

---

# Лекция 1: Введение, динамическое программирование

Алексей Скрынник  
Артем Латышев

# План лекции

- 1 Обучение с подкреплением: определения и примеры
- 2 Постановка задачи: вознаграждение, состояние, наблюдаемость
- 3 Схема RL-агента: стратегия, полезность, модель
- 4 Подзадачи в обучении с подкреплением
- 5 Марковское свойство и марковские процессы
- 6 Уравнение Беллмана для полезности
- 7 Функция полезности
- 8 Оптимальные функция полезности и стратегия
- 9 Алгоритм итерации по стратегиям
- 10 Итерация по полезностям

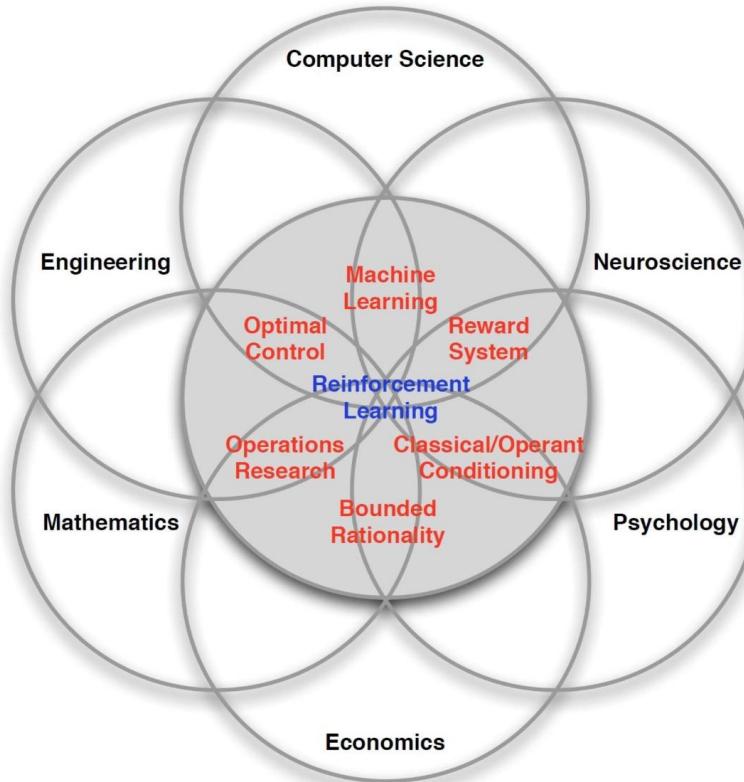
---

# Обучение с подкреплением: определения и примеры

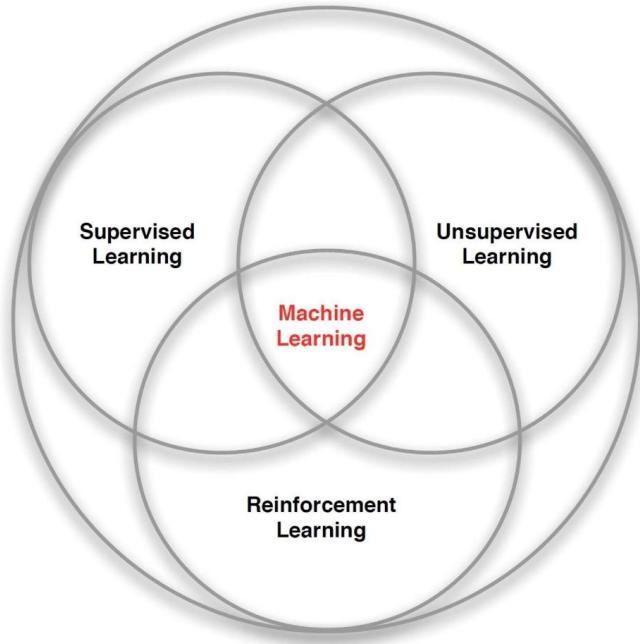
# Обучение с подкреплением

Обучение интеллектуального агента  
хорошой последовательности принятия решения  
в среде с неполной информацией

# Обучение с подкреплением и другие науки



# Особый вид машинного обучения



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

# Примеры обучения с подкреплением

- AlphaGo - игра в Go лучше человека
- Игра в игры Atari, Starcraft, Dota, MineRL - иногда лучше человека
- Управление движением мобильного и человекоподобного робота
- Управление энергетической станцией
- Управление инвестиционным пакетом
- Автоматизация «холодных звонков»
- Реализация сложных движений вертолета

---

Постановка задачи:  
вознаграждение, состояние,  
наблюдаемость

# Вознаграждение:

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

В обучении с подкреплением принята следующая гипотеза:

## Definition

Все цели могут быть описаны через максимизацию суммарного вознаграждения.

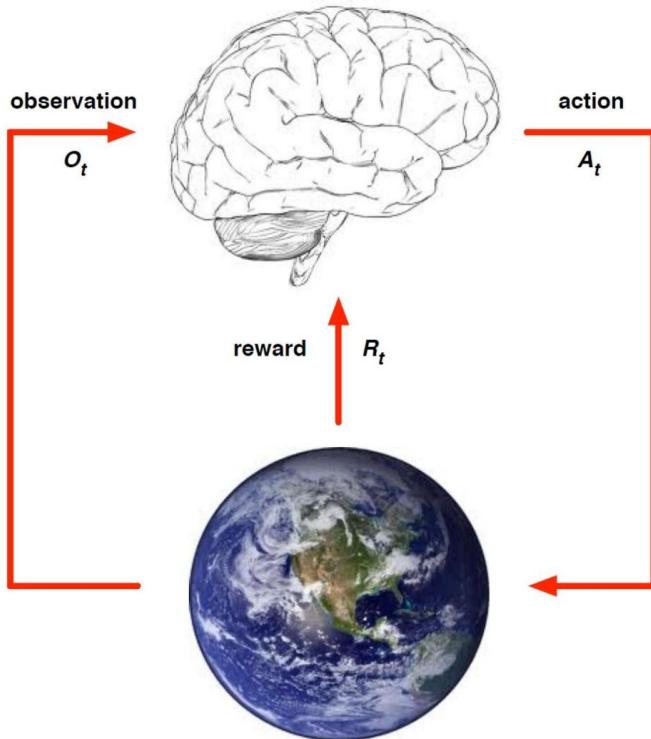
Примеры:

- *AlphaGo*:  $+r$  за победу в партии,  $-r$  за проигрыш
- *Atari*:  $+r$  за увеличение счета,  $-r$  за уменьшение счета
- *Робот*:  $+r$  за прямой шаг,  $-r$  за падение
- *Энергетическая станция*:  $+r$  за генерацию энергии,  $-r$  за превышение ограничений безопасности
- *Инвестиционный пакет*:  $+r$  за каждый заработанный рубль,  $-r$  за каждый потерянный рубль
- *Автоматизация «холодных звонков»*:  $+r$  за привлечение клиента,  $-r$  плата за звонок
- *Вертолет*:  $+r$  за следование намеченной траектории,  $-r$  за падение

# Последовательное принятие решений

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

# Взаимодействие агенты и среды



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

# История и состояние

- История (взаимодействия) это последовательность наблюдений, действий и вознаграждений:

$$H_t = \langle o_1, r_1, a_1, \dots, a_{t-1}, o_t, r_t \rangle$$

Например: все наблюдаемые переменные к моменту времени  $t$  или сенсомоторный поток робота.

- То, что случится далее зависит от истории:
  - ▶ агент выберет действие,
  - ▶ среда сгенерирует наблюдение и вознаграждение.
- Состояние – это информация используемая для определения того, что произойдет далее:

$$s_t = f(H_t)$$

# Марковское состояние

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

## Definition

Состояние  $s_t$  называется марковским, если и только если

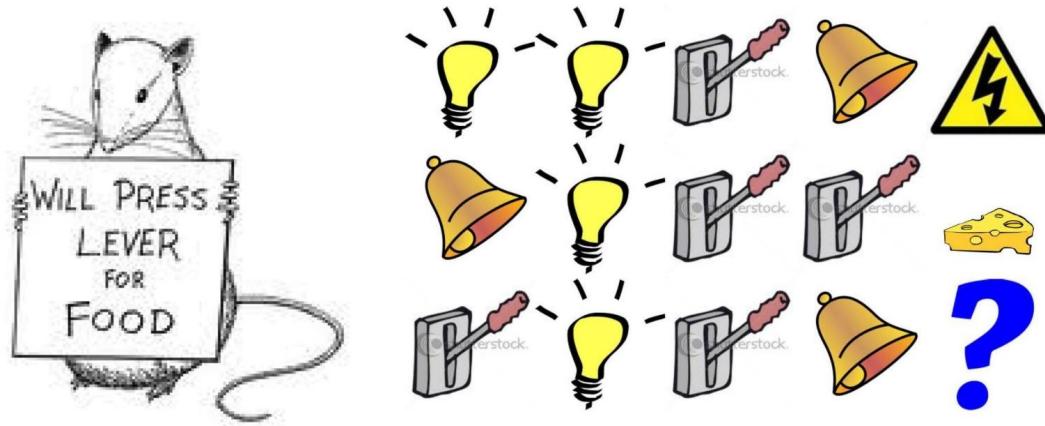
$$\mathbb{P}[s_{t+1}|s_t] = \mathbb{P}[s_{t+1}|s_1, \dots, s_t]$$

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

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

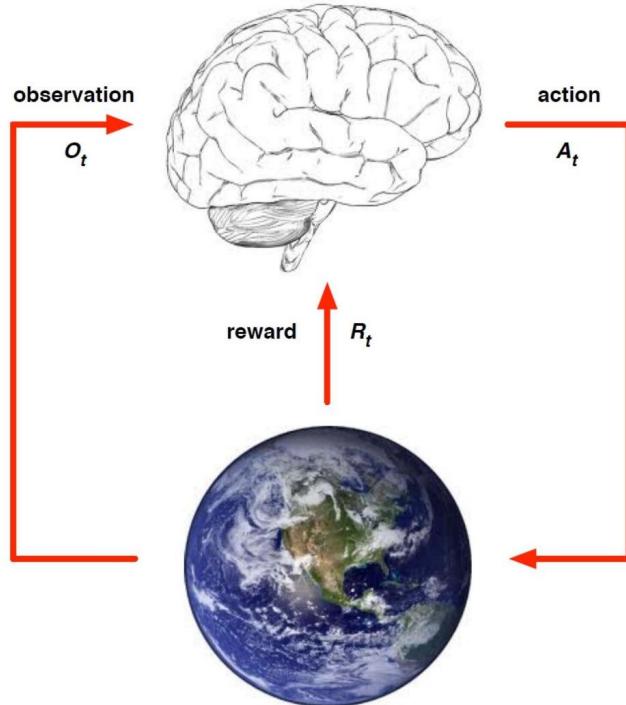
- Если известно текущее состояние, история может быть отброшена
- Текущее состояние содержит всю необходимую статистику о будущем
- Предполагаем, что состояние  $s_t$  является марковским

# Мышь и сыр



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

# Полностью наблюдаемые среды



- Полная наблюдаемость: агенту напрямую доступно состояние среды:  
$$o_t = s_t$$
- Формально такой случай взаимодействия называется *марковским процессом принятия решений* (МППР, Markov decision process, MDP)

# Частично наблюдаемые среды

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

---

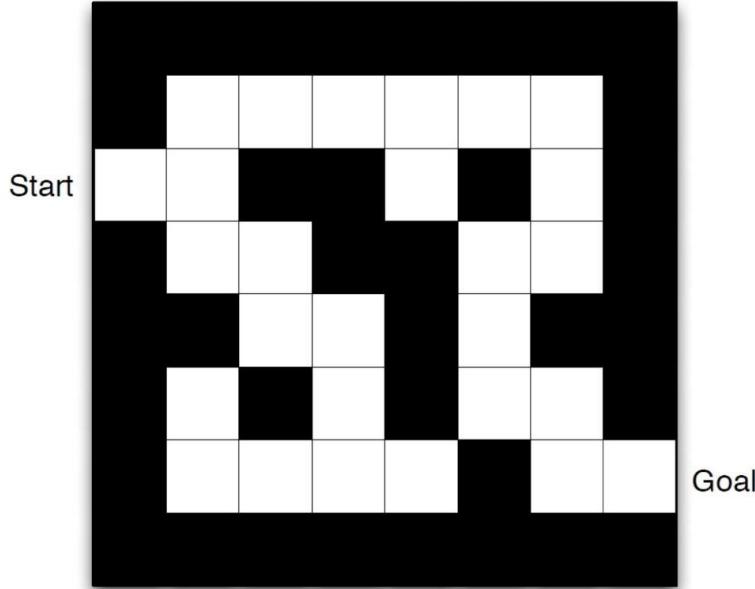
# Схема RL-агента: стратегия, полезность, модель

# Строение RL агента

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

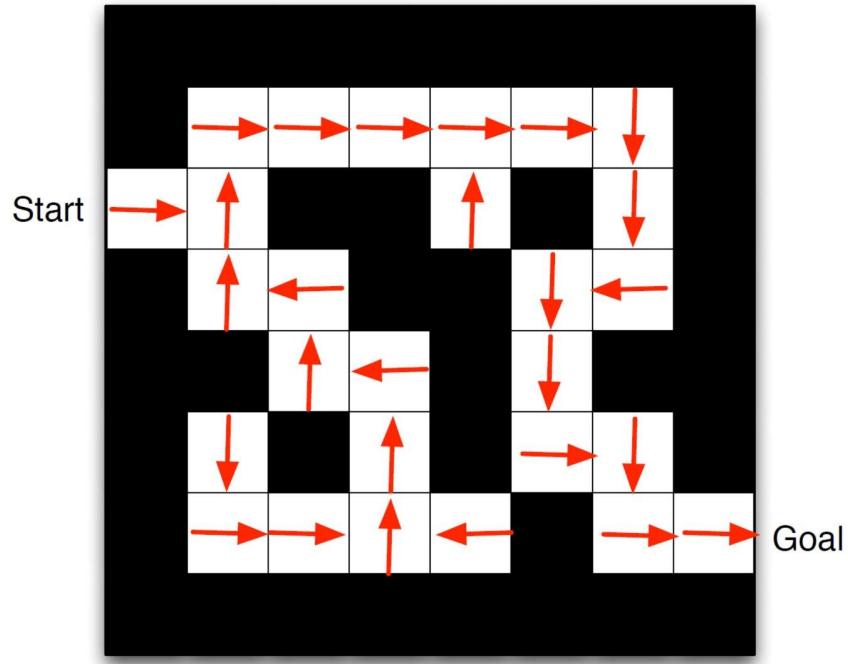
- **стратегия** (policy): функция поведения агента, обычно это отображение из множества состояний в множество действий (детерминированная стратегия  $a = \pi(s)$ , стохастическая стратегия  $\pi(a|s) = \mathbb{P}[a_t = a|s_t = s]$ ),
- **функция полезности** (value function): оценка того, насколько полезно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

# Лабиринт



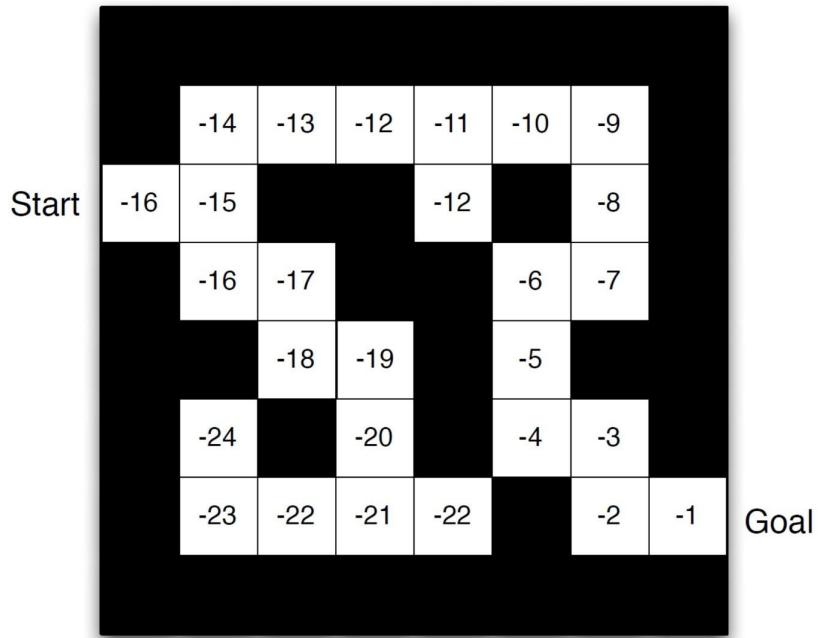
- Вознаграждения: -1 за каждый шаг
- Действия:  $\uparrow, \leftarrow, \rightarrow, \downarrow$
- Состояния: местоположение агента

# Лабиринт: стратегия



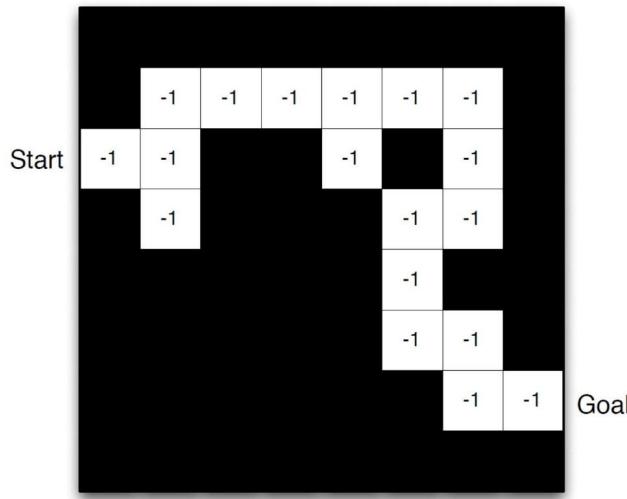
Стрелки отображают стратегию  $\pi(s)$  для каждого состояния  $s$

# Лабиринт: функция полезности



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

# Лабиринт: модель



- Клетки отображают модель переходов  $\mathcal{P}_{ss'}^a$
- Числа представляют следующее вознаграждение  $\mathcal{R}_s^a$  для каждого состояния  $s$

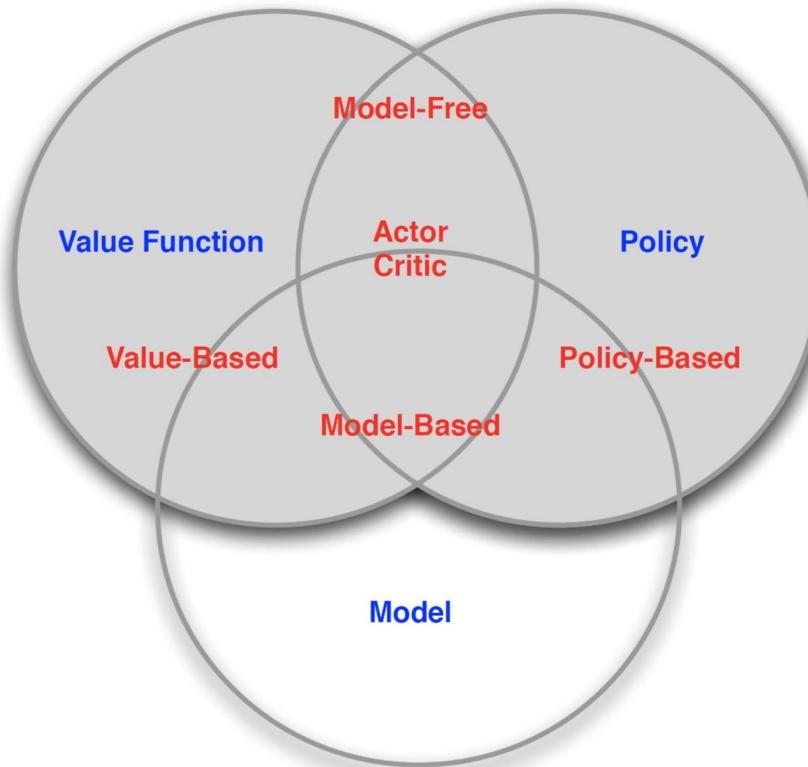
- Агент может обладать внутренней моделью среды
- Динамика: как действия меняют среду
- Вознаграждения: какое вознаграждение может быть получено в каждом состоянии
- Модель может быть неточной

# Типизация RL агентов

- Оценивающие функцию полезности (value based):
    - ▶ стратегия не представлена явно,
    - ▶ функция полезности.
  - Оценивающие стратегию (policy based):
    - ▶ стратегия,
    - ▶ функция полезности не вычисляется явно.
  - Актор-критик (actor-critic):
    - ▶ стратегия,
    - ▶ функция полезности.
- 

- Безмодельные (model free):
  - ▶ стратегия и/или функция полезности,
  - ▶ нет модели.
- Основанные на модели (model based):
  - ▶ стратегия и/или функция полезности,
  - ▶ строят модель.

# Таксономия RL агентов



---

# Подзадачи в обучении с подкреплением

# Планирование и обучение

В теории последовательного принятия решений существует две основных постановки задачи

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

# Исследование и применение

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

Примеры:

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

---

# Марковское свойство и марковские процессы

# Марковский процесс принятия решений

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

# Матрица переходов

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

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

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

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

Каждая строчка матрицы в сумме дает 1

# Марковский процесс

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

## Definition

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

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

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

# Суммарное вознаграждение

## Definition

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

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

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

# ФУНКЦИЯ ПОЛЕЗНОСТИ

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

## Definition

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

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

---

# Уравнение Беллмана для полезности

# Уравнение Беллмана

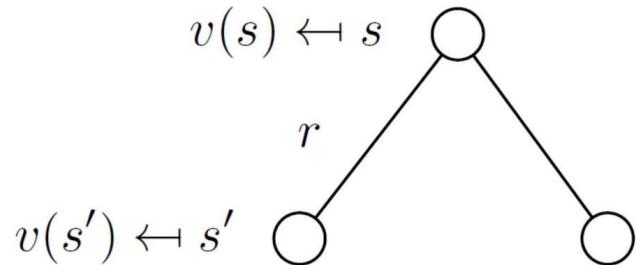
Функцию полезности можно представить в виде двух слагаемых:

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

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

# Уравнение Беллмана

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



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

# Уравнение Беллмана в матричной форме

Мы можем записать уравнение Беллмана в матричной форме:

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

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

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

# Решение уравнения Беллмана

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

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

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

# Марковский процесс принятия решений

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

## Definition

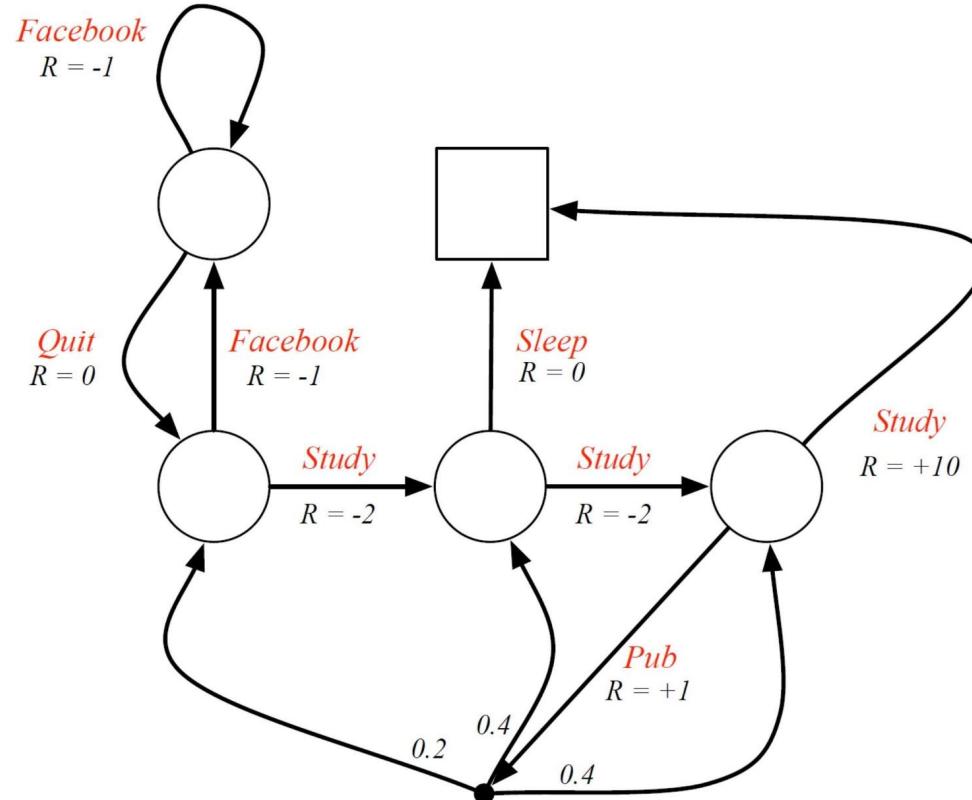
Марковский процесс принятия решений (МППР) – это кортеж  $\langle S, A, \mathcal{P}, \mathcal{R}, \gamma \rangle$ , где

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

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

- $\mathcal{R}$  – 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

## Definition

Стратегией  $\pi$  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  $\pi$
- Последовательность состояний  $s_1, s_2, \dots$  – марковский процесс  $\langle S, \mathcal{P}^\pi \rangle$
- Последовательность состояний и вознаграждений  $s_1, r_1, s_2, \dots$  – марковский процесс вознаграждений  $\langle S, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$ , где

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

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

---

# Функция полезности

# ФУНКЦИЯ ПОЛЕЗНОСТИ

## Definition

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

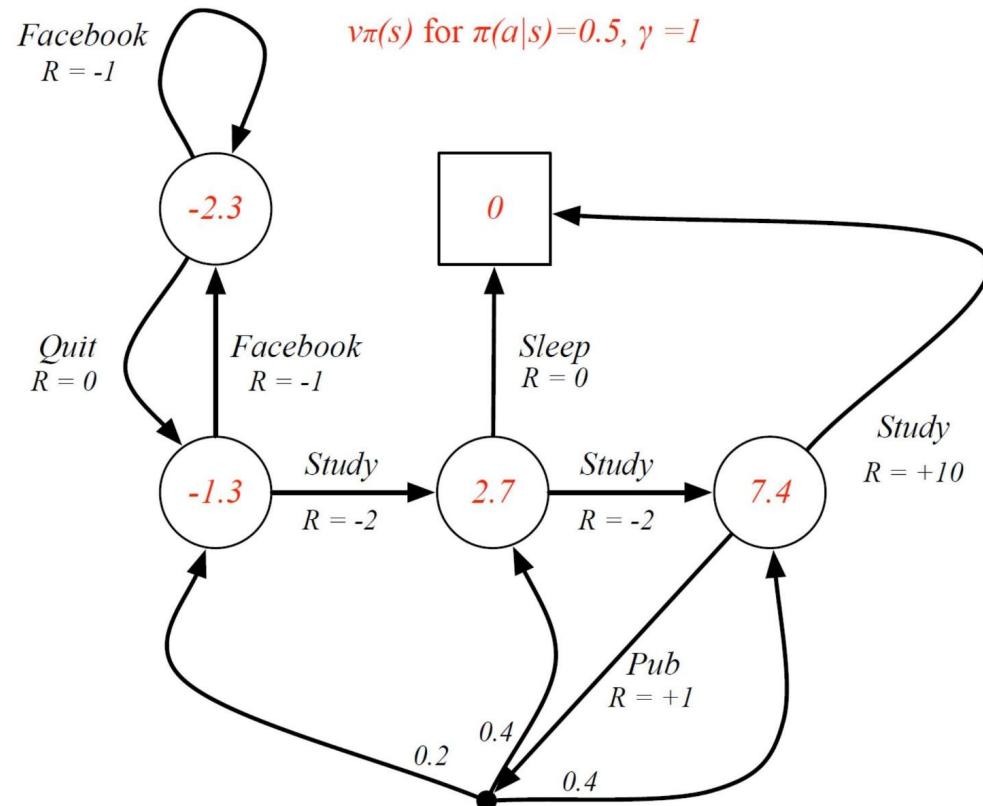
$$V^\pi(s) = \mathbb{E}_\pi[R_t | s_t = s]$$

## Definition

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  $\pi$ :

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

# Пример: функция полезности для студенческого МППР



# Уравнение Беллмана для стратегии и МППР

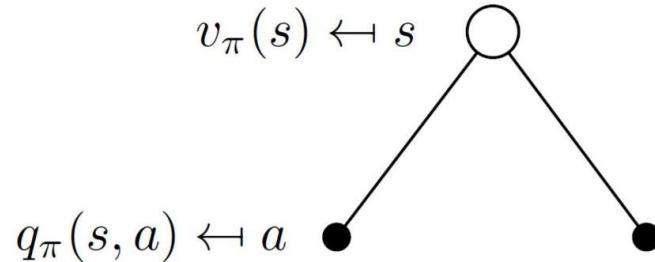
Функция полезности состояний может быть разложена на немедленное вознаграждение и дисконтируемую ценность следующего состояния:

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

Для функции полезности действий аналогично:

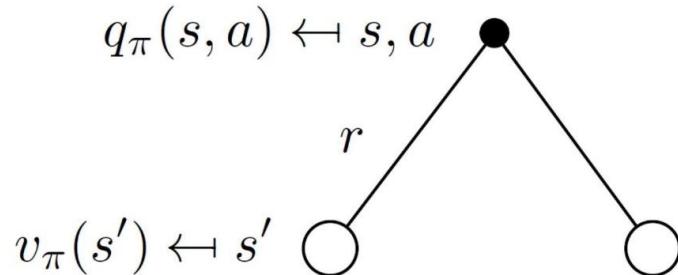
$$Q^\pi(s, a) = \mathbb{E}_\pi[r_{t+1} + \gamma Q^\pi(s_{t+1}, a_{t+1}) | s_t = s, a_t = a]$$

# Уравнение Беллмана $V$



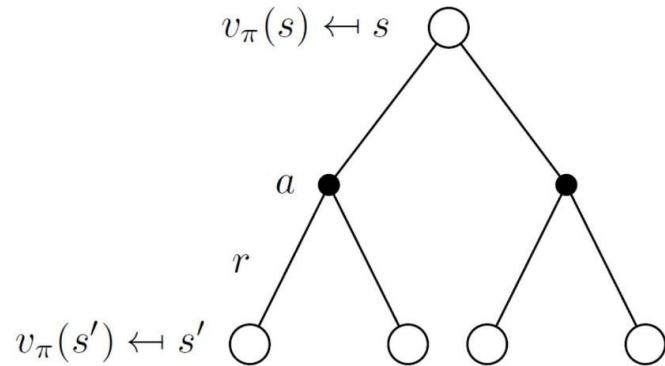
$$V^\pi(s) = \sum_{a \in A} \pi(a|s) Q^\pi(s, a)$$

# Уравнение Беллмана для Q



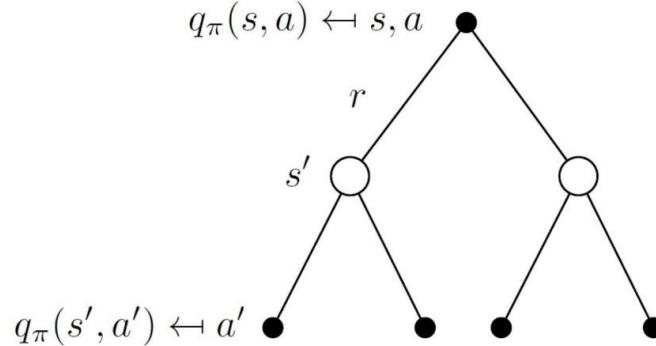
$$Q^{\pi}(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a V^{\pi}(s')$$

# Уравнение Беллмана



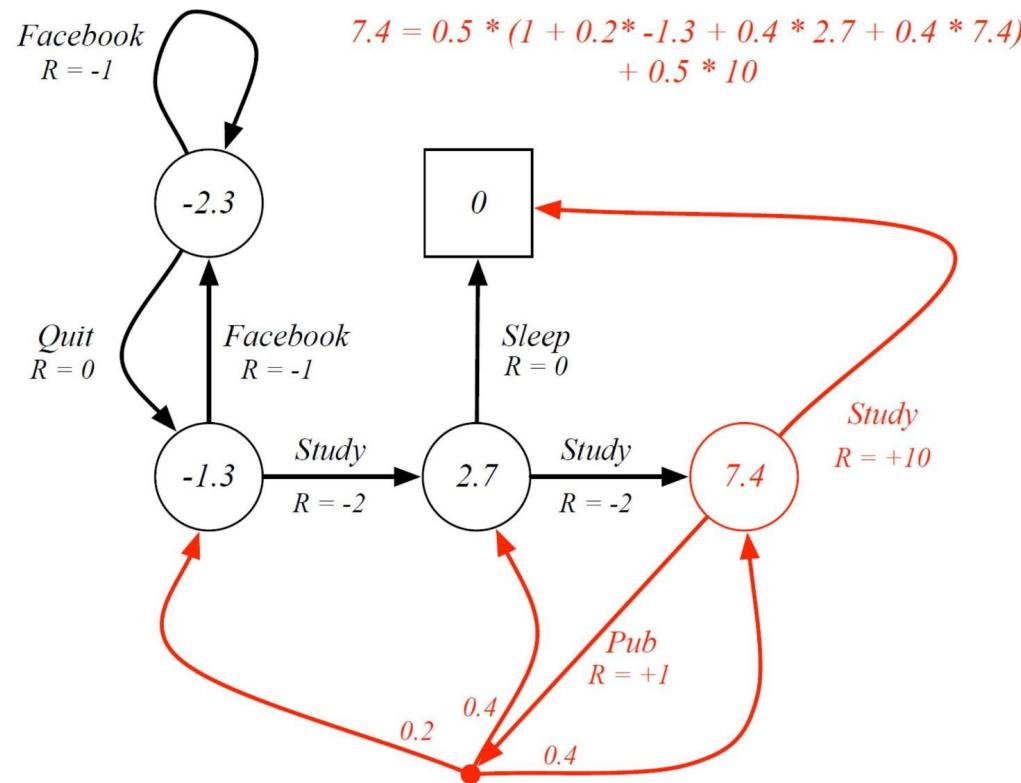
$$V^\pi(s) = \sum_{a \in A} \pi(a|s) \left( \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a V^\pi(s') \right)$$

# Уравнение Беллмана



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

# Пример: уравнение Беллмана для студенческого МППР



# Матричная форма уравнения Беллмана для МППР

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

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

Аналитическое решение:

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

---

# Оптимальные функции полезности и стратегия

# Оптимальная функция полезности

## Definition

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

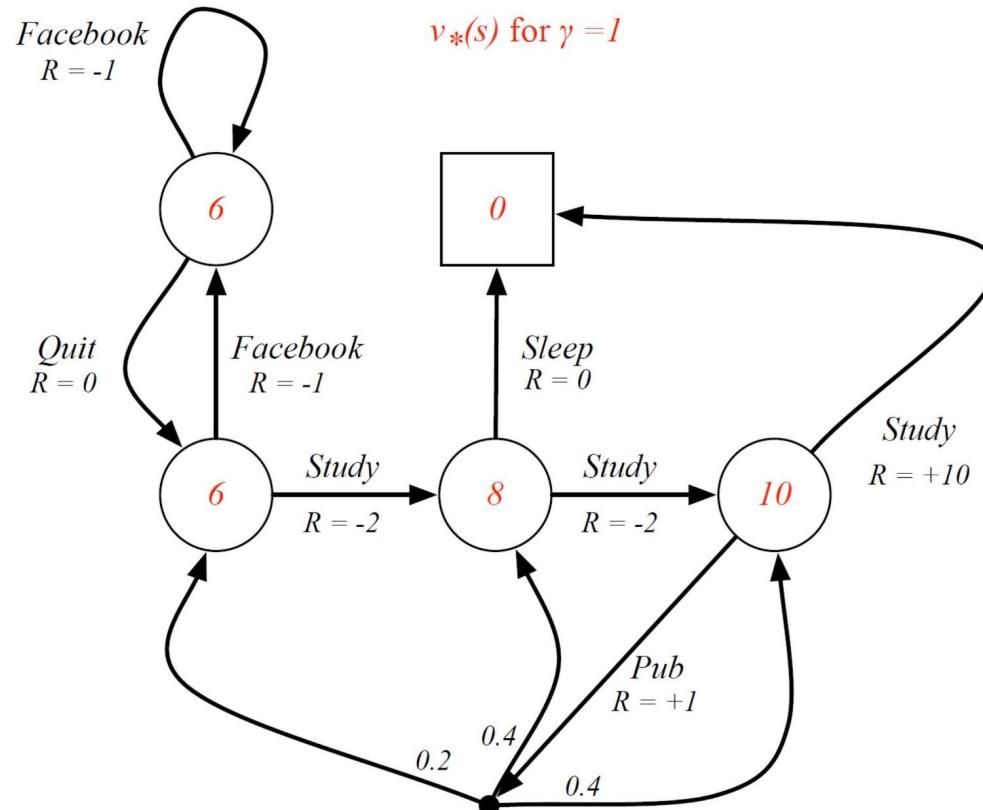
$$V^*(s) = \max_{\pi} V^{\pi}(s).$$

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

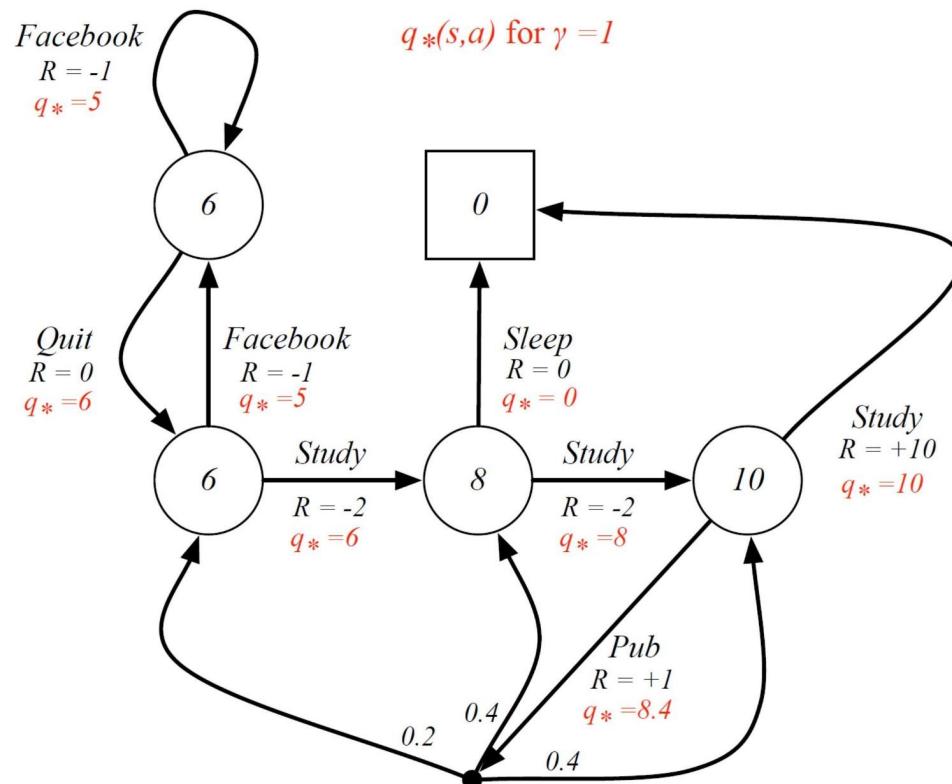
$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a).$$

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

# Пример оптимальной функции полезности V



# Пример оптимальной функции полезности Q



# Оптимальная стратегия

Определим частичный порядок на множестве стратегий:

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

## Theorem

Для любого марковского процесса принятия решений:

- существует оптимальная стратегия  $\pi^*$ , 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:  $Q^{\pi^*}(s, a) = Q^*(s, a)$

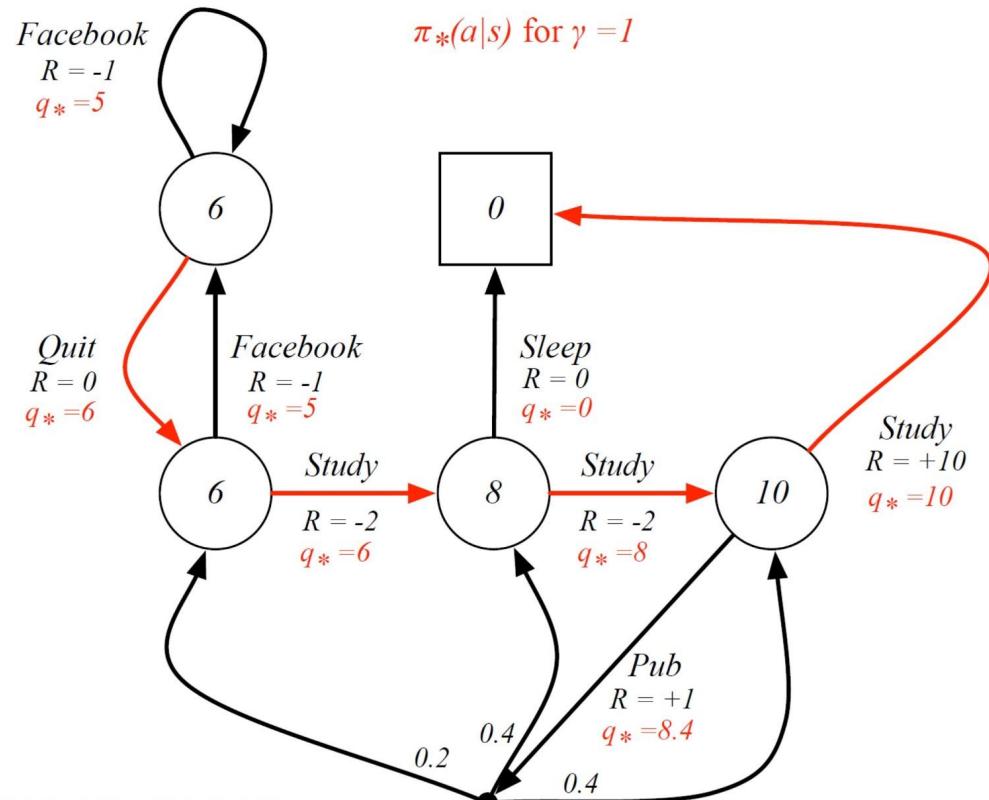
# Поиск оптимальной стратегии

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

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

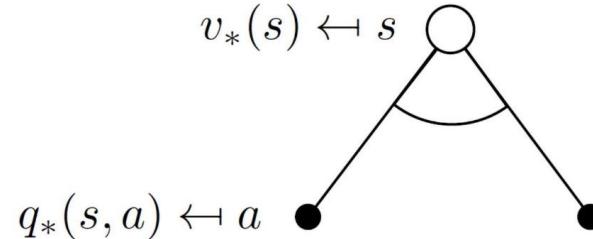
- Для любого MDP существует оптимальная детерминированная стратегия
- Если известна  $Q^*(s, a)$ , то мы одновременно получаем и оптимальную стратегию

# Пример: оптимальная стратегия для студенческого МППР



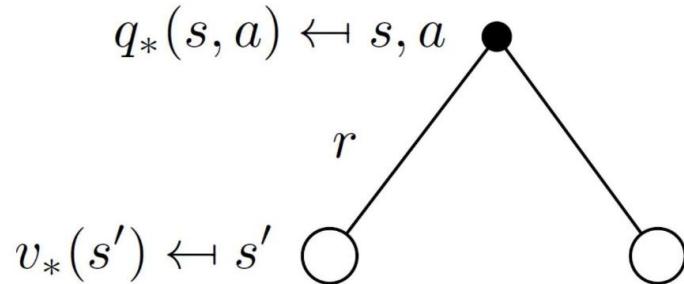
# Уравнение Беллмана для оптимальной $V$

Оптимальные значения функции полезности связаны уравнением оптимальности Беллмана:



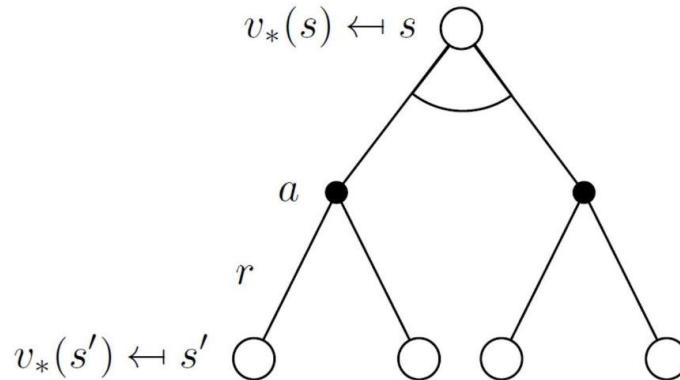
$$V^*(s) = \max_{a \in A} Q^*(s, a)$$

# Уравнение Беллмана для оптимальной Q



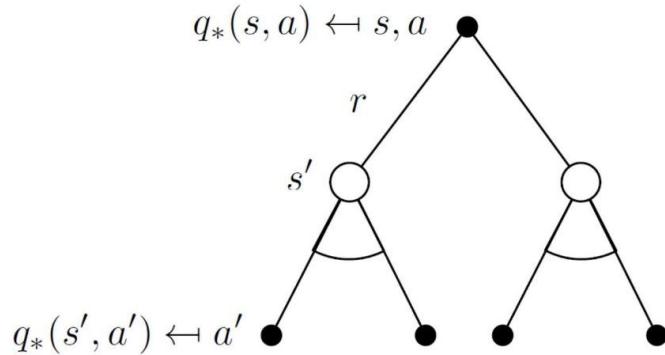
$$Q^*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a V^*(s')$$

# Уравнение Беллмана для оптимальной $V$



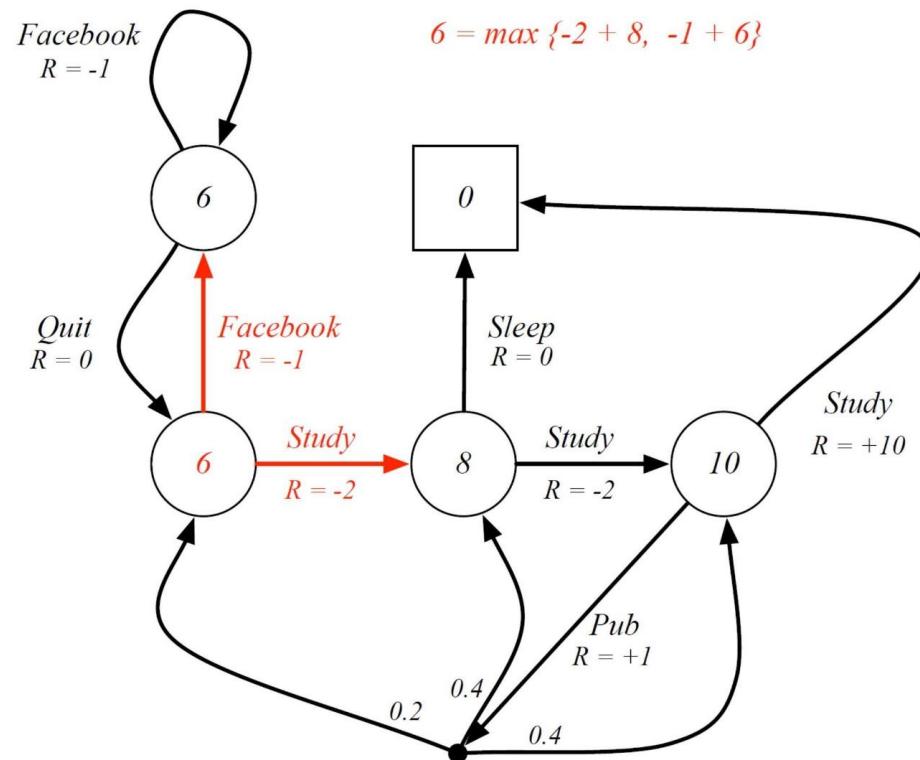
$$V^*(s) = \max_{a \in A} (R_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a V^*(s'))$$

# Уравнение Беллмана для оптимальной 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

---

# 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 повторяются 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:

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

или

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

- Для задачи поиска оптимальной стратегии:

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

# Итерационная оценка стратегии

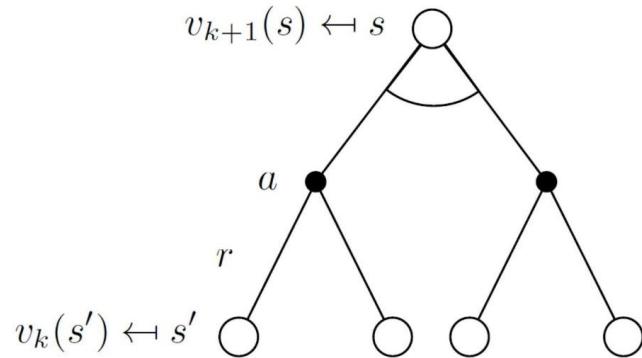
**Задача:** оценить текущую стратегию  $\pi$

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

$$V^1 \rightarrow V^2 \rightarrow \dots \rightarrow V^\pi$$

- 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  $V^\pi$

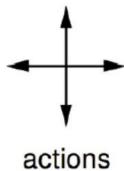
# Итерационная оценка стратегии



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

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

# Пример: случайная стратегия в клеточном мире



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

$r = -1$   
on all transitions

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

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

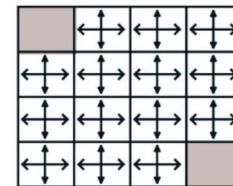
# Пример: итерационная оценка стратегии

$V^k$  для случайной стратегии

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

$k = 0$

Жадная стратегия по  $V^k$

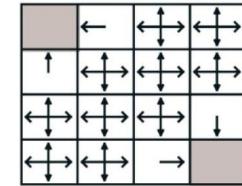


← случайная  
стратегия

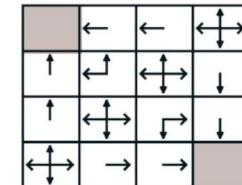
$k = 1$

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

$k = 2$



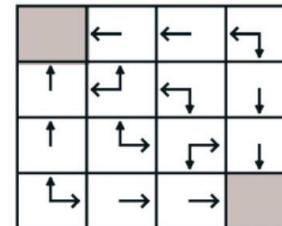
0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



# Пример: итерационная оценка стратегии

$k = 3$

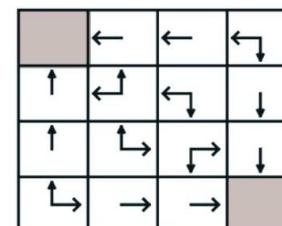
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0



оптимальная  
стратегия

$k = 10$

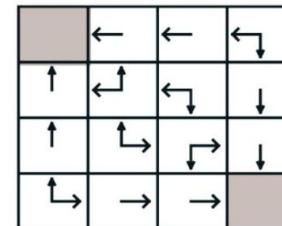
0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0



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

- Данна стратегия  $\pi$ :

- ▶ Оценить (evaluate) стратегию  $\pi$ :

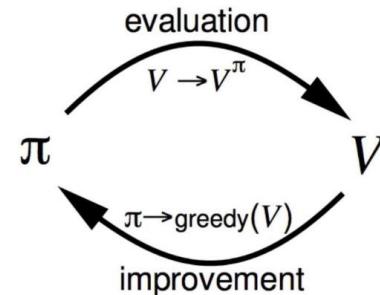
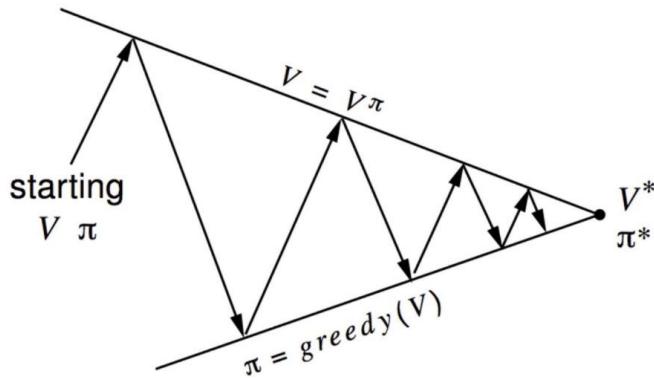
$$V^\pi(s) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \dots | s_t = s]$$

- ▶ Улучшить (improve) стратегию, выбирая действия жадно (greedy), но в соответствии с  $V^\pi$ :

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

- 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  $\pi^*$

# Итерация по стратегиям (Policy Iteration - PI)



**Оценка стратегии** – вычисление  $V^\pi$

Итеративная оценка стратегии

**Улучшение стратегии** – генерация

$$\pi' \geq \pi$$

Жадное обновление стратегии

$$\pi^* \xleftrightarrow{V^*} V^*$$

---

# 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  $s'$ .

## 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  $s$ ,  $\pi$  достигает оптимальной оценки состояния  $s'$ :  $V^\pi(s') = V^*(s')$ .

# Детерминированные итерации по ценностям

- 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

$V_7$

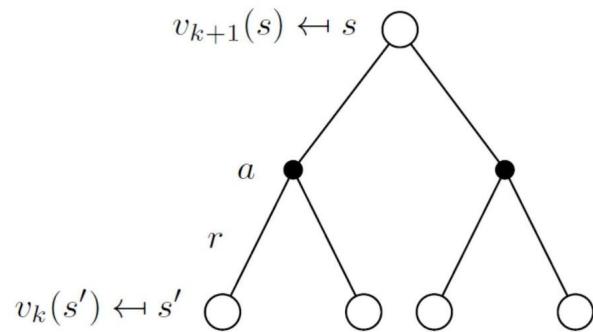
# Итерация по полезностям

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

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

- 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)

# Итог:

- Обучение с подкреплением (RL) – особый раздел машинного обучения, в котором важно последовательное поступление данных из среды, меняющейся в следствие действий агента
- В постановке задачи RL особую роль играет способ задания вознаграждения и состояния, доступного агенту
- Все RL-агенты имеют общие компоненты: стратегию, аппроксиматор полезности и, возможно, модель среды
- В обучении с подкреплением можно выделить подзадачи, связанные с долгосрочным планированием, исследование среды и обновлением стратегии
- Марковский процесс принятия решений – формальное описание процесса взаимодействия агента и среды
- Ключевое понятие динамического программирования – уравнение Беллмана
- Основная задача – поиск оптимальной функции стратегии или стратегии
- Аналитически решить уравнение оптимальности Беллмана невозможно – используем приближенные методы
- Базовые алгоритмы при известной функции переходов: итерации по стратегиям и полезностям

---

Спасибо за внимание!