RL: MDP

План

- Марковские процессы
- Уравнение Беллмана
- Функция полезности
- Оптимальности
- Итерации по стратегиям
- Итерации по полезностям

Марковский процесс принятия решений

- Марковский процесс принятия решения (МППР, MDP) моделирует взаимодействие агента и среды
- Предполагается что среда полностью наблюдаема (fully observable)
- Текущее состояние полностью характеризует весь процесс взаимодействия
- Почти все задачи RL могут быть сведены к задаче с MDP
  - Оптимальное управление — MDP непрерывным множеством состояний и действий
  - Частично наблюдаемы среды могут быть сведены к MDP
  - Игровые автоматы — пример MDP с одним состоянием

Марковское свойство

Будущее не зависит от прошлого и определяется только настоящим

Определение

- *Состояние s_t называется марковским если и только если*

$$P[s_{t+1} | s_t] = P[s_{t+1} | s_1, s_2, \dots, s_t]$$

- Текущее состояние содержит всю информация из истории взаимодействия
- Если есть текущее состояние, история далее может не учитываться
- Состояние содержит достаточно статистики для определения будущего

Матрица переходов

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

$$P_{ss'} = P[s_{t+1} = s' | s_t = s]$$

Матрица переходов P определяет вероятности переходов между всеми возможными состояниями

$$P = \begin{bmatrix} P_{11} & \dots & P_{1n} \\ \dots & \dots & \dots \\ P_{n1} & .. & P_{nn} \end{bmatrix}$$

Каждая строчка матрицы в сумме дает 1

Марковский процесс (принятия решений)

- Марковский процесс (Markov process, Markov chain) - это случайный процесс без памяти, т.е. Последовательность случайных состояний s1, s2, ..., обладающая свойством Маркова.

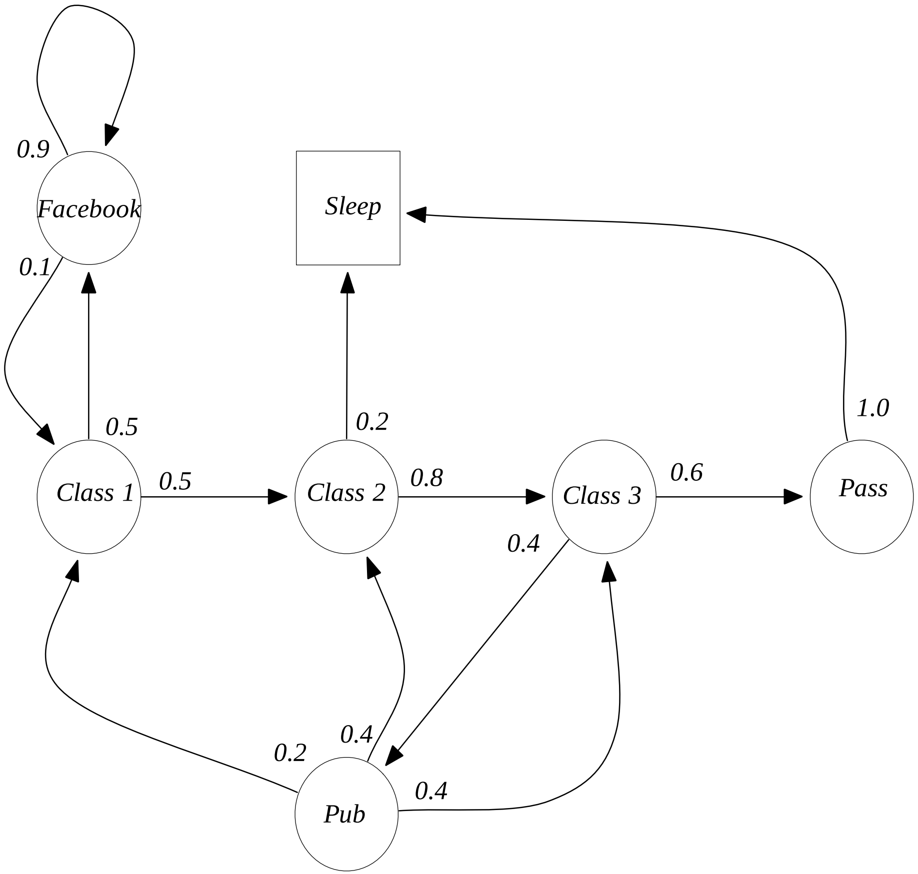
Определение

Марковский процесс (или цепь Маркова) - это пара <*S*,*P*>, где

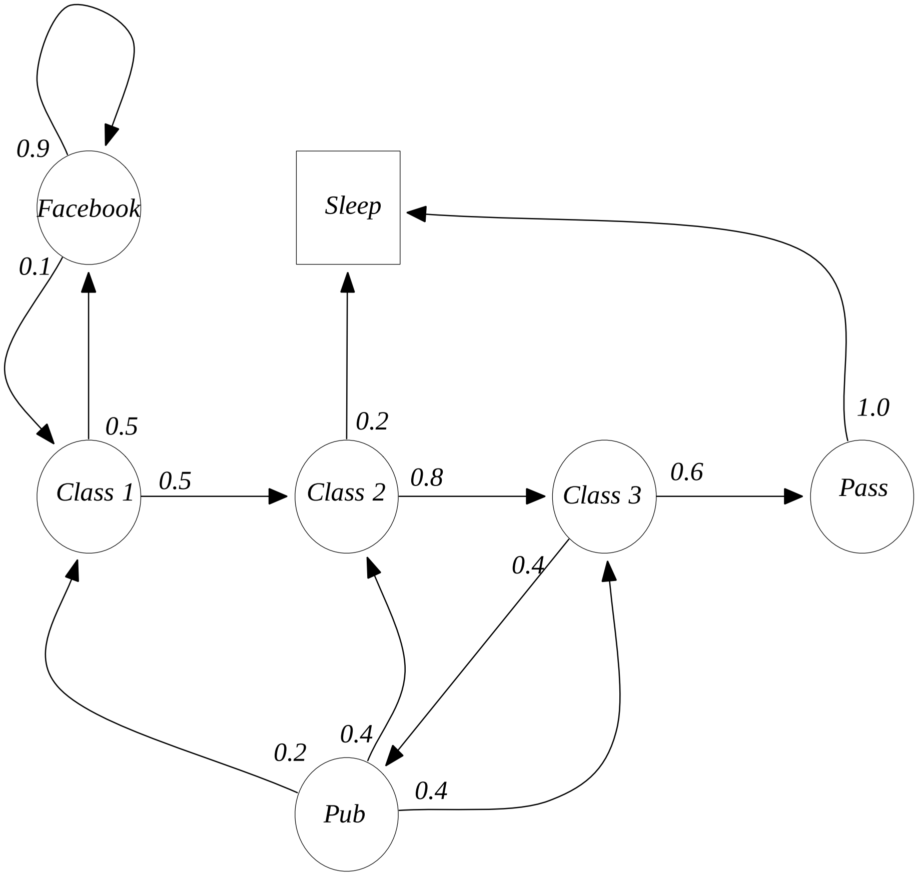
- S - (конечное) множество состояний
- P - матрица вероятностей перехода между состояниями,

$$P_{ss'} = P[s_{t+1} = s' | s_t = s]$$

Пример: студенческая марковская цепь



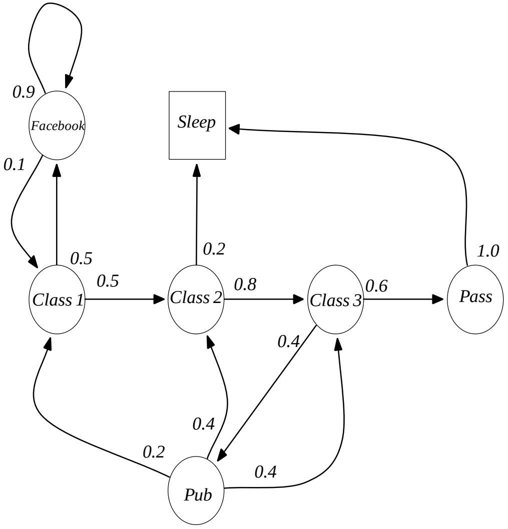
Пример: эпизоды студенческой марковской цепи



Выборка эпизодов для студенческой марковской цепи, начиная с s1 = 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

Пример: переходы студенческой марковской цепи



$$P =$$

	C1	C2	C3	Pass	Pub	FB	Sleep
C1		0.5				0.5	
C2			0.8				0.2
C3				0.6	0.4		
Pass							1
Pub	0.2	0.4	0.4				
FB	0.1					0.9	
Sleep							1.0

Марковский процесс вознаграждения

- Марковский процесс вознаграждения - это марковская цепь с дополнительными значениями вознаграждений за переходы

Определение

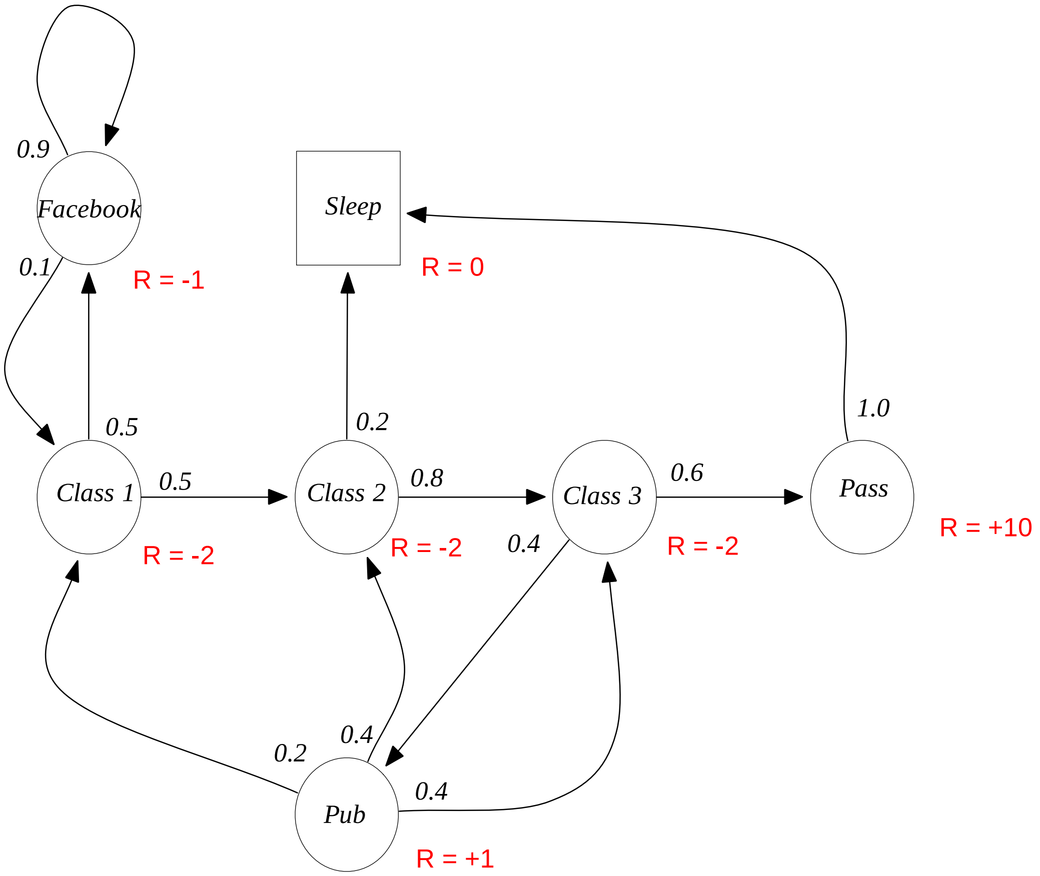
Марковский процесс вознаграждений - это тройка <*S*,*P*, *R*, *γ*>, где

- S - (конечное) множество состояний
- P - матрица вероятностей перехода между состояниями,

$$P_{ss'} = P[s_{t+1} = s' | s_t = s]$$

- R – функция вознаграждения $R_s = E[r_{t+1} | s_t = s]$
- $\gamma \in [0,1]$ – дисконтирующий множитель

Пример: студенческий MRP



Суммарное вознаграждение

Определение

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

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

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

Дисконтирующий множитель

- Математически удобно дисконтировать вознаграждения
- Позволяет избежать бесконечной отдачи в марковских процессах с циклами
- Отражение неопределенности в отношении будущего
- Если вознаграждение, например финансовое, немедленное вознаграждение может принести больше процентов, чем отложенное вознаграждение
- Поведение животных/людей показывает предпочтение немедленного вознаграждения
- В некоторых случаях удобно установить не дисконтированнное вознаграждение (т.е. $\gamma = 1$)

Функция полезности

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

Определение

Функция полезности V(s) марковского процесса вознаграждения - это ожидаемая отдача, которое получает агент, начиная с состояния s

$$V(s) = E[R_t | s_t = s]$$

Пример

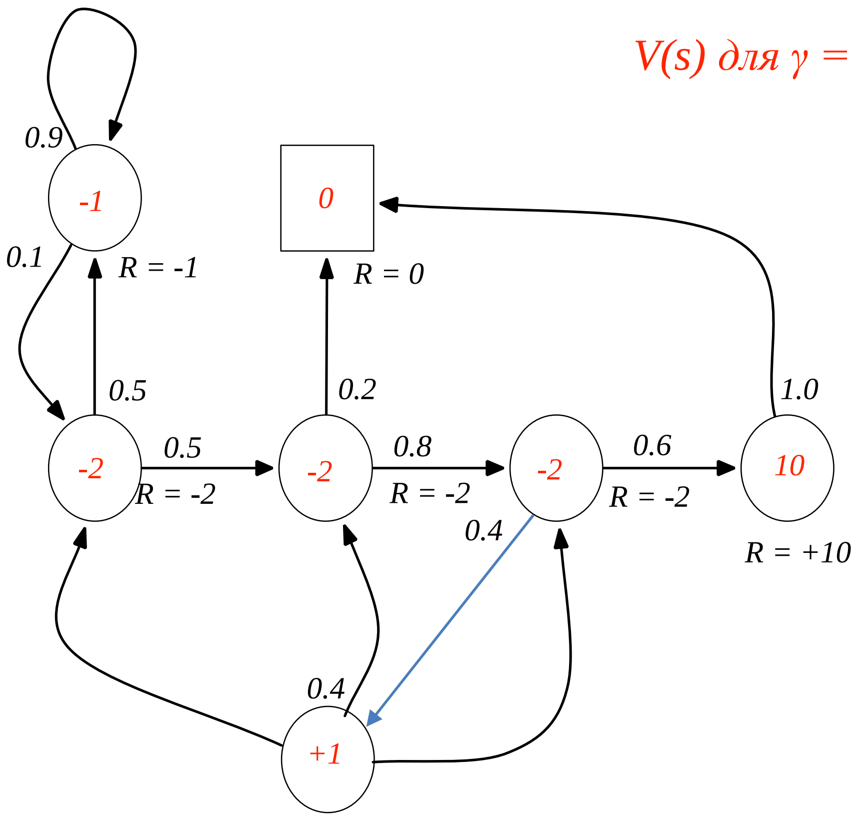
Выборка отдач для студенческого MRP, начиная с s1 = C1 с $\gamma = \frac{1}{2}$:

$$R_1 = r_2 + \gamma r_3 + \cdots + \gamma^{t-2} r_t$$

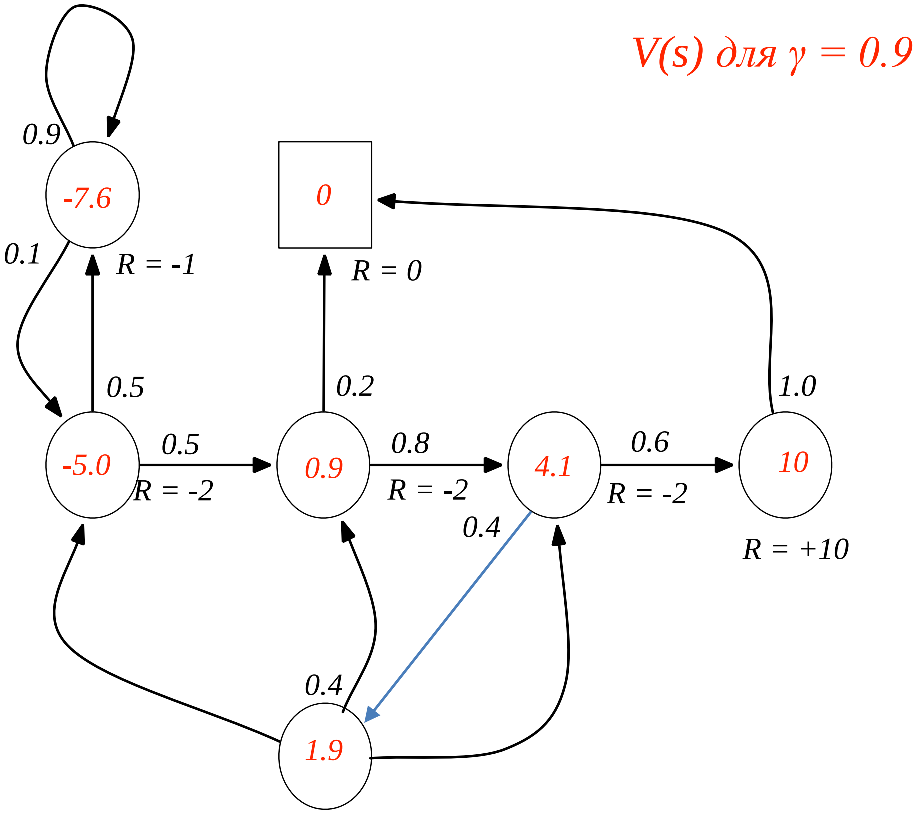
C1 C2 C3 Pass Sleep	v1 = -2 – 2*1/2 – 2*1/4 + 10*1/8	= -2.25
C1 C2 C3 Pub C2 C3 Pass Sleep	v1 = -2 – 1*1/2 – 1*1/4 – 2*1/8 – 2*1/16	= -2.125
C1 FB FB C1 C2 Sleep	v1 = -2 – 2*1/2 – 2*1/4 +1*1/8 – 2*1/16 …	= -3.41
C1 FB FB C1 C2 C3 Pub C1	v1 = -2 – 1*1/2 – 1*1/4 – 2*1/8 – 2*1/16 …	= -3.41
FB FB FB C1 C2 C3 Pub C2 Sleep		

Пример

V(s) для $\gamma = 0$

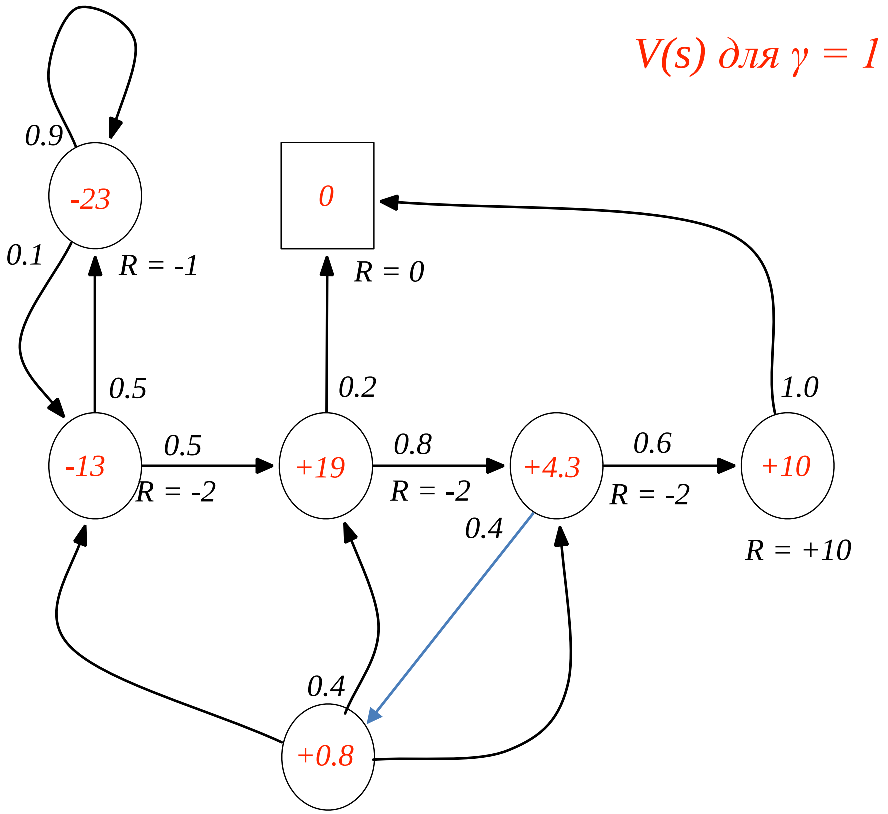


Пример



V(s) для γ = 0.9
-7.6
0.9
0.1
R = -1
0
R = 0
0.5
-5.0
0.5
R = -2
0.9
0.2
0.8
R = -2
4.1
0.4
0.6
R = -2
10
1.0
R = +10
0.4
1.9

Пример



V(s) для γ = 1
0.9
-23
0.1
R = -1
0
R = 0
0.5
0.2
1.0
-13
0.5
R = -2
+19
0.8
R = -2
+4.3
0.6
R = -2
+10
R = +10
0.4
0.4
+0.8

Уравнение Беллмана для MRP

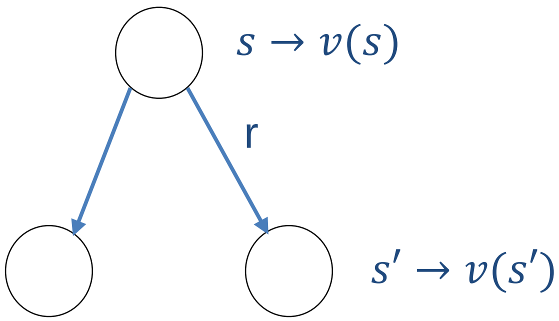
Функцию полезности V(s) можно представить в виде суммы двух слагаемых:

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

$$V(s) = E[R_t | s_t = s] =$$
$$E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s] =$$
$$E[r_{t+1} + \gamma (r_{t+2} + \gamma r_{t+3} + \cdots) | s_t = s] =$$
$$E[r_{t+1} + \gamma R_{t+1} | s_t = s] =$$
$$E[r_{t+1} + \gamma V(s_{t+1}) | s_t = s]$$

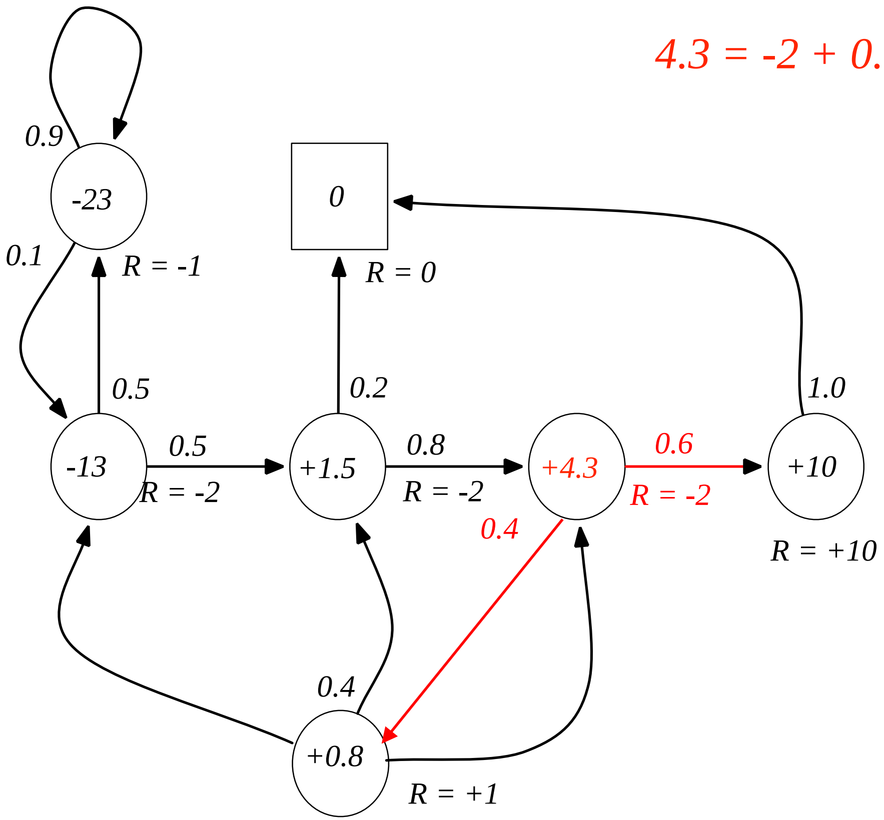
Уравнение Беллмана для MRP

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



$$V(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} V(s')$$

Уравнение Беллмана для студенческого MRP



*4.3 = -2 + 0.6*10 + 0.4*0.8*

Уравнение Беллмана в матричной форме

$$V = R + \gamma P V$$

Где V – это вектор колонка с одной компонентой на состояние

$$\begin{pmatrix} V(1) \\ \dots \\ V(n) \end{pmatrix} = \begin{pmatrix} R(1) \\ \dots \\ R(n) \end{pmatrix} + \gamma \begin{pmatrix} P_{11} & \dots & P_{1n} \\ \dots & \dots & \dots \\ P_{n1} & \dots & P_{nn} \end{pmatrix} \begin{pmatrix} V(1) \\ \dots \\ V(n) \end{pmatrix}$$

$$V(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} V(s')$$

Решение уравнения Беллмана

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

$$V = R + \gamma PV$$

$$(1 - \gamma P)V = R$$

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

- Вычислительная сложность $O(n^3)$, где n число состояний
- Аналитическое решение возможно только для небольших задач MDP
- Но есть множество приближенных итерационных методов
  - Динамическое программирование
  - Монте-Карло
  - Метод временных различий

Марковский процесс принятия решений

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

❑ Определение

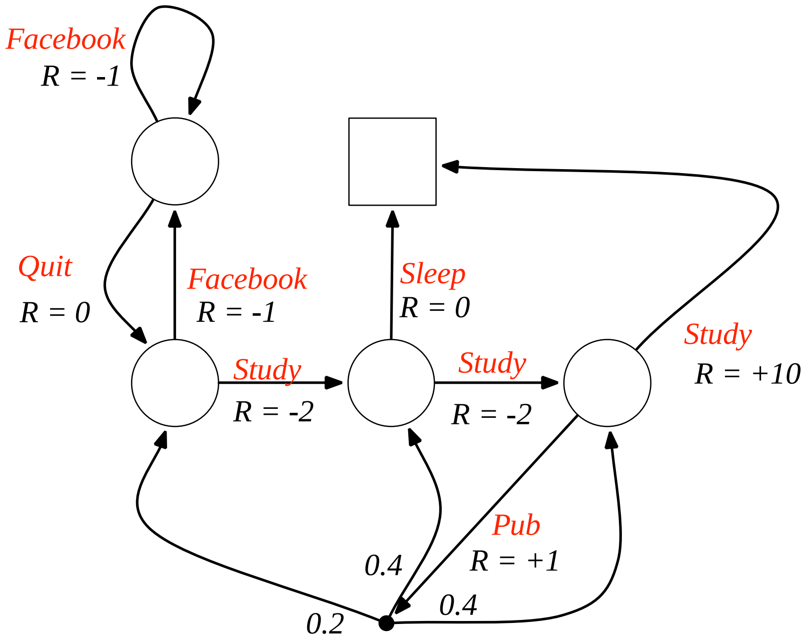
Марковский процесс принятия решений – это кортеж <*S,A,P,R,ɣ*>, *где*

- *S – конечное множество состояний*
- *A – конечное множество действий*
- *P – матрица перехода*

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

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

Студенческий MDP



Стратегия

❑ Определение

Стратегией π будем называть вероятностное распределение на множестве действий действиям

при текущем состоянии s:

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

- *Стратегия полностью определяет поведение агента*
- *MDP стратегии зависят от текущего состояния (не от истории).*
- *Стратегии стационарны (не зависят от времени)*

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

Стратегия

- Пусть дан MDP $\mathrm{M} = \langle S, A, P, R, \gamma \rangle$ и стратегия π.
- *Последовательность состояний s1,s2,... является марковским процессом* $\langle S, P^{\pi} \rangle$
- *Последовательность состояний и вознаграждений s1,r2,s2,... представляет собой марковский процесс вознаграждения* $\langle S, P^{\pi}, R^{\pi}, \gamma \rangle$

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

$$R^{\pi}_{ss'} = \sum_{a \in A} \pi\ (a|s) R^{a}_{ss'}$$

Стратегия

- Определение
  - *Функция полезности состояний MDP* $V^{\pi}(s)$ *- это математическое ожидание отдачи, начиная с состояния s, при выполнении политики π:*

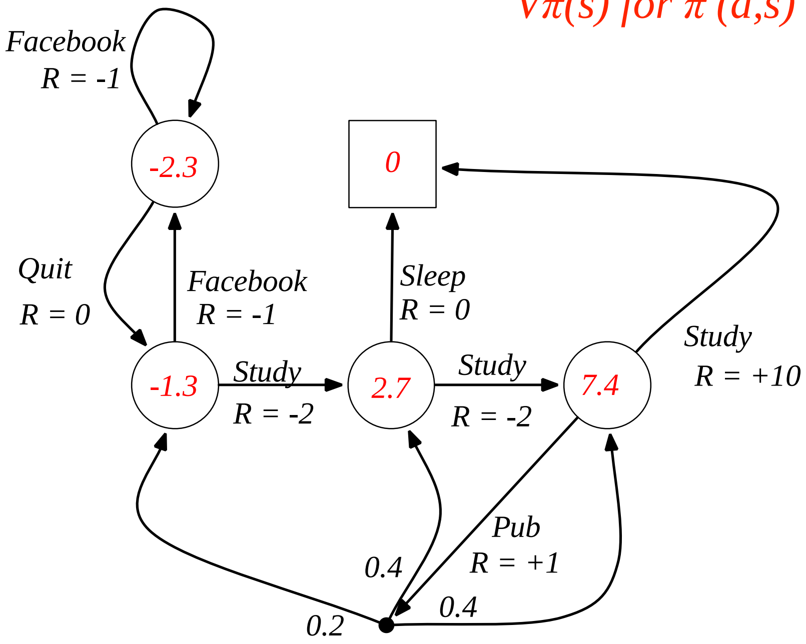
$$V^{\pi}(s) = E_{\Pi}[R_t | s_t = s]$$

- Определение
  - *Функция полезности действия MDP* $Q^{\pi}(s, a)$ *- это математическое ожидание отдачи, начиная с состояния s, выбранного действия a, при выполнении стратегии π*

$$Q^{\pi}(s, a) = E_{\Pi}[R_t | s_t = s, a_t = a]$$

Студенческий MDP

$V_\pi(s)$ for $\pi(a,s) = 0.5$, $\gamma = 1$



Уравнение Беллмана для MDP

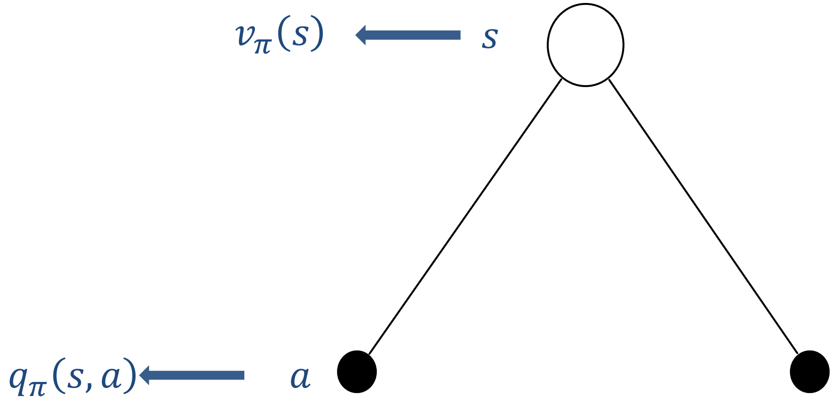
- *Функция полезности состояния может быть разложена на немедленное вознаграждение плюс дисконтированная стоимость следующего состояния*

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

- *Аналогично можно разложить функцию полезности действия,*

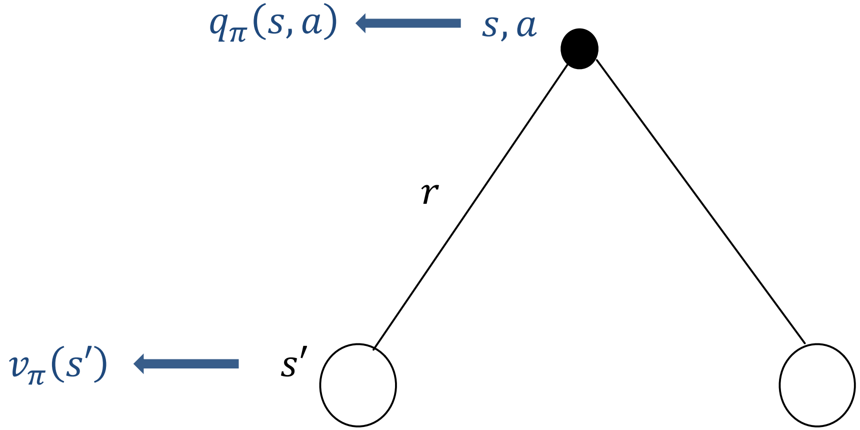
$$Q^{\pi}(s,a) = E_{\Pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1})|s_t = s, a_t = a]$$

Уравнение Беллмана для V^{π}



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

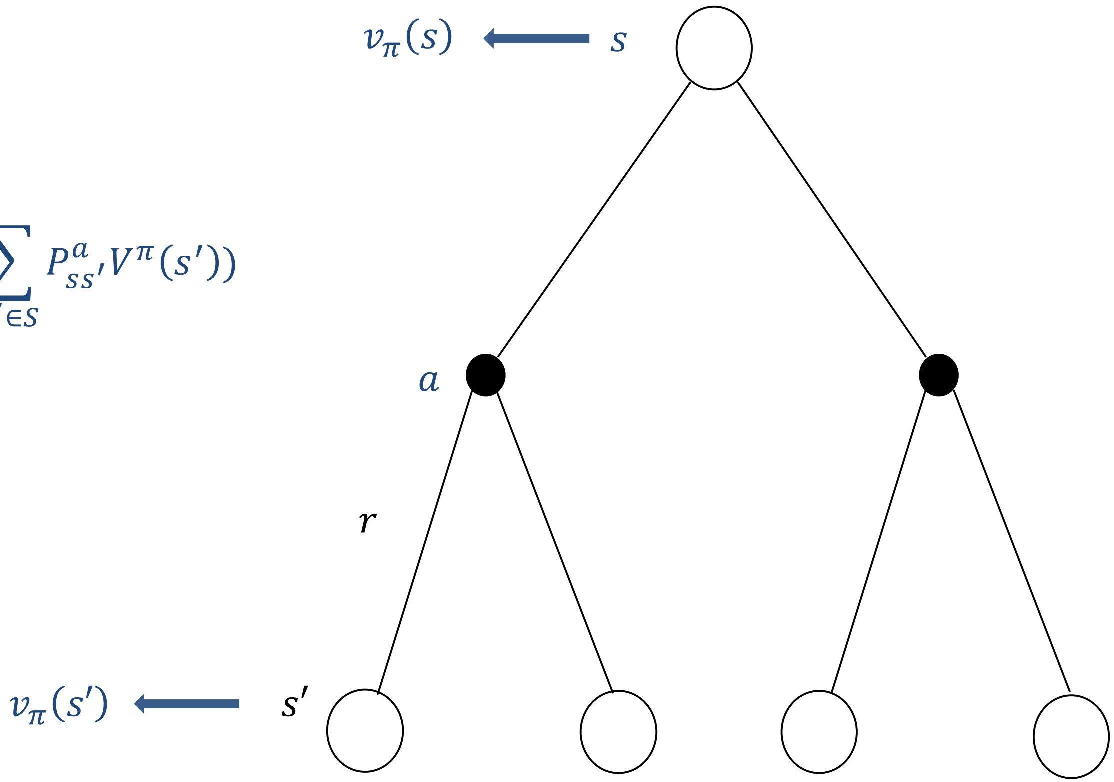
Уравнение Беллмана для Q^{π}



$$Q^{\pi}(s,a) = R_s^a + \Upsilon \sum_{s' \in S} P_{ss'}^a V^{\pi}(s')$$

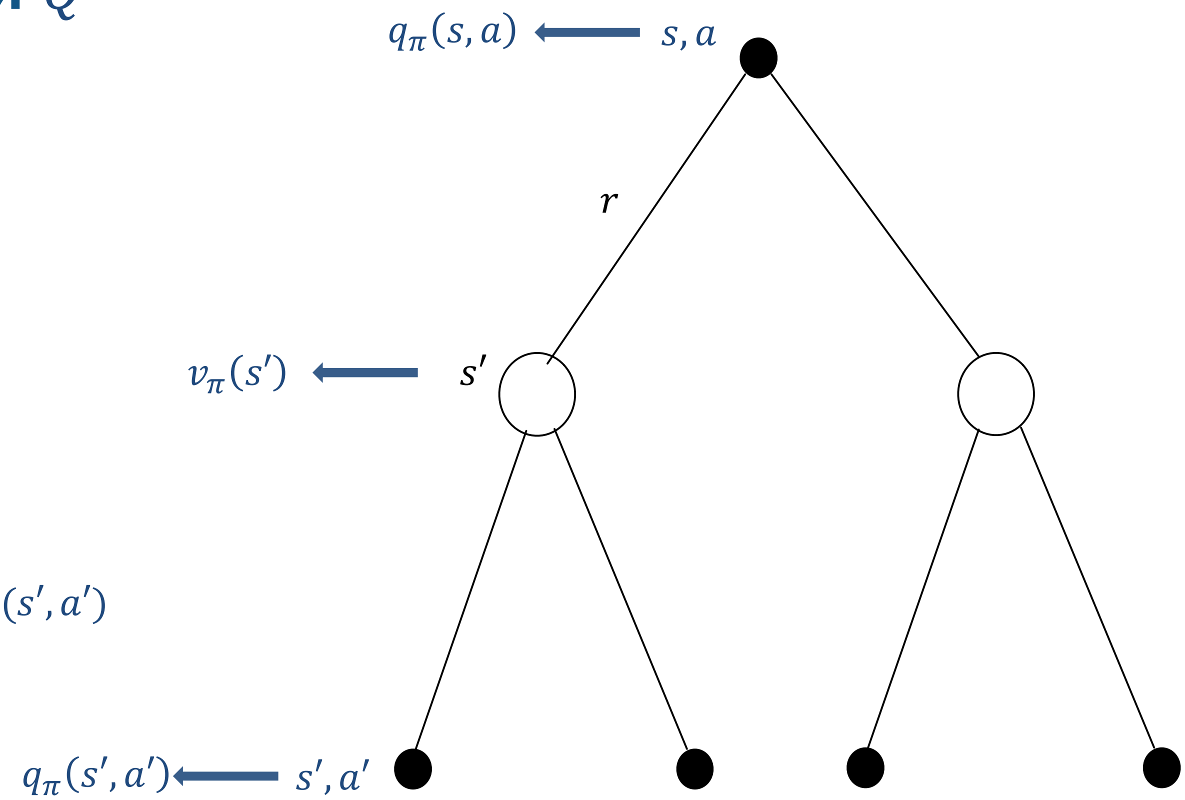
Уравнение Беллмана для V^{π}

$$V^{\pi}(s) = \sum_{a \in A} \pi(a|s)(R_s^a + \Upsilon \sum_{s' \in S} P_{ss'}^a V^{\pi}(s'))$$



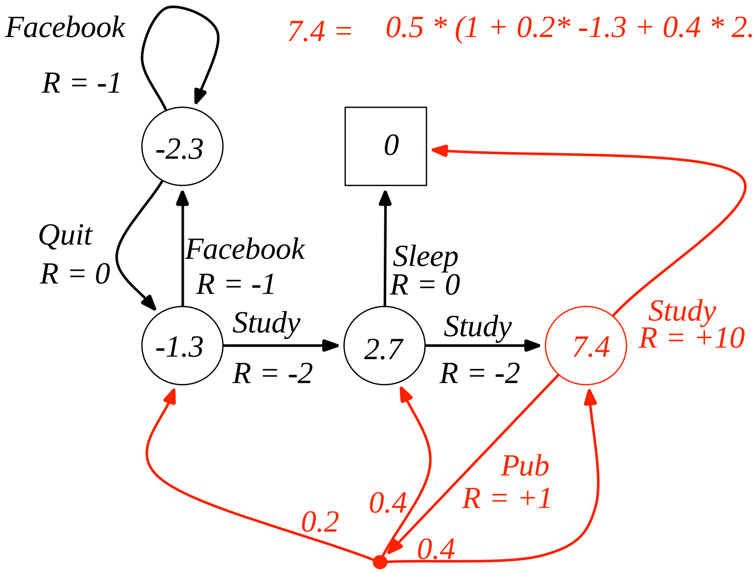
Уравнение Беллмана для Q^{π}

$$Q^{\pi}(s,a) = R_s^a + \Upsilon \sum_{s' \in S} P_{ss'}^a \sum_{a \in A} \pi(a'|s') Q^{\pi}(s',a')$$



Пример: уравнение Беллмана для студенческого MDP

$$7.4 = 0.5 * (1 + 0.2 * -1.3 + 0.4 * 2.7 + 0.4 * 7.4) + 0.5 * 10$$



Матричная форма уравнения Беллмана для MDP

- *Уравнение Беллмана с матожиданиями может быть кратко записано по аналогии с MRP:*

$$V^{\pi}(s) = R^{\pi} + \gamma P^{\pi} V^{\pi}$$

- *Аналитическое решение:*

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

Оптимальная функция полезности

- *Определение*

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

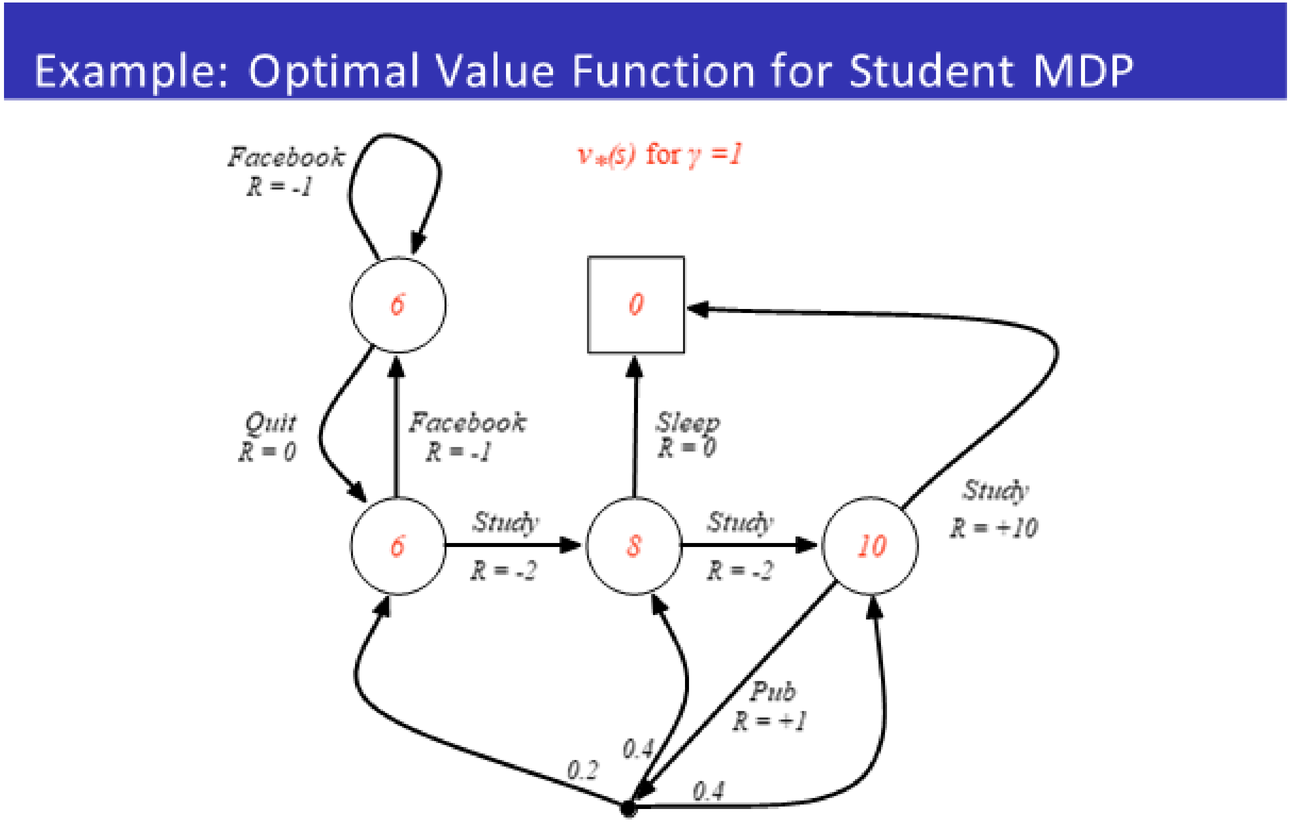
$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

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

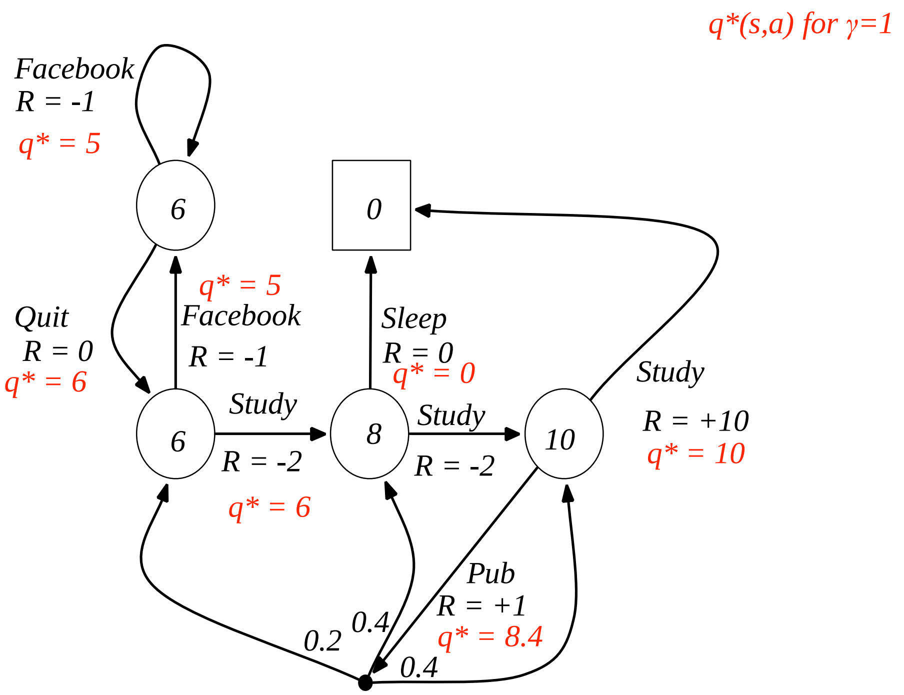
$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

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

Пример: уравнение Беллмана для студенческого MDP



Пример: уравнение Беллмана для студенческого MDP



Оптимальная стратегия

Определим частичный порядок на множестве стратегий:

$$\pi \geq \pi' \; if \; V^{\pi}(s) \geq V^{\pi'}(s), \forall s$$

- *Теорема*

Для любого MDP

- *Существует оптимальная стратегия,* π_* *которая лучше или равна всем другим стратегиям* $\pi_* \geq \pi, \forall \pi$
- *Все оптимальные стратегии доставляют оптимум функции (ценности) состояния - действия* $V^{\pi^*}(s) \geq V^{\pi}(s)$ $\quad Q^{\pi^*}(s,a) \geq Q^{\pi}(s,a)$

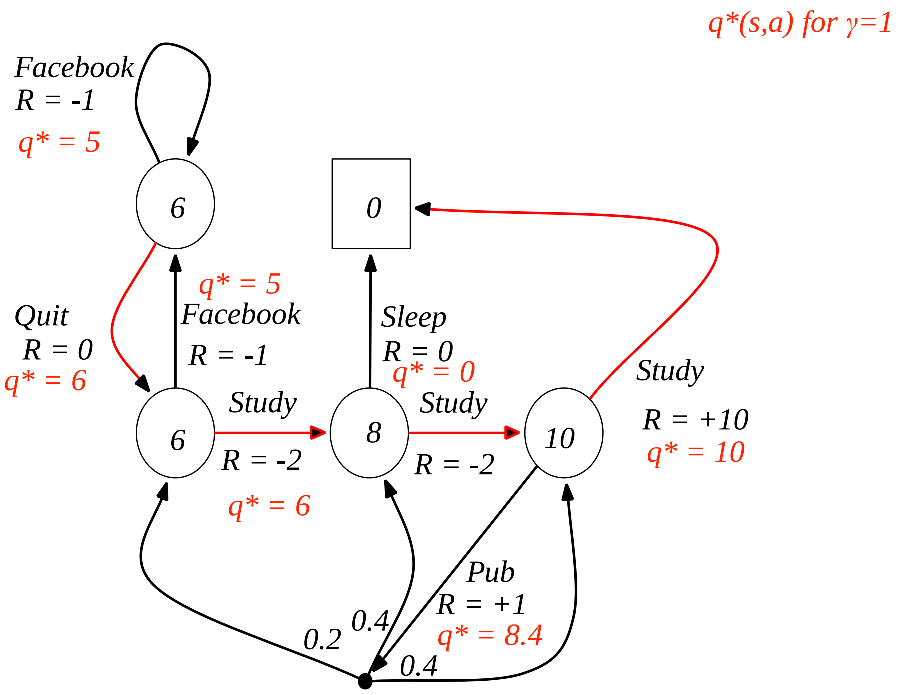
Поиск оптимальной стратегии

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

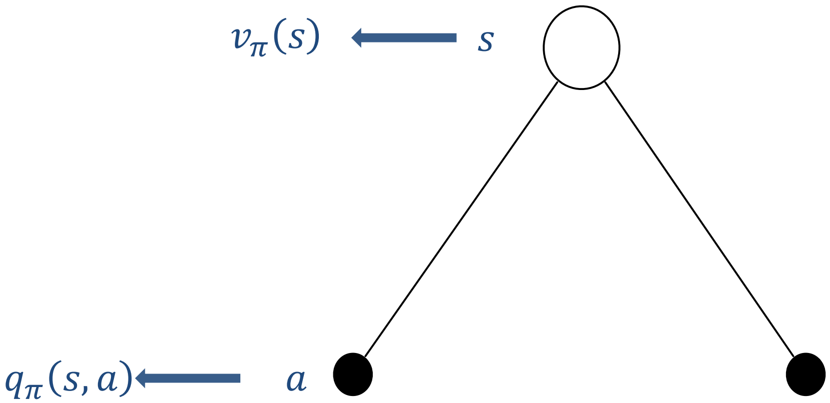
$$\pi^*(a|s) = \begin{cases} 1, \text{если } a = arg\max_{a \in A} Q^*(s,a) \\ 0, \text{иначе} \end{cases}$$

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

Пример: оптимальная стратегия для студенческого MDP

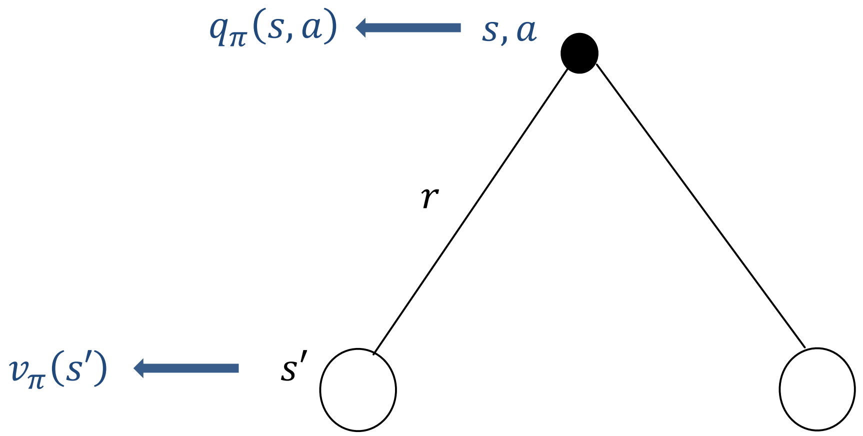


Уравнение Беллмана для V^*



$$V^*(s) = \max Q^*(s, a)$$

Уравнение Беллмана для Q^*

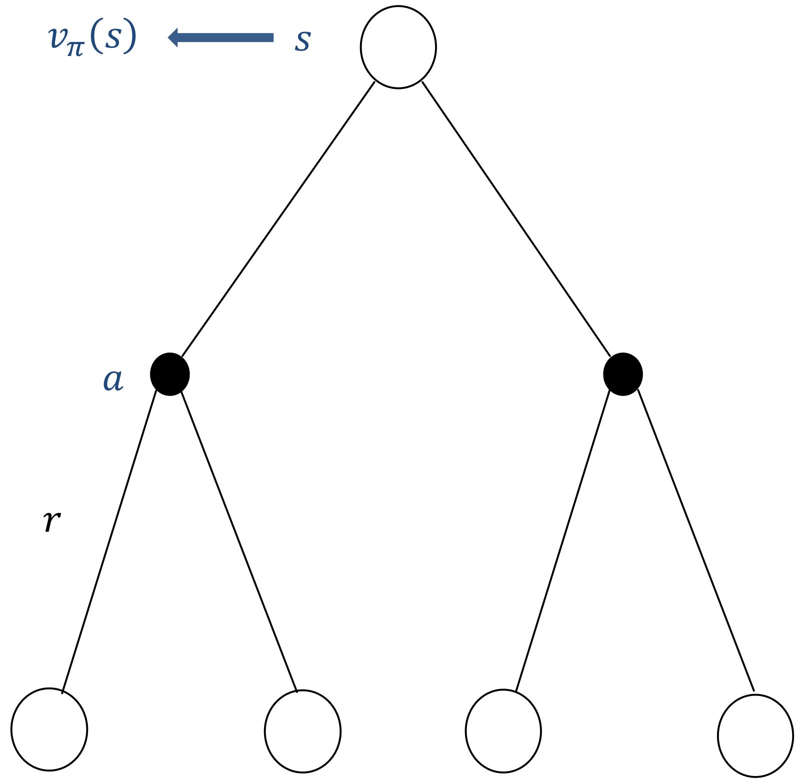


$$V^*(s) = R_s^a + \Upsilon \sum_{s' \in S} P_{ss^0}^a V^*(s^0)$$

Уравнение Беллмана для V^*

$$V^{\pi}(s) = \sum_{a \in A} \pi(a|s)\left(R_s^a + \Upsilon \sum_{s' \in S} P_{ss'}^a V^{\pi}(s')\right)$$

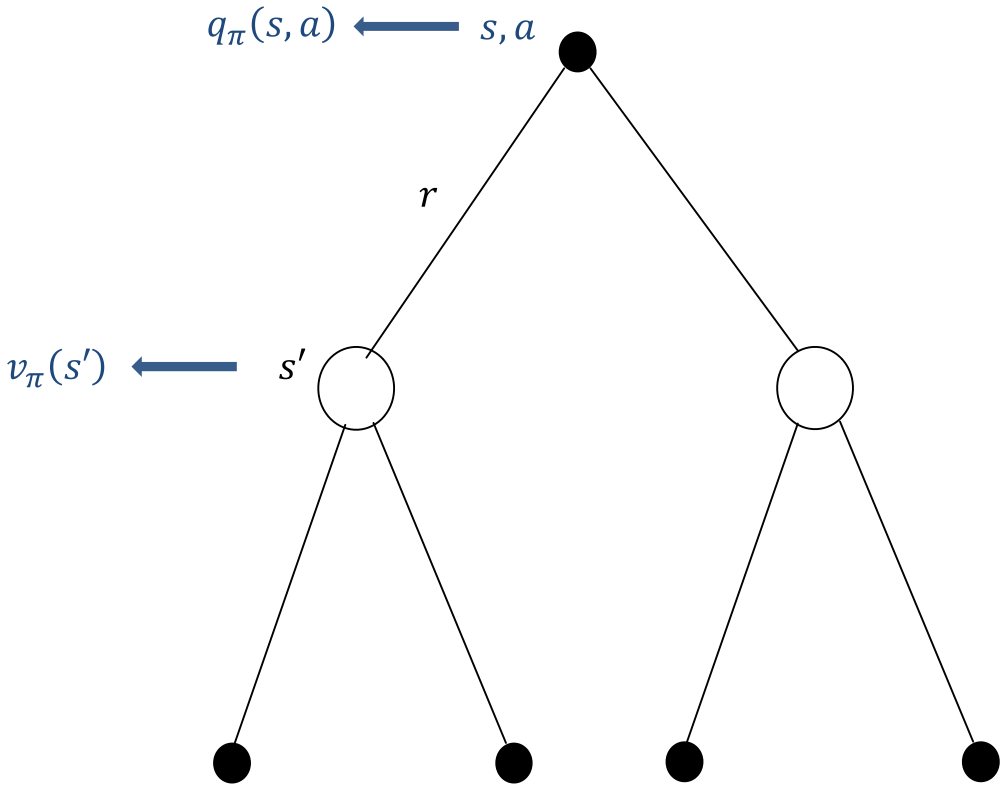
$$V^{\pi}(s) = \max_{a \in A}\left(R_s^a + \Upsilon \sum_{s' \in S} P_{ss'}^a V^{*}(s')\right)$$



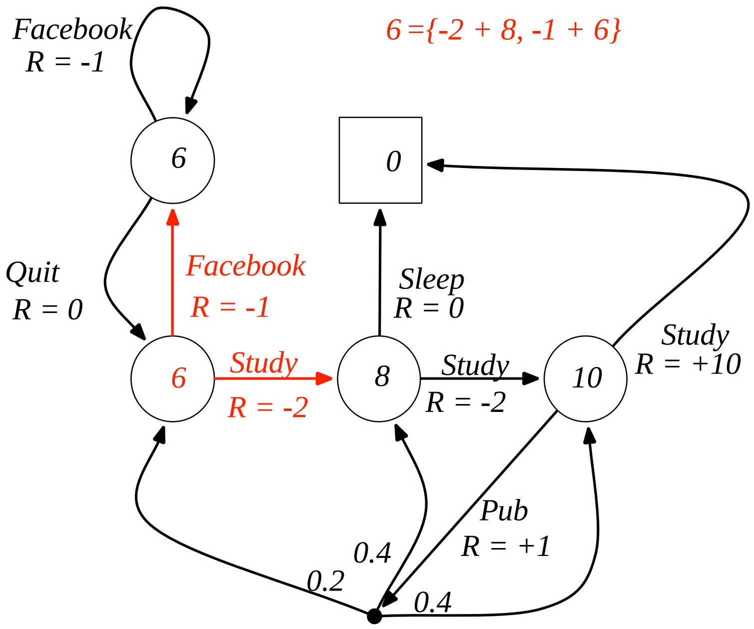
Уравнение Беллмана для Q^*

$$Q^{\pi}(s,a) = R_s^a + \Upsilon \sum_{s' \in S} P_{ss'}^a \sum_{a \in A} \pi(a|s) Q^{\pi}(s,a)$$

$$Q^{\pi}(s,a) = R_s^a + \Upsilon \sum_{s' \in S} P_{ss'}^a \max_{a' \in A} Q^{*}(s',a')$$



Пример: оптимальная стратегия для студенческого MDP



Оптимальное решение уравнение Беллмана

- Уравнение оптимальности Беллмана – это нелинейное уравнение
- Нет замкнутых (аналитических) формул для решения (в общем случае)
- Но есть множество приближенных итерационных методов
  - Итерации по функции ценности
  - Итерации по стратегиям
  - Q learning
  - SARSA

Динамическое программирование

- Динамическая часть задачи – динамические последовательности
- Программирование – оптимизация «программы», например стратегии
- Метод решения сложных задач путем выделения подзадач:
  - Решаем подзадачи
  - Комбинируем найденные решения

Требования динамического программирования

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

- Оптимальной структурой:
    - Применим принцип оптимальности,
    - Оптимальное решение может быть декомпозировано в подзадачи;
- Перекрывающиеся подзадачи:
    - Подзадачи повторяются многократно,
    - Решения могут быть 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 DP

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

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

или

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

- Для задачи управления:

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

Итерационная оценка стратегии

- Задача: оценить текущую стратегию π
- Решение: итеративное применение уравнения Беллмана:

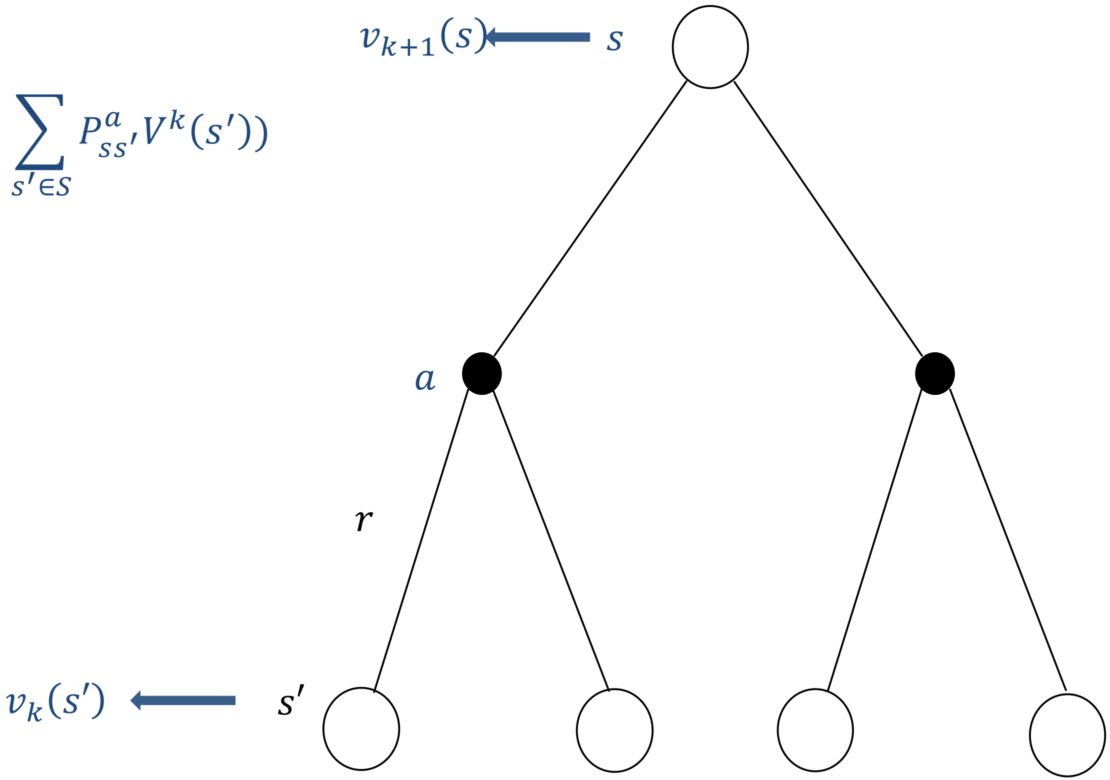
$$V^1 \to V^2 \to V^3 \to \cdots \to V^n$$

- Использование синхронных шагов:
  - Для каждой итерации *k + 1:*
  - Для каждого состояния $a \in A$
  - 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

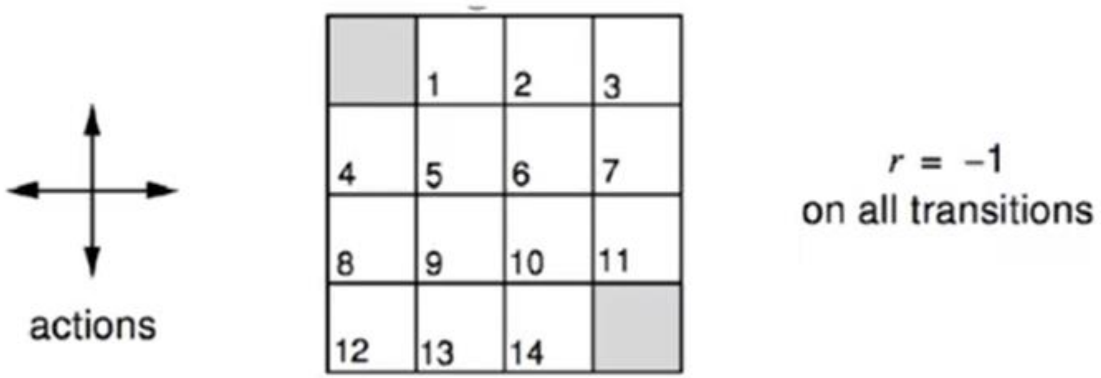
Итерационная оценка стратегии

$$V^{k+1}(s) = \sum_{a \in A} \pi(a|s)(R_s^a + \Upsilon \sum_{s' \in S} P_{ss'}^a V^k(s'))$$

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

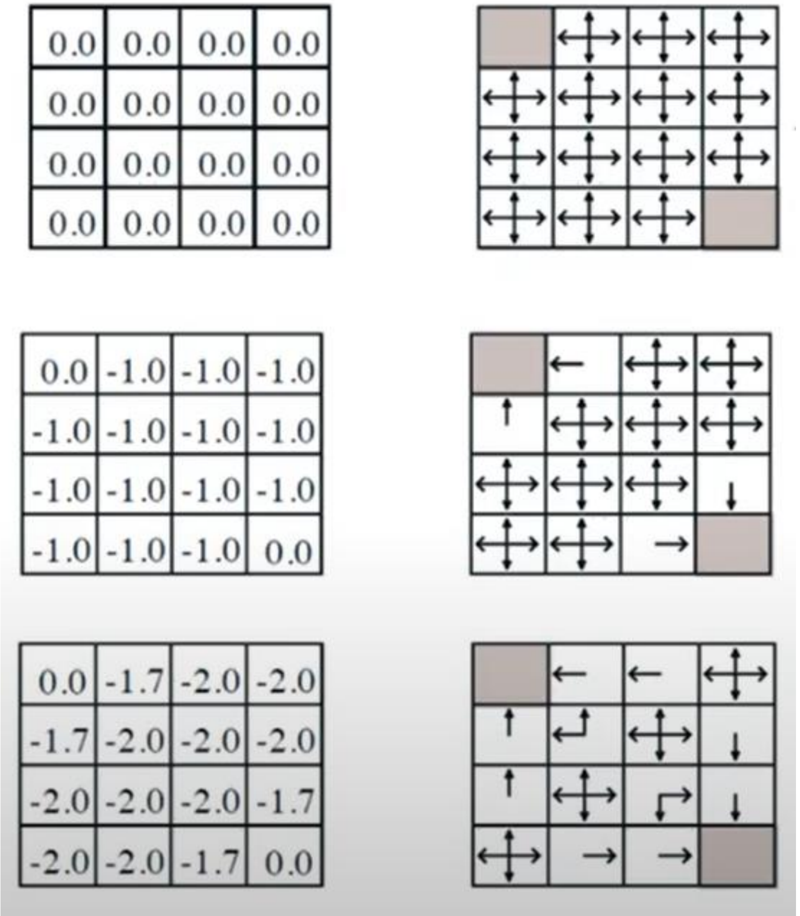


Пример: клеточный мир



Пример: клеточный мир

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



Улучшение стратегии

- Дана стратегия π:
  - Оценить стратегию π

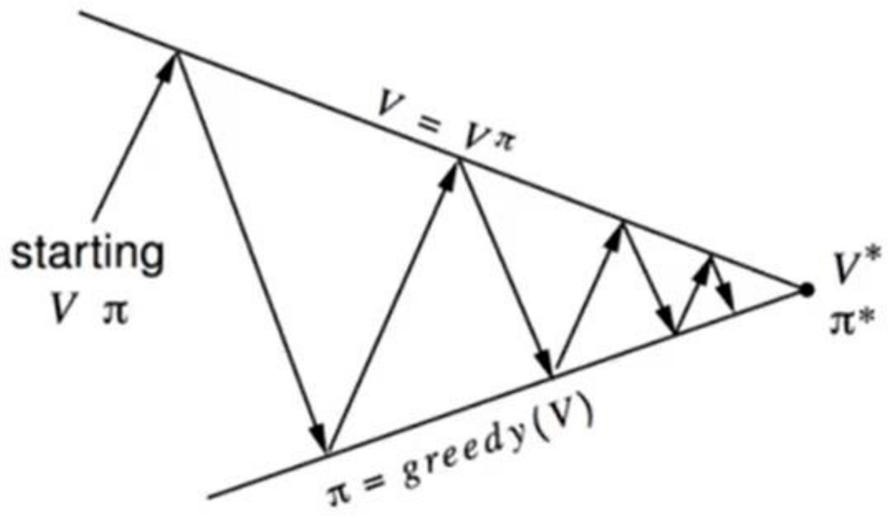
$$V^{\pi}(s) = E[r_{t+1} + \Upsilon r_{t+2} + \cdots | s_t = s]$$

  - Улучшить стратегию, выбирая жадное действие, 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 итеративный процесс по стратегии всегда сходится к оптимальной стратегии π^*

Итерация по стратегиям

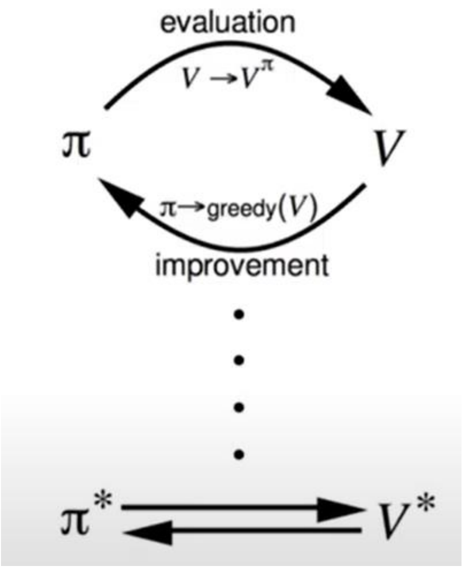


Оценка стратегии - вычисление V^{π}

Итеративная оценка стратегии

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

Жадное обновление стратегии



Улучшение стратегии

- Рассмотрим детерминированную стратегию $a = \pi(s)$:
- Мы можем улучшить стратегию, действуя жадно:

$$\pi'(s) = \underset{a \in A}{\operatorname{argmax}} Q^{\pi}(s, a)$$

  - Это улучшает полезность от любого состояния *s* на один шаг

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

  - Таким образом мы улучшаем функцию полезности:

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

Улучшение стратегии

- Когда процесс улучшения останавливается, мы получаем $a = \pi(s)$:

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

  - Тогда уравнение оптимальности Беллмана будет удовлетворено:

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

  - А это означает, что $V^\pi(s) = V^*(s)$ для всех $s \in S$)
  - Стратегия π будет оптимальной

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': $V^\pi(s') =$

$V^*(s')$

Детерменированные итерации по полезностям

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

$$V^*(s) \leftarrow \max_{a \in A}\left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^*(s')\right)$$

- 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

Итерации по полезностям

- Задача: найти оптимальную стратегия π^*
- Решение: итеративное применение уравнения оптимальности Беллмана:

$$V^1 \to V^2 \to V^3 \to \cdots \to V^n$$

- 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*
- Сходится к истинным значениям V^*
- В отличии от итерации по стратегиям, мы не получаем стратегию в явном виде
- Промежуточные значения полезностей могут не соответствовать ни одной стратегии

Итерации по полезностям

$$V^{k+1}(s) = \max_{a \in A}(R_s^a + \Upsilon \sum_{s' \in S} P_{ss'}^a V^k(s'))$$

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