

# Напоминалка: обучение с учителем

## Supervised learning

$x_i$  - изображение



$y_i$  - метка класса



СОБАКА



КОШКА

# Напоминалка: обучение с учителем

## Supervised learning

$x_i$  - изображение



$y_i$  - метка класса



СОБАКА



КОШКА

### Стандартное предположение:

- Правильные ответы  $y_i$  известны

Учитель

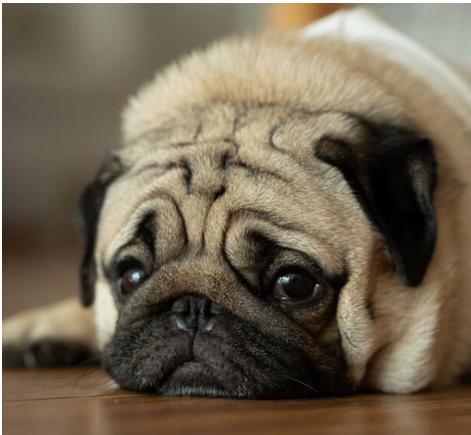
# Напоминалка: обучение с учителем

## Supervised learning

- Имеется выборка:

$$D := \{(x_i, y_i)\}$$

$x_i$  - изображение



$y_i$  - метка класса



СОБАКА

### Стандартное предположение:

- Правильные ответы  $y_i$  известны



КОШКА

Учитель

# Напоминалка: обучение с учителем

## Supervised learning

- Имеется выборка:

$$D := \{(x_i, y_i)\}$$

- Нужно выучить отображение:

$$\hat{y}_i = f(x_i)$$

$x_i$  - изображение



$y_i$  - метка класса



СОБАКА

### Стандартное предположение:

- Правильные ответы  $y_i$  известны



КОШКА

Учитель

# Напоминалка: обучение с учителем

## Supervised learning

- Имеется выборка:

$$D := \{(x_i, y_i)\}$$

- Нужно выучить отображение:

$$\hat{y}_i = f(x_i)$$

- Такое, что:

$$\hat{y}_i \approx y_i$$

### Стандартное предположение:

- Правильные ответы  $y_i$  известны

$x_i$  - изображение



$y_i$  - метка класса



СОБАКА



КОШКА

Учитель

# Процесс принятия решений

Decision Process (a self-driving car)



# Процесс принятия решений

Decision Process

Наблюдения:

- Изображения с видеокамеры
- Данные с датчиков

$s_i$  - изображение



# Процесс принятия решений

Decision Process

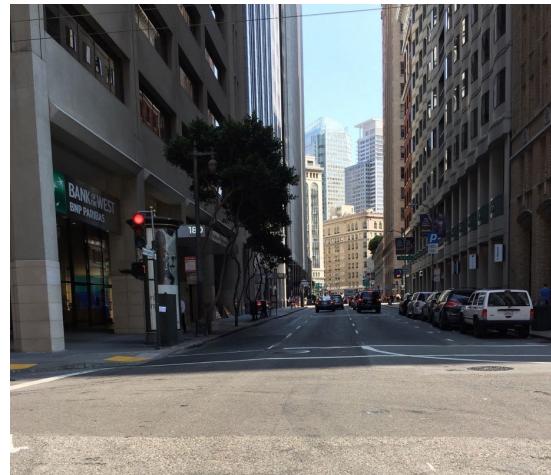
Наблюдения:

- Изображения с видеокамеры
- Данные с датчиков

Действия:

- Газ \ тормоз
- Поворот руля

$s_i$  - изображение



$a_i$  - действие

ДЕЙСТВИЕ 1



ДЕЙСТВИЕ 2

# Процесс принятия решений

Decision Process

Наблюдения:

- Изображения с видеокамеры
- Данные с датчиков

Действия:

- Газ \ тормоз
- Поворот руля

Цель:

- ехать по маршруту

$s_i$  - изображение



$a_i$  - действие

ДЕЙСТВИЕ 1



ДЕЙСТВИЕ 2

# Процесс принятия решений

Decision Process

Наблюдения:

- Изображения с видеокамеры
- Данные с датчиков

Действия:

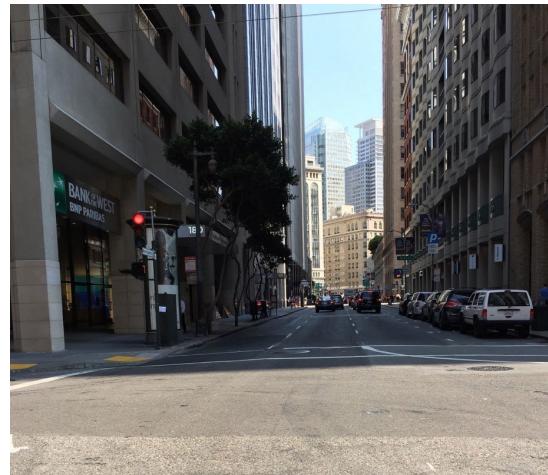
- Газ \ тормоз
- Поворот руля

Цель:

- ехать по маршруту

? кто разметит выборку ?

$s_i$  - изображение



$a_i$  - действие

??

→ ДЕЙСТВИЕ 1



??

→ ДЕЙСТВИЕ 2

# Процесс принятия решений

Decision Process

Наблюдения:

- Изображения с видеокамеры
- Данные с датчиков

Действия:

- Газ \ тормоз
- Поворот руля

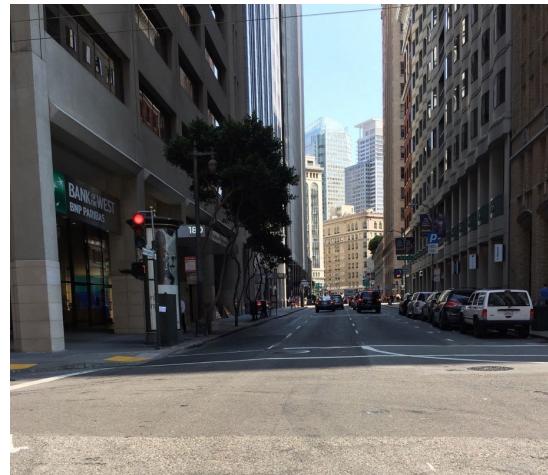
Цель:

- ехать по маршруту

? кто разметит выборку ?

!! действия влияют на наблюдения !!

$s_i$  - изображение



$a_i$  - действие

??

ДЕЙСТВИЕ 1



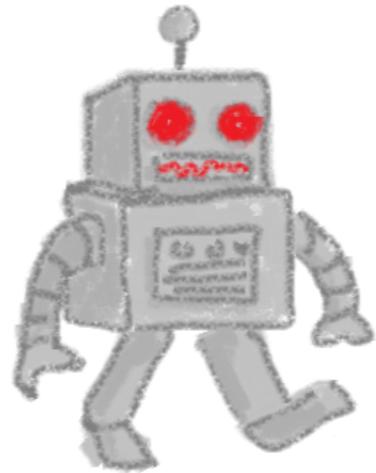
??

ДЕЙСТВИЕ 2

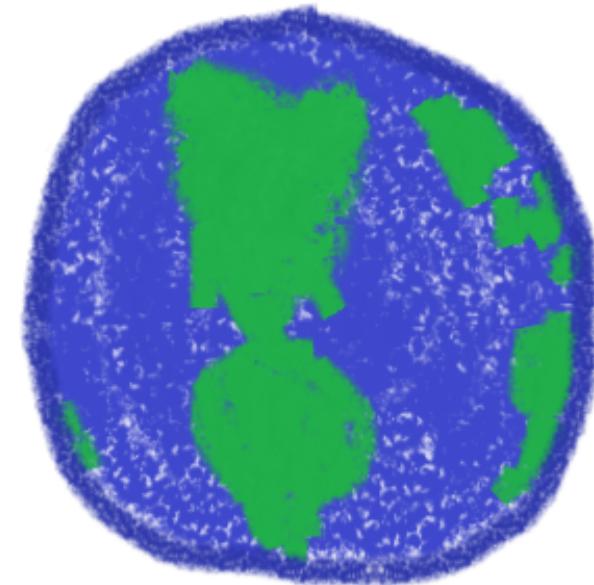
# Процесс принятия решений

Decision Process

агент



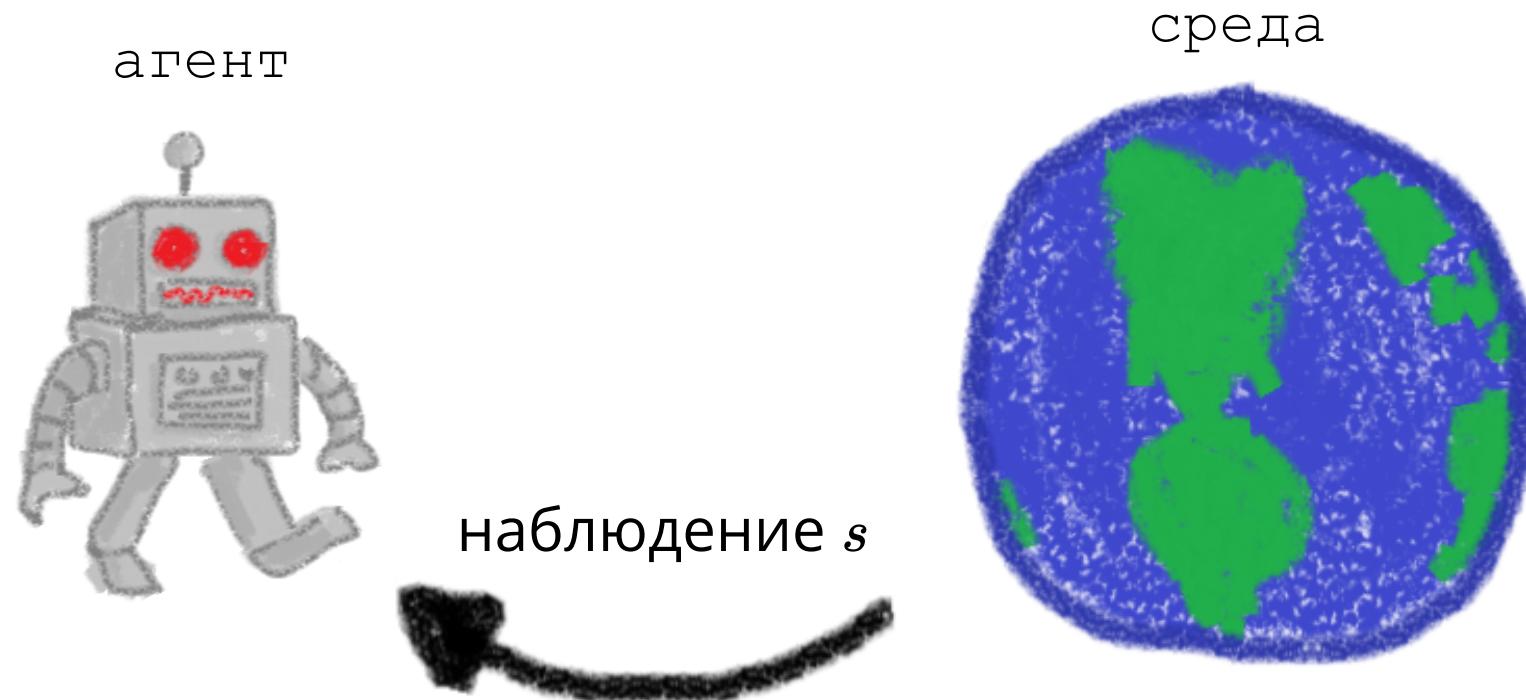
среда



Decision Process - выбор действий по наблюдениям

# Процесс принятия решений

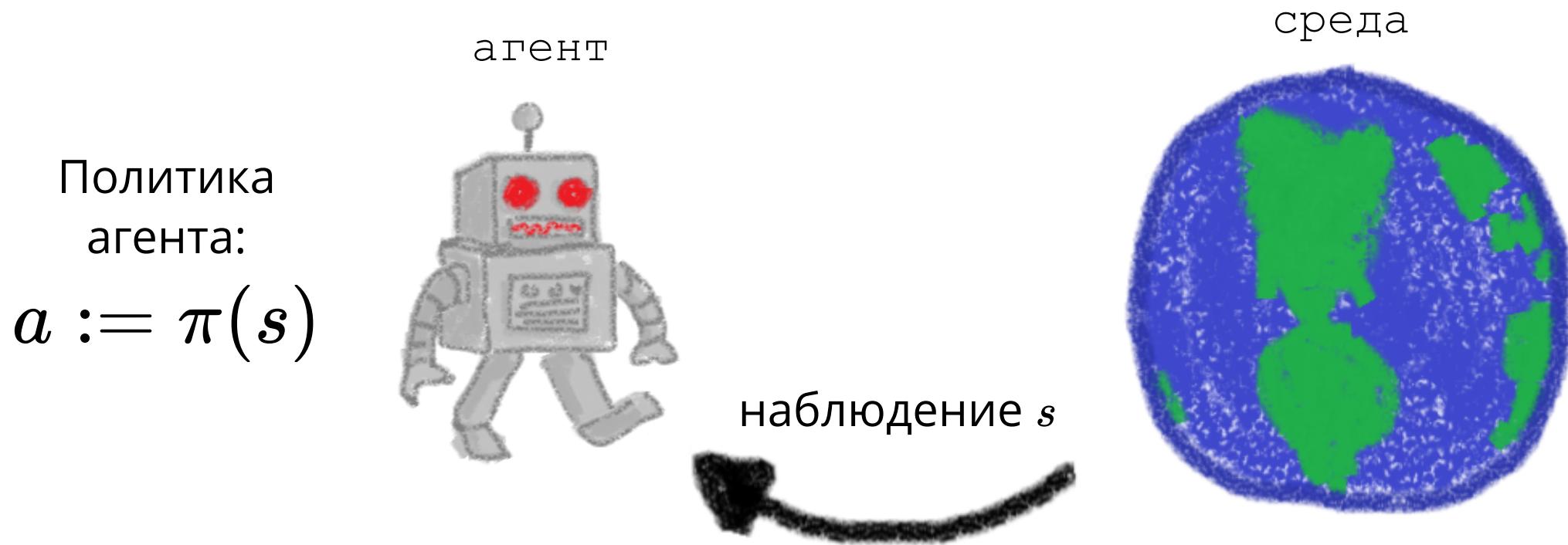
Decision Process



Decision Process - выбор действий по наблюдениям

# Процесс принятия решений

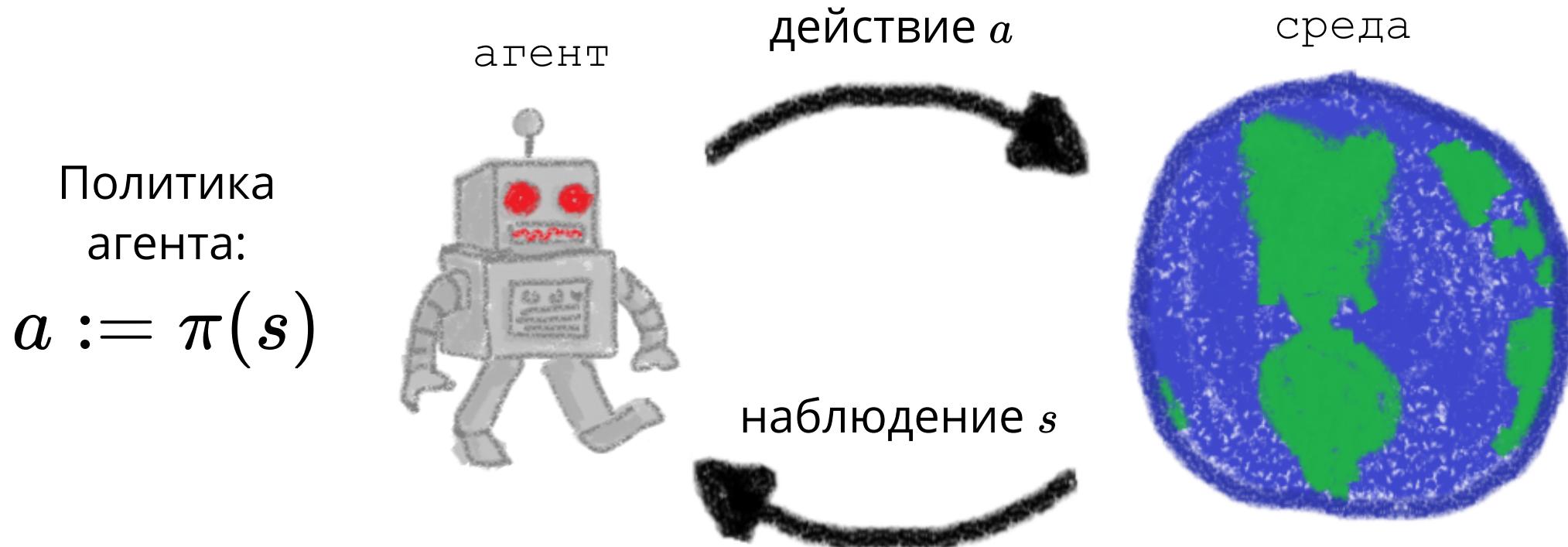
Decision Process



Decision Process - выбор действий по наблюдениям

# Процесс принятия решений

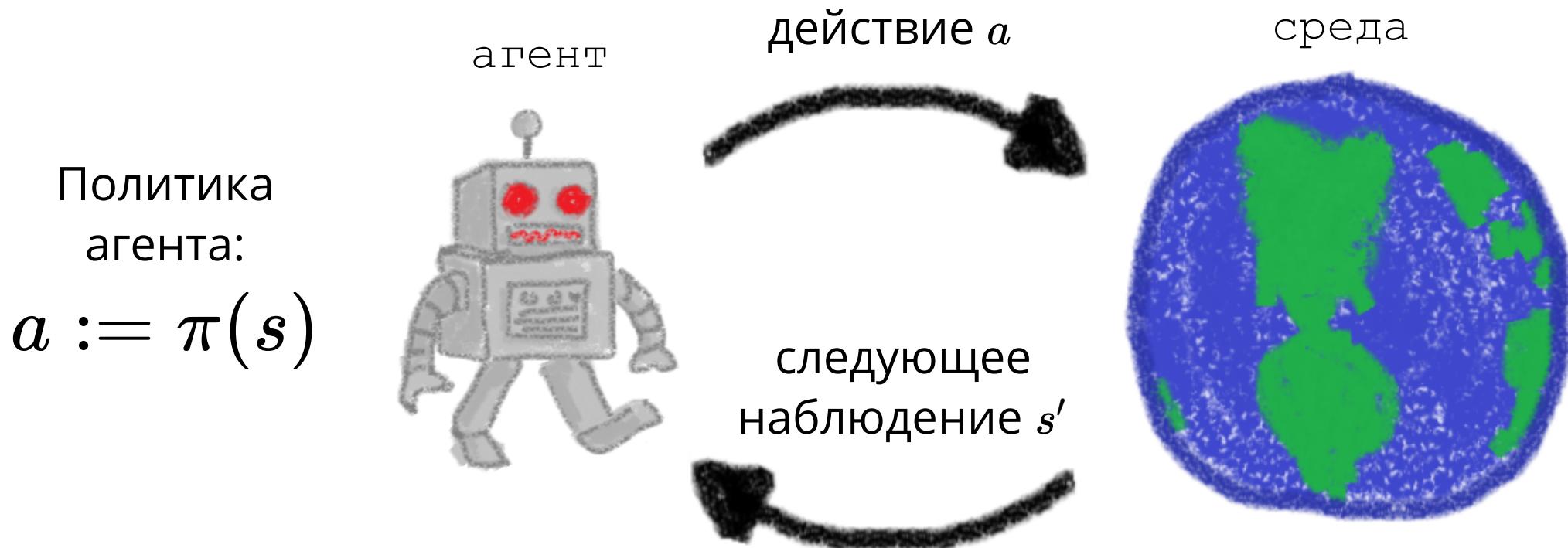
Decision Process



Decision Process - выбор действий по наблюдениям

# Процесс принятия решений

Decision Process



Decision Process - выбор действий по наблюдениям

# Behavior Cloning

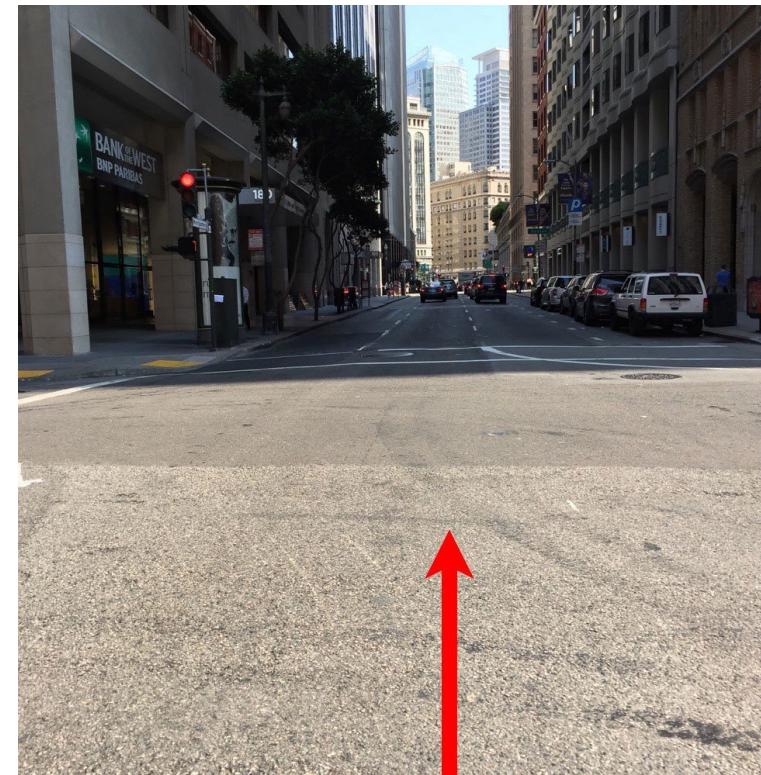
Что, если попросить эксперта сказать, какие действия хорошие?

А затем применить обучение с учителем.

**На примере self-driving cars (беспилотный автомобиль):**

1. Берем хорошего водителя
2. Отправляем ездить по маршруту
3. Записываем видео его траекторий ( $s_i$ )
4. Сохраняем историю его действий ( $a_i$ )
5. Обучаем ML модель  $\pi$ :

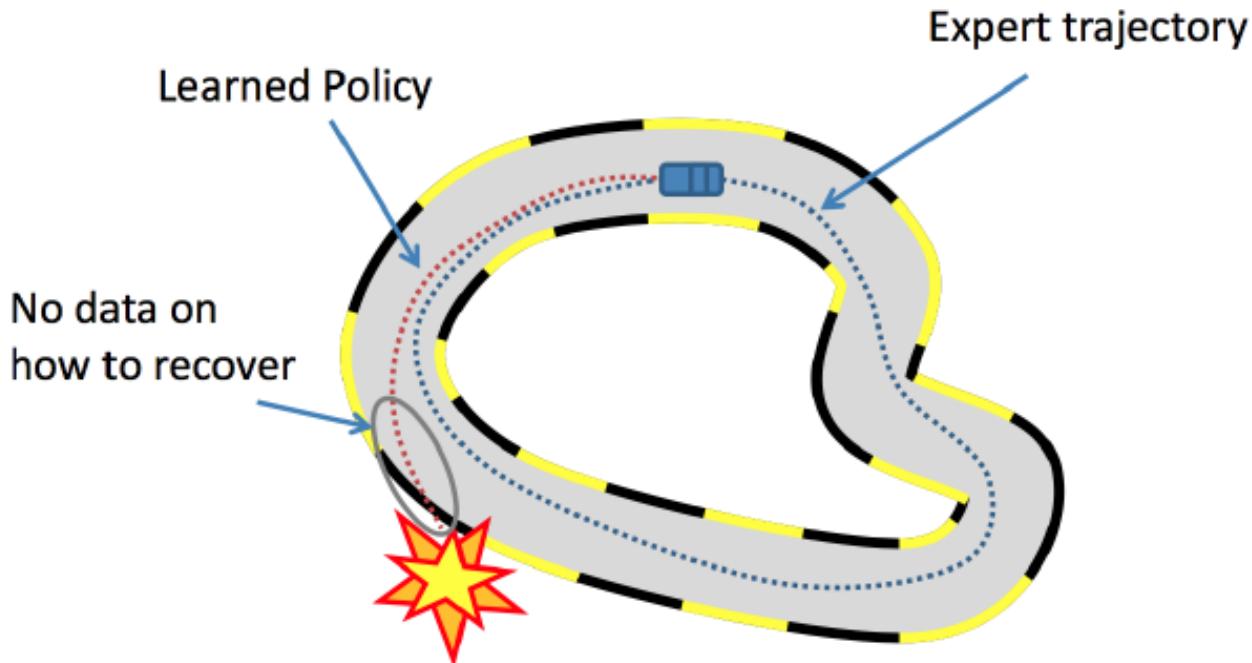
$$a_i \approx \pi(s_i)$$



# Behavior Cloning

Неизбежно, наш агент заедет туда, где эксперт никогда не был.

Агент не знает, как себя там вести.



Проблема **Distributional Shift**:

Наши наблюдения меняются с изменением стратегии!

# DAGGER

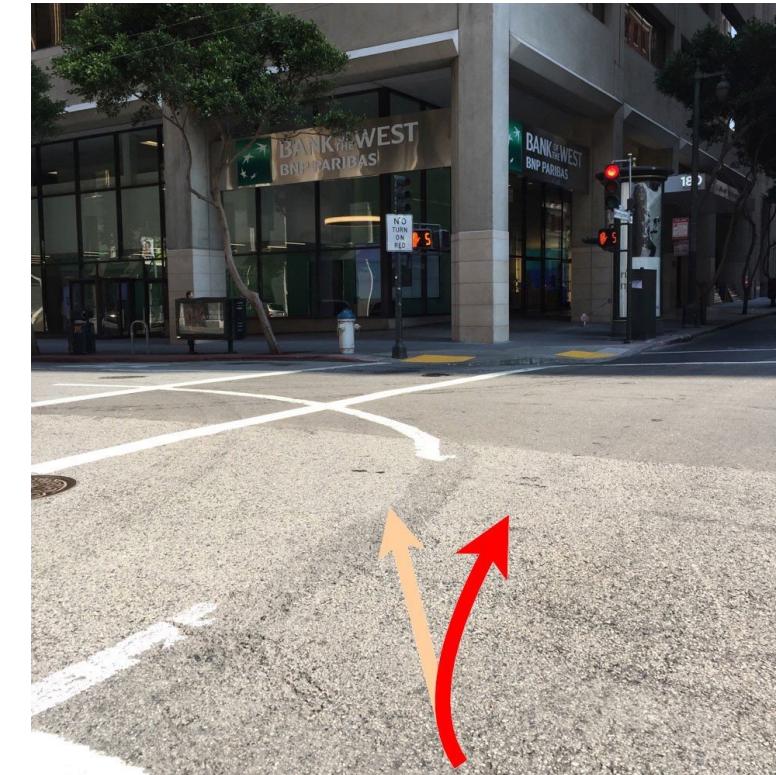
Можно просить эксперта возвращать агента на путь истинный.

**На примере self-driving cars (беспилотный автомобиль):**

- Отправляем водителя ездить по маршруту
- Записываем его траектории  $s_i$  и действия  $a_i$
- Обучаем ML модель  $\pi : a_i \approx \pi(s_i)$

Делаем, пока не удовлетворены:

- Пускаем агента в город, собираем траектории  $s_i$
- Просим эксперта дать правильные действия  $a_i$  для собранных траекторий  $s_i$



# DAGGER

## Плюсы:

- Очень прост
- Иногда хорошо работает

## Минусы:

- Эксперт нужен в режиме онлайн
- Агент не будет лучше эксперта
- Не всегда эксперт вообще знает, что делать!



Действия: напряжение, подаваемое на моторчики в суставах

# DAGGER

<https://www.youtube.com/embed/YuyT2SDcYrU?t=137&enablejsapi=1>

# Примеры Decision Process

Робототехника

Наблюдения:

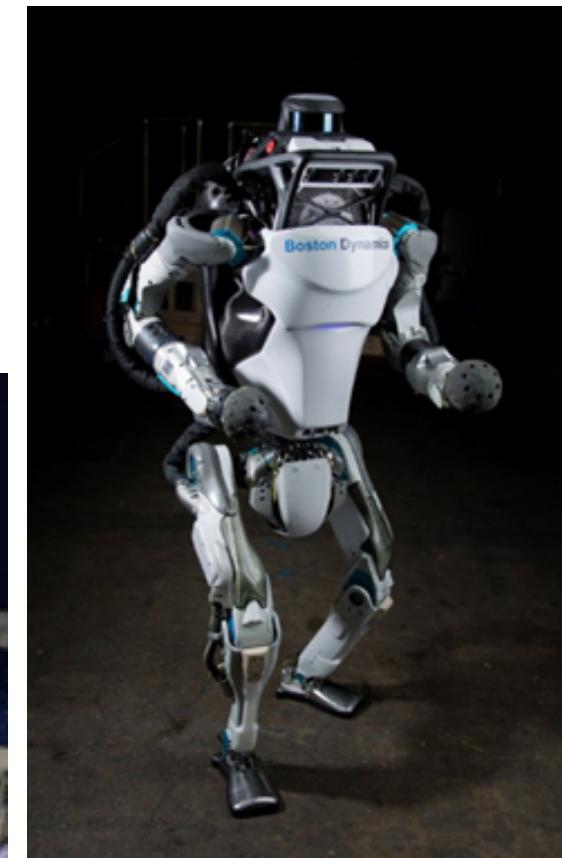
- Изображения с видеокамеры
- Данные с датчиков

Действия:

- Усилие подаваемое на сочленения робота

Цели:

- Движение вперед
- Решение составных задач (перенос предметов)
- ...



# Примеры Decision Process

Шахматы (или другие настольные игры)

Наблюдения:

- Расстановка фигур на доске

Действия:

- Выбор фигуры и хода ей

Цели:

- Победа
- Или, хотя бы, ничья



# Награда за достижение целей

Вместо учителя



# Награда за достижение целей

Вместо учителя



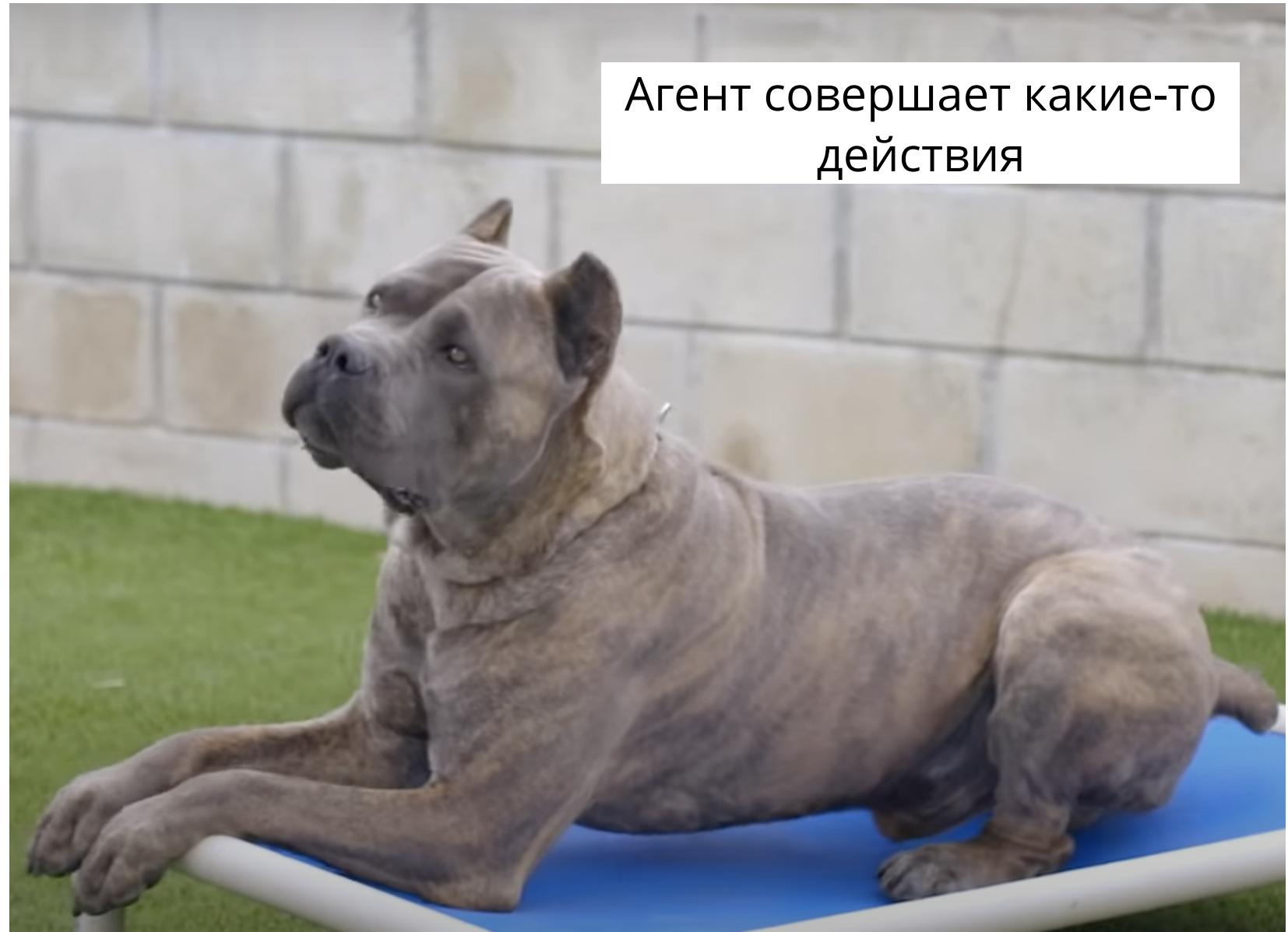
# Награда за достижение целей

Вместо учителя



# Награда за достижение целей

Вместо учителя



Агент совершает какие-то  
действия

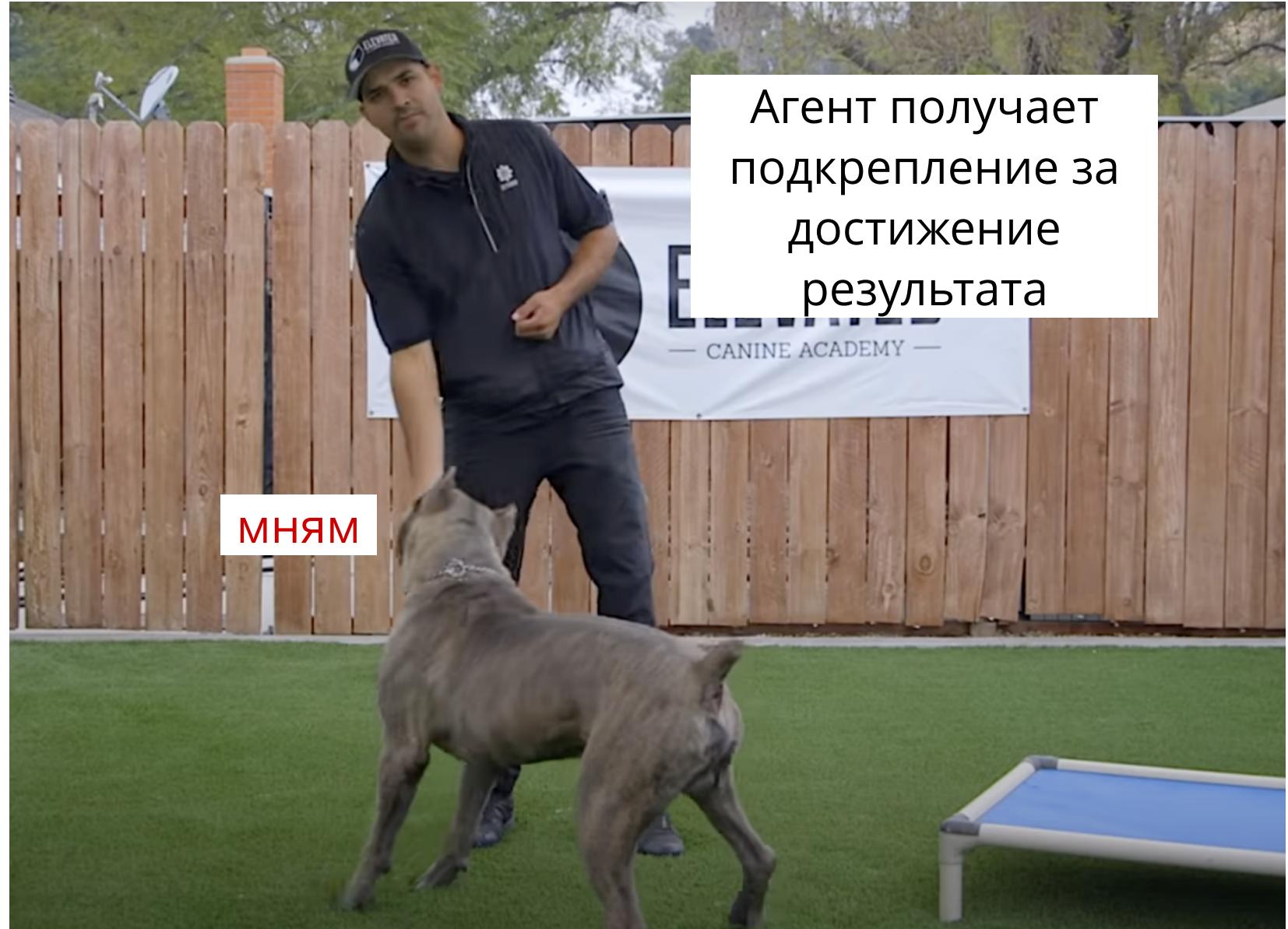
# Награда за достижение целей

Вместо учителя



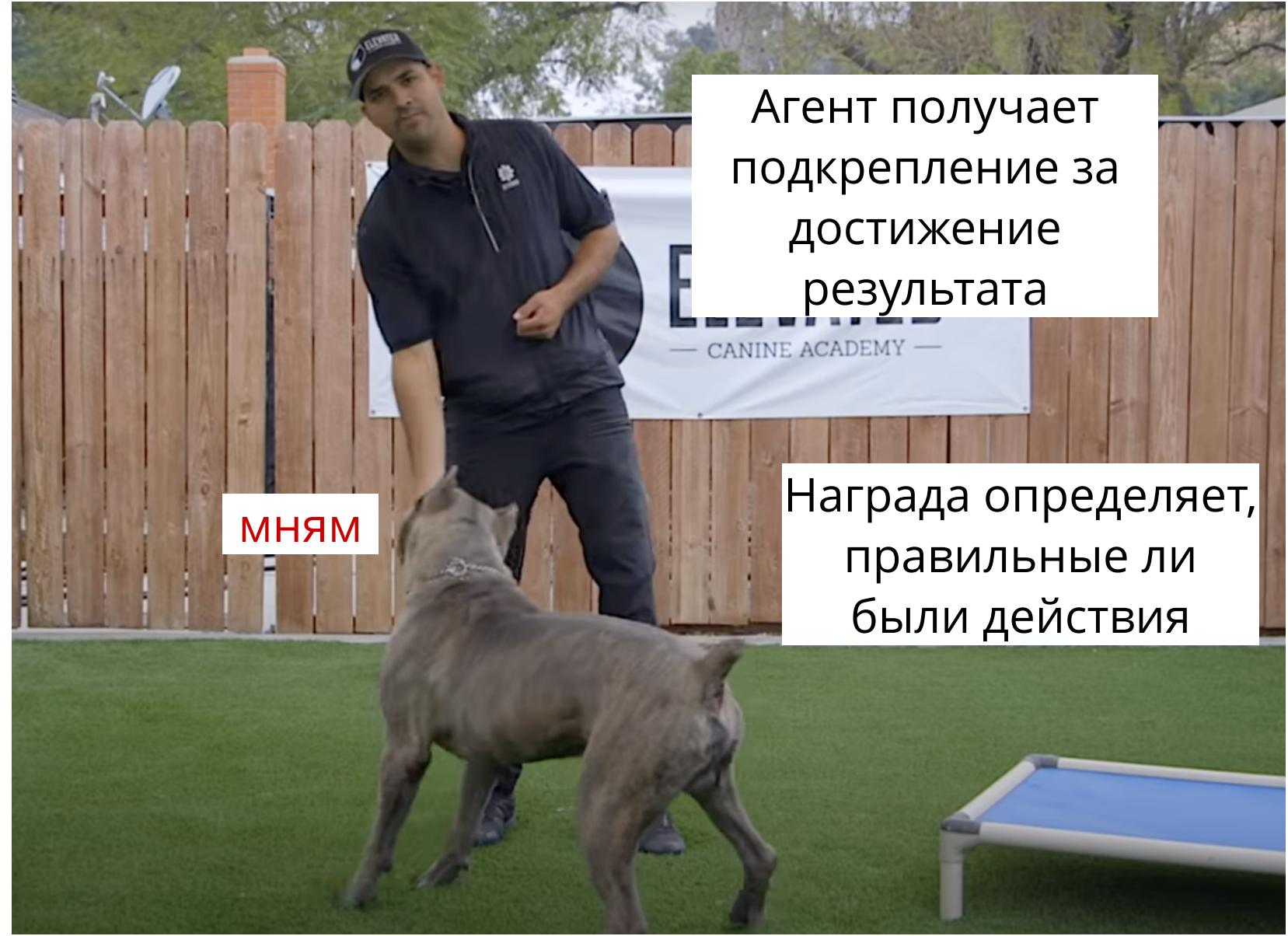
# Награда за достижение целей

Вместо учителя



# Награда за достижение целей

Вместо учителя



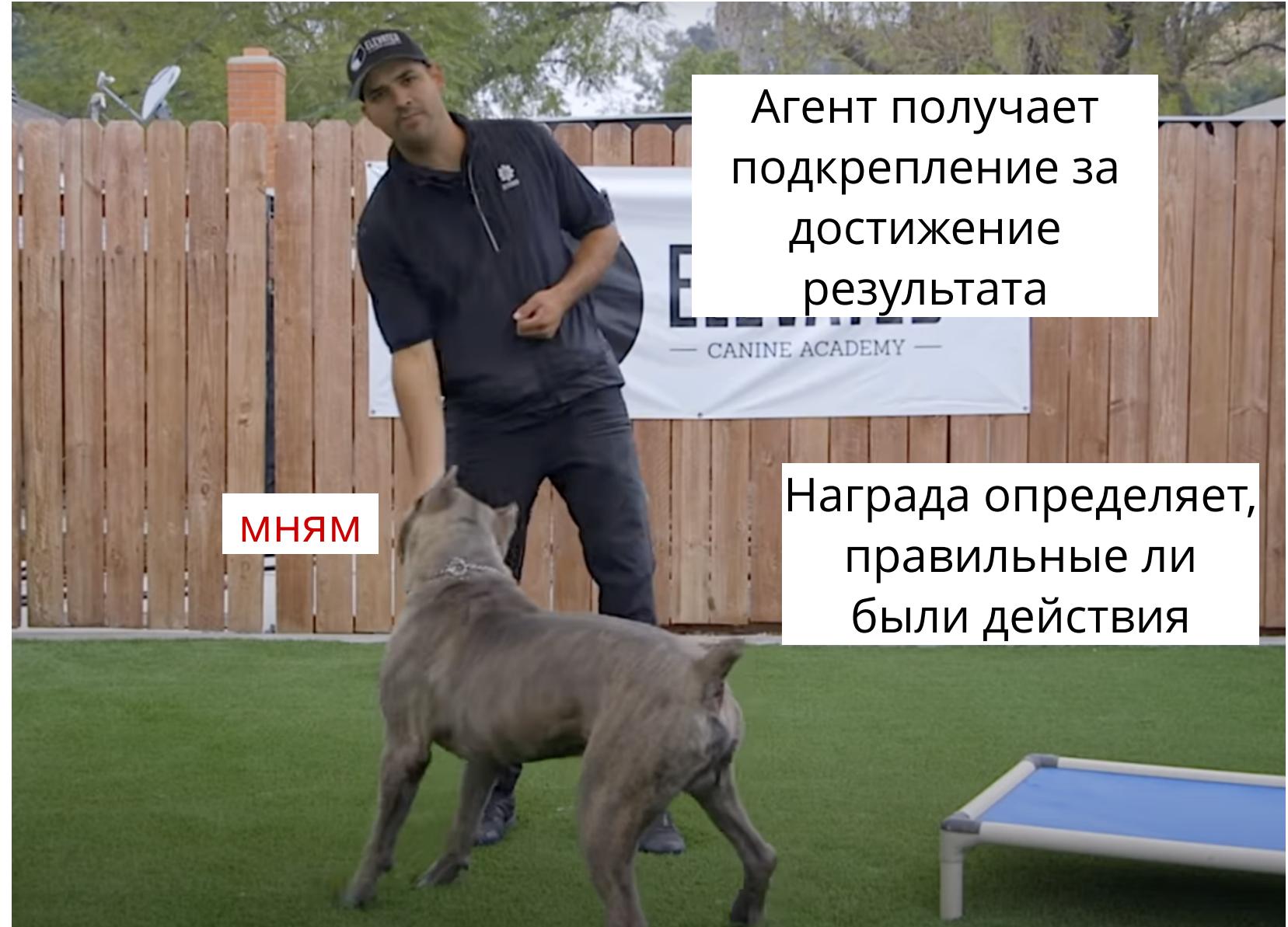
# Награда за достижение целей

Вместо учителя

Это и есть

Обучение с  
подкреплением

**Reinforcement  
Learning**



# Обучение с подкреплением

Если вы знаете, **чего** вы хотите, но не знаете, **как** этого достичь...

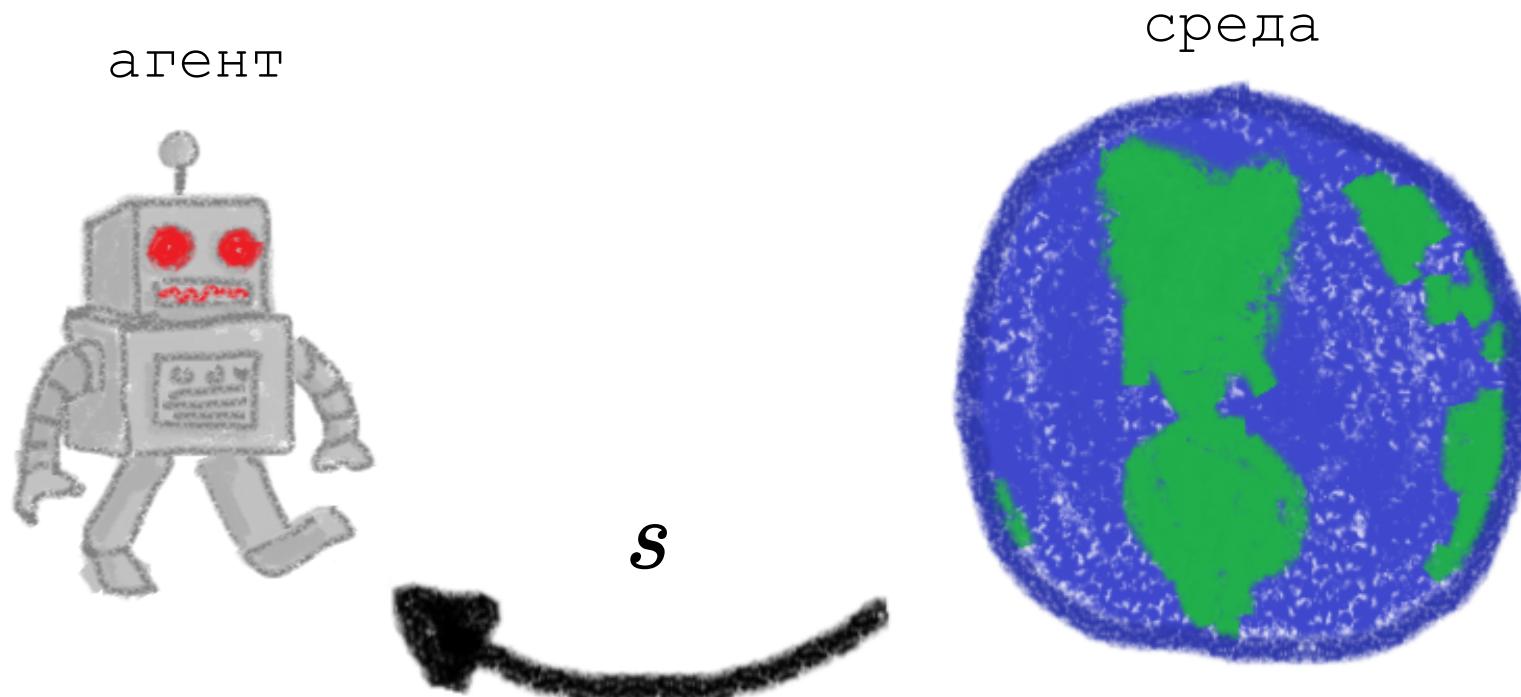
Используйте **НАГРАДЫ** (rewards)

- Победа: +1
- Проигрыш: -1
- Ничья: 0



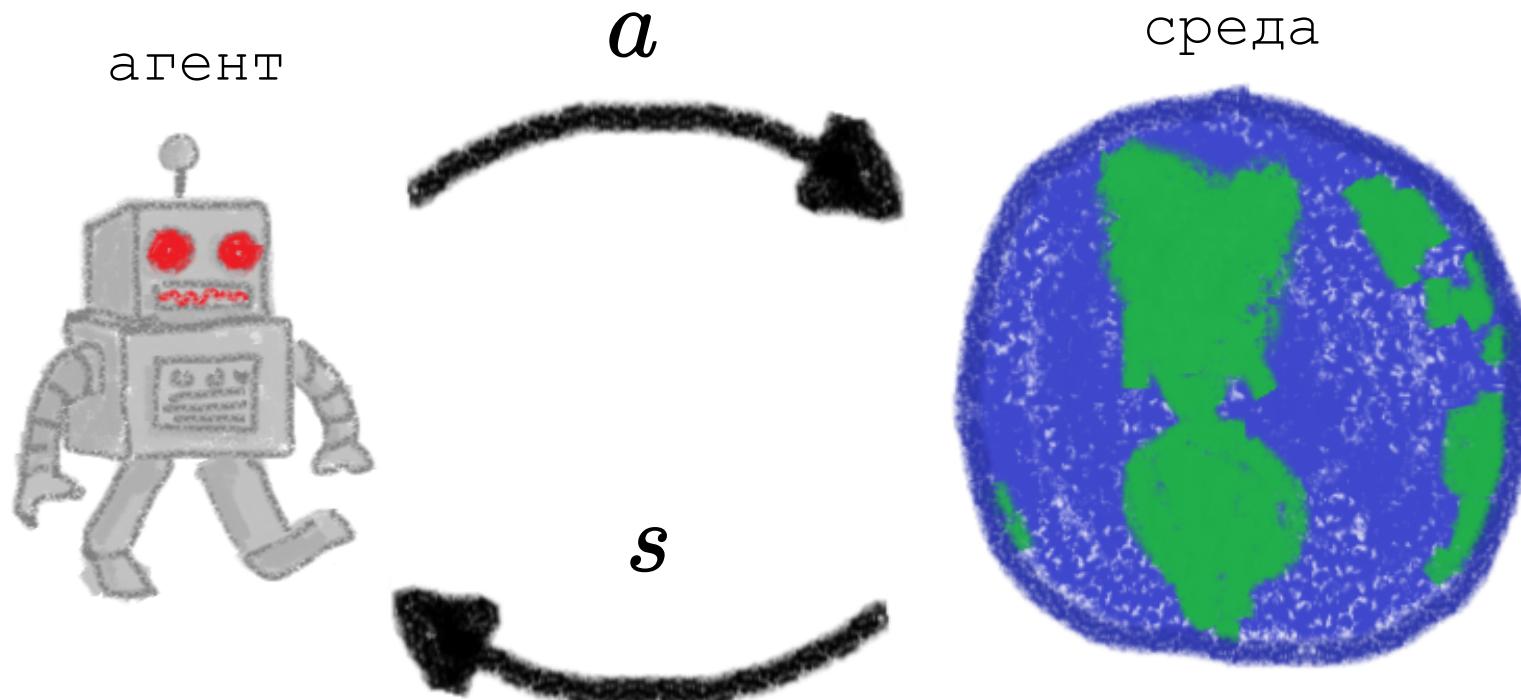
# Задача Reinforcement Learning

- Агент взаимодействует со средой - наблюдает ее состояние  $s$



# Задача Reinforcement Learning

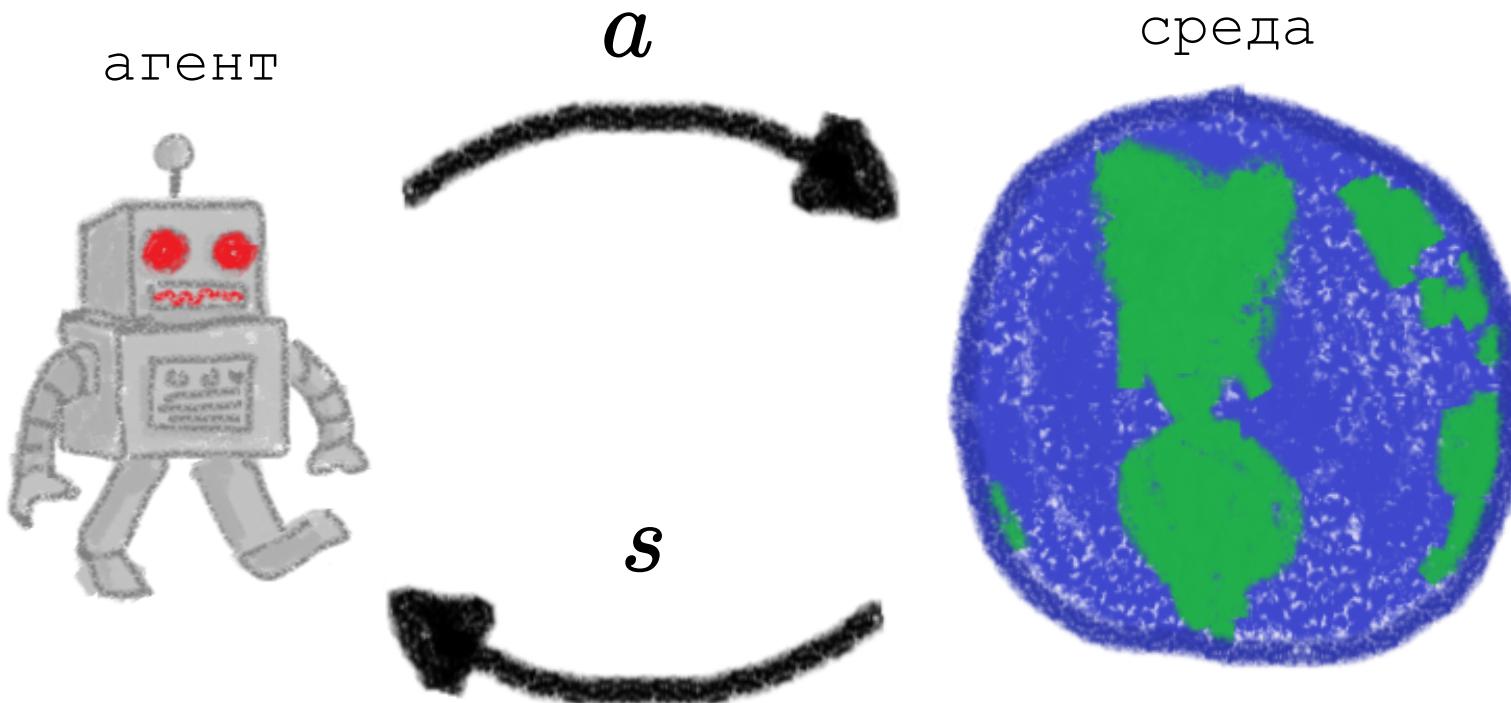
- Агент взаимодействует со средой - наблюдает ее состояние  $s$
- Агент посыпает в среду действие  $a$



# Задача Reinforcement Learning

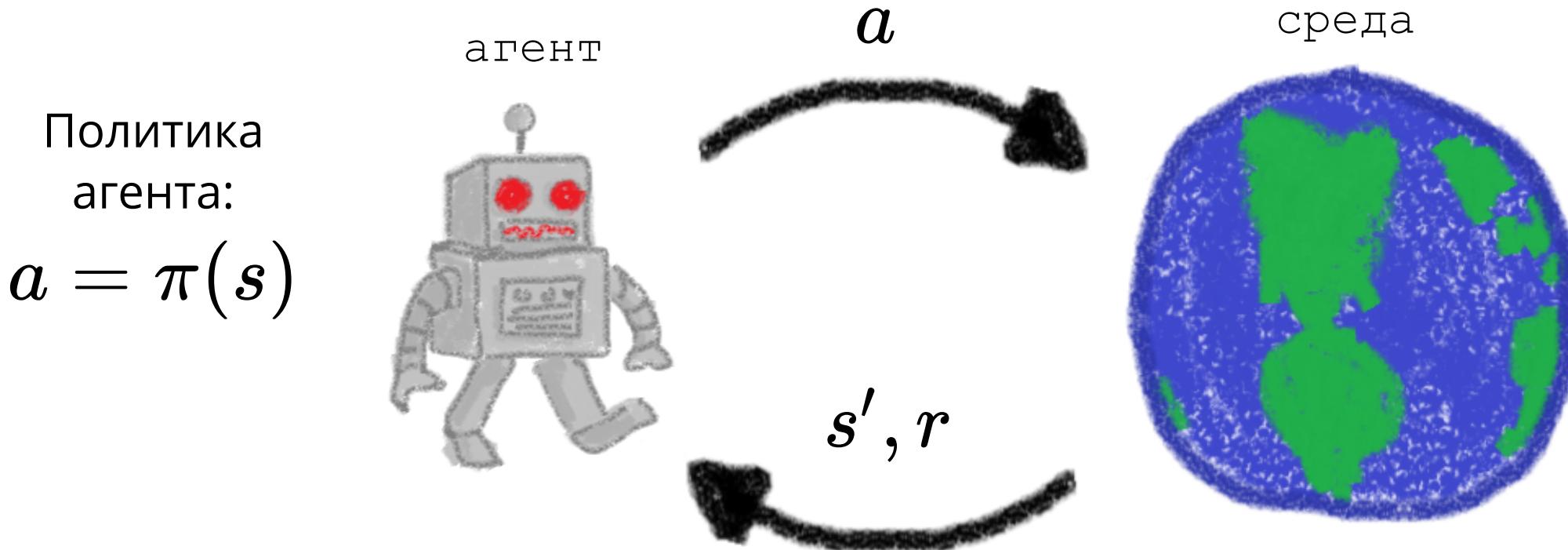
- Агент взаимодействует со средой - наблюдает ее состояние  $s$
- Агент посыпает в среду действие  $a$

Политика  
агента:  
 $a = \pi(s)$



# Задача Reinforcement Learning

- Агент взаимодействует со средой - наблюдает ее состояние  $s$
- Агент посыпает в среду действие  $a$
- Среда возвращает агенту следующее состояние  $s'$  и награду  $r$



# Среды и наблюдения

Пример: **робот-пылесос**



# Среды и наблюдения

Пример: **робот-пылесос**

**Цель:** пропылесосить всю квартиру

**Действия  $a$ :** повороты, скорость вращения колес



# Среды и наблюдения

Пример: **робот-пылесос**

**Цель:** пропылесосить всю квартиру

**Действия**  $a$ : повороты, скорость вращения колес

Какие могут быть **наблюдения**  $s$  у робота?



# Среды и наблюдения

Пример: **робот-пылесос**

**Цель:** пропылесосить всю квартиру

**Действия**  $a$ : повороты, скорость вращения колес

Какие могут быть **наблюдения**  $s$  у робота?

- Данные лидаров и др. сенсоров
- Карта помещения
- Локация робота на карте



# Среды и наблюдения

Пример: **робот-пылесос**

**Цель:** пропылесосить всю квартиру

**Действия**  $a$ : повороты, скорость вращения колес

Какие могут быть **наблюдения**  $s$  у робота?

- Данные лидаров и др. сенсоров
- Карта помещения
- Локация робота на карте

Нужно ли что-то еще?



# Среды и наблюдения

Пример: **робот-пылесос**

**Цель:** пропылесосить всю квартиру

**Действия**  $a$ : повороты, скорость вращения колес

Какие могут быть **наблюдения**  $s$  у робота?

- Данные лидаров и др. сенсоров
- Карта помещения
- Локация робота на карте

Нужно ли что-то еще?

Хранение предыдущих наблюдений?



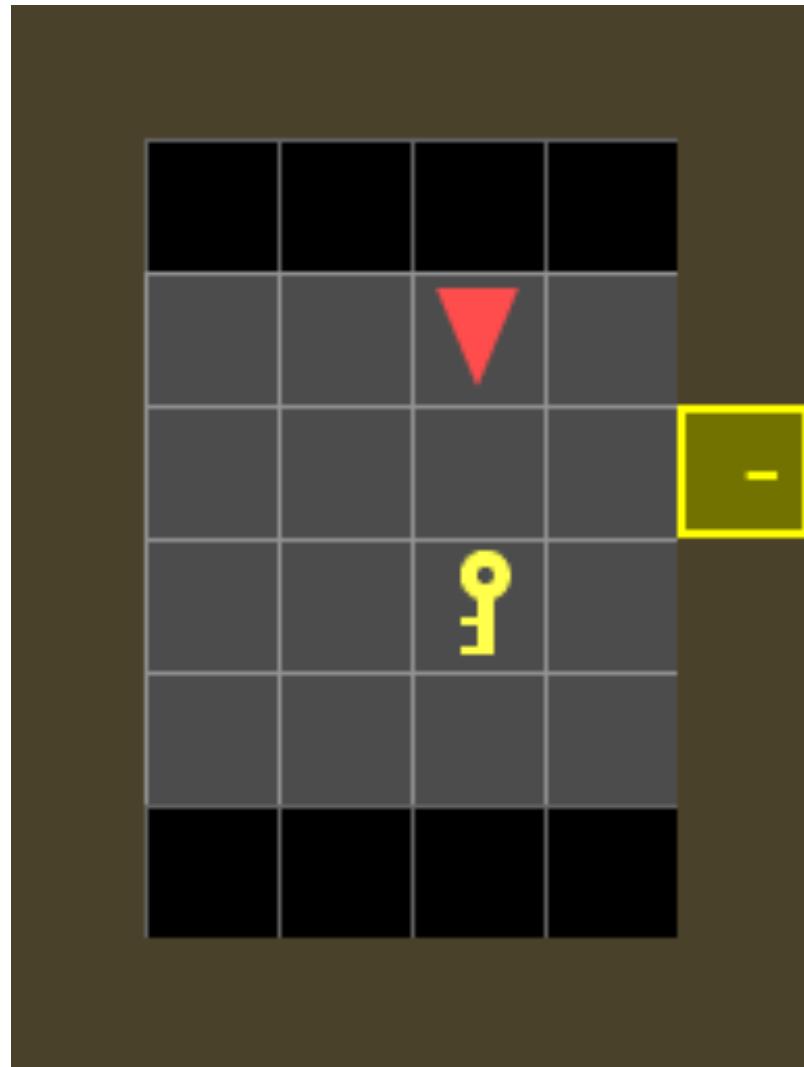
# Марковское свойство

## Markov Decision Process (MDP)

**Задача:** открыть дверь ключом

**Действия:**

- вверх \ вниз \ влево \ вправо
- взять объект
- использовать объект  
(если находишься у двери)

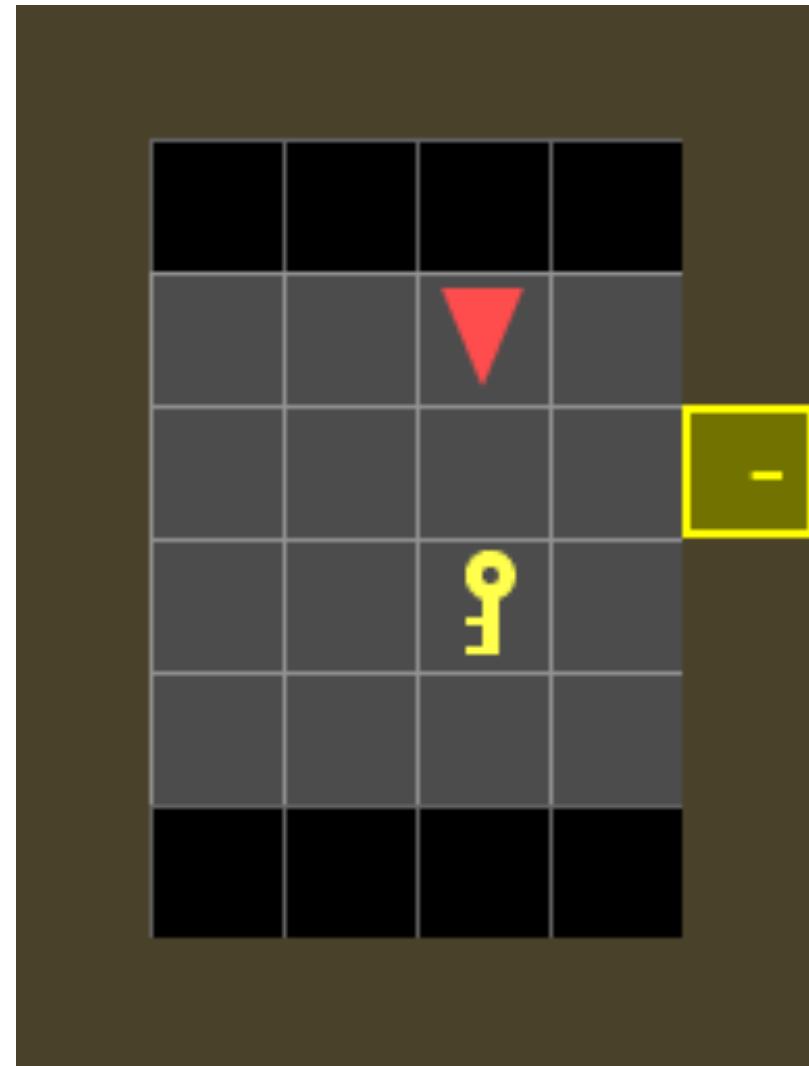


# Марковское свойство

## Markov Decision Process (MDP)

Нужно ли агенту помнить свою историю?

**Возможные наблюдения:**



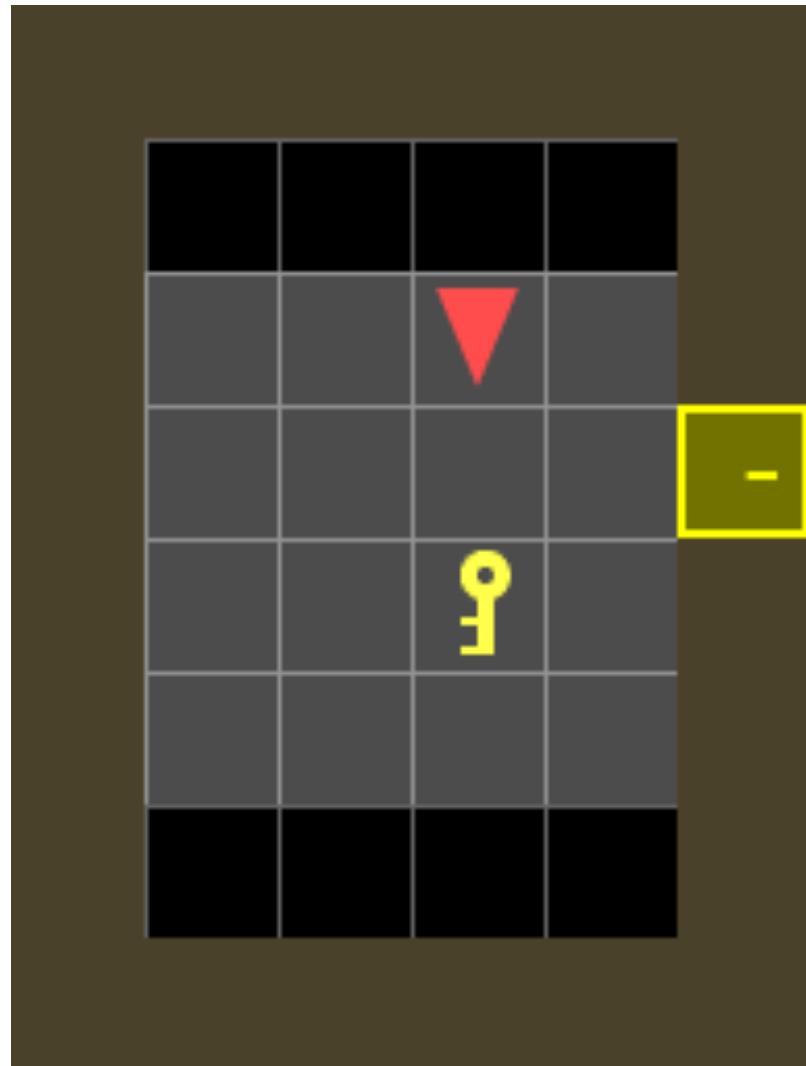
# Марковское свойство

## Markov Decision Process (MDP)

Нужно ли агенту помнить свою историю?

### Возможные наблюдения:

1. координаты агента



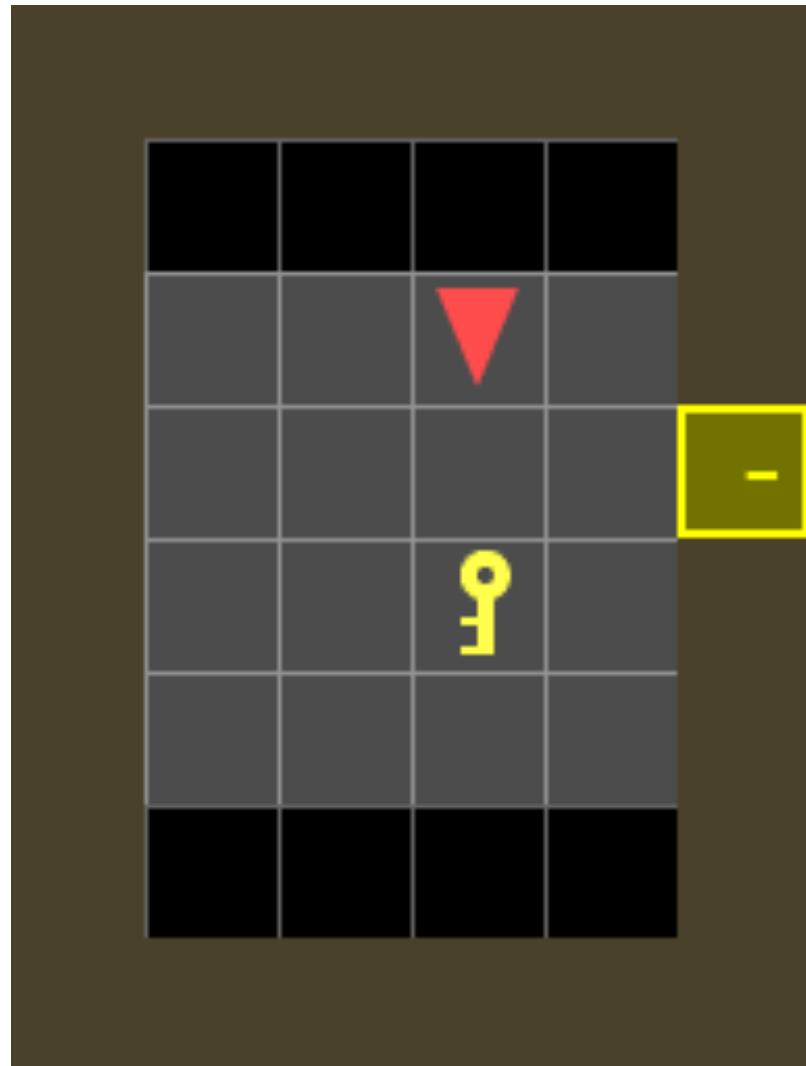
# Марковское свойство

## Markov Decision Process (MDP)

Нужно ли агенту помнить свою историю?

### Возможные наблюдения:

1. координаты агента
2. полная картинка лабиринта



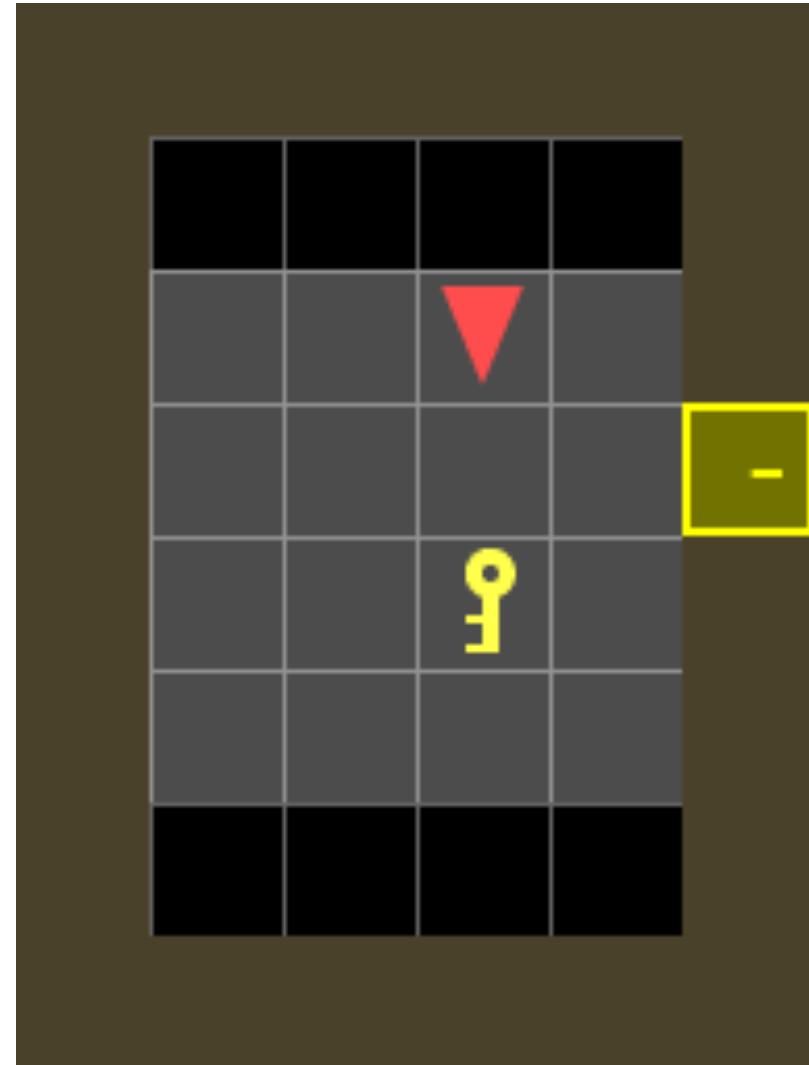
# Марковское свойство

## Markov Decision Process (MDP)

Нужно ли агенту помнить свою историю?

### Возможные наблюдения:

1. координаты агента
2. полная картинка лабиринта
3. координаты агента + есть ли у него ключ



# Марковское свойство

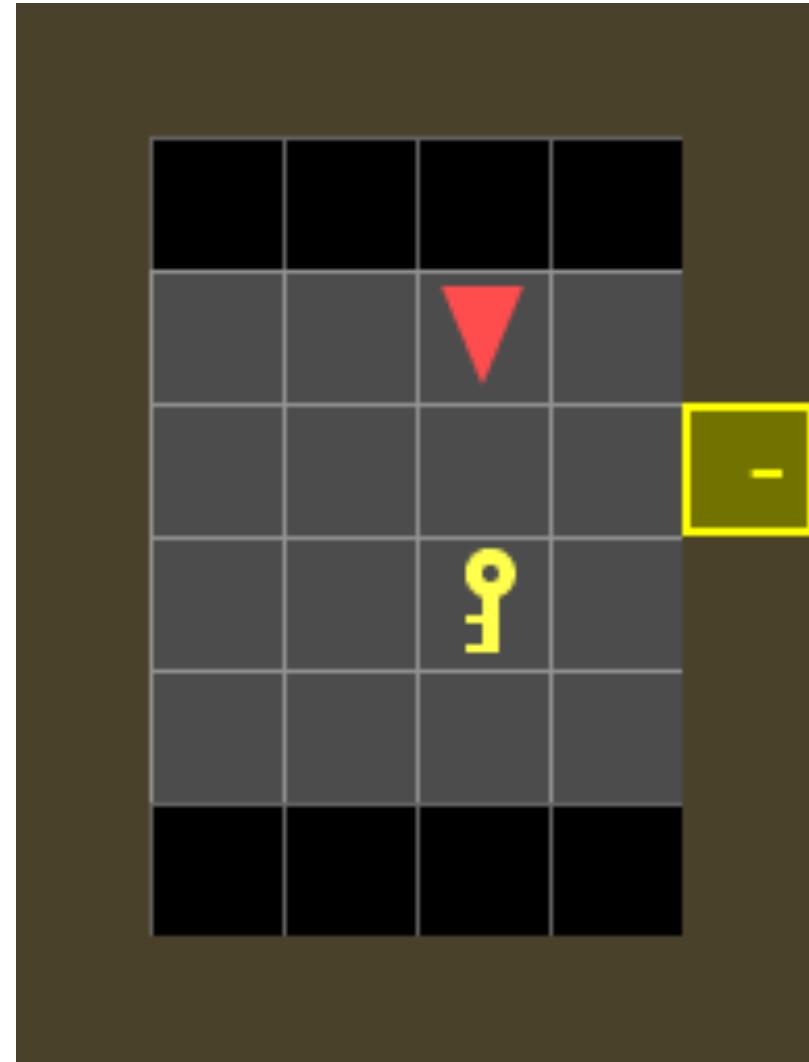
## Markov Decision Process (MDP)

Нужно ли агенту помнить свою историю?

### Возможные наблюдения:

1. координаты агента
2. полная картинка лабиринта
3. координаты агента + есть ли у него ключ

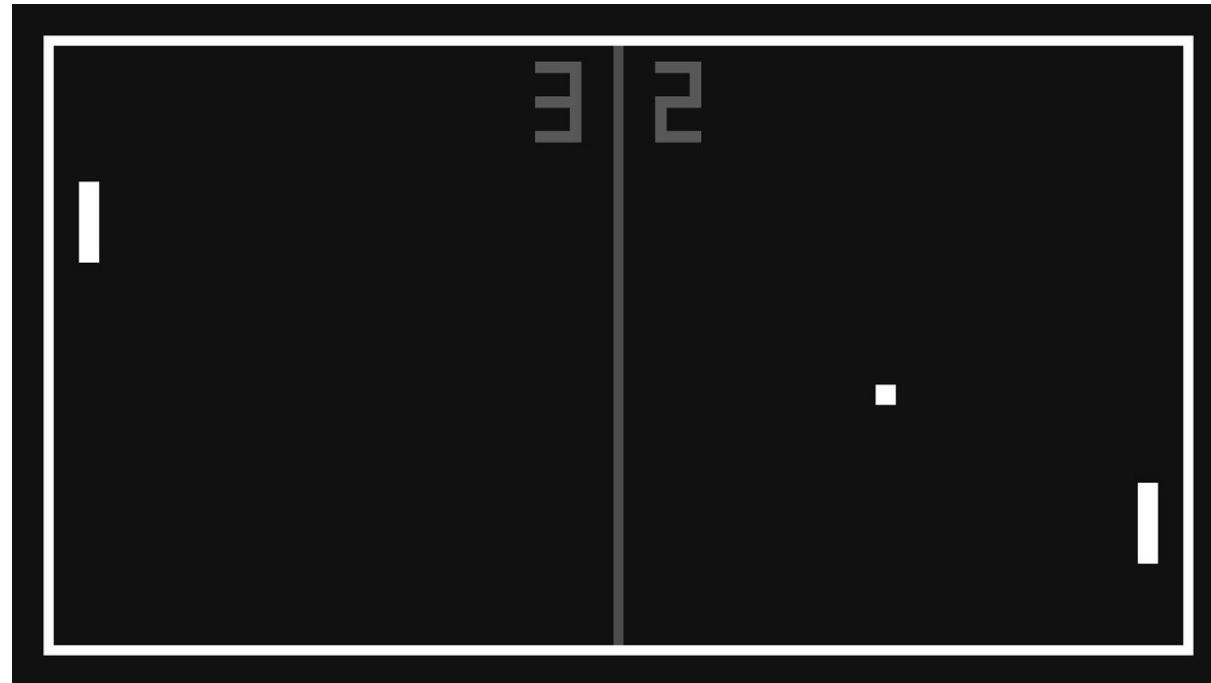
2 и 3 не требуют хранения истории



Марковость: "Будущее не зависит от прошлого, если известно настоящее"

# Пример немарковости

Куда летит шарик?



# Задача Reinforcement Learning

# Как выбрать награду?

На примере шахмат

Какой выбор награды лучше?

- победа: +1
  - проигрыш: -1
  - ничья: 0
- 
- победа: +1
  - проигрыш: -1
  - ничья: 0
  - взятие фигуры: +1
  - потеря фигуры: -1



Почему?

# Как выбрать награду?

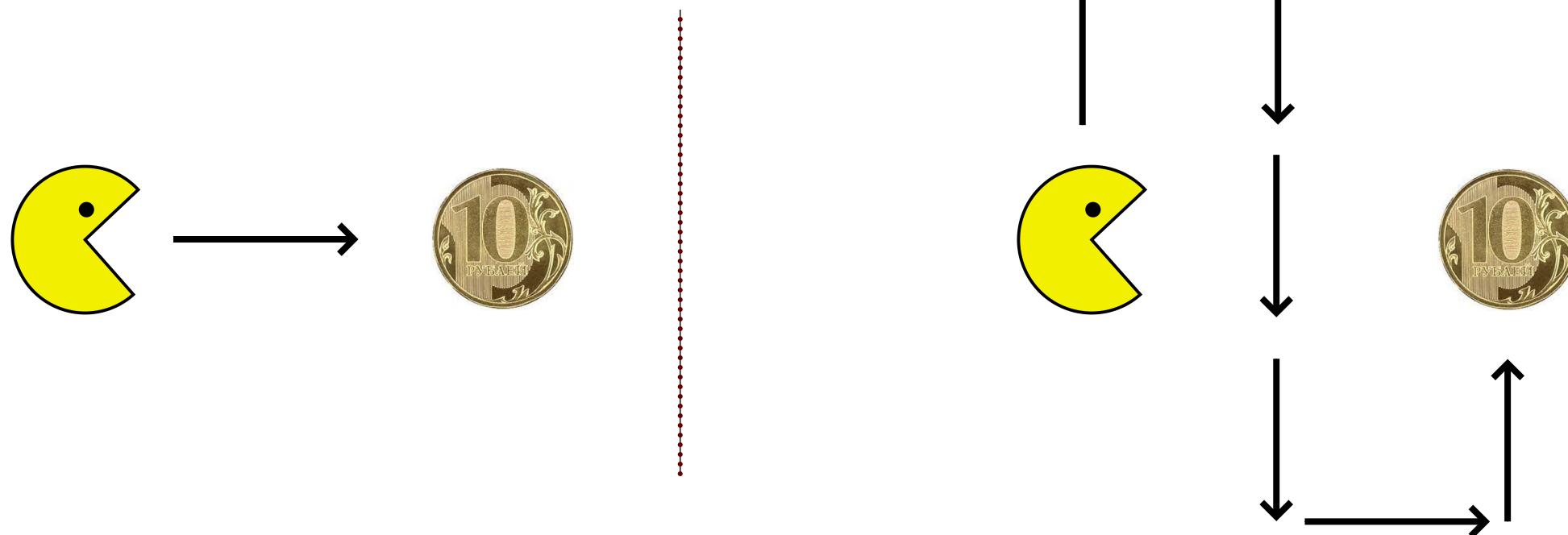
Агент, стремящийся получить максимум награды, должен лучше всего решать вашу задачу!

<https://www.youtube.com/embed/tI0IHko8ySg?enablejsapi=1>

# Дисконтирование награды

Сравним эти две траектории агента

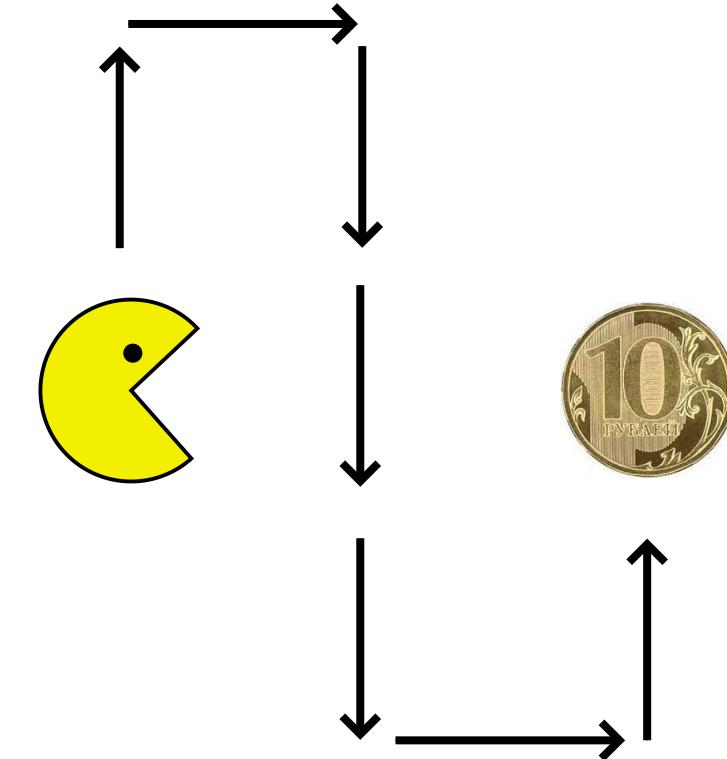
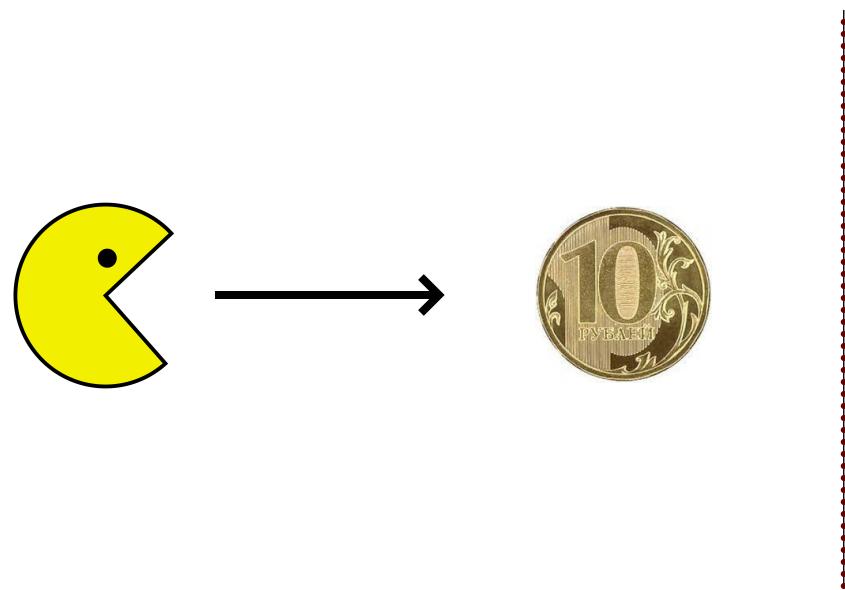
Какая лучше?



# Дисконтирование награды

Сравним эти две траектории агента

Какая лучше?



Награда сегодня лучше, чем награда завтра!

Дисконтирование награды - уменьшение значимости наград, которые далеко

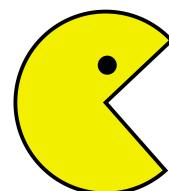
$$R_\pi = \sum_{t=0}^T \gamma^t r_t \quad \text{где } 0 < \gamma < 1$$

# Дисконтирование награды

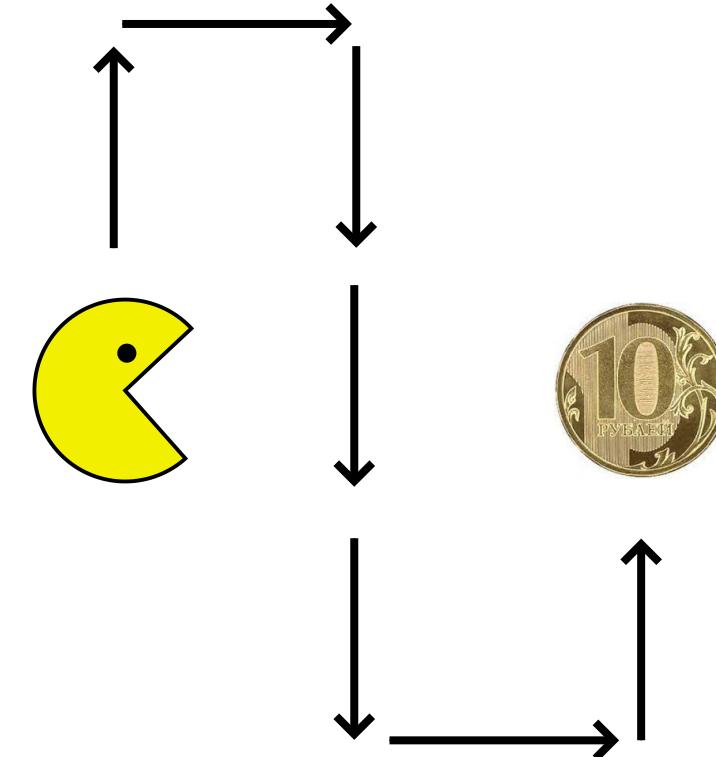
Сравним эти две траектории агента

Какая лучше?

пусть  $\gamma = 0.9$



$r = 1$



Награда сегодня лучше, чем награда завтра!

Дисконтирование награды - уменьшение значимости наград, которые далеко

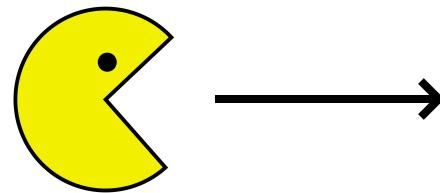
$$R_\pi = \sum_{t=0}^T \gamma^t r_t \quad \text{где } 0 < \gamma < 1$$

# Дисконтирование награды

Сравним эти две траектории агента

Какая лучше?

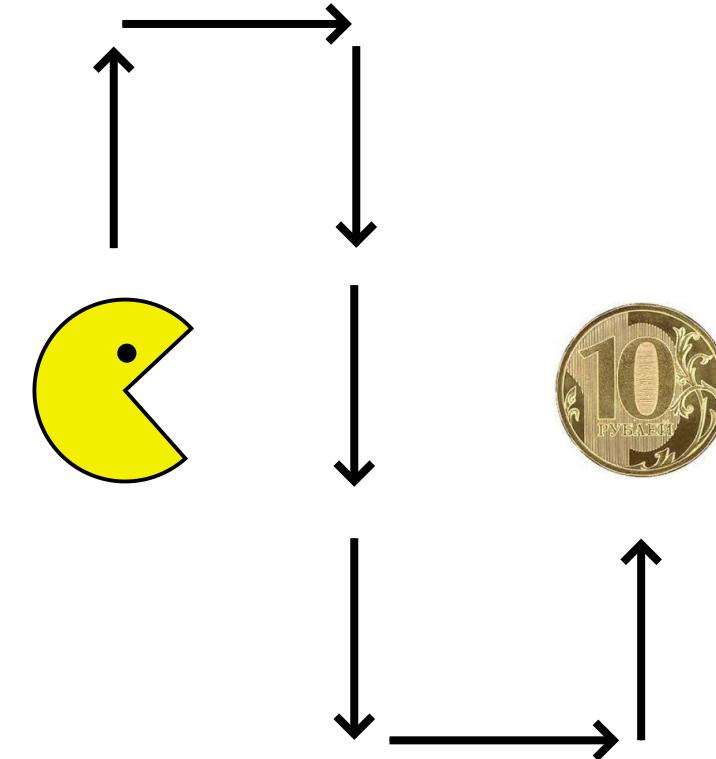
пусть  $\gamma = 0.9$



$r = 1$



$$R = \gamma^1 r = 0.9$$



Награда сегодня лучше, чем награда завтра!

Дисконтирование награды - уменьшение значимости наград, которые далеко

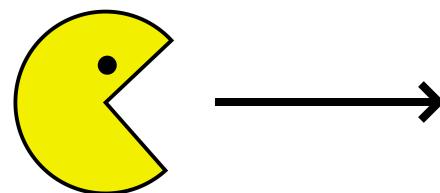
$$R_\pi = \sum_{t=0}^T \gamma^t r_t \quad \text{где } 0 < \gamma < 1$$

# Дисконтирование награды

Сравним эти две траектории агента

Какая лучше?

пусть  $\gamma = 0.9$



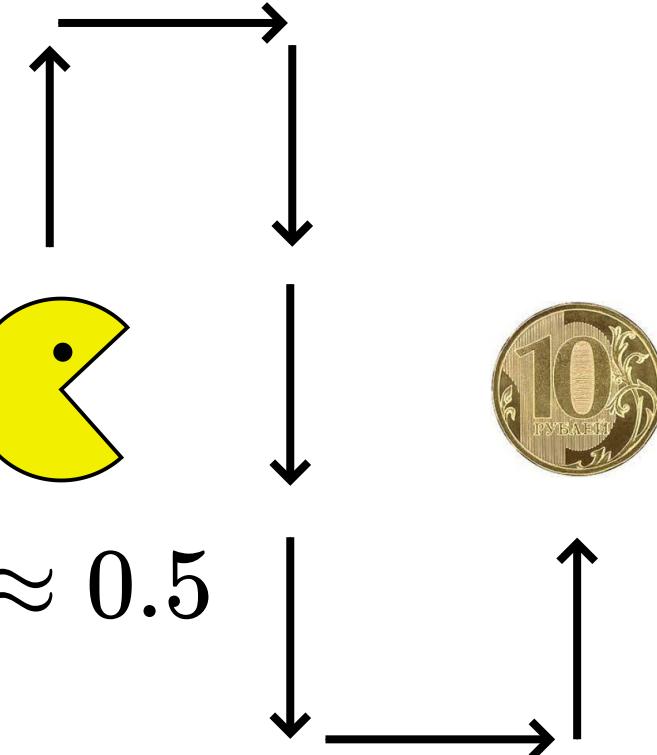
$r = 1$



$$R = \gamma^1 r = 0.9$$



$$R = \gamma^7 r \approx 0.5$$



Награда сегодня лучше, чем награда завтра!

Дисконтирование награды - уменьшение значимости наград, которые далеко

$$R_\pi = \sum_{t=0}^T \gamma^t r_t \quad \text{где } 0 < \gamma < 1$$

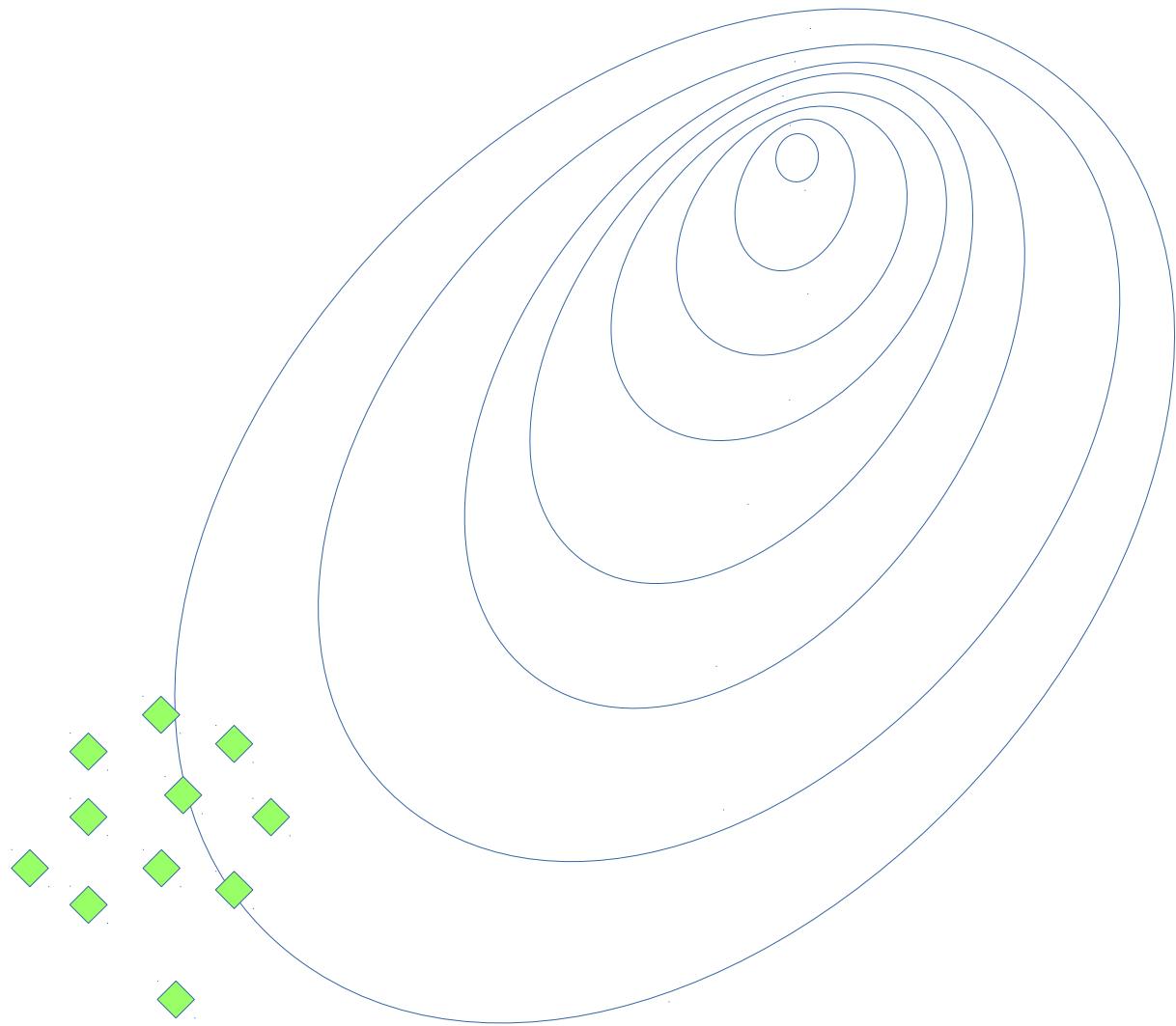
# Crossentropy method

Initialize policy

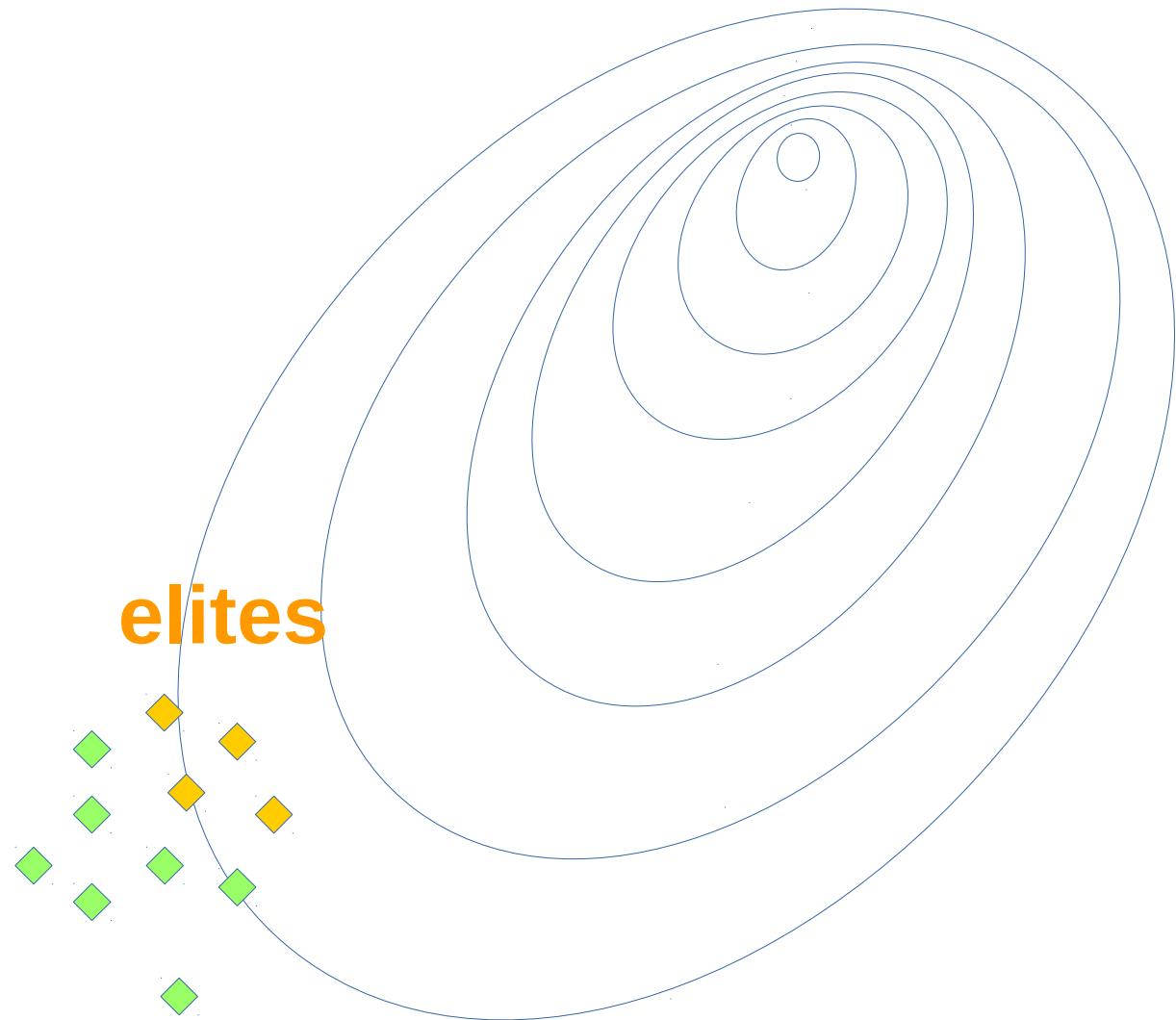
Repeat:

- Sample  $N[100]$  sessions
- Pick  $M[25]$  best sessions, called **elite** sessions
- Change policy so that it prioritizes actions from elite sessions

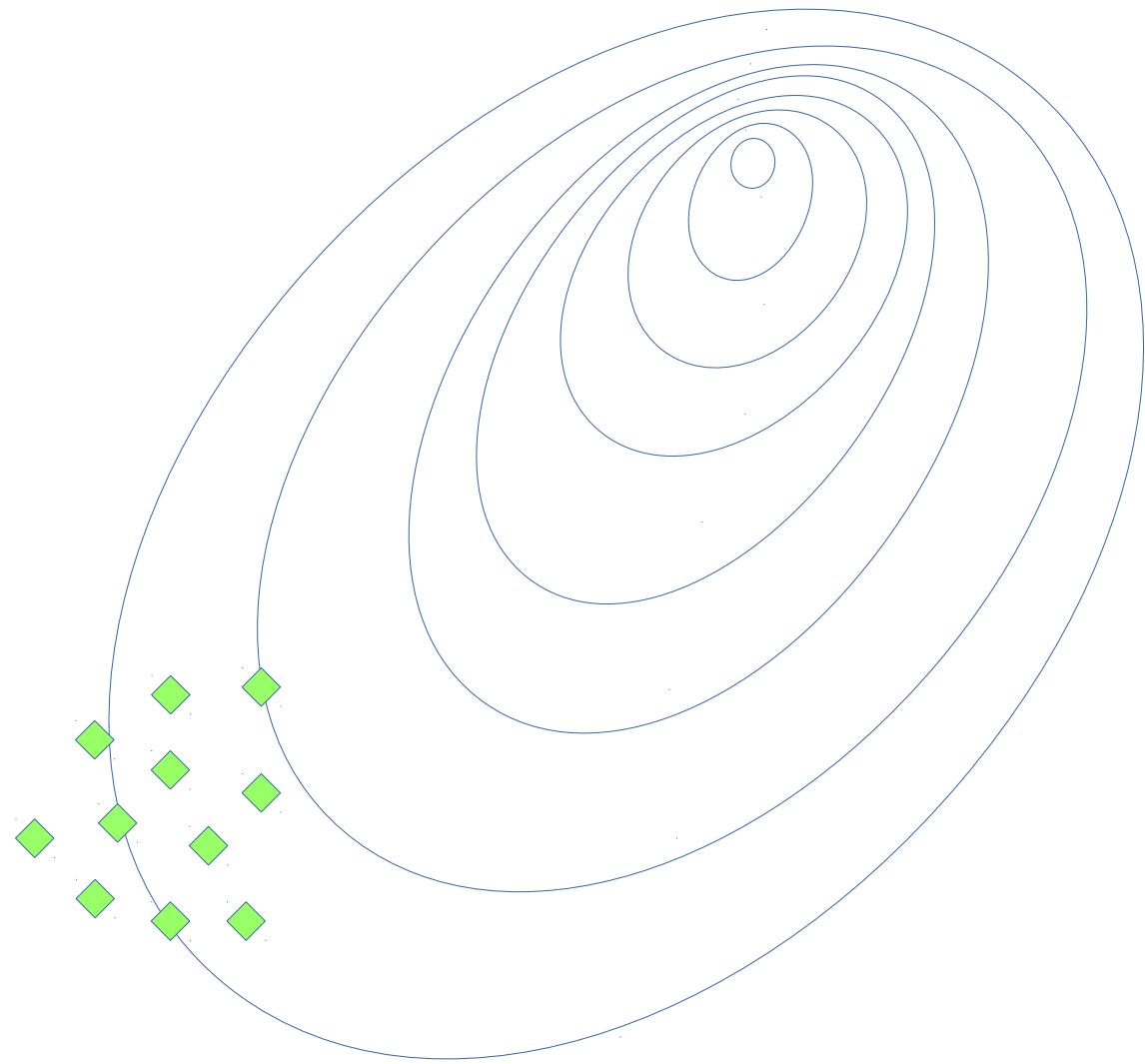
# Step-by-step view



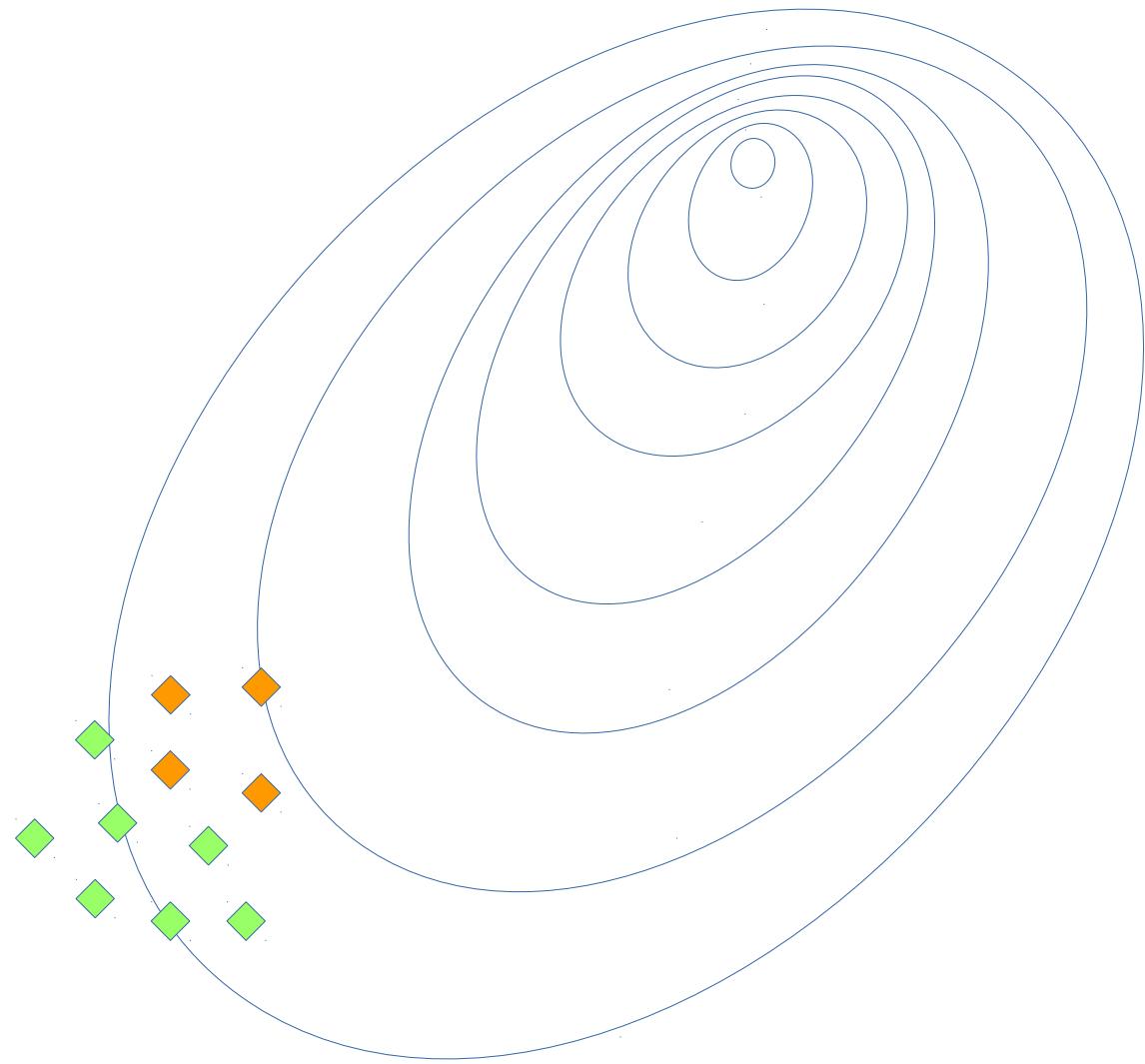
# Step-by-step view



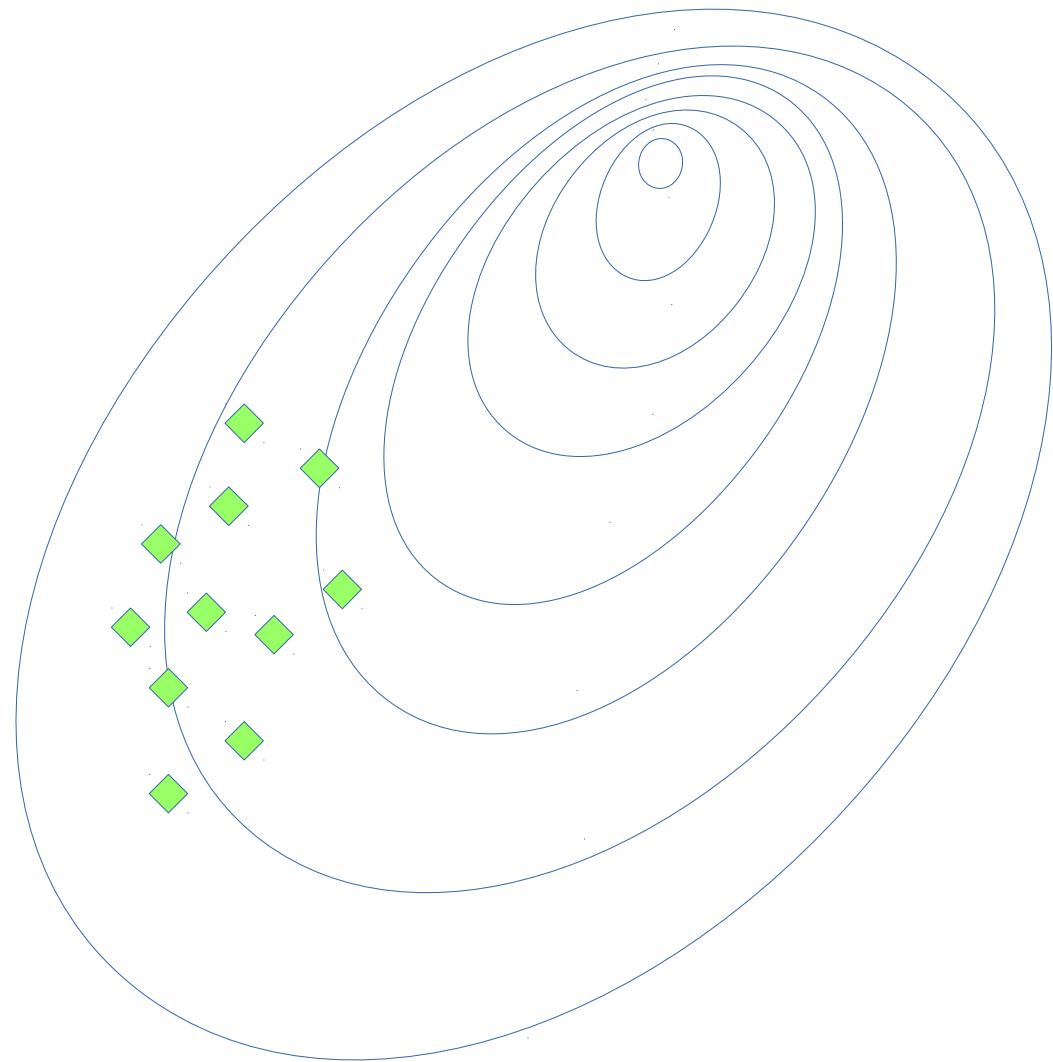
# Step-by-step view



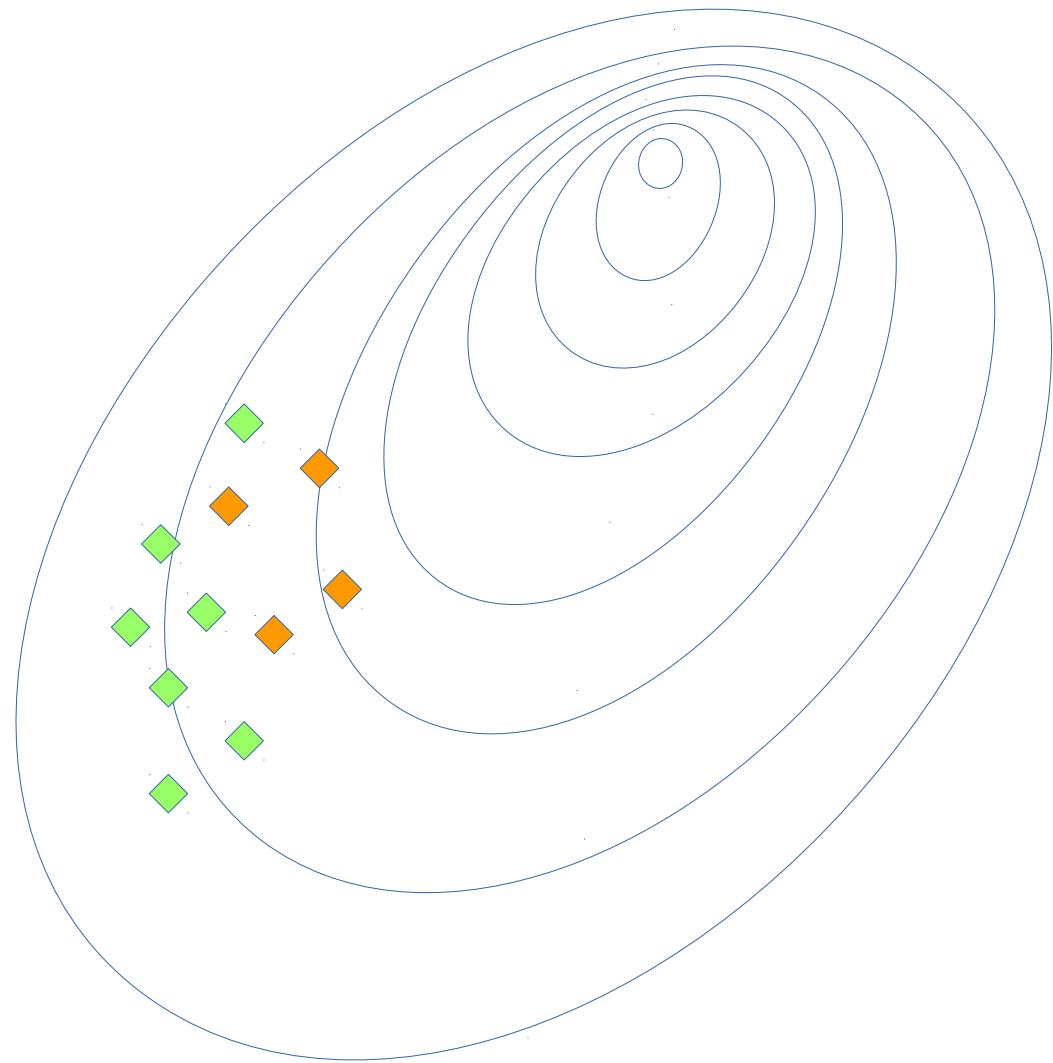
# Step-by-step view



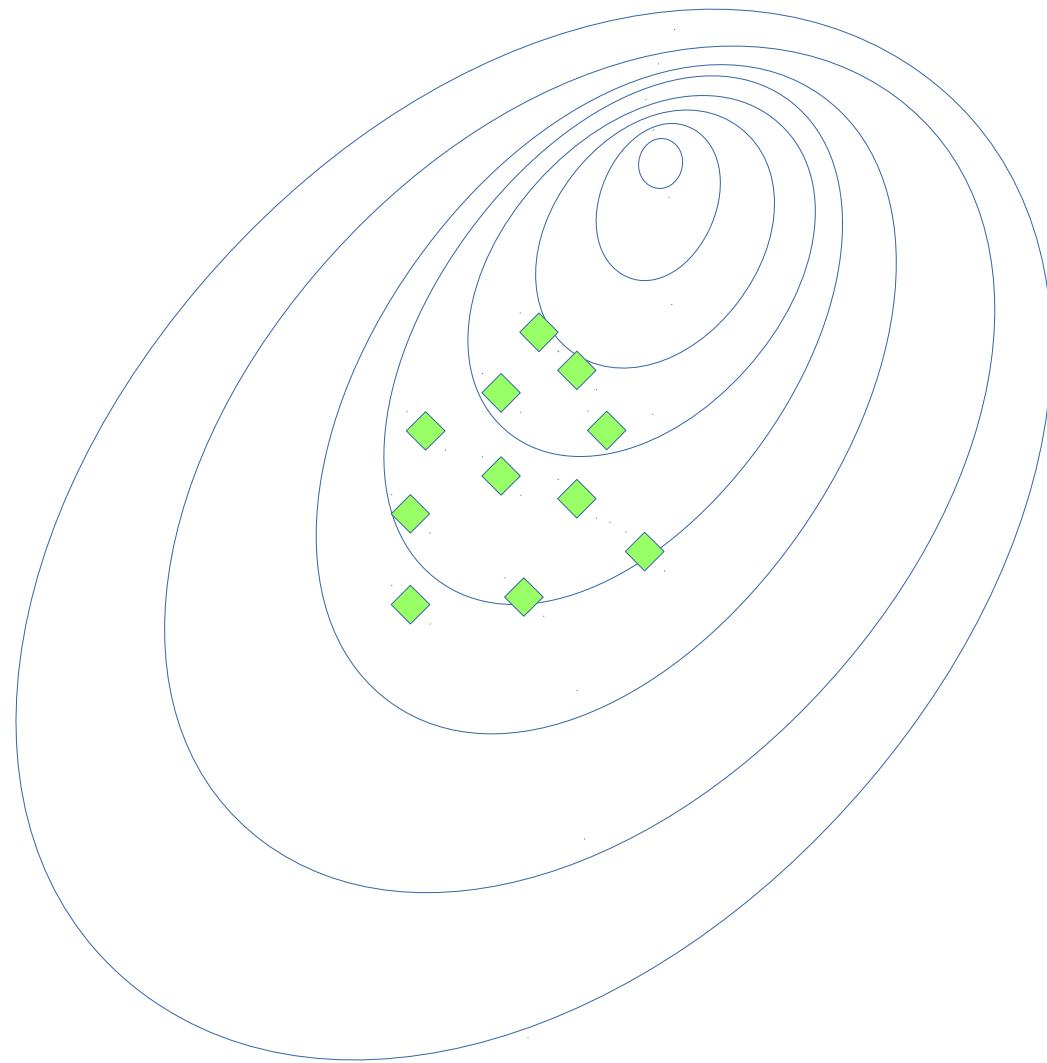
# Step-by-step view



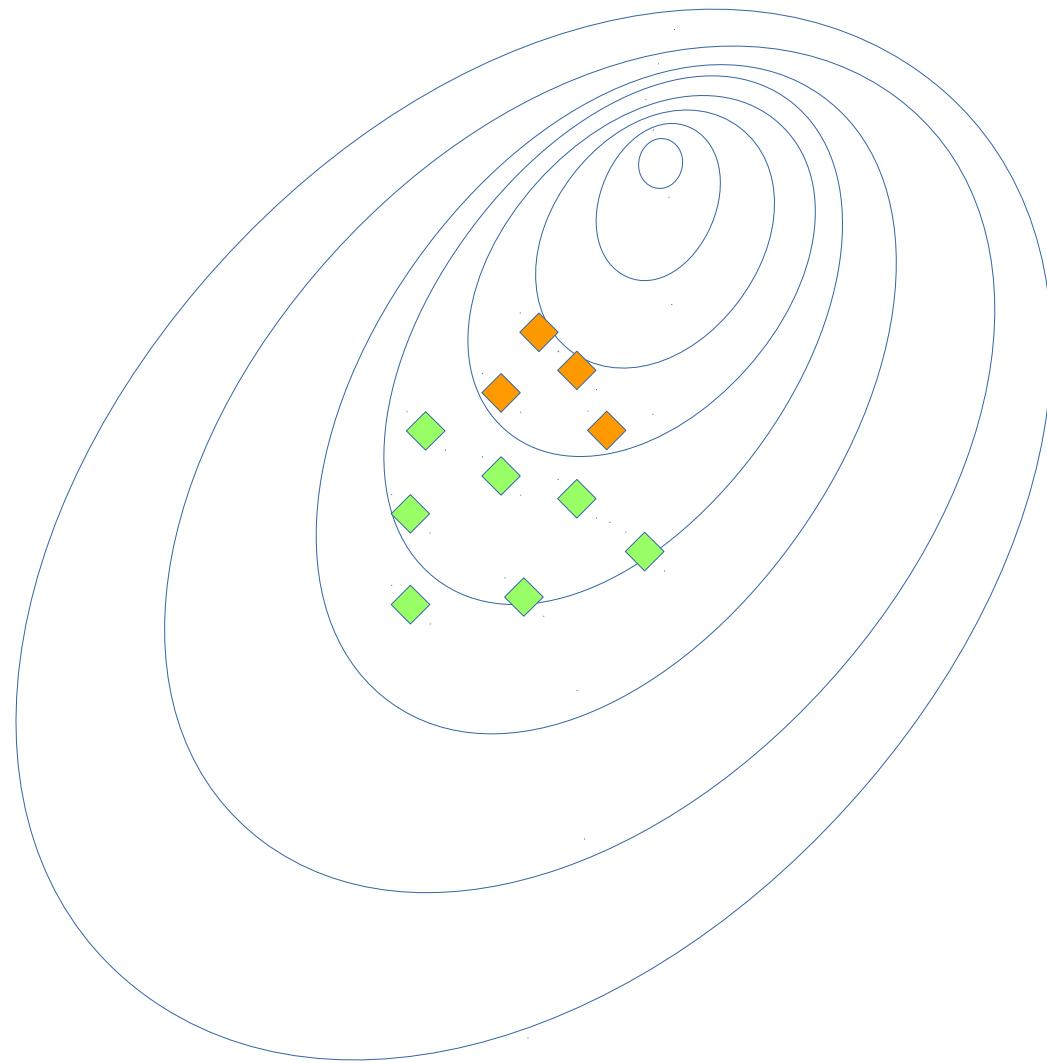
# Step-by-step view



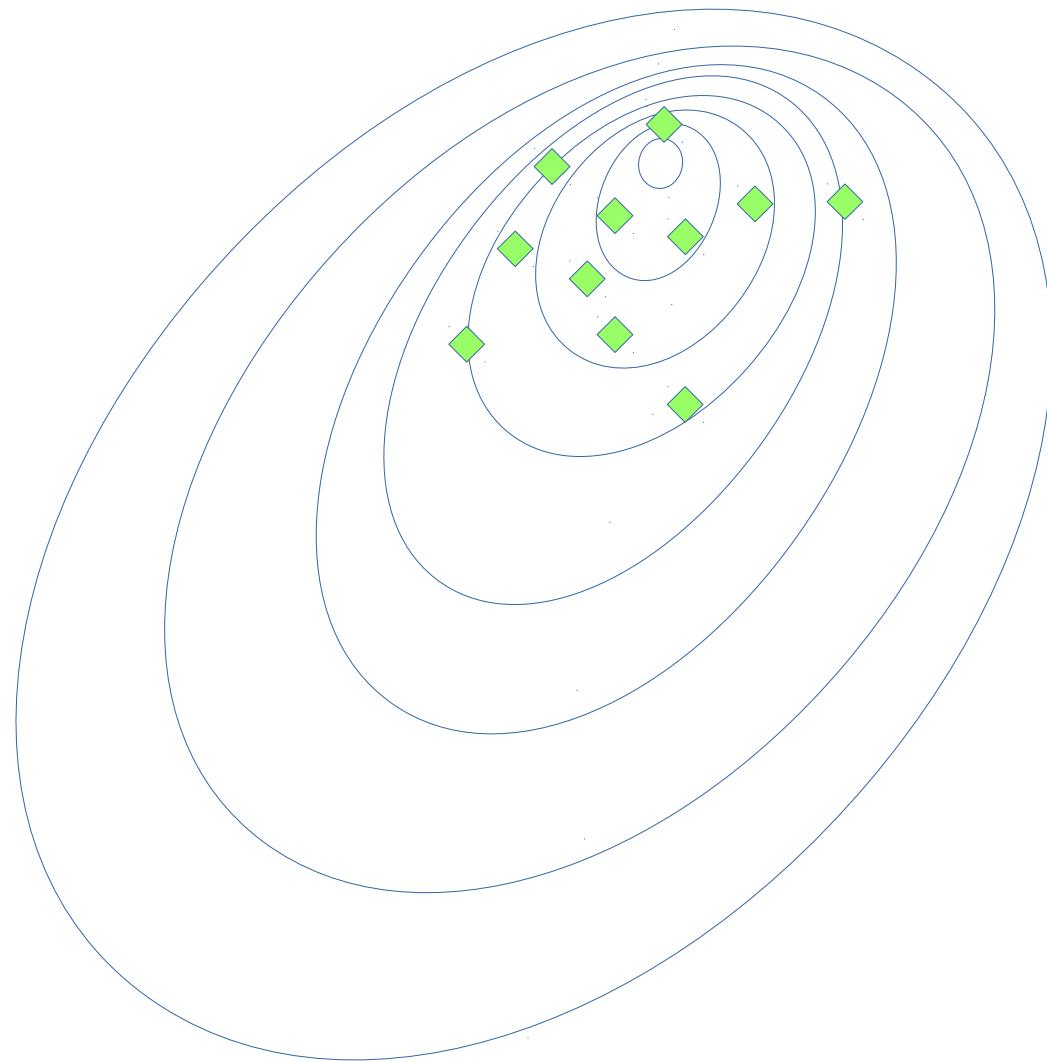
# Step-by-step view



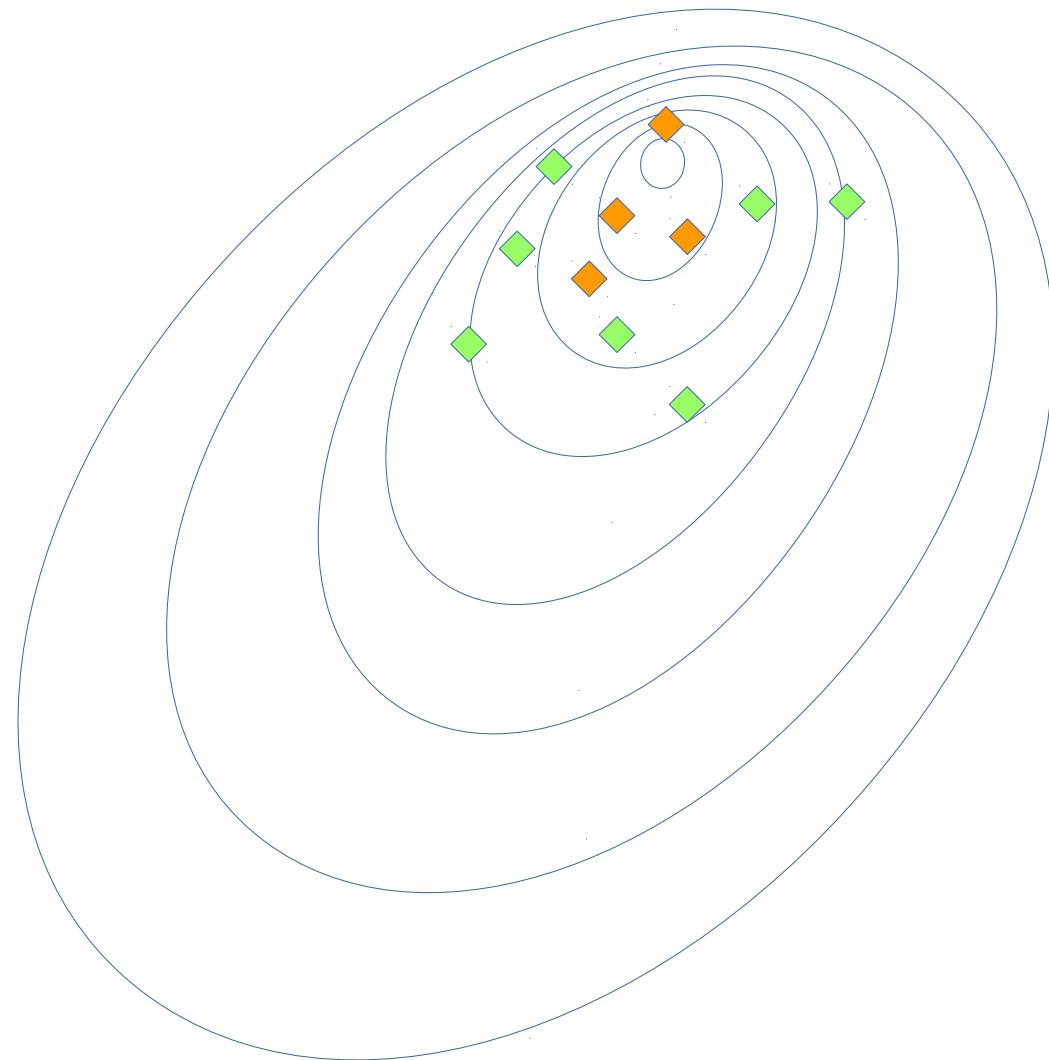
# Step-by-step view



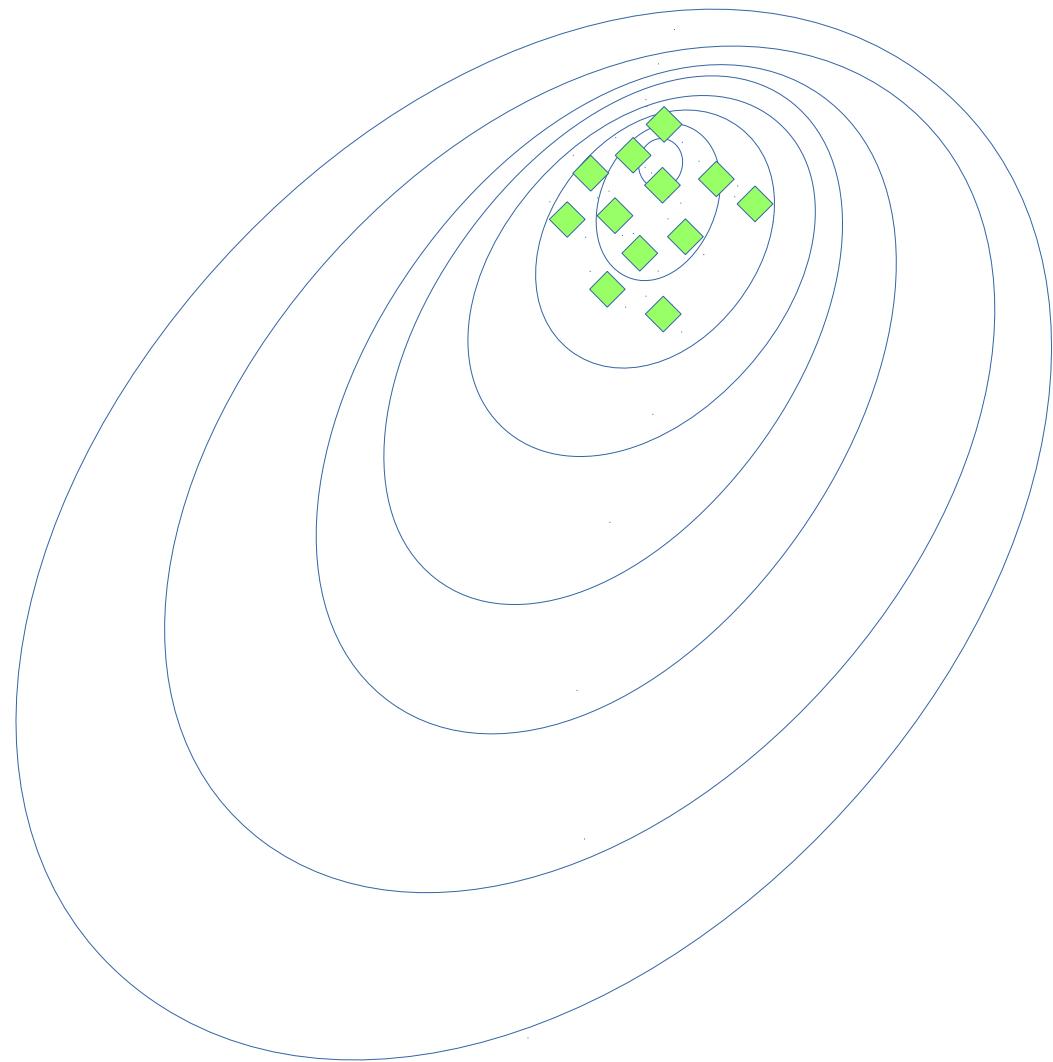
# Step-by-step view



# Step-by-step view



# Step-by-step view



# Tabular crossentropy method

- Policy is a matrix

$$\pi(a|s) = A_{s,a}$$

- Sample N games with that policy
- Get M best sessions (elites)

$$Elite = [(s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_k, a_k)]$$

# Tabular crossentropy method

- Policy is a matrix

$$\pi(a|s) = A_{s,a}$$

- Sample N games with that policy
- Take M best sessions (elite)
- Aggregate by states

$$\pi(a|s) = \frac{\text{took } a \text{ at } s}{\text{was at } s}$$

**In M best games**

# Crossentropy method

Initialize policy

Repeat:

- Sample  $N[100]$  sessions
- Pick  $M[25]$  best sessions, called **elite** sessions
- Change policy so that it prioritizes actions from elite sessions

# Grim reality

If your environment has infinite/large state space

