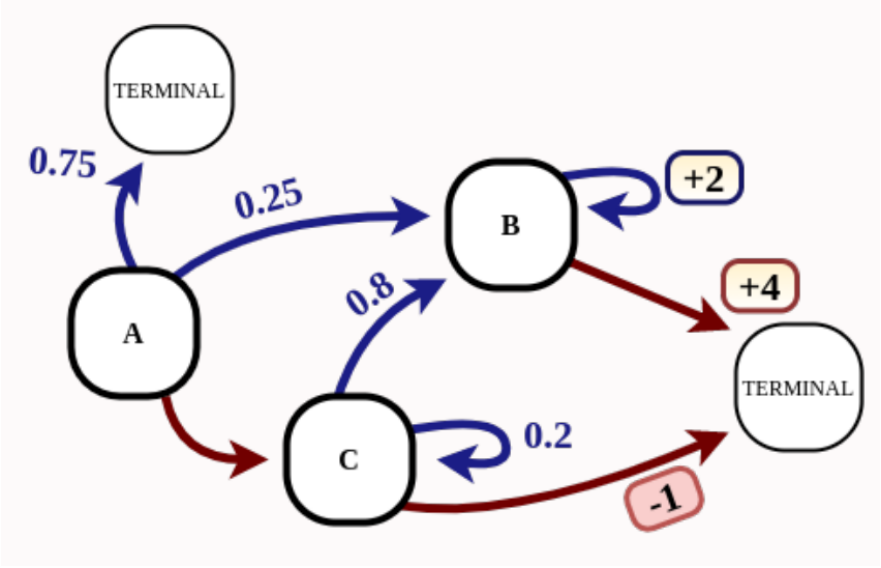
$$V^{\pi}(A) = 0.25^2 \cdot \gamma V^{\pi}(B) + 0.75 \cdot \cdot \gamma V^{\pi}(C)$$

$$V^{\pi}(B) = 0.5\,(2 + \gamma V^{\pi}(B)) + 0.5 \cdot 0.4$$

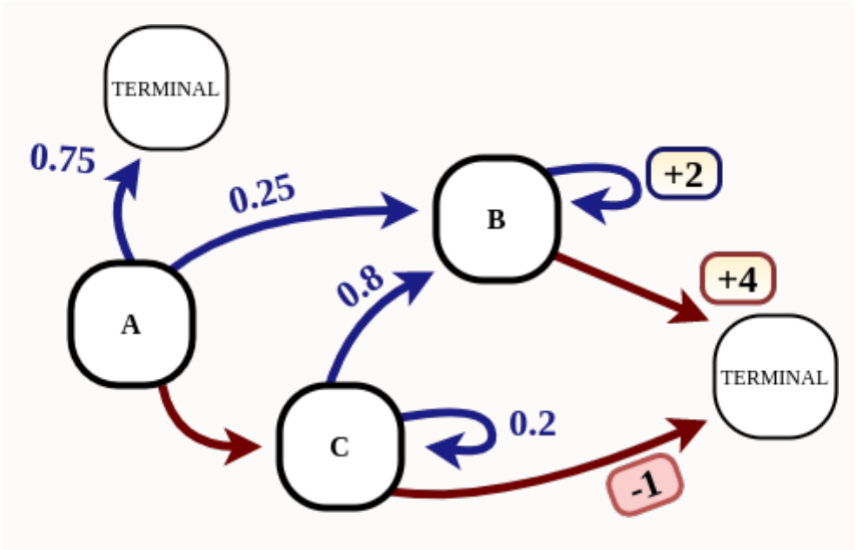
$$V^{\pi}(C) = -1$$

Пример 3-1 (Оптимальная стратегия)

- Q-функция получает на вход пару состояние-действие и ничего не говорится о том, что это действие должно быть как-то связано с оцениваемой стратегией π

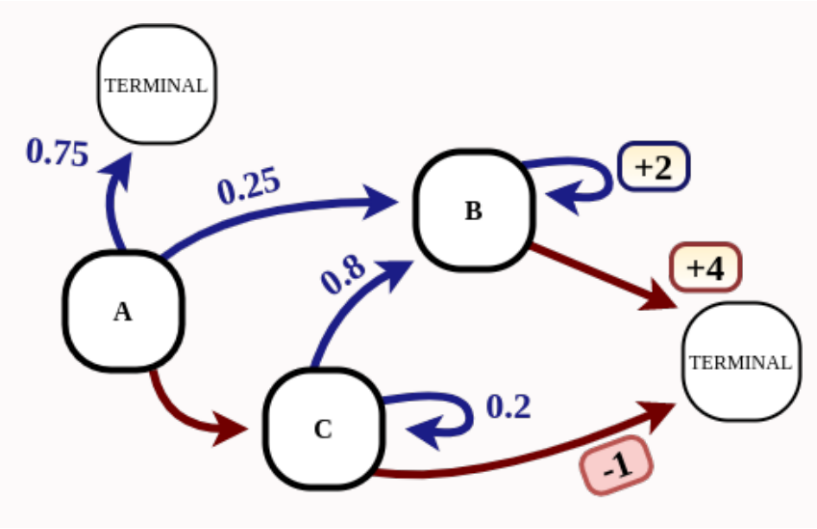


Пример 3-2



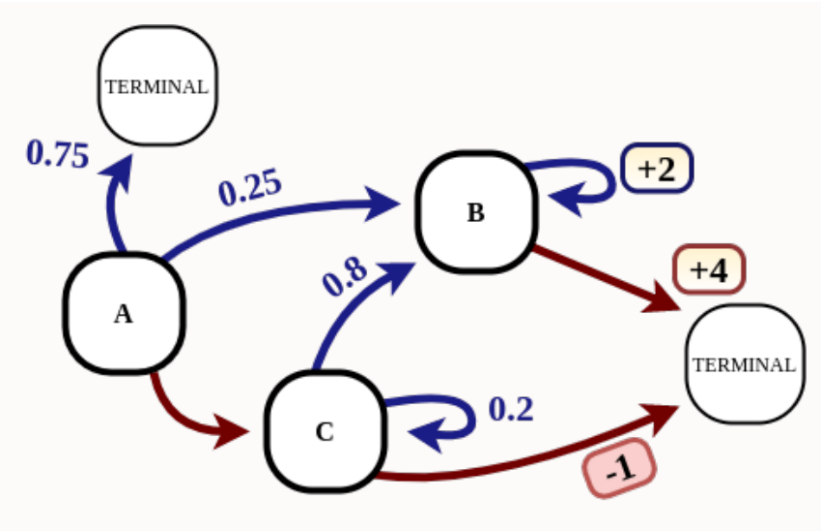
- В MDP рассмотрим стратегию π, которая всегда детерминировано выбирает действие “к”.
- Тем не менее можем вычислить $Q^{\pi}(s,\text{“c”})$ для любых состояний (например, для терминальных это значение формально равно нулю)

Пример 3-3



- $Q^{\pi}(s = A, \text{“}c\text{”}) = 0.25\gamma V^{\pi}(s = B)$
- $Q^{\pi}(s = B, \text{“}c\text{”}) = 2 + \gamma V^{\pi}(s = B)$
- $Q^{\pi}(s = C, \text{“}c\text{”}) = 0.8\gamma V^{\pi}(s = B) + 0.2\gamma V^{\pi}(s = C)$

Пример 3-4

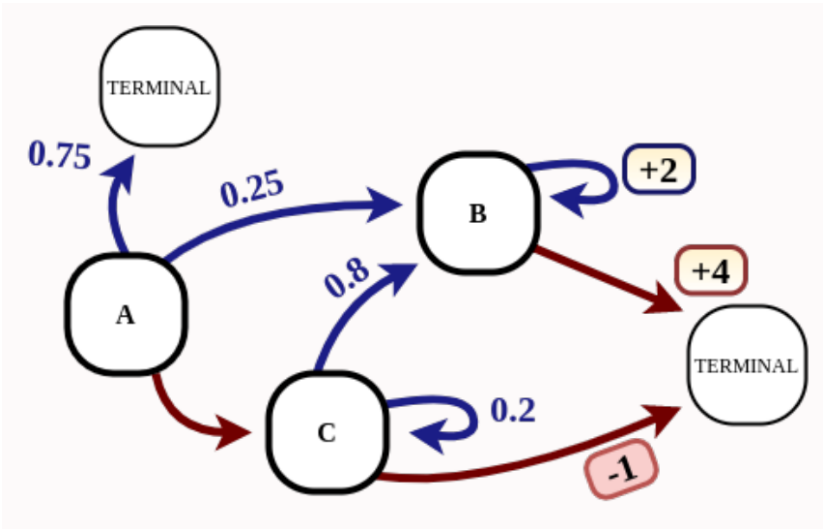


Внутри V^π сидит дальнейшее поведение при помощи стратегии π, то есть выбор исключительно действий “k”: соответственно,

$$V^\pi(s = B) = 4, V^\pi(s = C) = -1$$

Пример 4-1

- Сформулируем для MDP уравнения оптимальности Беллмана для V^*. Мы получим систему из трҷх уравнений с трҷмя неизвестными.



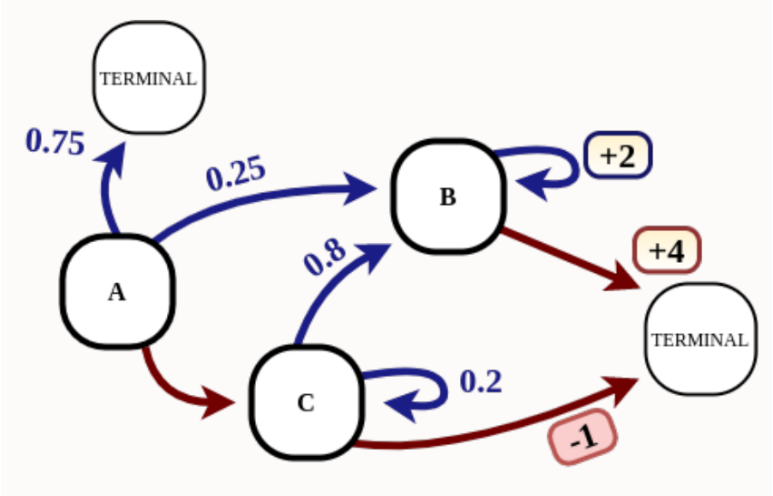
Пример 4-2

- Уравнения оптимальности Беллмана

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'} Q^*(s',a'),$$

$$V^*(s) = \max_a [r(s,a) + \mathbb{E}_{s'} V^*(s')]$$

Пример 4-3



$$V^*(s = A) = \max\left(0.25 \cdot \gamma V^*(s = B),\ \gamma V^*(s = C)\right)$$

$$V^*(s = B) = \max\left(2 + \gamma V^*(s = B),\ 4\right)$$

$$V^*(s = C) = \max\left(0.8 \cdot \gamma V^*(s = B) + 0.2 \cdot \gamma V^*(s = C),\ -1\right)$$

Пример 4-4

$$V^*(s = A) = \max\left(0.25 \cdot \gamma V^*(s = B),\ \gamma V^*(s = C)\right)$$

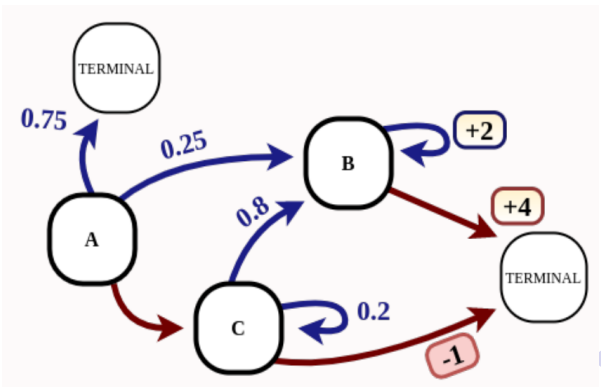
$$V^*(s = B) = \max\left(2 + \gamma V^*(s = B),\ 4\right)$$

$$V^*(s = C) = \max\left(0.8 \cdot \gamma V^*(s = B) + 0.2 \cdot \gamma V^*(s = C),\ -1\right)$$

Заметим, что в полученных уравнениях не присутствует мат.ожиданий по самим оптимальным стратегиям - предположение дальнейшей оптимальности поведения по сути «заменяет» их на взятие максимума по действиям. Вместо поиска оптимальной стратегии можно искать оптимальные оценочные функции.

Пример 5-1

- Найдем оптимальные стратегии в MDP для $\gamma = 0.5$, используя критерий оптимальности Беллмана
- В состоянии B оптимально или выбирать какое-то одно из двух действий с вероятностью 1, или действия эквивалентны, и тогда оптимально любое поведение.
- Допустим, мы будем выбирать всегда “c”, тогда $2/(1-\gamma) = 4$; (из $V^*(s = B) = \max(2 + \gamma V^*(s = B), 4)$)
- если же будем выбирать “k”, то получим +4. Действия эквивалентны и $V^*(s = B) = 4$.

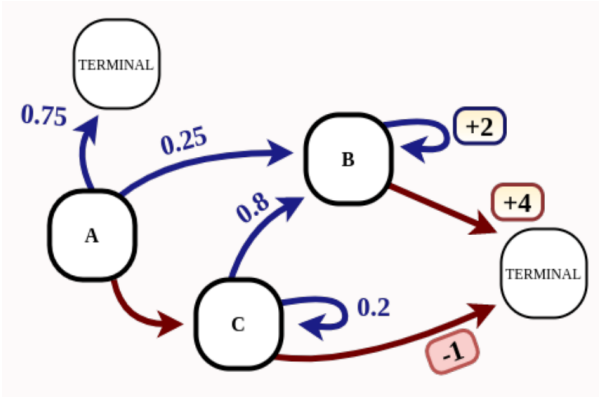


Пример 5-2

- Аналогичное рассуждение для состояния C. Если опти-мально действие “c”, то

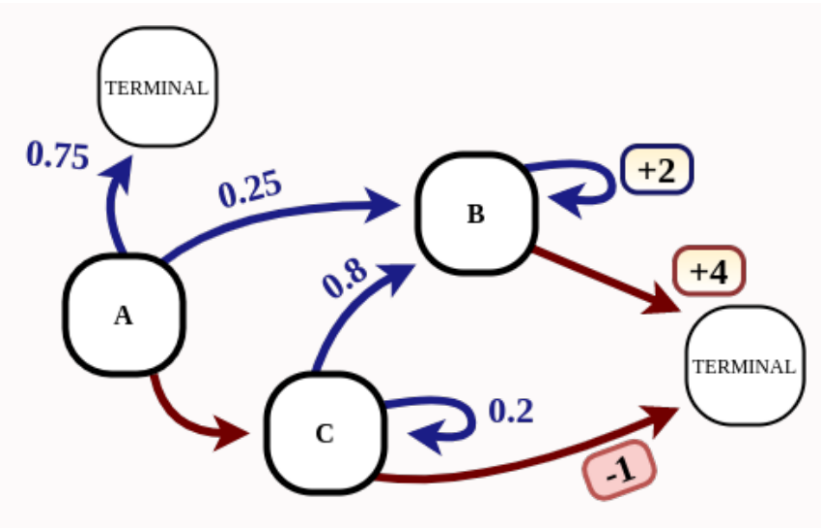
$$Q^*(s = C, a = \text{“}c\text{”}) = 0.2\gamma Q^*(s = C, a = \text{“}c\text{”}) + 0.8\gamma V^*(s = B)$$

- Решая уравнение относительно неизвестного $Q^*(s = C, a = \text{“}c\text{”})$, получаем $16/9 > Q^*(s = C, a = \text{“}k\text{”}) = -1$. Значит в $s = C$ оптимальная стратегия обязана выбирать “c” и $V^*(C) = 16/9$.



Пример 5-3

- Для состояния A достаточно сравнить $Q^*(s = A, a = \text{“}c\text{”}) = 0.25 \cdot \gamma V^*(s = B) = 1/4$ и $Q^*(s = A, a = \text{“}k\text{”}) = \gamma V^*(s = C) = 8/9$. Опт. стратегий “k”.



Монте-Карло подход к RL - оценка стратегии

- Цель: построить V^π по эпизодам взаимодействия по стратегии π

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

- Отдача (return) - суммарное вознаграждение

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

- Функция полезности - это мат.ожидание отдачи

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

- Монте-Карло оценка стратегии использует эмпирическое среднее вместо мат. ожидания

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 первую отдачу: $V(s) = N(s)/S(s)$
- По закону больших чисел

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

- Смещенная состоятельная оценка с большой дисперсией

Монте-Карло: среднее приращение

Средние последовательности могут вычисляться последовательно:

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

Монте-Карло для приращений

- Обновим $V(s)$ с приращением для эпизода $s_1, a_1.r_1, \ldots, s_T$
- 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 можно 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, TD) обучается напрямую по эпизодам взаимможействия со средой
- TD - безмодельный подход (model-free): модель переходов MDP и функция вознаграждения не известны
- TD обучается по эпизодам, используя бутстреп
- TD приближает значение на основе предыдущего приближения

Монте-Карло и TD

- Цель: построить V^π интерактивно (online) по эпизодам взаимодействия по стратегии π
- M-K с каждым посещением для приращений: обновляем $V(s_t)$ на основе текущей отдачи R_t

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

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

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

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

Монте-Карло и TD

- TD может обучаться до того, как стала известна итоговая отдача
  - TD может обучаться интерактивно на каждом шаге
  - M-K должен дождаться окончания эпизода, когда становится известной отдача
- TD может обучаться без информации об итоговой отдаче
  - TD может обучаться на неполных эпизодах
  - M-K может обучаться только на полных эпизодах
  - TD работает и для бесконечных окружениях (без терминальных вершин)
  - M-K работает только в эпизодических окружениях (с терминальными вершинами)

Смещенность и дисперсия

- Отдача $R_t = r_{t+1} + \gamma r_{t+2} + ... + \gamma^{T-1} r_T$ является несмещенной оценкой для $V^\pi(s)$
- Истинный TD показатель $r_{t+1} + \gamma V^\pi(s_{t+1})$ является несмещенной оценкой для $V^\pi(s)$
- TD показатель $r_{t+1} + \gamma V(s_{t+1})$ является смещенной оценкой для $V^\pi(s)$
- TD показатель имеет меньшую дисперсию, чем отдача:
  - отдача зависит от *большого количества* случайных действий, переходов, вознаграждений
  - показатель зависит от *одного* случайного действия, перехода, вознаграждения

Смещенность и дисперсия

- M-K обладает большей дисперсией и нулевым смещением:
  - не очень сильно зависит от начального приближения
  - очень прост для понимания и использования
- TD обладает меньшей дисперсией и ненулевым смещением:
  - обычно более эффективен, чем M-K
  - TD(0) сходится к $V^{\pi}(s)$
  - более чувствителен к начальному приближению

Пакетные M-K и TD

- M-K и TD сходятся к $V(s) \to V^{\pi}(s)$, если опыт $\to \infty$
- А если применим пакетный (batch) подход для конечного опыта?

$$s_1^1, a_1^1.r_1^1, ..., s_T^1$$

$$\dots$$

$$s_1^K, a_1^K.r_1^K, ..., s_T^K$$

- Например, многократно выбирать эпизод $k \in [1, K]$
- Применять M-K или TD к эпизоду k

Пример: AB

Два состояния A и B, нет дисконтирования, 8 эпизодов опыта, вознаграждения 0 и 1:

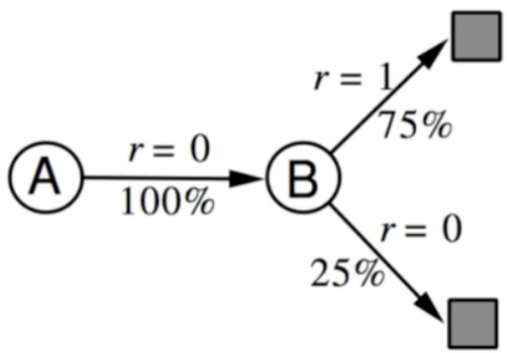
A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0

Какие значения $V(A)$ и $V(B)$?

Пример: AB

Два состояния A и B, нет дисконтирования, 8 эпизодов опыта:

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0



M-K: $V(A) = 0$ и $V(B) = 6/8$ TD: $V(A) = 0.75$ и $V(B) = 6/8$

Эквивалентность

- M-K сходится к решению с минимальной СКО
  - наилучшее приближение к наблюдаемой отдаче

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

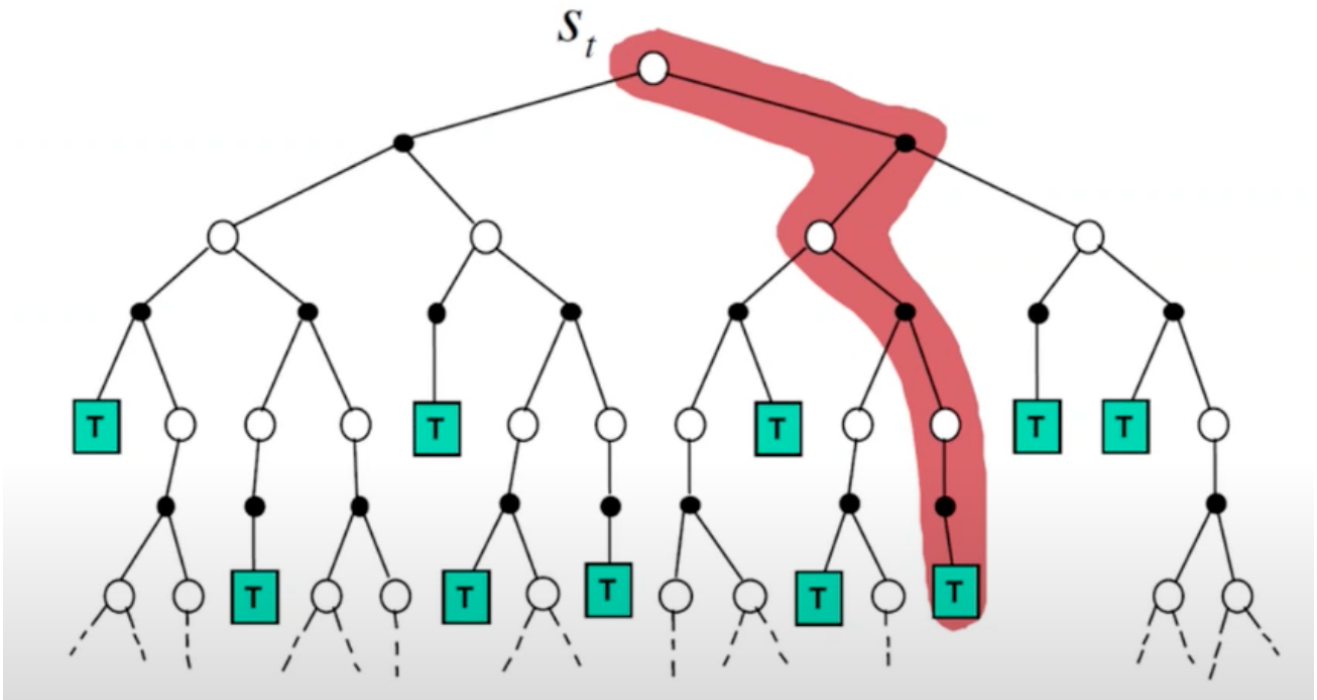
  - для AB примера $V(A) = 0$
- TD сходится к решению максимально правдоподобной марковской модели
  - решение для MDP $\left\langle \mathcal{S}, A, \hat{\mathcal{P}}, \hat{\mathcal{R}}, \gamma \right\rangle$, который лучше всего удовлетворяет данным

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

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

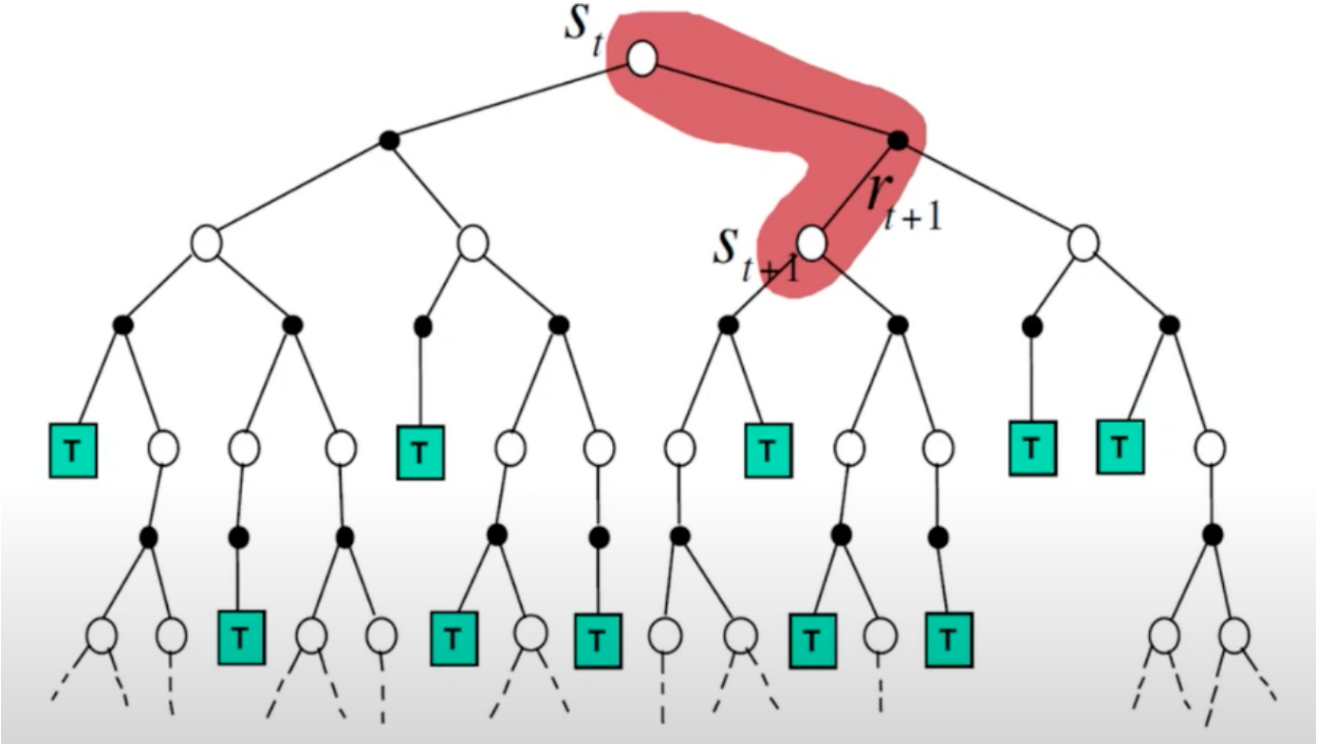
Монте-Карло обновление полезности

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



Обновление полезности TD

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

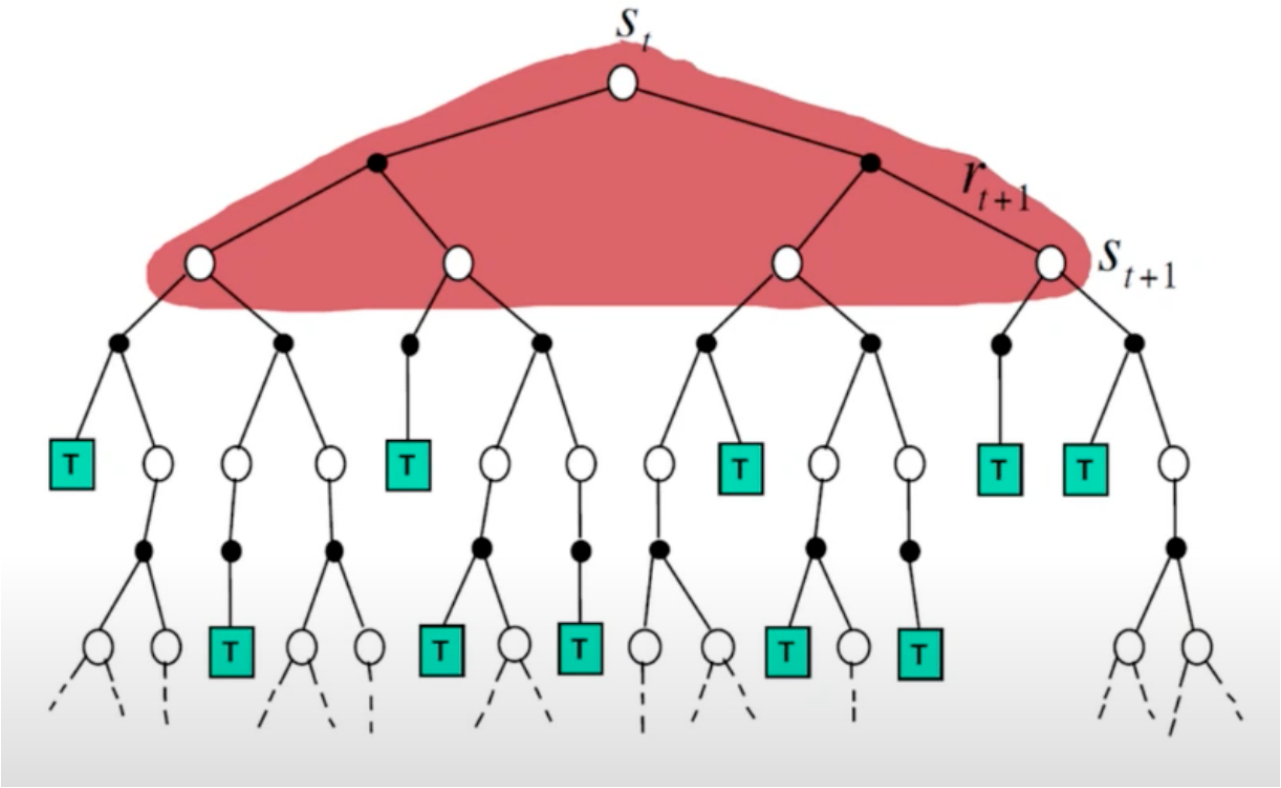


TD (пример)

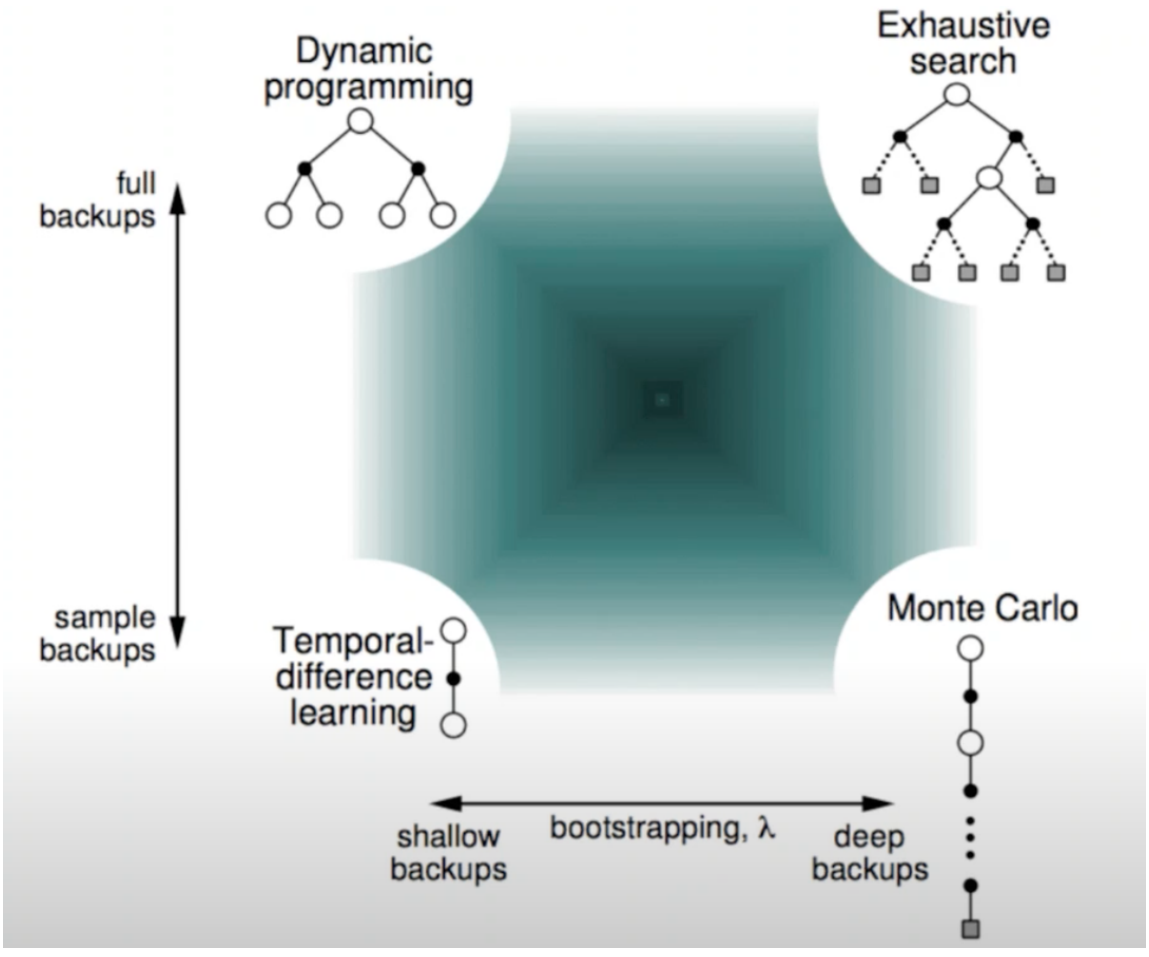
Вы сидите в кафе (s) и хотите вернуться домой. Вы знаете, что в среднем этой займет $-Q(s,a) = 30$ мин. Вы тратите одну минуту ($-r$) на выход из кафе и обнаруживаете пробку ($s\prime$). За сччт этой новой информации вы можете дать более точную оценку времени до возвращения до дома: $-Q(s\prime, a\prime) = 40$ мин. Как можно в будущем откорректировать наши прогнозы? Можно засечь время, сколько в итоге заняла поездка, доиграть эпизод до конца и посчитать Монте-Карло оценку. А можно уже сделать вывод о том, что случилась некоторая «временная разность», ошибка за один шаг, равная $41 - 30 = 11$ мин. Заменять исходное приближение расстояния от кафе до дома $-Q(s,a)$на 41 минуту, конечно же, слишком грубо: но временная разность говорит, что 30 минут было заниженной оценкой и ее надо чуть-чуть увеличить.

Обновление динамического программирования

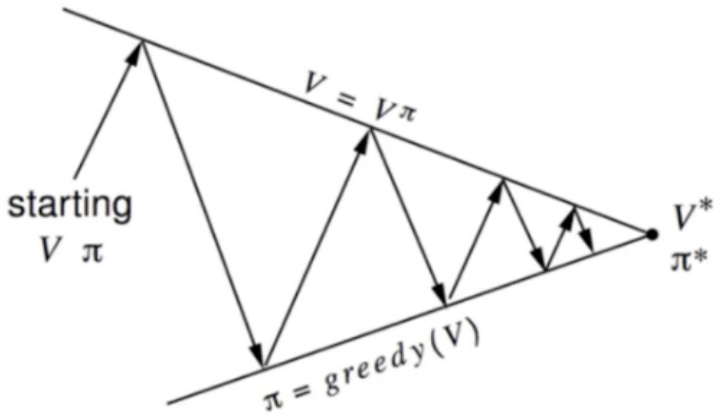
$$V(s_t) \leftarrow \mathbb{E}_\pi \left[r_{t+1} + \gamma V(s_{t+1}) \right]$$

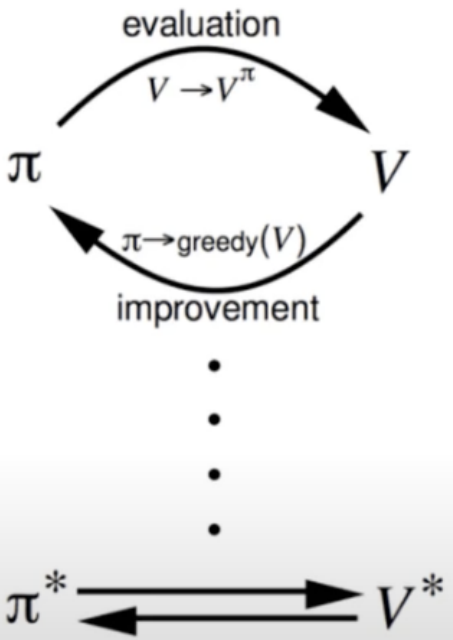


Общая схема


Dynamic programming
Exhaustive search
full backups
sample backups
Temporal-difference learning
Monte Carlo
shallow backups
bootstrapping, λ
deep backups

Итерации по стратегиям





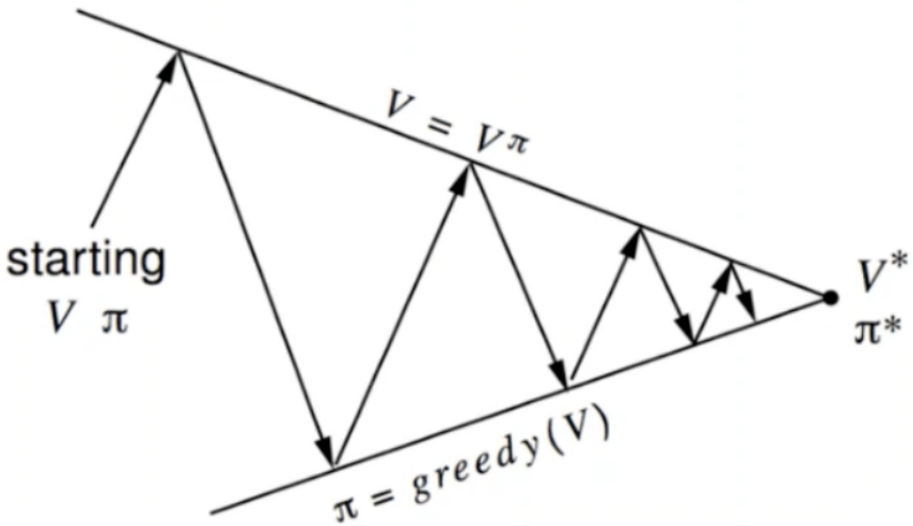
Оценка стратегии – вычисление V^π

Итеративная оценка стратегии

Улучшение стратегии – генерация $\pi' \geq \pi$

Жадное обновление стратегии

Обобщенные итерации по стратегиям с M-K оценкой



Оценка стратегии: M-K оценка стратегии по $V = V^{\pi}$?
Улучшение стратегии: жадное обновление стратегии?

Метод Q-обучения

Аппроксимируем оптимальную функцию ценности действия экспоненциальным скользящим средним:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right)$$

Алгоритм:

1. инициализация стратегии $\pi_1(a|s)$ и состояния среды s_1
2. для всех $t = 1, ..., T, ...$
   1. агент выбирает действие $a_t \sim \pi_t(a|s_t)$, например, $a_t = \arg\max_{a \in A} Q(s_t, a)$ - жадная стратегия;
   2. среда генерирует $r_t \sim p(r|s_t, a_t)$ и $s_{t+1} \sim p(s|s_t, a_t)$;
   3. $Q(s_t, a_t) = Q(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right)$

Метод Q-обучения

- Для каждого состояния введем 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

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

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 - 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*: Попытаться пойти в новый ресторан
- Интернет-банеры
  - *Эксплуатация*: Показать наиболее успешное объявление
  - *Разведка*: Показать различные объявления

Дилемма Разведка-Эксплуатация (3)

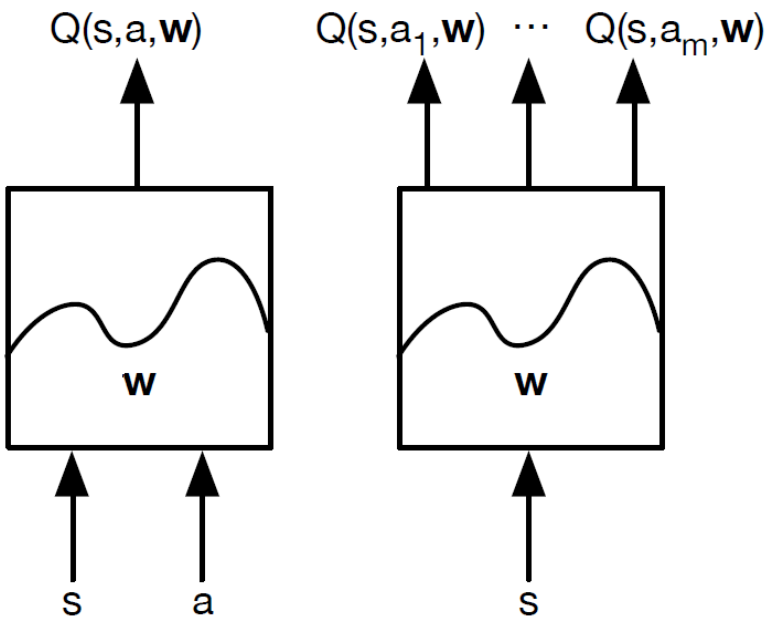
- Бурение нефтяных скважин
  - *Эксплуатация*: Бурение в лучших известных местах
  - *Разведка*: Бурение в лучшем месте
- Игра
  - *Эксплуатация*: Делай ход, в который веришь сам больше всего
  - *Разведка*: Делай экспериментальный ход

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

Q-сеть

- Представим значение функции при помощи Q-сети с весами w
- $Q(s, a, \mathbf{w}) \approx Q^*(s, a)$



Q-обучение

- Оптимальное значение Q удовлетворяет уравнению Беллмана

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

- Используем правую часть $r + \gamma \max_{a'} Q(s',a',\mathbf{w})$ в качестве цели
- Минимизируем MSE-потери градиентным спуском

$$l = \left(r + \gamma \max_{a} Q(s',a',\mathbf{w}) - Q(s,a,\mathbf{w})\right)^2$$

- Сходится к Q^*
- Но расходятся при использовании нейронных сетей из-за
  - корреляции между примерами
  - нестационарной цели

Вопросы

?