Лекция 13
Reinforcement learning pt.1

Галков Михаил, Храбров Кузьма

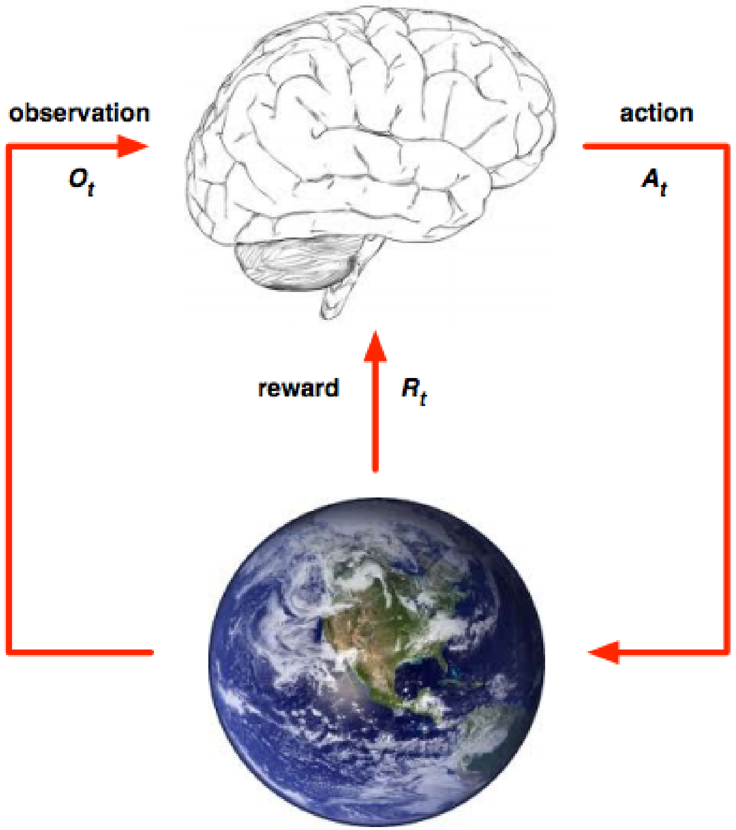
27 ноября 2017 г.

План лекции

Reinforcement learning

- Very brief introduction
- Markov process
- Markov reward process
- Markov decision process
- Policy iteration

Introduction



- Agent
  - Получает reward R_t
  - Получает наблюдение O_t
  - Совершает действие A_t
- Environment
  - Получает действие A_t
  - Генерирует состояние O_{t+1}
  - Генерирует reward R_{t+1}

Examples

- Portfolio management
- Chess (+1 win -1 loss)
- Robotics (+ за движение по траектории)
  - за отклонение

Какие еще задачи можно формализовать в этих терминах?

Main concepts: Rewards

- R_t - скаляр
- Задача агента максимизировать среднюю сумму полученных R_τ

Definition (Reward hypothesis)

Любая задача может быть сформулирована в виде максимизации суммы R_τ

Main concepts: State

В процессе взаимодействия со средой агент накапливает историю $H_t = R_1, O_1, A_1, ..., R_t, O_t, A_t$. Очевидно, что для принятия решения хранение всей истории крайне избыточно:

- Games: O_t - скриншот экрана (1200x700x3)
- Markets: оборот NYSE - 474m акций в день

Мы хотим иметь такое представление истории $S_t = f(H_t)$, которое было бы "достаточной статистикой"для будущего.

Markov property

Повторение: Мы хотим иметь такое представление истории $S_t = f(H_t)$, которое было бы "достаточной статистикой"для будущего.
Более формально: распределение состояний в будущем должно зависеть только от текущего и не зависеть от прошлых.

Definition (Markov property)

Пусть S_t - последовательность случайных величин (векторов, элементов). Последовательность обладает марковским свойством, если

$$Pr(S_{t+1}|S_t) = Pr(S_{t+1}|S_t, S_{t-1}, .., S_1) \tag{1}$$

Т.е. S_t достаточно для предсказания будущих состояний.

Transition matrix

Пусть S_t - последовательность дискретных состояний. (Пример будет очень скоро).
Поскольку последовательность задается распределением $Pr(S_{t+1}|S_t)$ естественно упорядочить его в матрицу.
$P_{ss'} = Pr(S_{t+1} = s'|S_t = s)$

$$\mathcal{P} = \begin{array}{cc} & \textit{to} \\ \textit{from} & \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \end{array}$$

Какие суммы вероятностей должны равняться единице?

Markov process

Definition

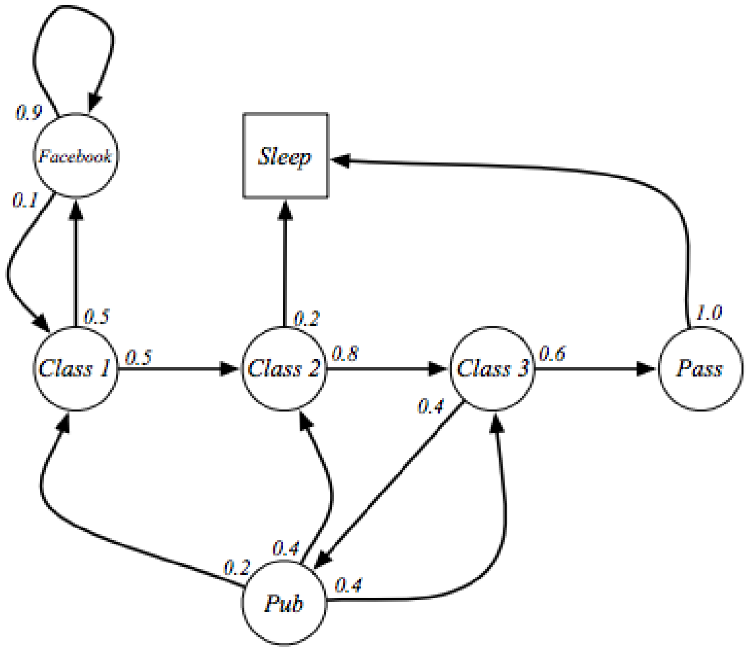
Марковский процесс (цепь) это кортеж (S, P), где

- S - принимает дискретные (конечные значения)
- P - матрица переходов (transition matrix)
  $P_{ss'} = Pr(S_{t+1} = s' | S_t = s)$

Строго говоря необходимо еще распределение начальных состояний (но мы предполагаем, что оно вырождено, т.е. мы знаем где начинаем с вероятностью 1).

Марковский процесс - основа для RL. Мы будем постепенно усложнять эту модель, добавляя rewards и actions.

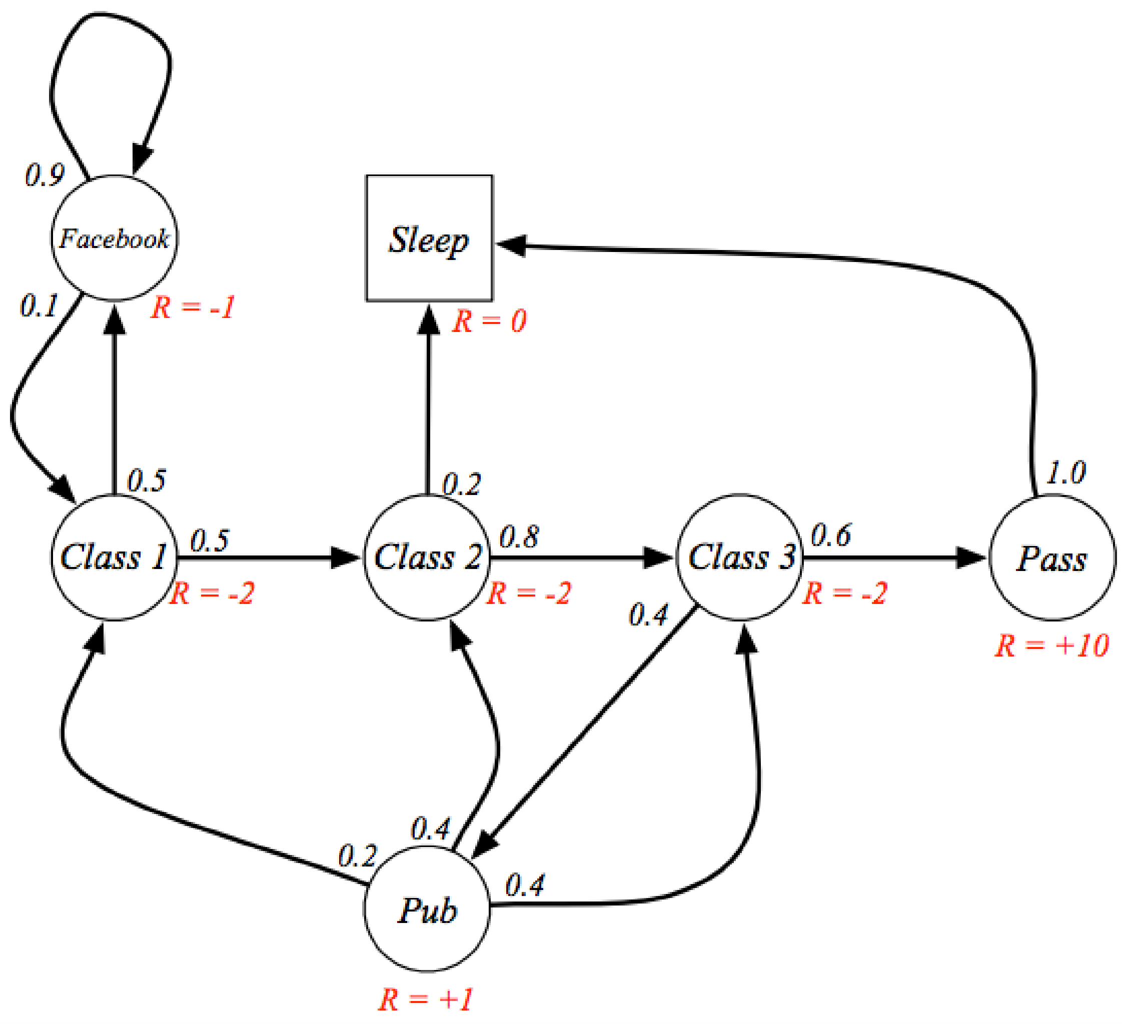
Example



$$
\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array}
\begin{array}{c}
\begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\
\left[\begin{array}{ccccccc}
 & 0.5 & & & & 0.5 & \\
 & & 0.8 & & & & 0.2 \\
 & & & 0.6 & 0.4 & & \\
 & & & & & & 1.0 \\
0.2 & 0.4 & 0.4 & & & & \\
0.1 & & & & & 0.9 & \\
 & & & & & & 1
\end{array}\right]
\end{array}
$$

Марковская цепь описывает блуждание по конечному (в нашем случае) пространству состояний, что не очень интересно. В нашем примере состояния не равноценны, добавим ценность нахождения в каждом при помощи rewards.

Example with rewards



Вспомним нашу основную цель: максимизировать получаемые rewards.

New concepts: return

Definition (Return)

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{i=0}^{\infty} \gamma^i R_{t+1+i} \tag{2}$$

Под G_t мы понимаем дисконтированную сумму всех будущих rewards.

- $\gamma \in [0, 1]$ - discount factor
- Единая форма для "конечных"и "бесконечных"моделей
- Обеспечивает сходимость ряда ($max|R_t| < C = const$)
- Нетерпеливость (impatience) - насколько важно получить reward сейчас, чем потом.

По определению G_t - случайная величина (не привязанная ни к чему). Покажем, как ее можно использовать.

New concepts: value

Definition (value of the state)

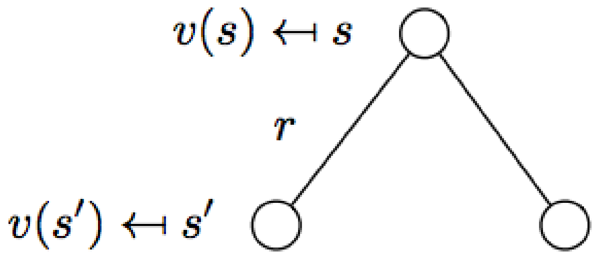
$$v(s) = \mathbb{E}[G_t | S_t = s] \tag{3}$$

Ценность состояния - ожидаемая сумма всех полученных rewards, если стартовать из s.

$$\begin{aligned} v(s) &= \mathbb{E}[G_t | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + ...) | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \end{aligned} \tag{4}$$

Bellman equation

$$v(s) = \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]$$

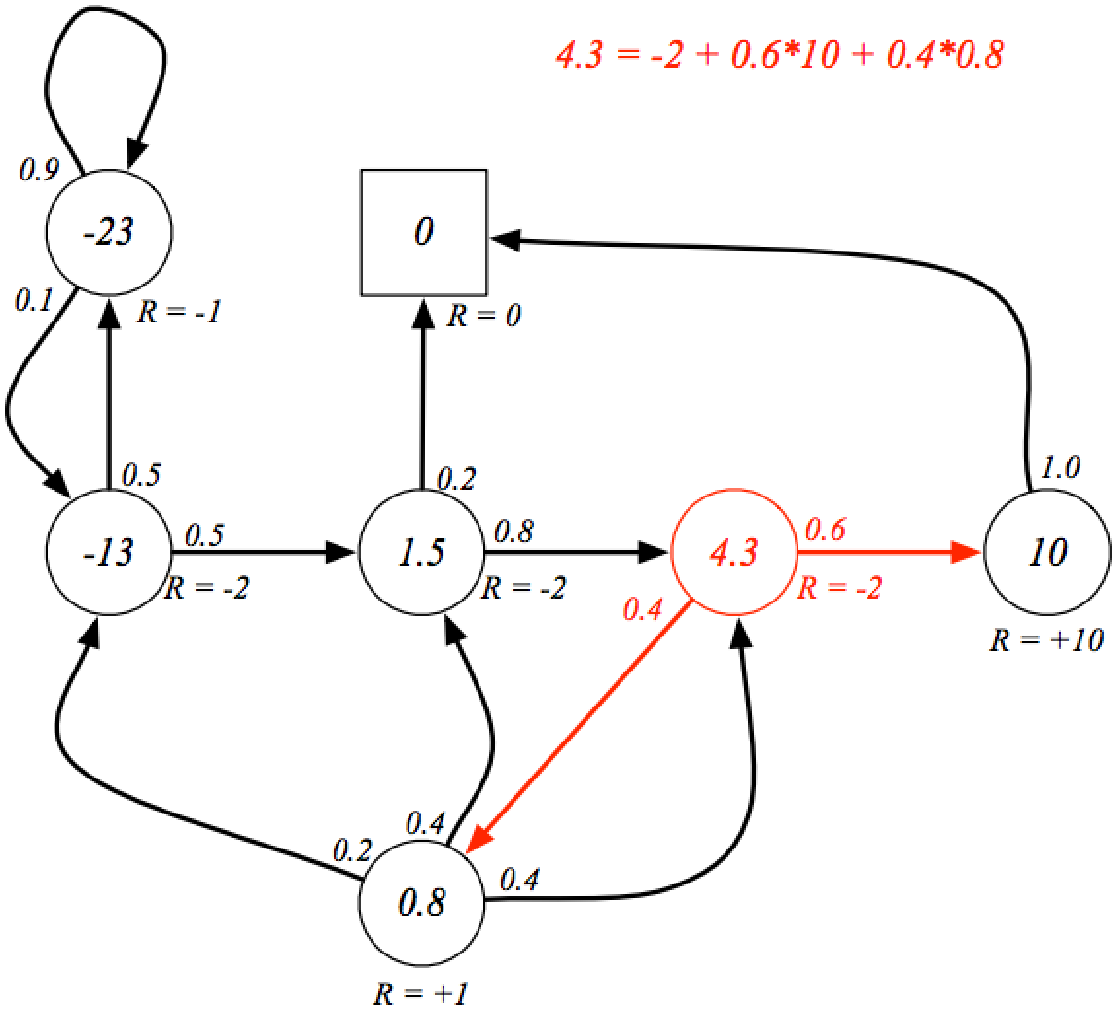


$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

Эти уравнения должны напоминать backprop (как минимум идейно). Из них очень легко найти value для любого состояния

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

Example



$\gamma = 1$

Markov reward process

Definition

MRP это кортеж (S, R, P, γ), где

- S - принимает дискретные (конечные значения)
- R - функция rewards, $R_s = \mathbb{E}[R_{t+1}|S_t = s]$
- P - матрица переходов (transition matrix)
  $P_{ss'} = Pr(S_{t+1} = s'|S_t = s)$
- γ - discount factor

Осталось добавить actions.

Markov decision process

Definition

MRP это кортеж (S, A, R, P, γ), где

- S - состояния (дискретное пространтсво)
- A - действия (дискретное пространтсво)
- R - функция rewards, $R_s^a = \mathbb{E}[R_{t+1}|S_t = s, A_t = a]$
- P - матрица переходов (transition matrix) $P_{ss'}^a = Pr(S_{t+1} = s'|S_t = s, A_t = a)$
- γ - discount factor

Поговорим о том, как меняются наши уравнения.

New concepts: policy q-function

Definition (Policy)

$\pi(a|s) = Pr(A_t = a|S_t = s)$ - стратегия, т.е. то как мы выбираем действия оказавшись в состоянии s.
NB: мы фиксируем нашу стратегию в начале каждой игры!

Definition (Value function)

$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$ - ценность состояния определяется еще и стратегией.

Definition (Q-function)

$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$ - ценность действия в состоянии s.

Почему ничего радикально не изменилось

Самое важное уравнение для понимания:

$$
\begin{aligned}
P_{ss'} &= Pr(S_{t+1} = s' | S_t = s) \\
&= \sum_a Pr(S_{t+1} = s', A_t = a | S_t = s) \\
&= \sum_a Pr(A_t = a | S_t = s) Pr(S_{t+1} = s' | S_t = s, A_t = a) \\
&= \sum_a \pi(a|s) P^a_{ss'}
\end{aligned} \tag{5}
$$

Bellman equations

Bellman equation для value-function

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s \right]$$

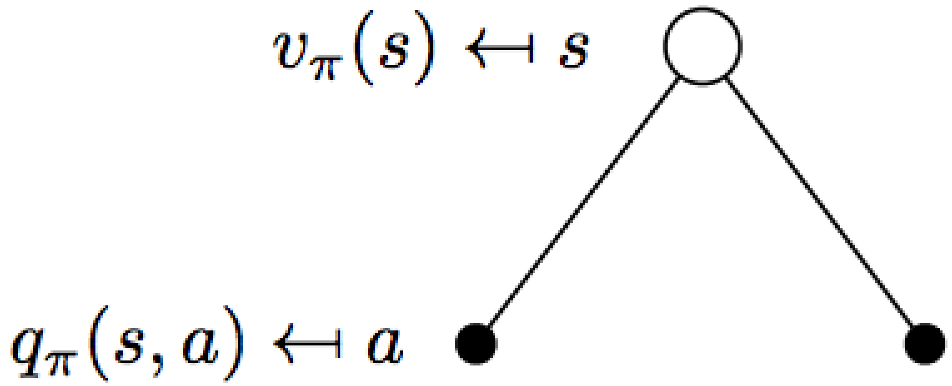
Bellman equation для q-function

$$q_\pi(s, a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right]$$

Задание: выведите их самостоятельно

Bellman equations visual 1

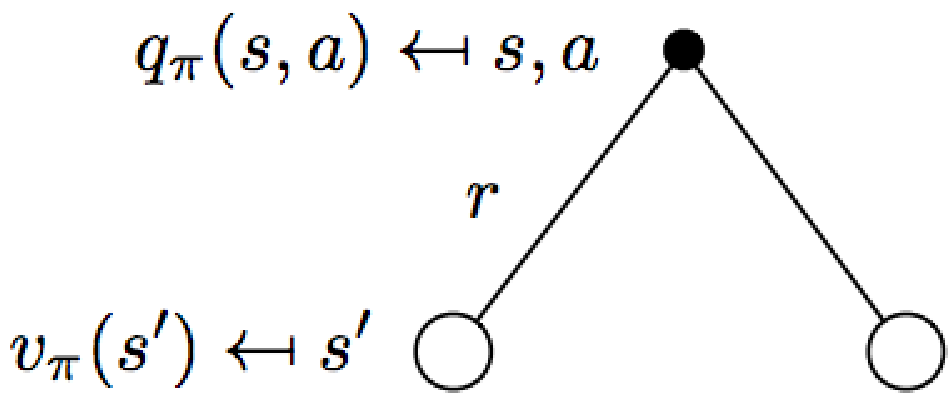
value function очевидно зависит от q-function (смотри самое важное уравнение)



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

Bellman equations visual 2

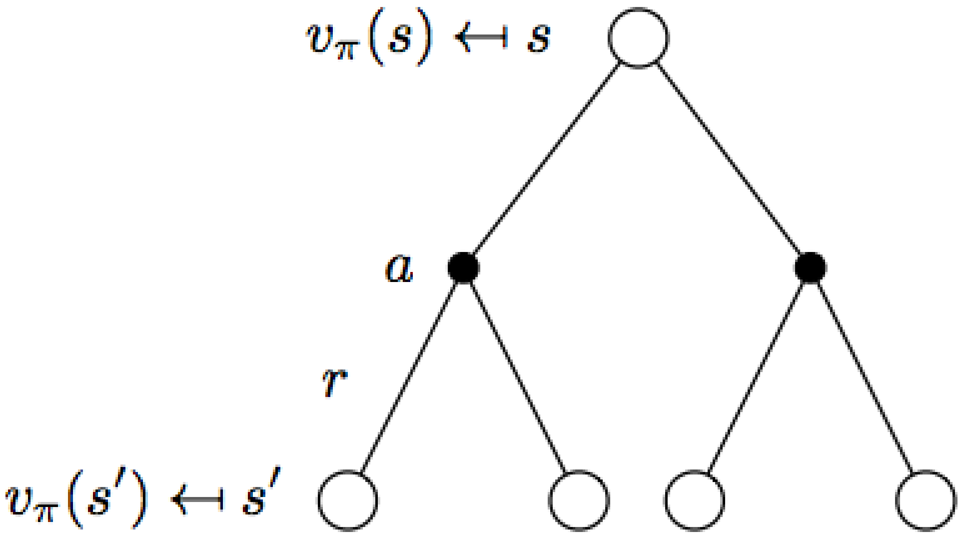
q-function в свою очередь зависит от value-function следующего состояния



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

Bellman equations visual 3

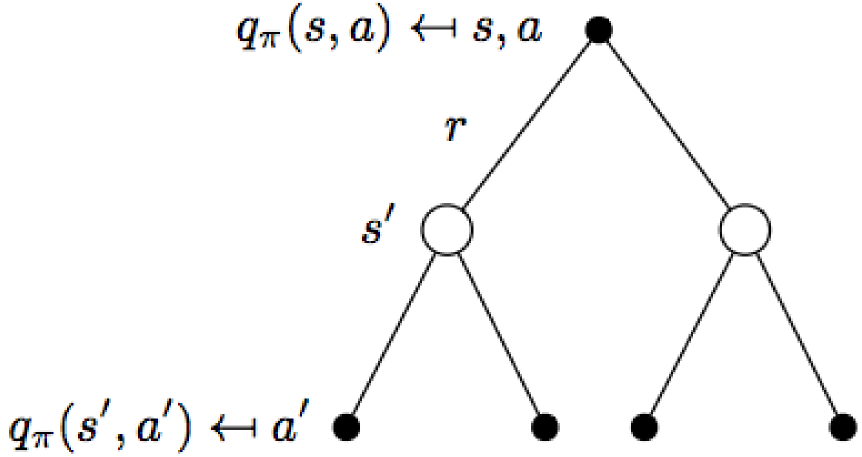
Соберем две предыдущие картинки вместе



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

Bellman equations visual 4

Точно так же можно вывести зависимость q-function



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

Выведите самостоятельно. Перейдем к самому интересному: как находить решения.

Moar Bellman equations

Зафиксируем некоторую policy.

Возьмем уравнение Беллмана в случае MRP

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

Возьмем уравнение Беллмана в случае MDP

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

И сделаем замену:

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

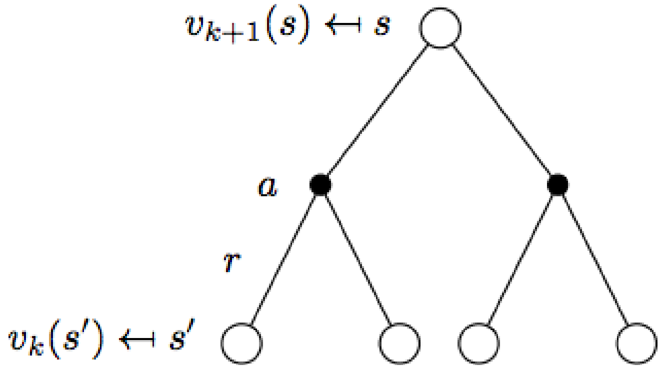
$$\mathcal{R}_s^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}_s^a$$

Value iteration

Получим следующее уравнение

$$\mathbf{v}_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi \mathbf{v}_\pi$$

И будем решать методом итераций:



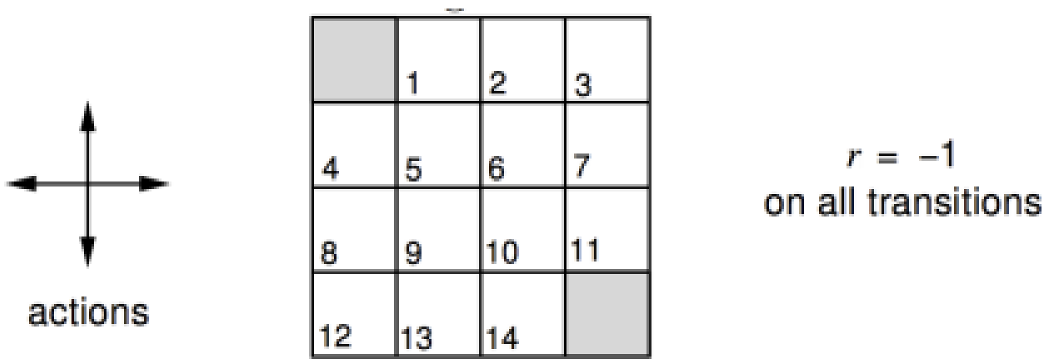
$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s') \right)$$

$$\mathbf{v}^{k+1} = \boldsymbol{\mathcal{R}}^{\boldsymbol{\pi}} + \gamma \boldsymbol{\mathcal{P}}^{\boldsymbol{\pi}} \mathbf{v}^k$$

Results

Легко доказывается, что это процесс сходится и в результате для каждого состояния мы получаем его $v(s)$
Теперь зададимся вопросом, а как улучшить нашу policy?

Game



Iteration 1

v_k for the Random Policy

$k = 0$

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

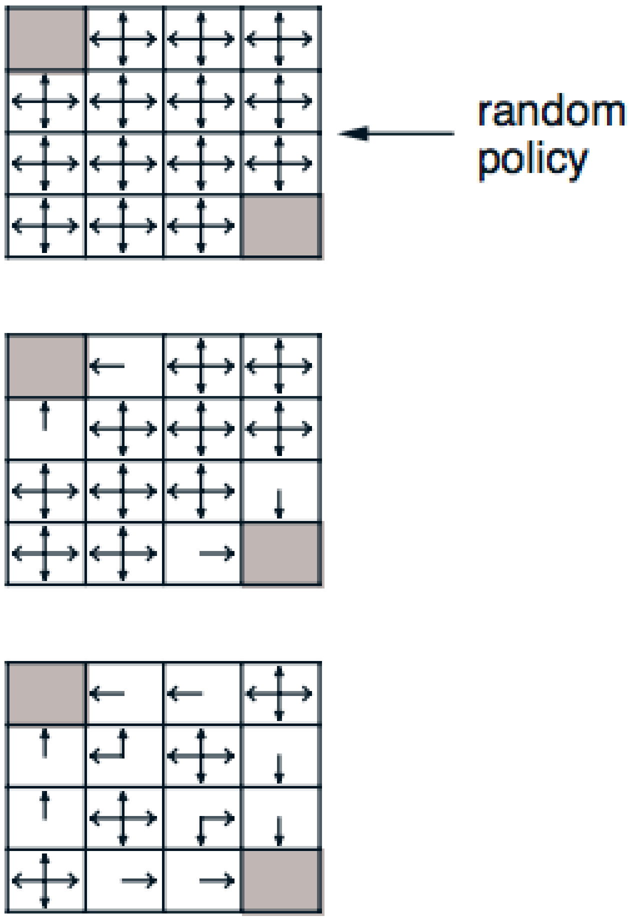
$k = 1$

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

Greedy Policy w.r.t. v_k



Iteration 2

$k = 3$

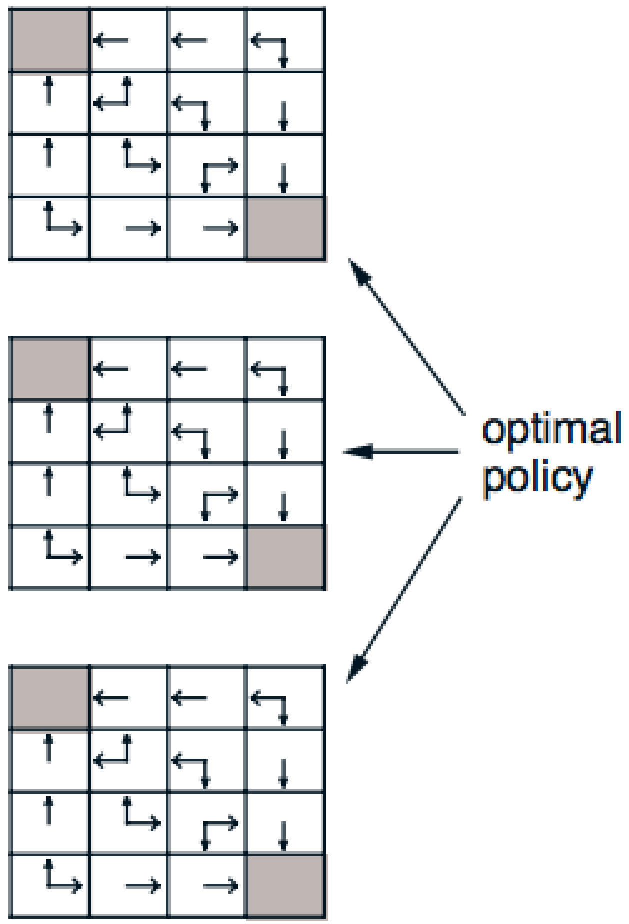
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

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

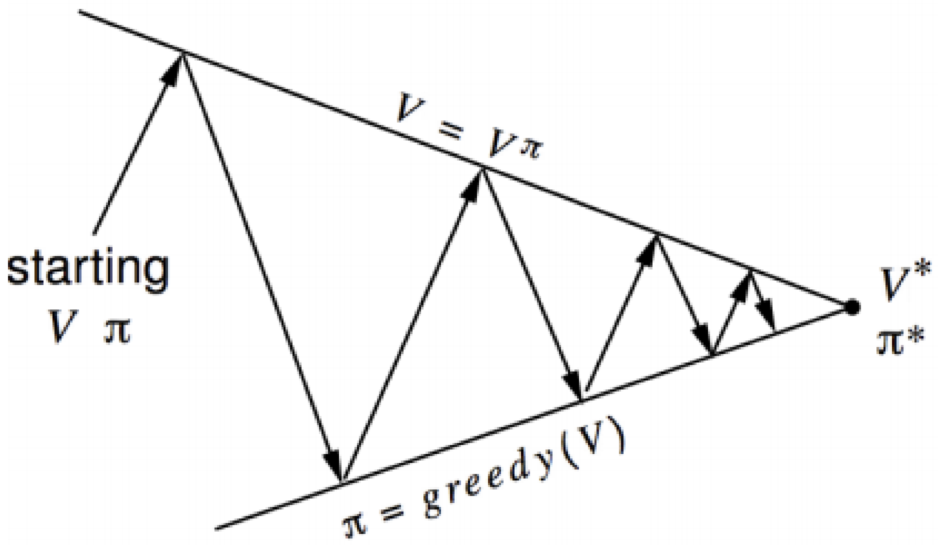
$k = \infty$

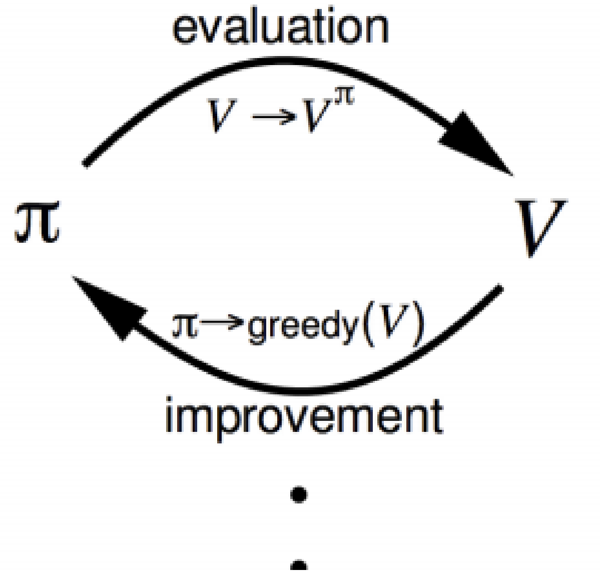
0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



Algorithm

Очевидно, что для того, чтобы улучшить policy надо выбирать действия, которые приводят в более выгодные состояния. Полученный алгоритм называется policy iteration.





Следующая лекция: более подробно о том, что значит "решить"MDP, много других интересных алгоритмов решения.

Вопросы