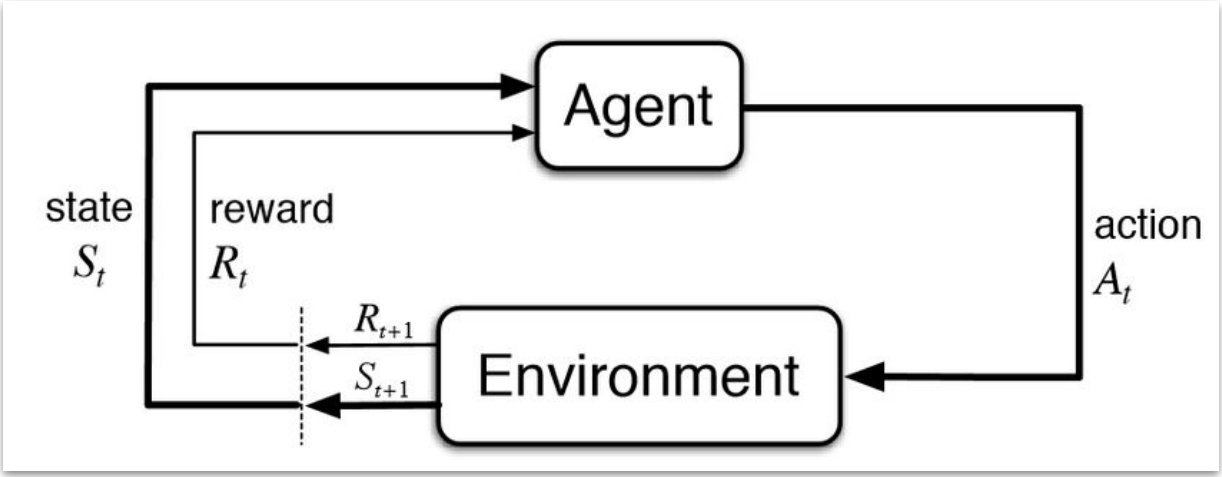
Policy и value iteration

Николаев Максим, 192

Постановка задачи



Хотим максимизировать награду G, получаемую агентом

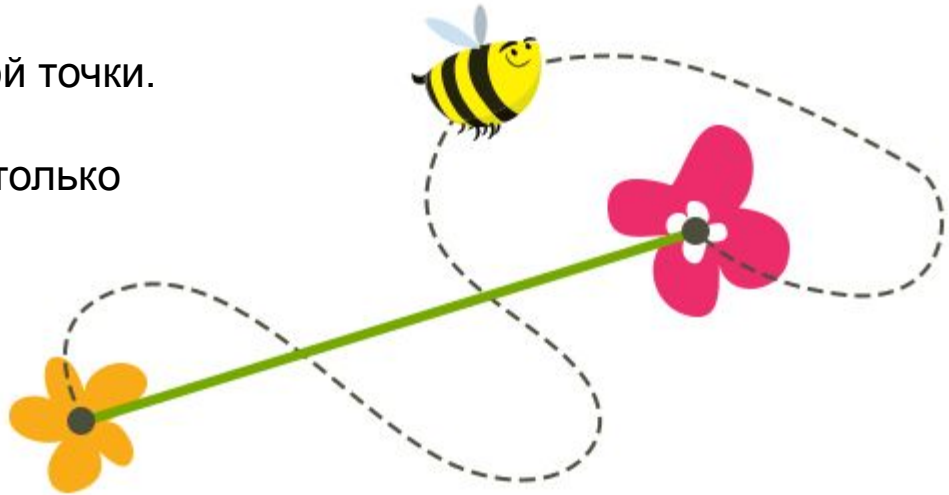
$$G = R_1 + \gamma R_2 + ... + \gamma^{T-1} R_T$$

Коэффициент дисконтирования: $0 \leqslant \gamma < 1$

А зачем дисконтирование?

Задача научить робота
добираться до финальной точки.

Робот получает награду только
по достижению финиша.



Без дисконтирования обе траектории для
робота одинаково хороши.

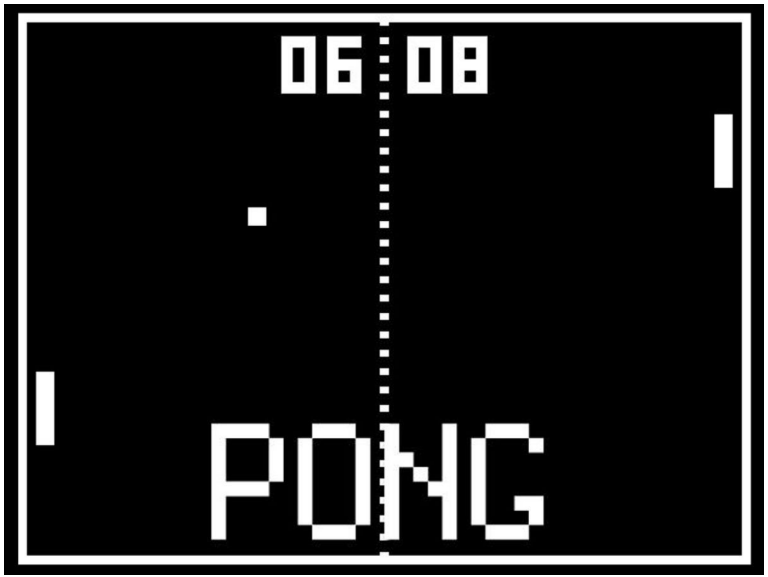
Марковский процесс принятия решений (MDP)

Следующее состояние и награда зависит только от текущего состояния и действия:

$$p(r_t, s_{t+1}|s_0, a_0, r_0, \ldots, s_t, a_t) = p(r_t, s_{t+1}|s_t, a_t)$$

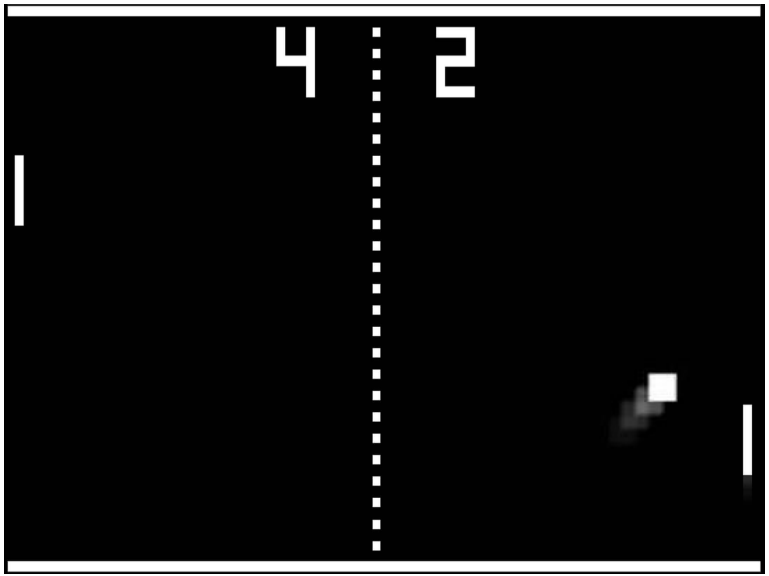
Примеры Марковского процесса

Не Марковский процесс



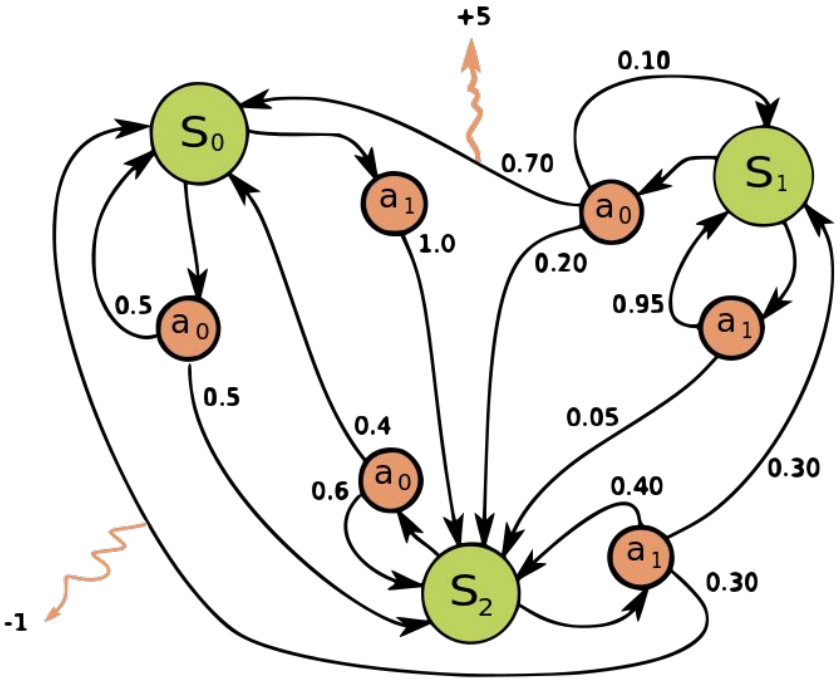
Без предыдущих состояний не знаем направление мяча

Марковский процесс



Можем узнать направление мяча по следу

Политика



Классический пример MDP

Пример политики (policy)

S	$\pi(a_0\|S)$	$\pi(a_1\|S)$
s_0	0.5	0.5
s_1	0.9	0.1
s_2	0	1

Решением задачи будет политика, которая максимизирует награду

Уравнения Беллмана

Награда за сессию с момента времени t:

$$G_t = R_t + \gamma R_{t+1} + ... + \gamma^{T-t} R_T = R_t + \gamma G_{t+1}$$

Action-value – это математическое ожидание наград оставшейся сессии, если сделаем действие a из состояния s

$$q_\pi(s,a) = \mathbb{E}_\pi[R_t + \gamma G_{t+1} | S_t = s, A_t = a] = \sum_{r,s'} p(r,s'|s,a) \cdot [r + \gamma v_\pi(s')]$$

State-value – это математическое ожидание наград оставшейся сессии, если сейчас находимся в состоянии s

$$v_\pi(s) = \mathbb{E}_\pi[R_t + \gamma G_{t+1} | S_t = s] = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \cdot \sum_{r,s'} p(r,s'|s,a)[r + \gamma v_\pi(s')]$$

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \cdot q_\pi(s,a)$$

Принцип оптимальности Беллмана

Можем выписать оценку сверху для state-value:

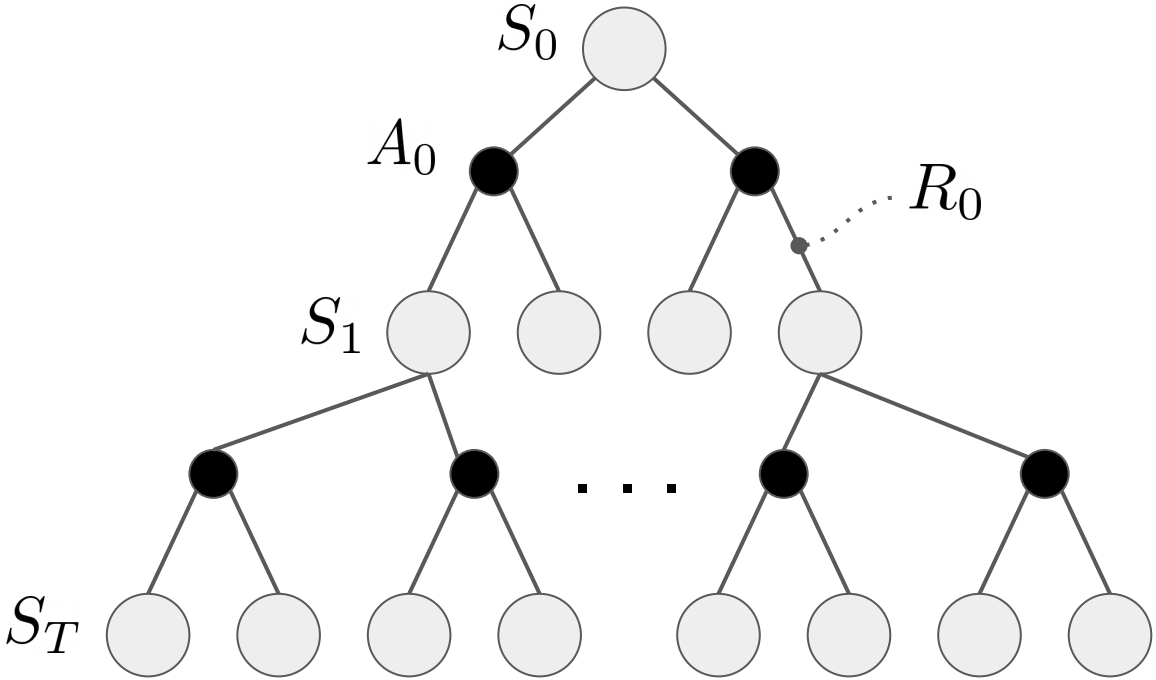
$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \cdot q_\pi(s,a) \leqslant \max_{a \in \mathcal{A}(s)} q_\pi(s,a) \left[\sum_{a \in \mathcal{A}(s)} \pi(a|s) \right] = \max_{a \in \mathcal{A}(s)} q_\pi(s,a)$$

Оценка достигается на новой оптимальной политике:

$$\pi_*(a|s) = \begin{cases} 1, & a = \operatorname*{argmax}_{a \in \mathcal{A}(s)} q_\pi(s,a) \\ 0, & \text{иначе} \end{cases}$$

$$v_{\pi_*}(s) = \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a)$$

Как найти оптимальную политику?



Backup Tree

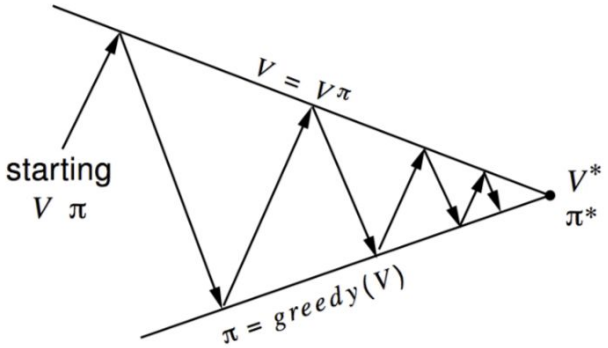
Идея generalized value iteration для решения MDP

Policy iteration

1) Вычисляем политику, пока сходимся с определенным допуском
2) Улучшаем политику

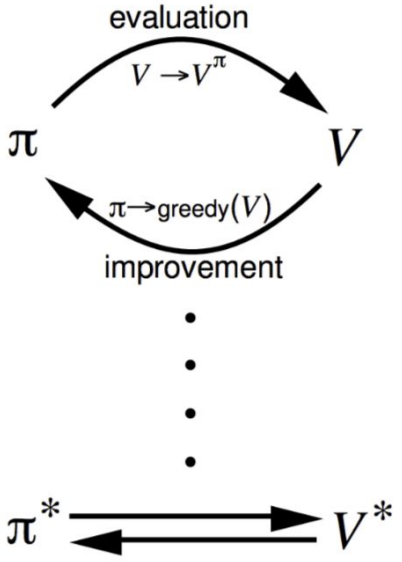
Value Iteration

1) Вычисляем политику только на одном шаге
2) Улучшаем политику



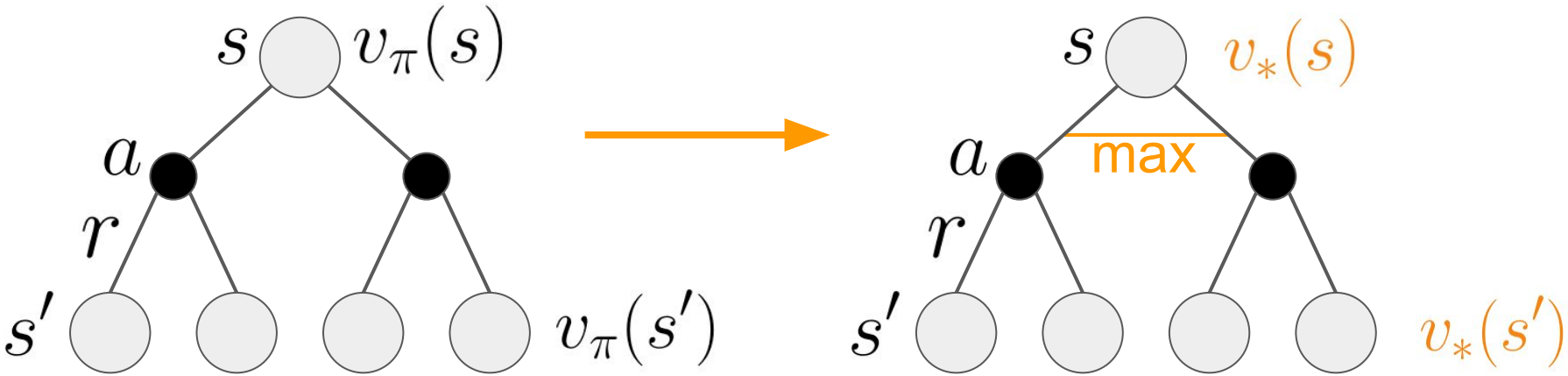
Policy evaluation Estimate v_π
Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
Greedy policy improvement



Идея generalized value iteration для решения MDP

Bellman optimality equation for **v(s)**



Bellman expectation equation for v(s)

Bellman optimality equation for $v_*(s)$

Policy iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

Policy iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V(s') \right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

Уравнение Беллмана для value-state v(s)

Policy iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
      $\Delta \leftarrow 0$
      Loop for each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V(s') \right]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      old-action $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma V(s') \right]$
      If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Уравнение Беллмана для value-state v(s)

Уравнение Беллмана для action-value q(s, a)

Value iteration

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$

action-value q(s, a)

Пример policy iteration

Нужно пересечь поле, попав в терминальные вершины

1	2	3	4
5	6 	7	8
9	10	11	12
13	14	15	16

Gridworld

actions 

Reward is -1 for all transition

Пример policy iteration

Делаем инициализацию value-state

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

Делаем инициализацию policy, с равновероятными действиями и делаем итерацию policy evaluation

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

$$v_1(6) = \sum_{a\in\{u,d,l,r\}} \pi(a|6) \sum_{s',r} p(s',r|6,a)[r+\gamma v_0(s')]$$

$$= \sum_{a\in\{u,d,l,r\}} \underbrace{\pi(a|6)}_{=0.25\ \forall a} \sum_{s'} p(s'|6,a)[\underbrace{r}_{=-1} + \gamma \underbrace{v_0(s')}_{=0\ \forall s'}]$$

$$= 0.25 * \{-p(2|6,u) - p(10|6,d) - p(5|6,l) - p(7|6,r)\}$$

$$= 0.25 * \{-1-1-1-1\}$$

$$= -1$$

Пример policy iteration

После первой итерации policy evaluation
получаем новые значения value-state

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

Пример policy iteration

Ещё одна итерация policy evaluation
коэффициент дисконтирования полагаем равным 1

$$v_2(6) = \sum_{a\in\{u,d,l,r\}} \underbrace{\pi(a|6)}_{=0.25\ \forall a} \sum_{s'} p(s'|6,a)[\underbrace{r}_{=-1} + \gamma \underbrace{v_1(s')}_{=\begin{cases}-1, s'\in S\\ 0, s'\in S^+\setminus S\end{cases}}]$$

$$= 0.25 * \{p(2|6,u)[-1-\gamma] + p(10|6,d)[-1-\gamma] + p(5|6,l)[-1-\gamma] + p(7|6,r)[-1-\gamma]\}$$

$$\overset{\gamma=1}{=} 0.25 * \{-2-2-2-2\}$$

$$= -2$$

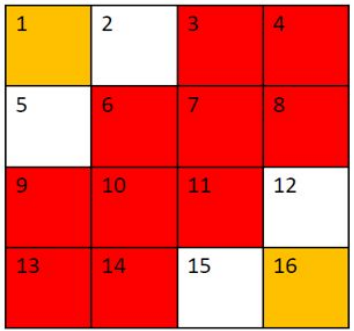
$$v_2(2) = \sum_{a\in\{u,d,l,r\}} \underbrace{\pi(a|2)}_{=0.25\ \forall a} \sum_{s'} p(s'|2,a)[\underbrace{r}_{=-1} + \gamma \underbrace{v_1(s')}_{=\begin{cases}-1, s'\in S\\ 0, s'\in S^+\setminus S\end{cases}}]$$

$$= 0.25 * \{p(2|2,u)[-1-\gamma] + p(6|2,d)[-1-\gamma] + p(1|2,l)[-1-\gamma*0] + p(3|2,r)[-1-\gamma]\}$$

$$\overset{\gamma=1}{=} 0.25 * \{-2-2-1-2\}$$

$$= -1.75$$

$$\Rightarrow v_2(2) = -1.75$$



$\Rightarrow v_2$ for the random policy:

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 policy iteration

Повторяем итерацию вычисления k раз

$k = 0$

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

$k = 1$

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

$\cdots$

$k = 10$

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

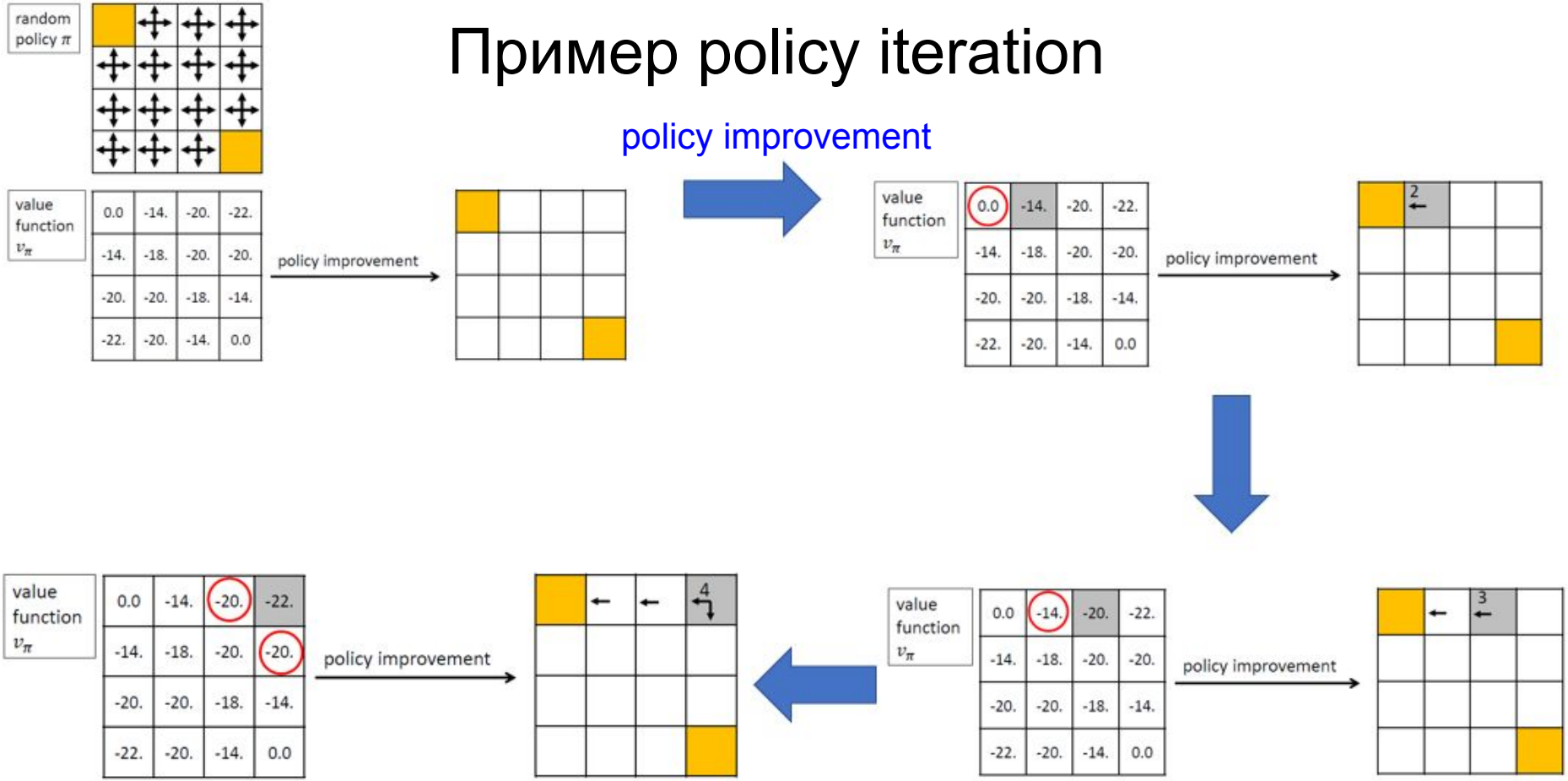
$\cdots$

$k = \infty$

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

$\leftarrow v_\pi$

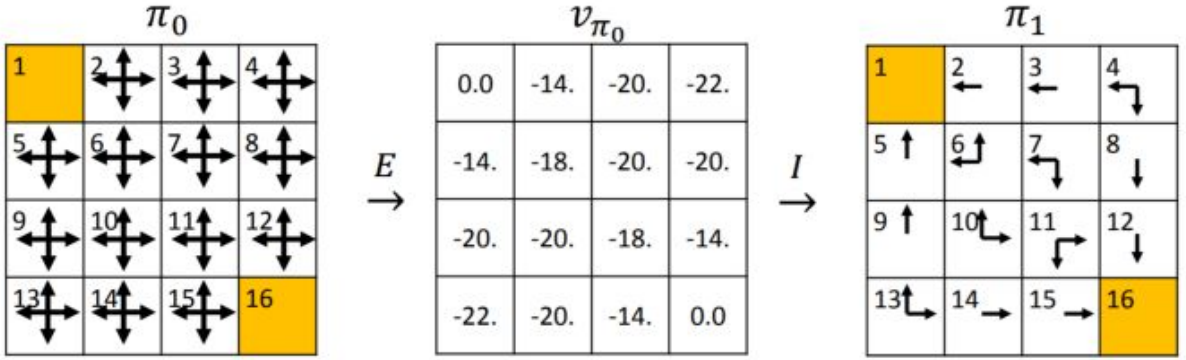
Пример policy iteration



Пример policy iteration

Получили финальное policy

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

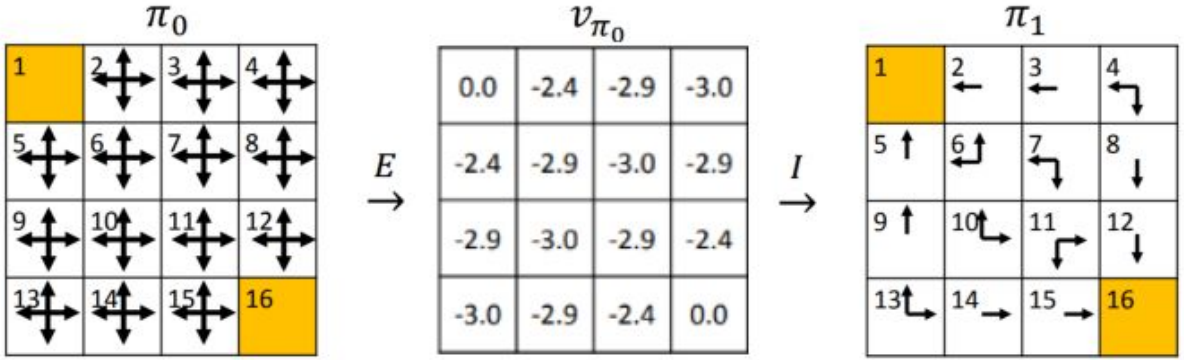


k=∞

Пример policy iteration

Могли получить тот же ответ, всего за 3 шага

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$



k=3

Policy iteration (PI) vs. Value iteration (VI)

- PI медленее, каждая итерация $O(|A||S|^2 + |S|^3)$
- PI требует мало итераций

- VI быстрее, каждая итерация $O(|A||S|^2)$
- VI требует много итераций

В общем случае оптимальнее использовать generalized value iteration, экспериментируя с количеством шагов для policy evaluation

А как решать более сложные задачи?

Хотим научиться играть в 8 bit игру, с разрешением 200x100 пикселей.

Можем оценить количество состояний как

$$|S| = 8^{20000}$$

Или, хотим научиться работать в среде, где действием агента является вещественное число

Расскажут на RL2

Список источников

- https://sites.ualberta.ca/~szepesva/rlbook.html
- http://incompleteideas.net/book/the-book-2nd.html
- https://www.wiley.com/en-us/Markov+Decision+Processes%3A+Discrete+Stochastic+Dynamic+Programming-p-9780471727828
- https://github.com/yandexdataschool/Practical_RL
- https://ai.plainenglish.io/understanding-policy-iteration-algorithm-for-reinforcement-learning-3bcb0930bcaf
- https://www.analyticsvidhya.com/blog/2018/09/reinforcement-learning-model-based-planning-dynamic-programming/