Динамическое программирование: итерация по стратегиям и по полезности

Алексей Скрынник

ПЛАН

01 МППР и уравнение Беллмана

02 Алгоритм итерации по стратегиям

03 Алгоритм итерации по полезностям

01

МППР и уравнение Беллмана

Марковский процесс принятия решений

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

Определение

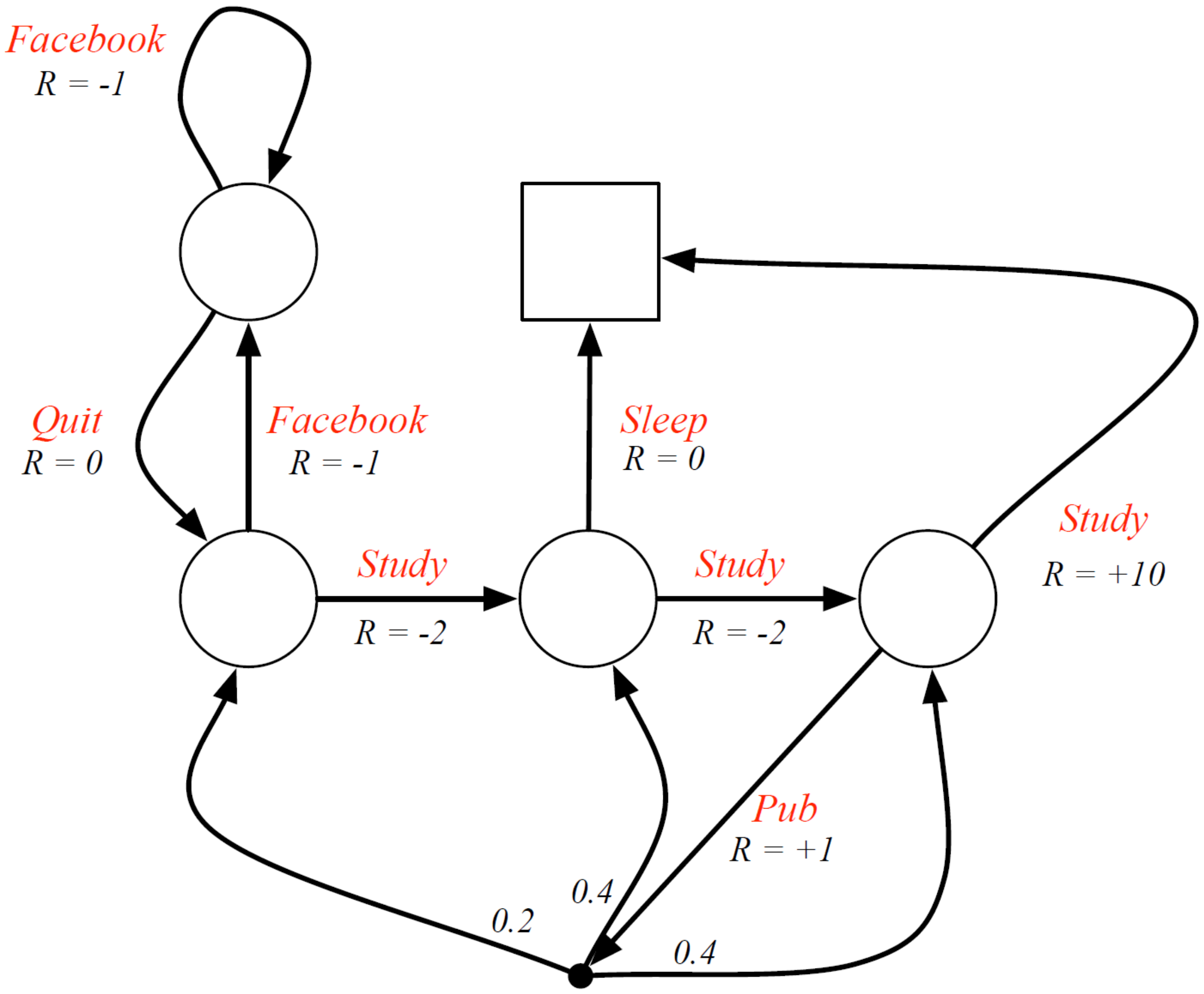
Марковский процесс принятия решений (МППР) – это кортеж $\langle S, A, \mathcal{P}, \mathcal{R}, \gamma \rangle$, где

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

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

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

Пример: студенческий МППР



Стратегии

Определение

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):

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

Стратегии

- Пусть дан МППР $\mathcal{M} = \langle S, A, \mathcal{P}, \mathcal{R}, \gamma \rangle$ и стратегия π
- Последовательность состояний и вознаграждений $s_1, r_1, s_2, \ldots$ – марковского процесса принятия решений $\langle S, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$, где

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

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

Функция Полезности

Функция полезности

Определение

Функцией полезности состояний МППР $V^\pi(s)$ будем называть математическое ожидание отдачи, полученной, начиная с состояния s, при выполнении стратегии π:

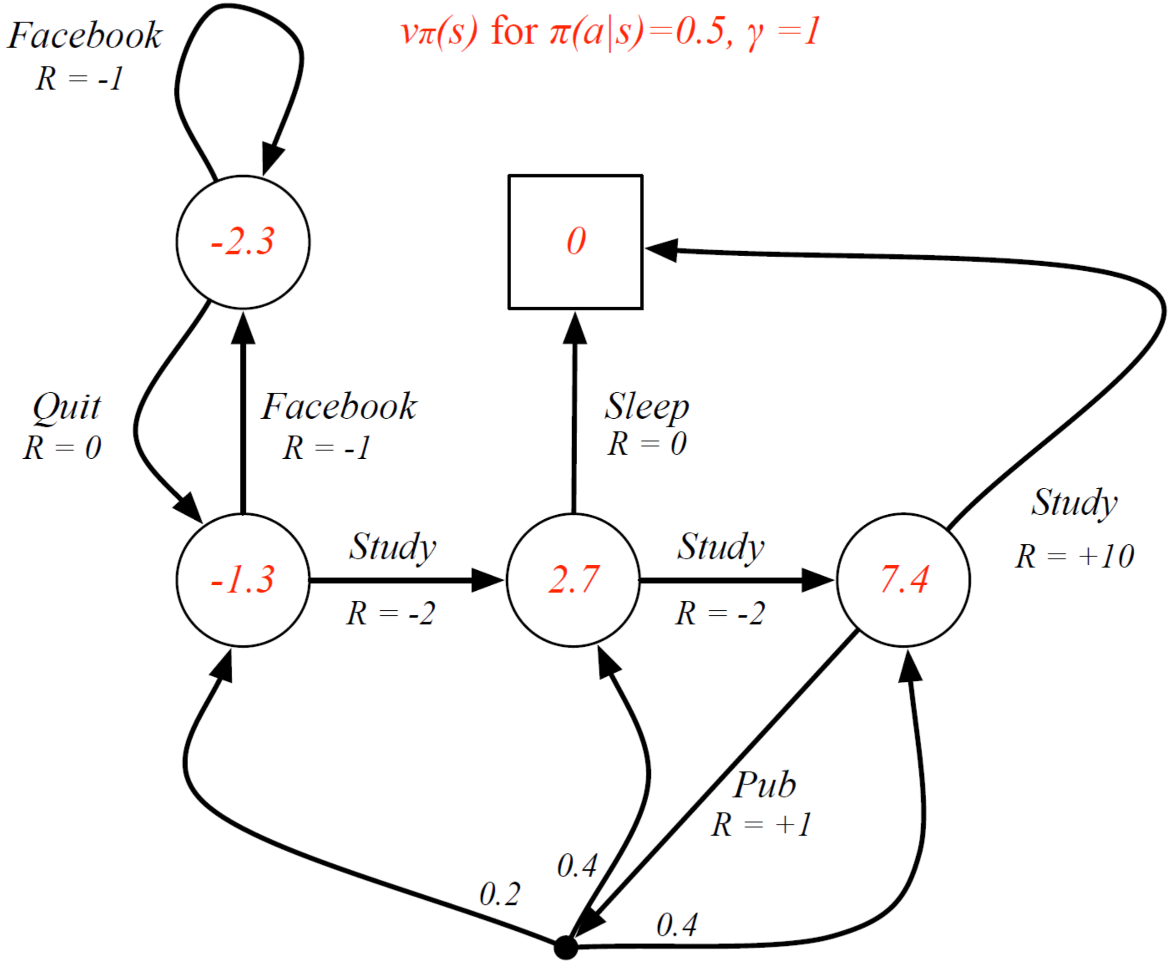
$$V^\pi(s) = \mathbb{E}_\pi[R_t | s_t = s]$$

Определение

Функцией полезности действий МППР $Q^\pi(s, a)$ будем называть математическое ожидание отдачи, полученной, начиная с состояния s и выбранного действия a, при выполнении стратегии π:

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

Пример: функция полезности для студенческого МППР



Уравнение Беллмана для МППР

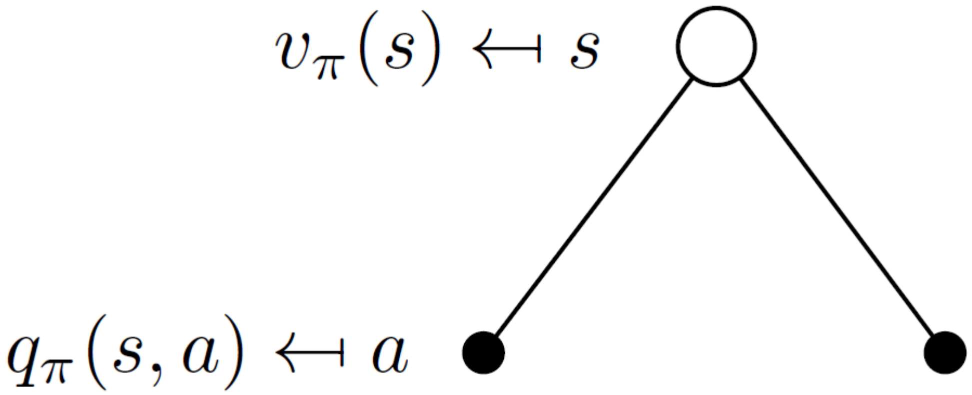
Функция полезности состояний может быть разложена на немедленное вознаграждение и дисконтированную ценность следующего состояния:

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

Для функции полезности действий аналогично:

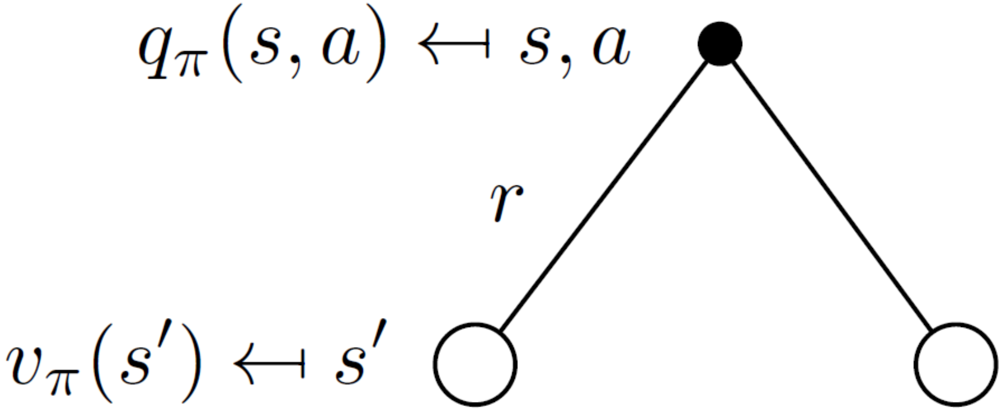
$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) | s_t = s, a_t = a]$$

Уравнение Беллмана для V^π



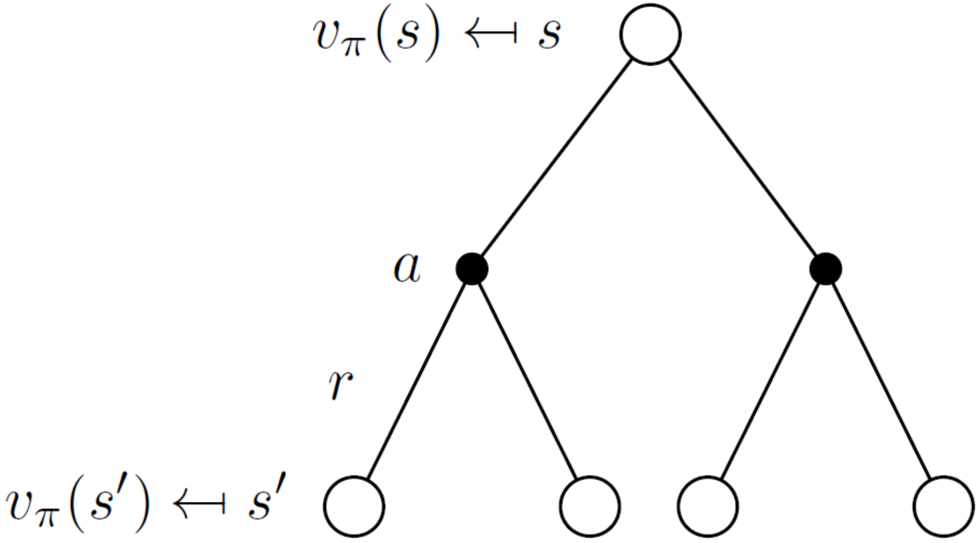
$$V^\pi(s) = \sum_{a \in A} \pi(a|s) Q^\pi(s, a)$$

Уравнение Беллмана для Q^π



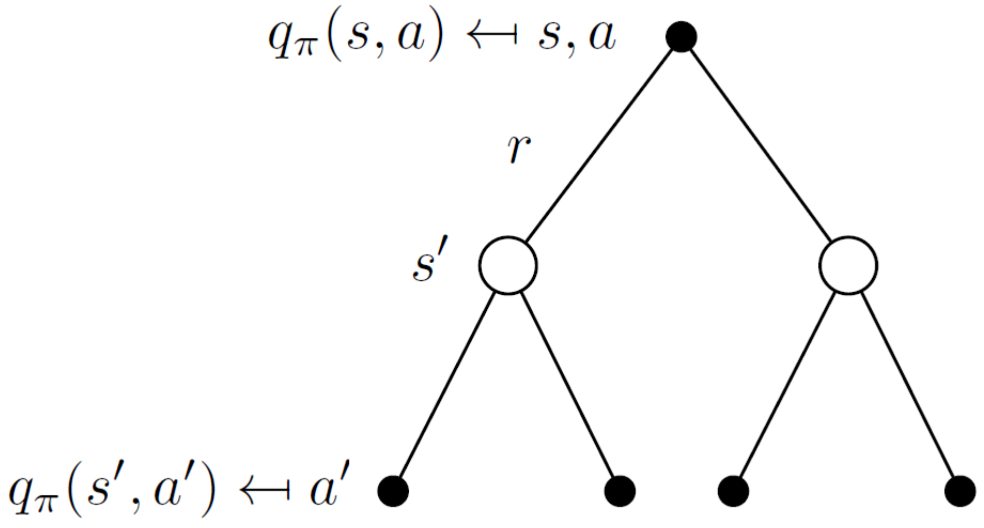
$$Q^\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a V^\pi(s')$$

Уравнение Беллмана для V^π



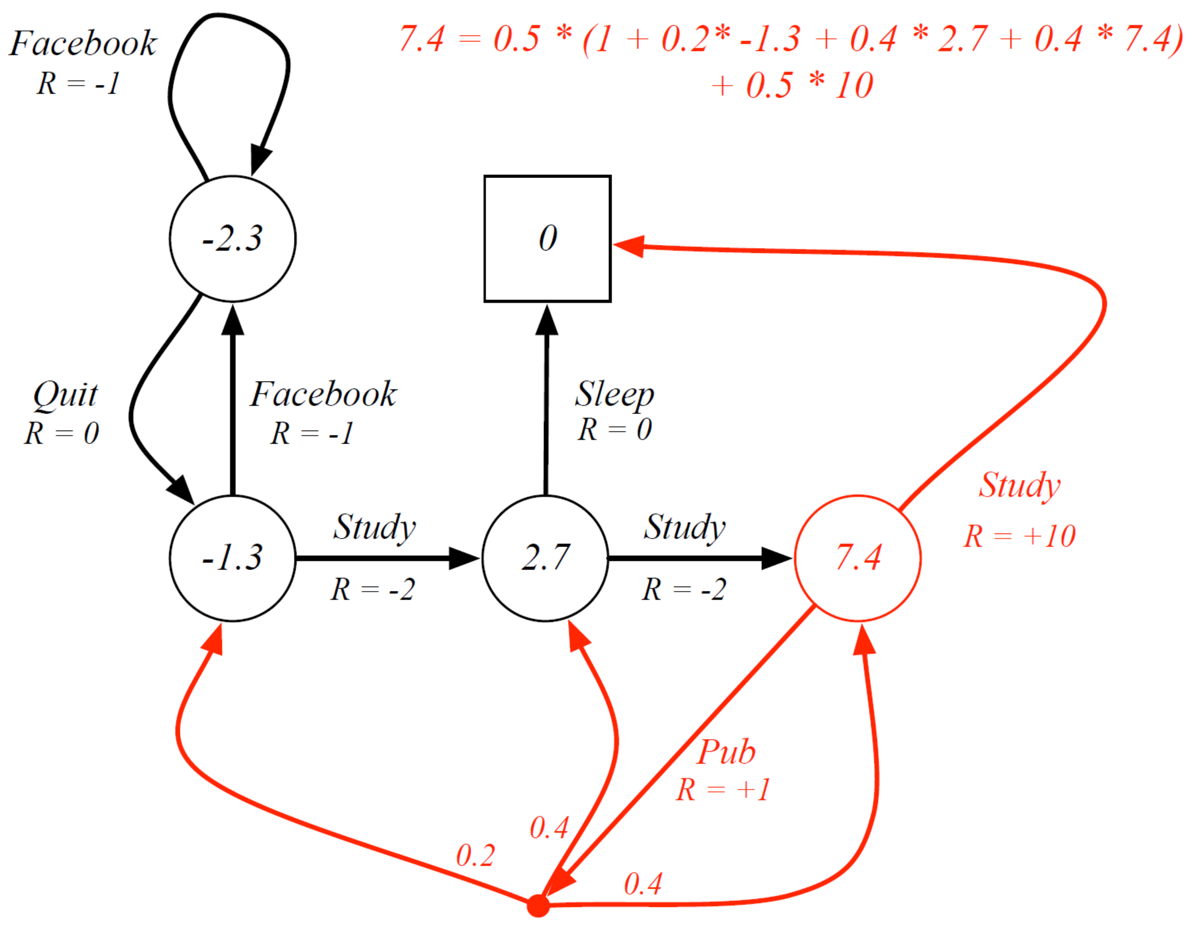
$$V^\pi(s) = \sum_{a \in A} \pi(a|s)\Big(\mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a V^\pi(s')\Big)$$

Уравнение Беллмана для Q^π



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

Пример: уравнение Беллмана для студенческого МППР



Матричная форма уравнения Беллмана для МППР

Уравнение Беллмана с матожиданиями может быть кратко записано:

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

Аналитическое решение:

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

Оптимальная функция полезности и стратегия

Оптимальная функция полезности

Определение

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

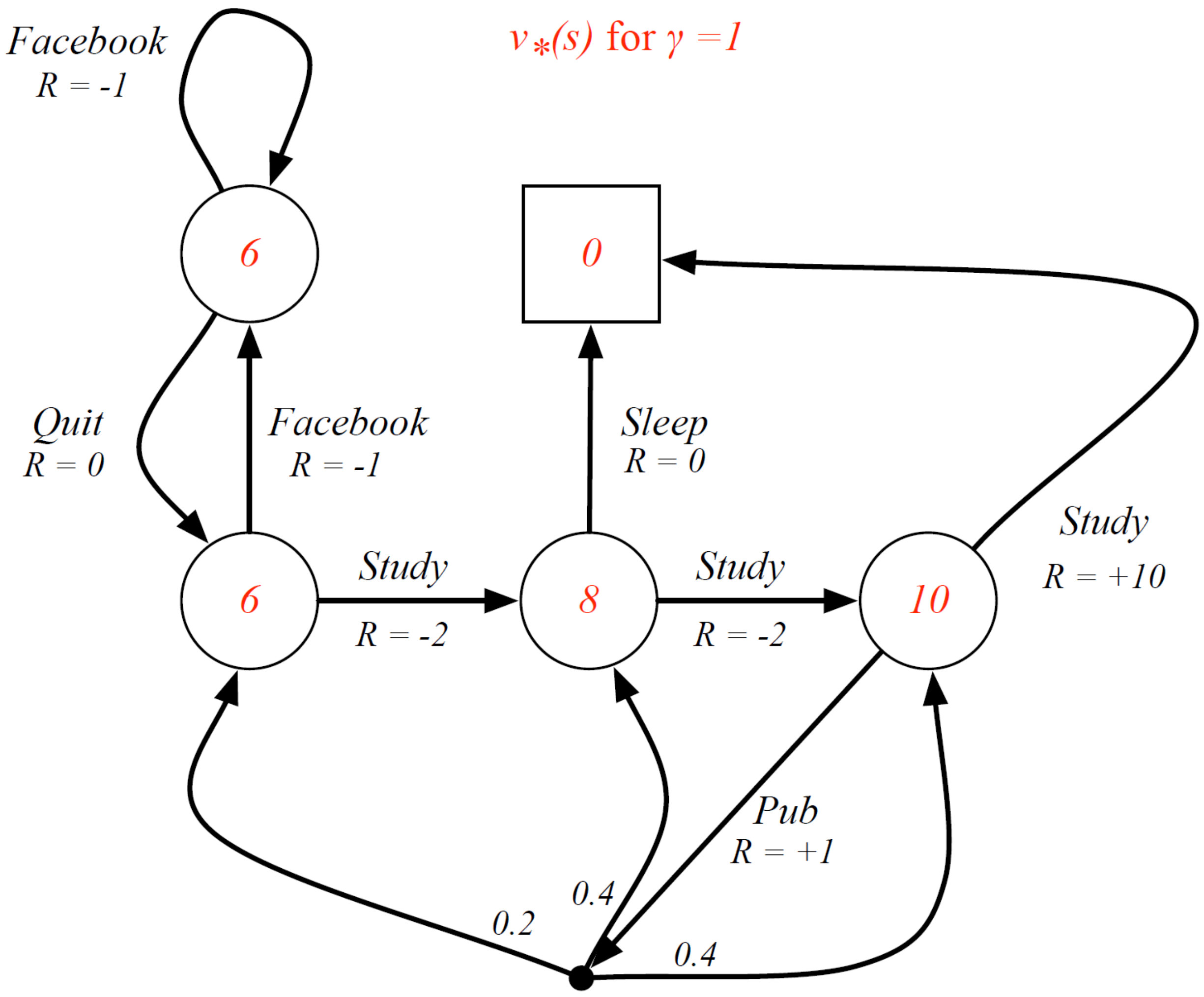
$$V^*(s) = \max_{\pi} V^{\pi}(s).$$

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

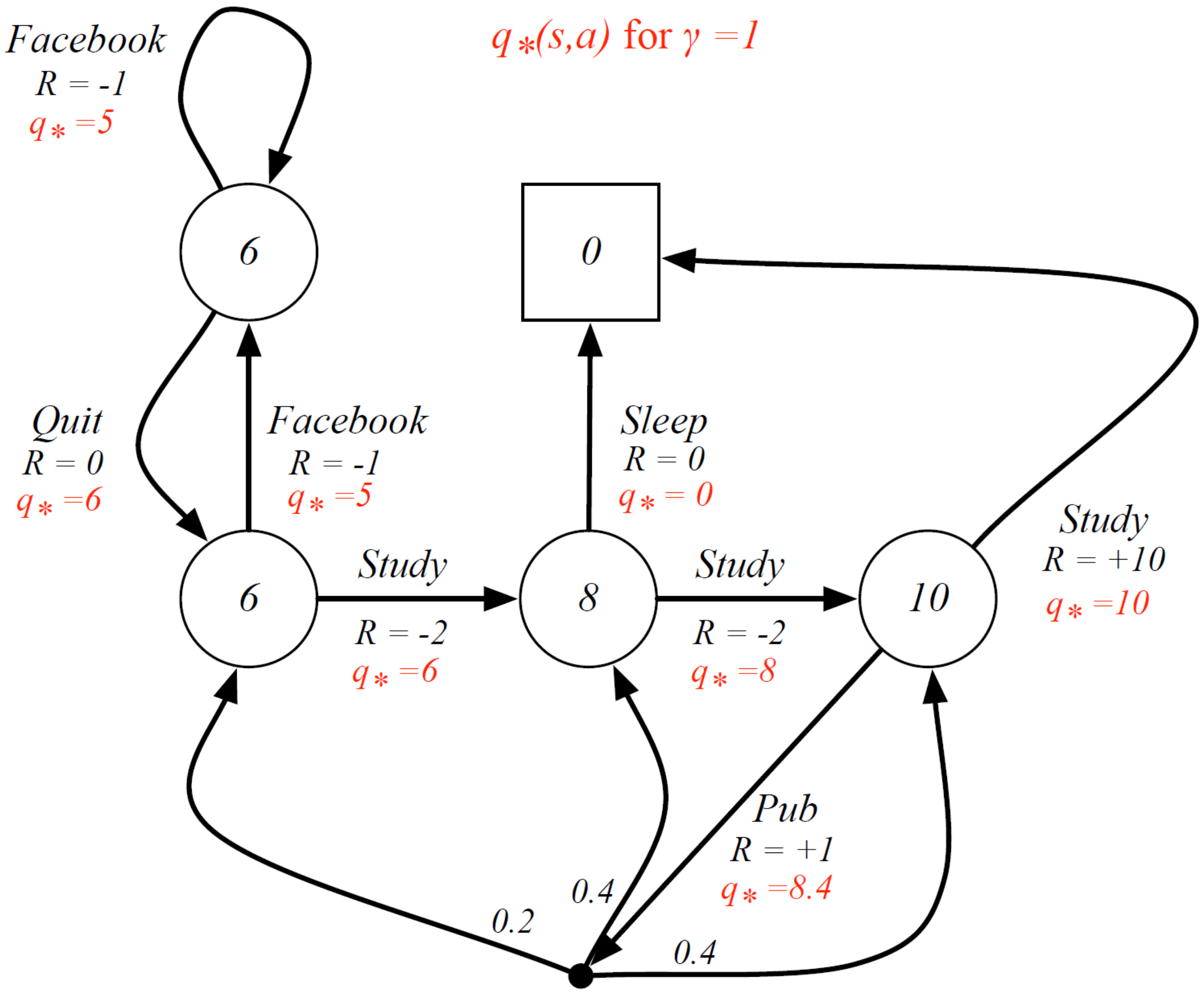
$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a).$$

- Оптимальные стратегии характеризуют лучшее поведение в MDP.
- Говорят, что задача MDP решена, когда найдена оптимальная функция полезности.

Пример: V^* для студенческого МППР



Пример: Q^* для студенческого МППР



Оптимальная стратегия

Определим частичный порядок на множестве стратегий:

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

Теорема

Для любого 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: $Q^{\pi^*}(s, a) = Q^*(s, a)$

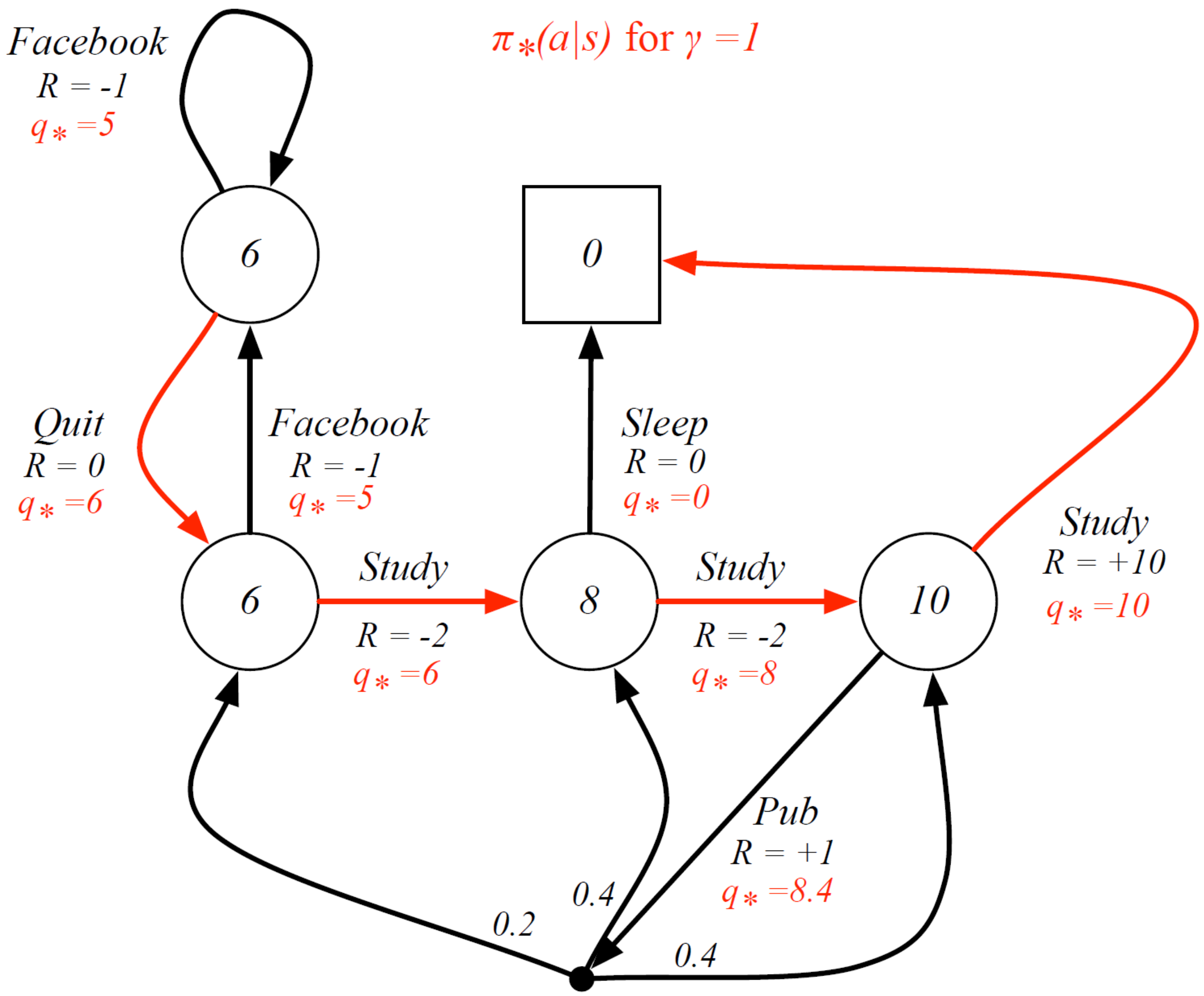
Поиск оптимальной стратегии

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

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

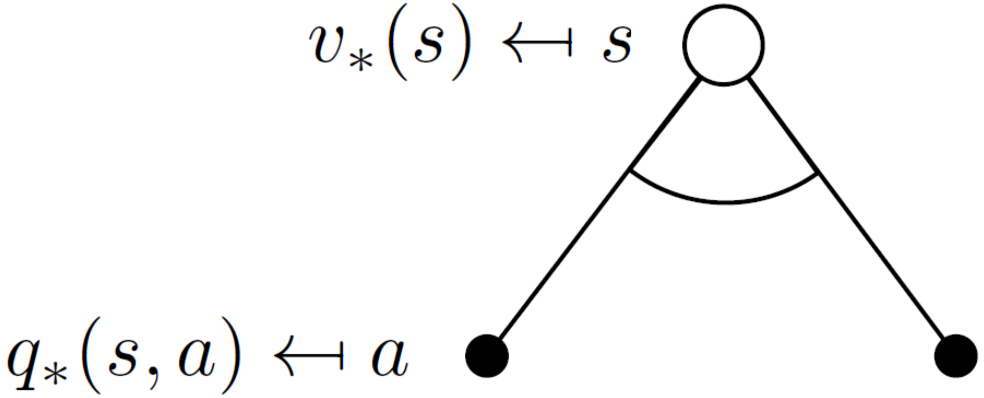
- Для любого MDP существует оптимальная детерминированная стратегия
- Если известна $Q^*(s,a)$, то мы одновременно получаем и оптимальную стратегию

Пример: оптимальная стратегия для студенческого МППР



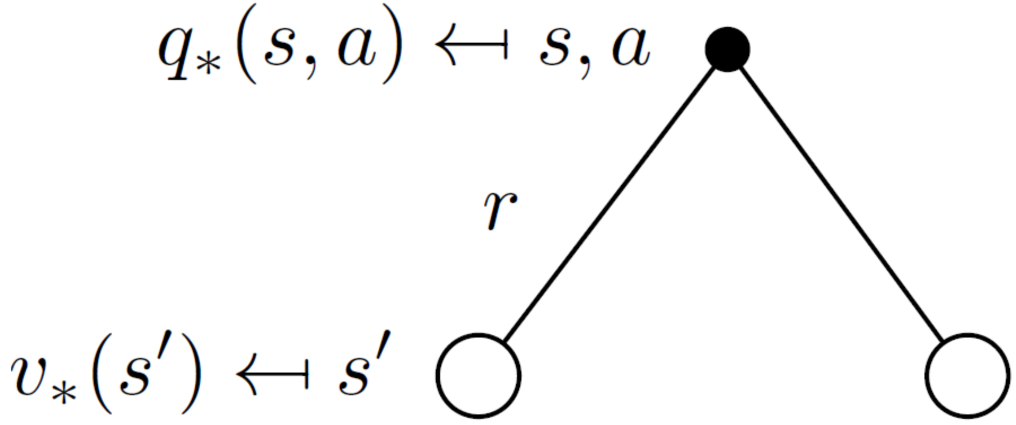
Уравнение Беллмана для V^*

Оптимальные значения функции полезности связаны уравнением оптимальности Беллмана:



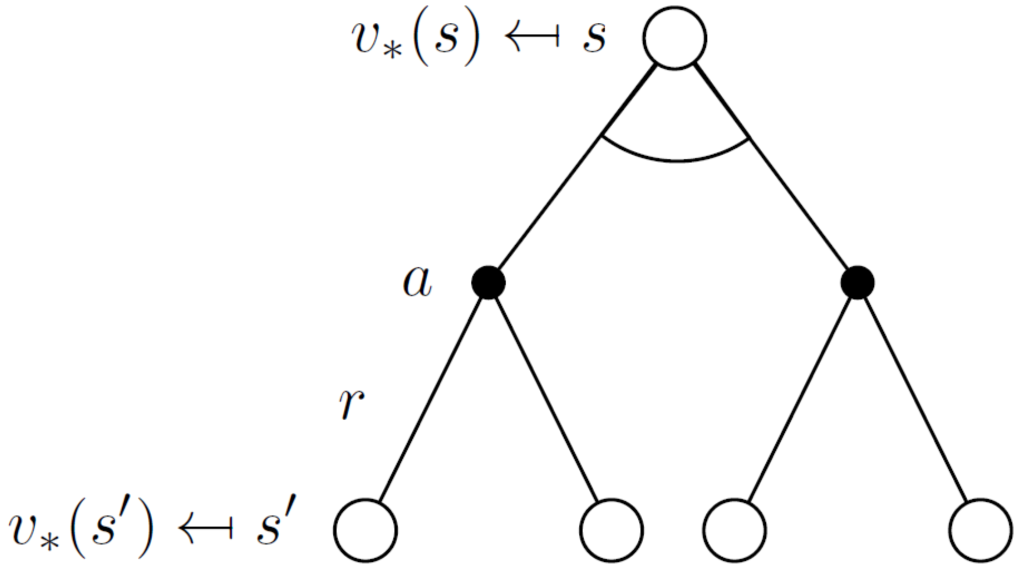
$$V^*(s) = \max_{a \in A} Q^*(s, a)$$

Уравнение Беллмана для Q^*



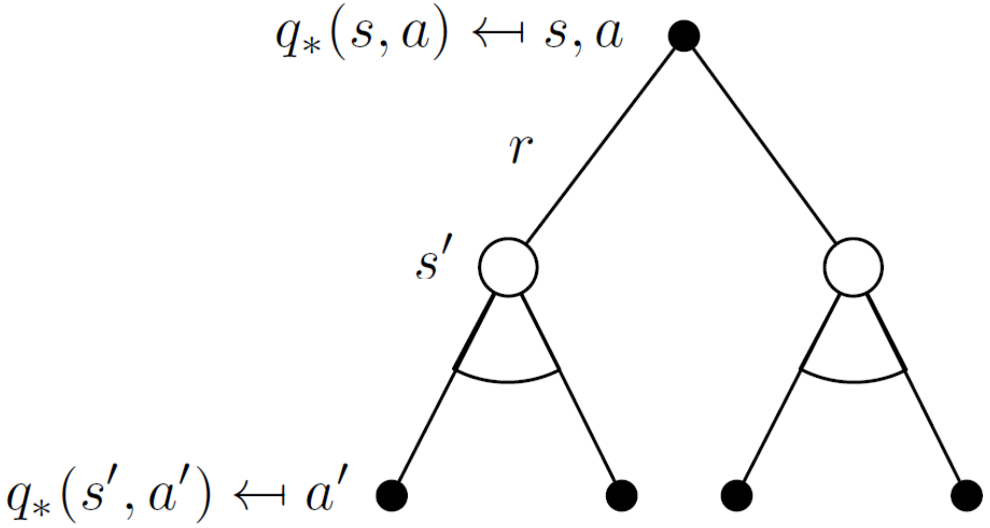
$$Q^*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a V^*(s')$$

Уравнение Беллмана для V^*



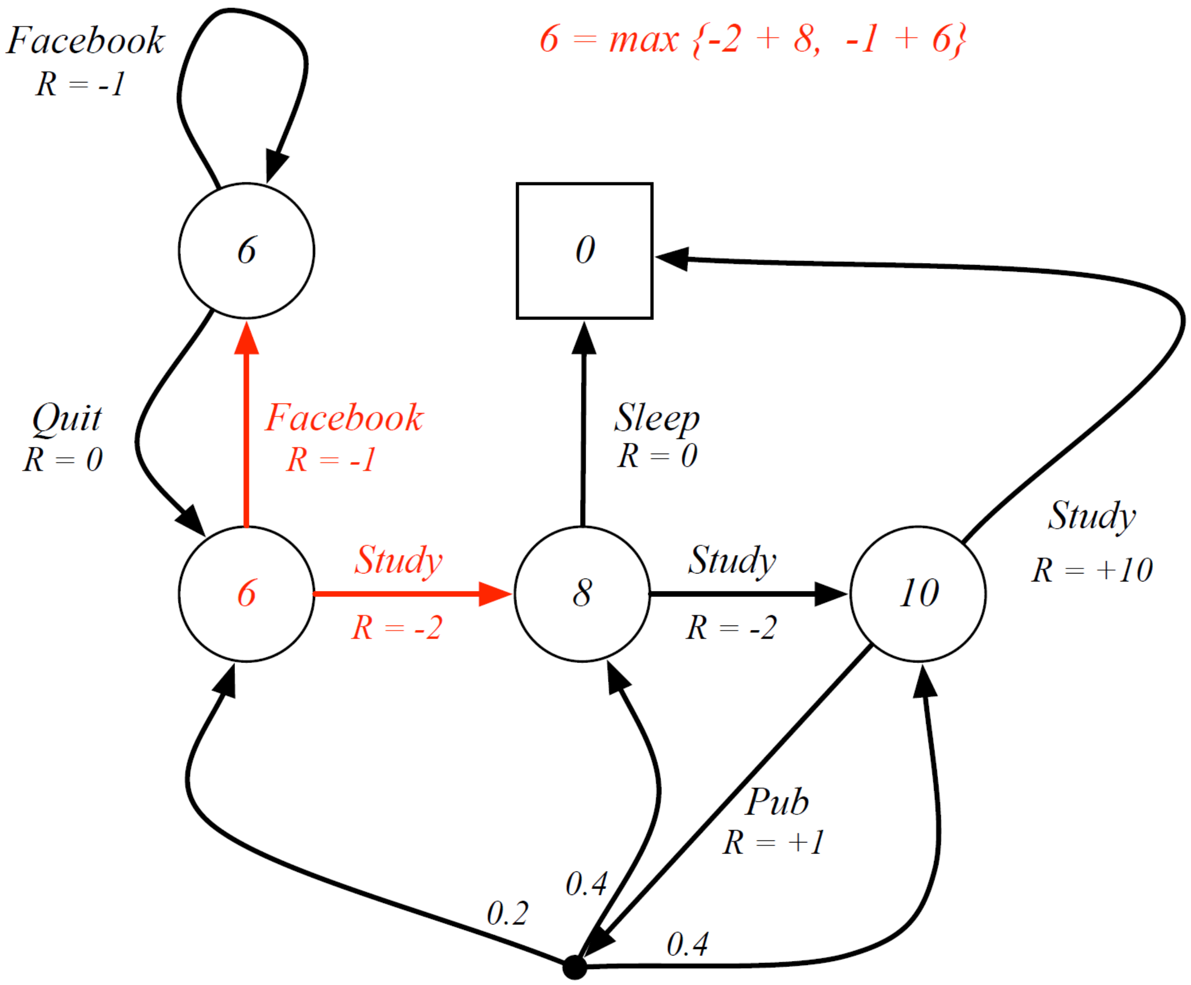
$$V^*(s) = \max_{a \in A} \left(R_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a V^*(s') \right)$$

Уравнение Беллмана для Q^*



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

Пример: уравнение оптимальности для студенческого МППР



Решение уравнения оптимальности Беллмана

- Уравнение оптимальности Беллмана нелинейно
- Аналитического решения в общем виде не существует
- Много итерационных методов:
  - итерации по ценностям,
  - итерации по стратегии,
  - Q-обучение,
  - SARSA

02

Алгоритм итерации по стратегиям

Требования динамического программирования

Динамическое программирование (dynamic programming, ДП) – очень общий способ решения задач, обладающих двумя свойствами:

- оптимальной структурой:
  - применим принцип оптимальности,
  - оптимальное решение может быть декомпозировано в подзадачи;
- перекрывающиеся подзадачи:
  - подзадачи повторяется многократно,
  - решения могут сохранены и переиспользованы.

Марковский процесс принятия решений удовлетворяет обоим свойствам:

- уравнение Беллмана задает рекурсивную структуру подзадач,
- функция полезности сохраняет и переиспользует решения.

Планирование с помощью ДП

- ДП предполагает использование всей информации о МППР
- В реальных системах используется на этапе планирования
- Для задачи оценки стратегии:

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

или

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

- Для задачи поиска оптимальной стратегии:

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

Итерационная оценка стратегии

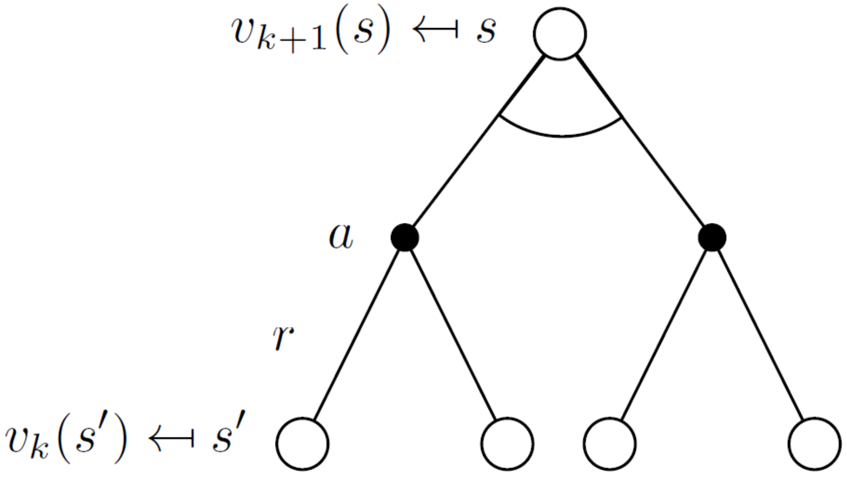
Задача: оценить текущую стратегию π

Решение: итеративное применение уравнения Беллмана в обратном направлении (Bellman backup):

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

- Использование *синхронных* шагов:
  - Инициализируем V^1 нулями:
  - для каждой итерации $k+1$:
  - для каждого состояния $s \in S$:
  - 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

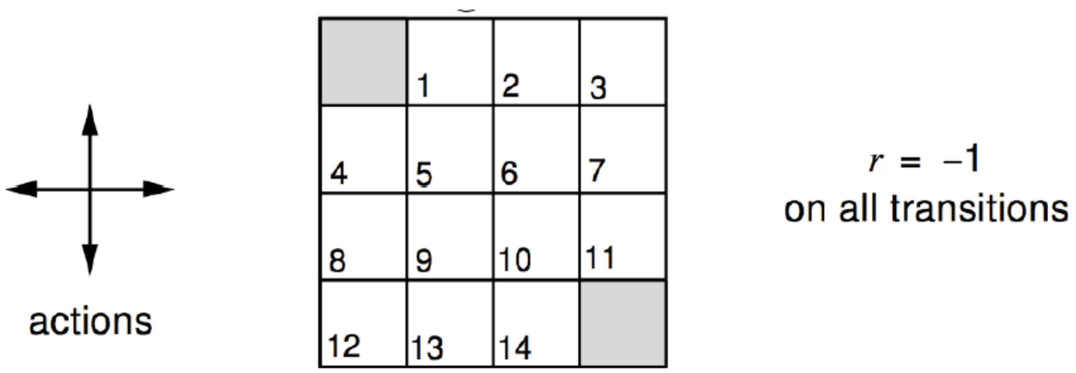
Итерационная оценка стратегии



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

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

Пример: случайная стратегия в клеточном мире



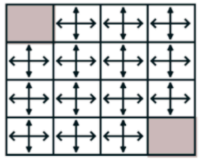
- Эпизодический МППР без дисконтирования ($\gamma = 1$)
- Нетерминальные состояния $1, 2, \ldots, 14$
- Одно терминальное состояние (серые квадраты)
- Действия, ведущие за пределы карты не меняют состояния
- Агент следует случайной стратегии:

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

Пример: случайная стратегия в клеточном мире

$k = 0$

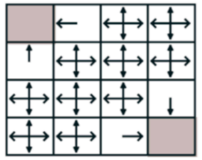
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



← случайная стратегия

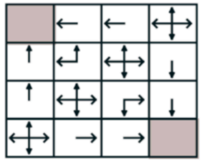
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



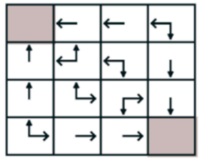
$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



$k = 3$

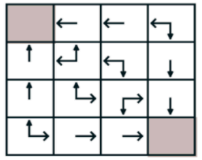
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0



оптимальная стратегия

$k = 10$

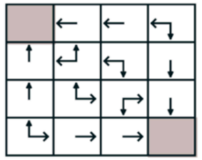
0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0



оптимальная стратегия

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



оптимальная стратегия

Улучшение стратегии

- Дана стратегия π:
  - *Оценить* (evaluate) стратегию π:

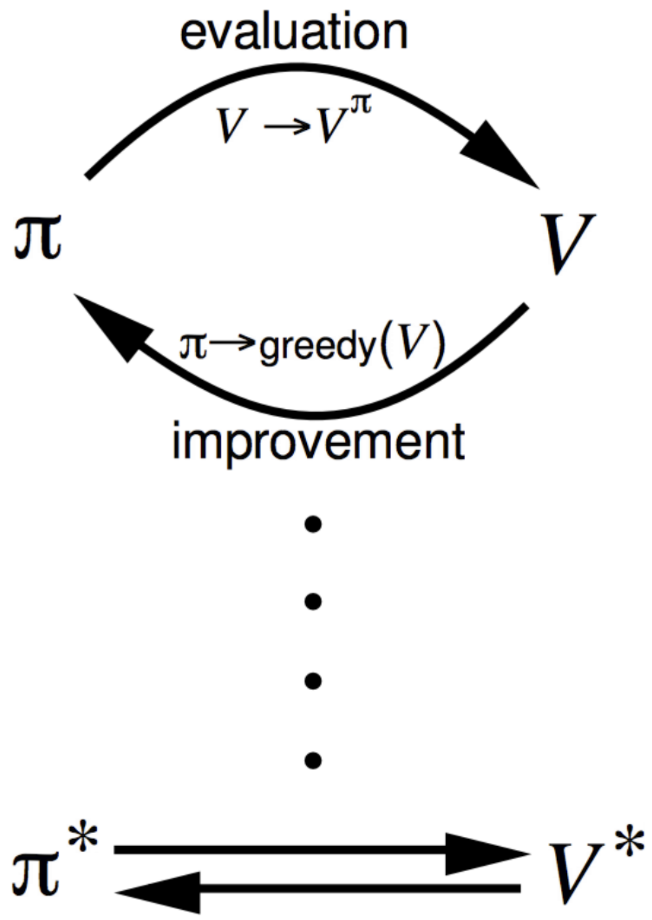
$$V^{\pi}(s) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \ldots | s_t = s]$$

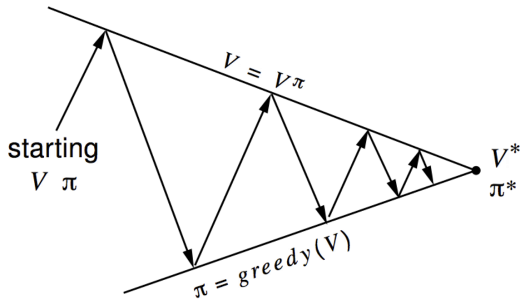
  - *Улучшить* (improve) стратегию, выбирая действия жадно (greedy), но в соответствии с V^{π}:

$$\pi' = greedy(V^{\pi})$$

- В клеточном мире улучшенная стратегия оказалась оптимальной: $\pi' = \pi^*$
- В общем случае нужно больше итераций
- Однако этот процесс *итерации по стратегии* всегда сходится к оптимальной стратегии π^*

Итерация по стратегиям (Policy Iteration - PI)





Оценка стратегии – вычисление V^π

Итеративная оценка стратегии

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

Жадное обновление стратегии

Псевдокод — итерация по стратегиям

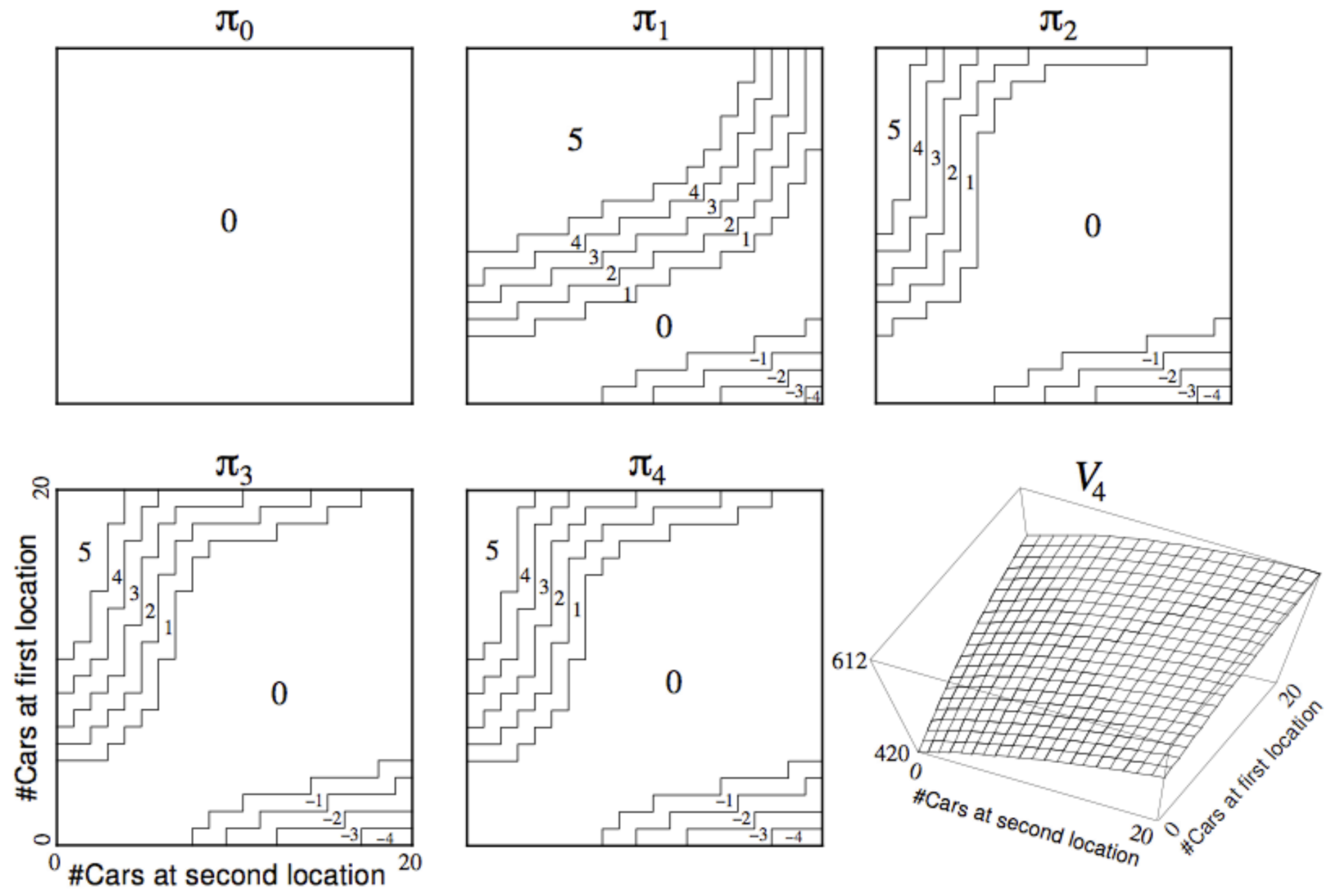
Algorithm 1 Итерация по стратегиям

Инициализировать policy π произвольно
repeat
    Оценка стратегии:
    repeat
        for каждый state $s \in \mathcal{S}$ **do**
            $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
        end for
    until сходимости value function
    Улучшение стратегии:
    policy-stable $\leftarrow$ **true**
    for каждый state $s \in \mathcal{S}$ **do**
        old-action $\leftarrow \arg\max_a \pi(a|s)$
        $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
        if old-action $\neq \pi(s)$ **then**
            policy-stable $\leftarrow$ **false**
        end if
    end for
until policy-stable
return π, V

Пример: аренда машин

- *Состояния*: две локации, максимально по 20 машин в каждой
- *Действия*: перегнать до 5 машин в течение ночи между локациями
- *Вознаграждение*: 10$ за каждую арендованную машину
- *Переходы*: машины возвращаются и арендуются случайно:
  - пуассоновское распределение: n запросов/возвратов с вероятностями $\frac{\lambda^n}{n!}e^{-\lambda}$,
  - 1 локация: среднее кол-во запросов 3, среднее кол-во возвратов 3,
  - 2 локация: среднее кол-во запросов 4, среднее кол-во возвратов 2

Пример: итерация по стратегиям для аренды машин



03

Алгоритм итерации по полезностям

Принцип оптимальности

Любая оптимальная стратегия может быть разделена на две части:

- оптимальный первый шаг a^*,
- следование оптимальной стратегии, начиная со следующего состояния s'.

Теорема 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': $V^\pi(s') = V^*(s')$.

Детерминированные итерации по полезностям

- Пусть мы знаем решение для подзадачи $V^*(s')$,
- тогда мы можем найти решение за один шаг:

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

- Идея итераций по ценностям – применять эти обновления рекурсивно.
- *Интуиция*: начать с конечных вознаграждений и двигаться назад.

Пример: кратчайший путь

g			

Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

V_5

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

V_7

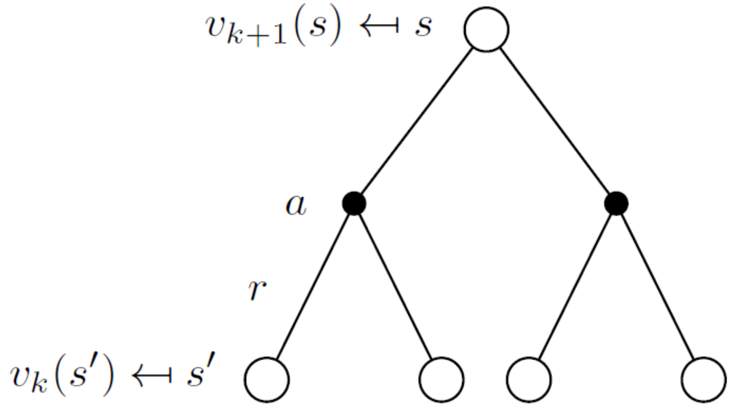
Итерация по полезностям

Задача: найти оптимальную стратегию π^*.
Решение: итеративное применение уравнения **оптимальности** Беллмана в обратном направлении:

$$V^1 \to V^2 \to \dots \to V^*$$

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

Итерация по полезностям



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

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

$V^{k+1} = \mathcal{B} V^k$ – опреатор Беллмана (Bellman backup operator)

Псевдокод — итерация по полезностям

Algorithm 2 Итерация по полезностям

Инициализировать value function $V(s)$ произвольно для всех $s \in \mathcal{S}$
repeat
    $\Delta \leftarrow 0$
    for каждый state $s \in \mathcal{S}$ **do**
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end for
until $\Delta < \theta$ ▷ порог сходимости
Определить policy $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
return π, V

Вопросы?