Deep Learning

Lecture 8.1

from really good course in AI masters (https://ozonmasters.ru/reinforcementlearning).

Recap

- Semantic segmentation problem
- Upsampling
- Architectures
- Panoptic / Instance segmentation

What is Reinforcement Learning?

Let’s start from...

Supervised Learning Problem

Supervised Learning case:

Given Dataset $D := \{(X_i, y_i)\}$

Learn a function that will predict y from X: $f_\theta: X \rightarrow y$

e.g. find parameters theta that will minimize: $L(f_\theta (X_i), y_i)$, where L is a loss function

Standard Assumptions:

- Samples in dataset are I.I.D
- We have ground truth labels y

No ground truth answers

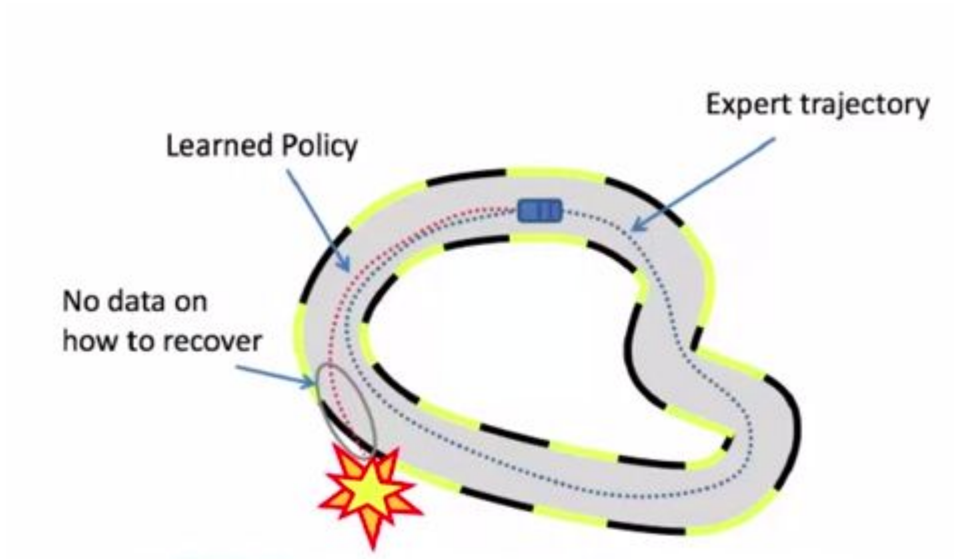
You don't have answers at all

Your answers are not good enough

Choice matters

Assume that we have expert trajectories, i.e. sufficiently good answers:

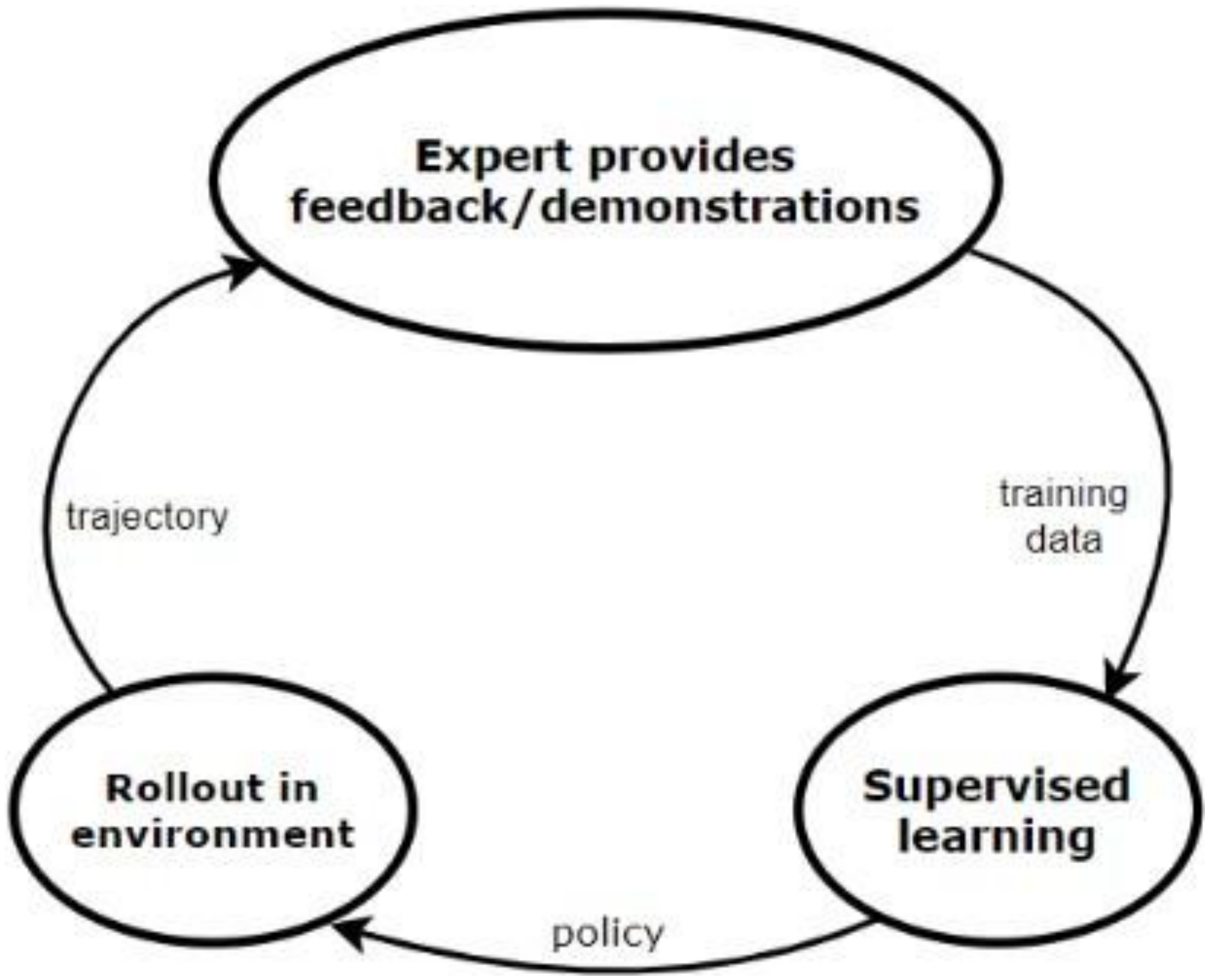
- Treat trajectories as a dataset:
  $D = \{(x_1, a_1), ..(x_N, a_N)\}$
- Train with Supervised Learning
- Done?:)



Choice matters

New Plan (DAGGER algorithm):

1. Train a model from human trajectories :
   $D_0 = \{(x_1, a_1), ..(x_N, a_N)\}$
2. Run the model to get new trajectories:
   $D' = \{(x_1, ?), ..(x_N, ?)\}$
3. Ask humans to label D' with actions a_t
4. Aggregate: $D1 \leftarrow D_0 \cup D'$
5. Repeat



Choice matters

But this is really hard to do: 3. Ask humans to label D' with actions a_t

Reinforcement learning

If you know what you want, but don't know how to do it...

USE REWARDS!

Assumptions:

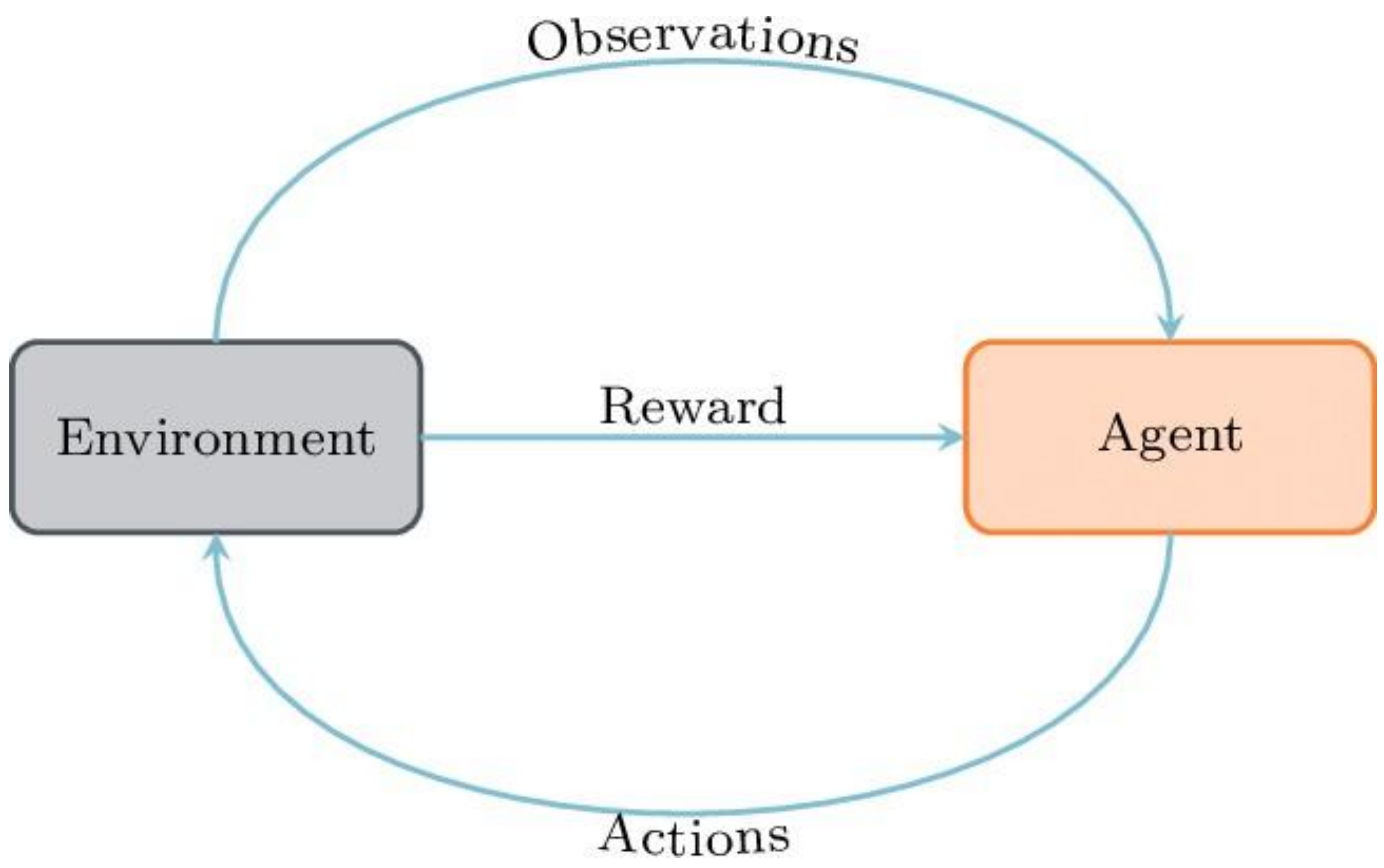
- **It's easy to compute reward**
- **You can express your goals with rewards!**

Reinforcement Learning Problem

You have **Agent** and **Environment** that interact with each other:

- Agent's actions change the state environment
- After each action agent receives new state and reward

Interaction with environment is typically divided into **episodes.**



Reinforcement Learning Problem

Agent has a policy: $\pi(\text{action} \mid \text{observations from env})$

Agent learns its policy via **Trial and Error**!

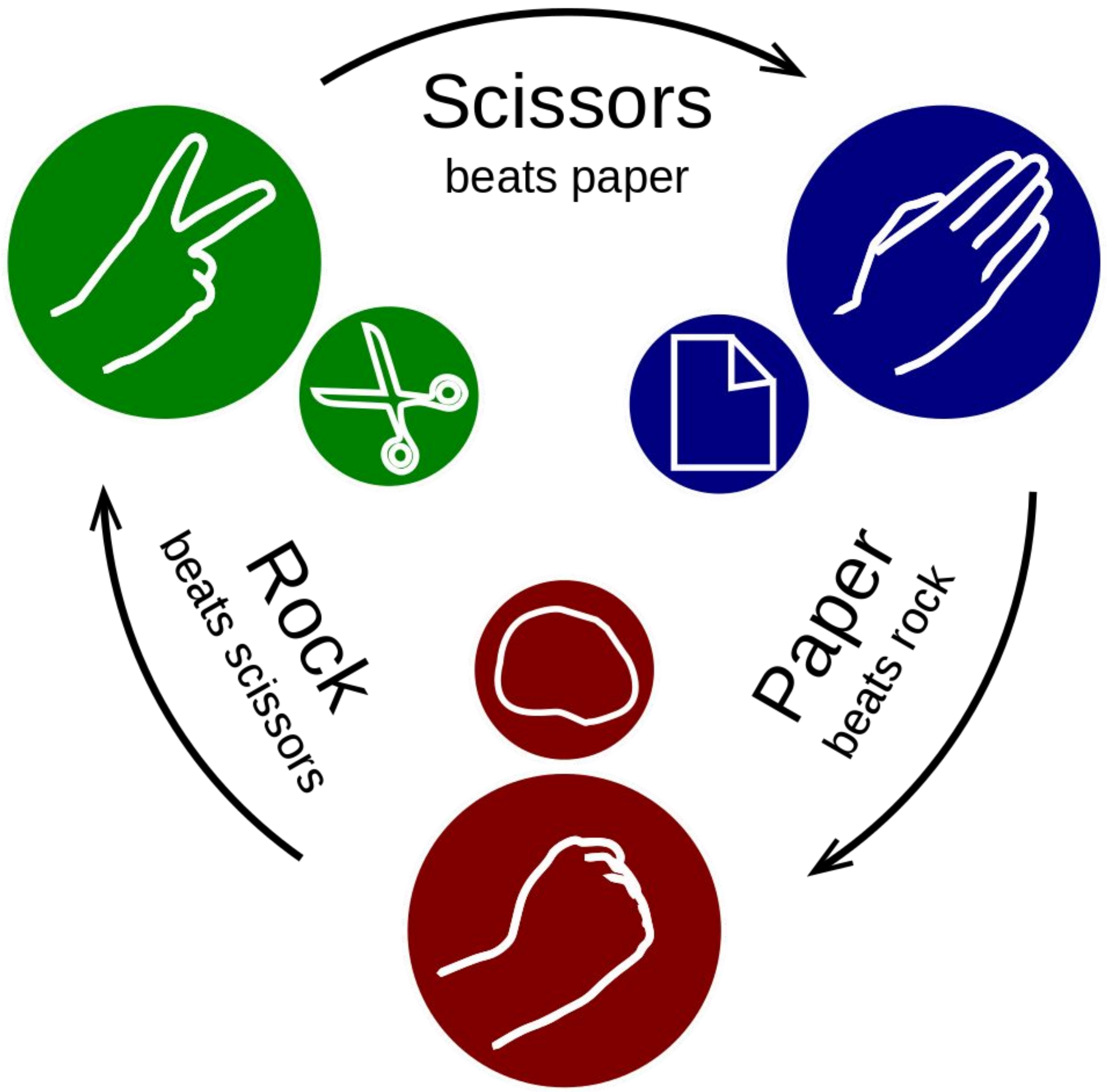
The goal is to find a policy that maximizes **total expected reward:**

$$\text{maximize}_{\pi} \mathrm{E}_{\pi}[\textstyle\sum^{T}_{t=0} r_t]$$

Why we need E_{π} ?

A non-deterministic policy or environment lead to

a distribution of total rewards!



Environment and Observation

What should an agent observe?

- Wheel speed
- Acceleration
- LiDAR
- Battery

- Map of the apartment
- Location

Is this enough?

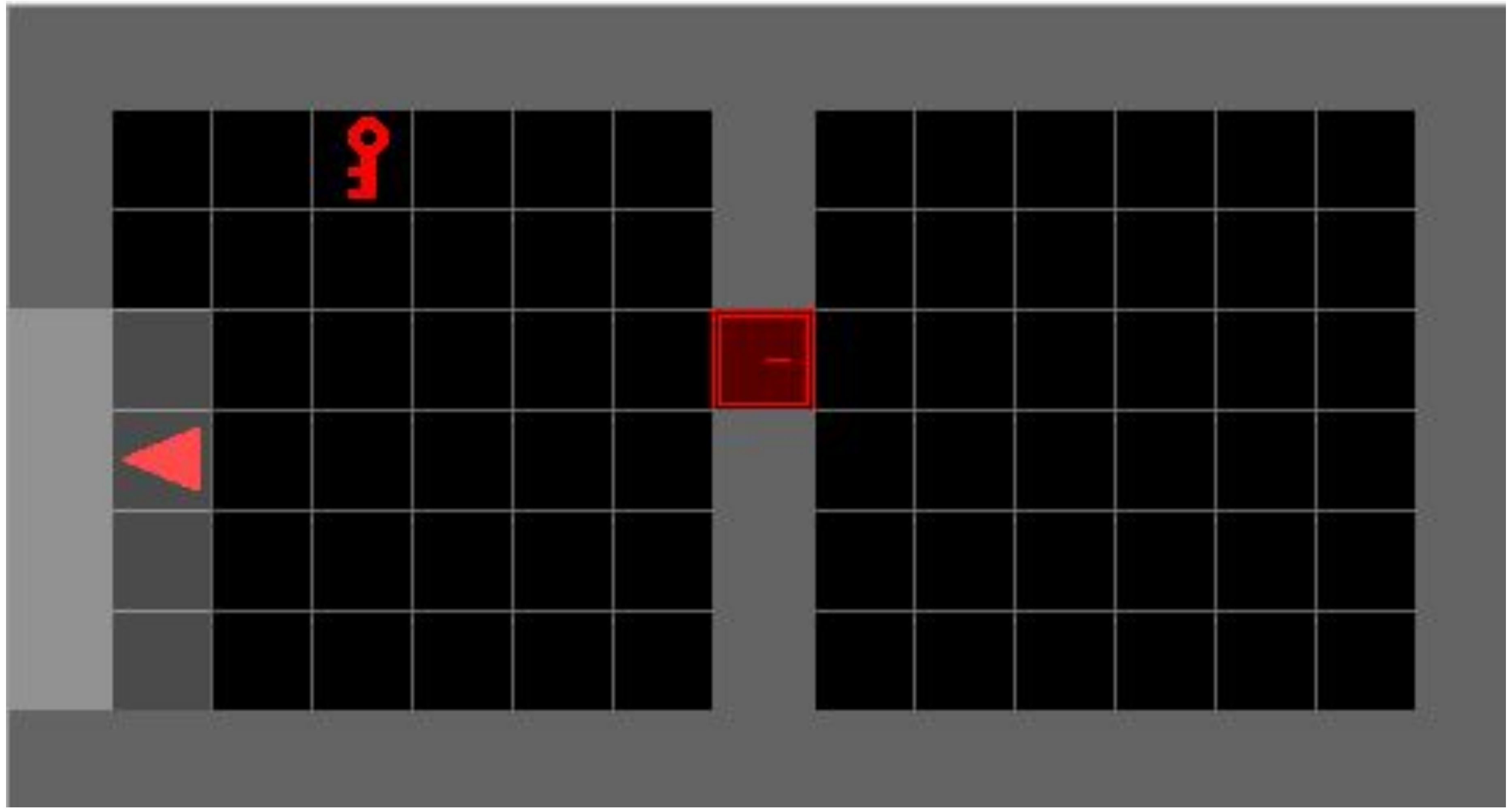
Does agent need past observations?

Markovian Property

Task: Open the red door with the key
Details: Agent starts at random location

Actions:

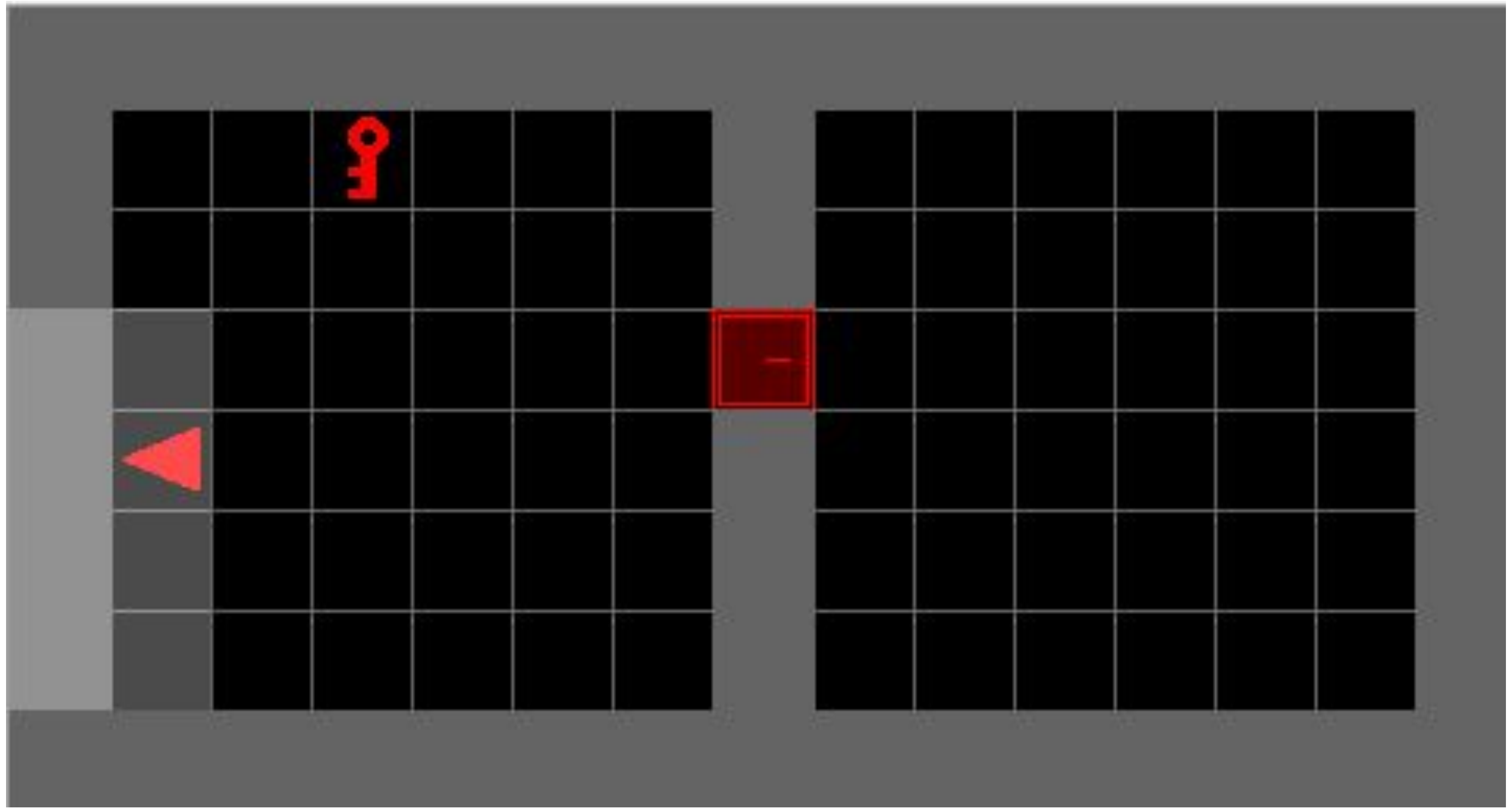
- move up/left/right/down
- pick up an object
- apply an object to the door (when near the door)



Markovian Property

Which observations are enough to learn the optimal policy?

1. Agent's coordinates, and previous action
2. Full image of the maze
3. Agent's coordinates and does it has key



For 2 and 3 agent doesn't need to remember it's history:

$$P\,(o_{t+1}, r_{t+1} \,|\, o_t, a_t) = P\,(o_{t+1}, r_{t+1} \,|\, o_t, a_t, \ldots, o_1, a_1, o_0, a_0)$$

Markovian property: **"The future is independent of the past given the present."**

Markov Decision Process

MDP is a 5-tuple $< S, A, R, T, \gamma >$:

- S is a set of states
- A is a set of actions
- $R : S \times A \rightarrow \mathrm{R}$ is a reward function
- $T : S \times A \times S \rightarrow [0, 1]$ is a transition function
  $T(s, a, s') = \;\; P(S_{t+1} = s' | S_t = s, A_t =$
- $a)$ $\gamma \in [0, 1]$ is a discount factor

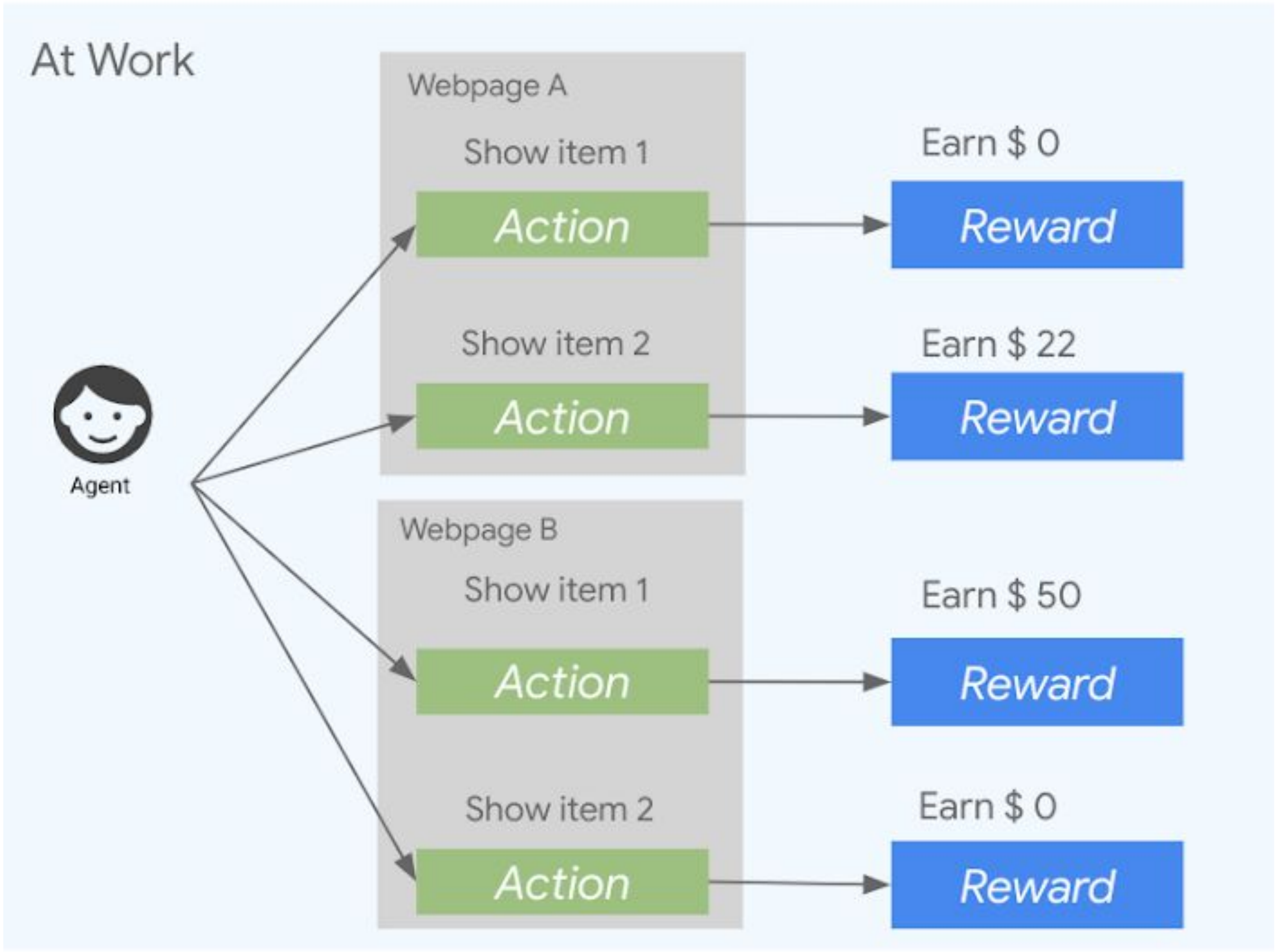
Discount factor γ determines how much we should care about the future!

Given Agent's policy π, RL objective become: $\mathrm{E}_{\pi} \sum^{\mathrm{T}}_{t=0} \gamma^t r_t$

Multi-Armed Bandits

MDP for Multi-Armed Bandits:

1. Only one state: $\pi(a \mid s) = \pi(a)$
2. Rewards are immediate
3. Rewards are stochastic



RL problems

- Local optimum

  Stuck in safe choices, missing the bigger rewards
- Delayed reward

  Actions now, consequences later
- Credit assignment problem

  Tracing rewards back to the right actions
- Exploration-exploitation trade-off

  Balancing discovery with reward

Delayed reward

- Agent makes a move at step 8
- At step 50 agent loses: R = -1
- Was it a good move?

Your data is not i.i.d. Previous actions affect future states and rewards.

Credit Assignment Problem:

How to determine which actions are responsible for the outcome?

Credit assignment problem

Goal: Train a bot to win the game!

Rewards:

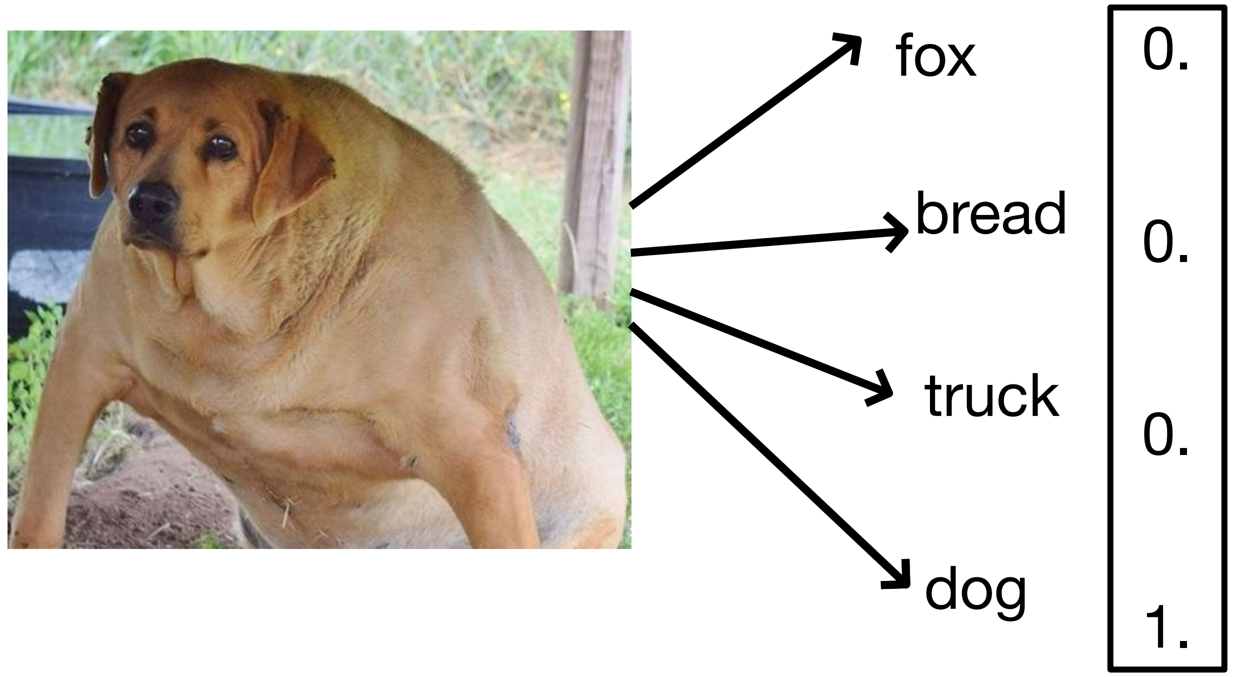
- +100 for the first place
- +5 for additional targets along the course

[Link]

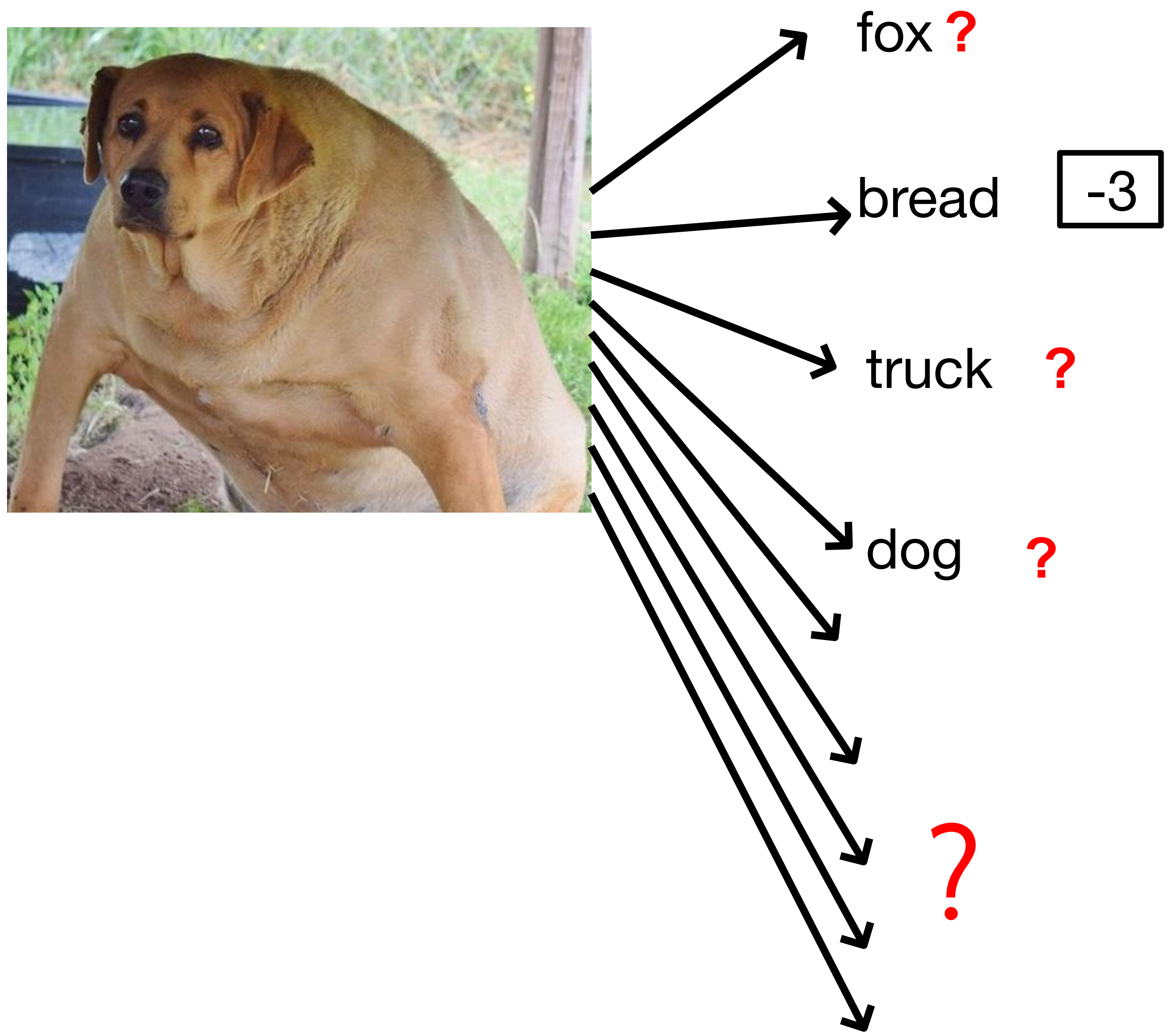
Reward is a proxy for you goal, but they are not the same!

Exploration-Exploitation Dilemma

We have ground truth labels



We have rewards



- Was that action optimal?
- Should you explore other actions?
- When you need to stop exploration?

Reward contains **less information** than a correct answer!

What we have discussed:

- What is RL?
- When do we need it?
- State, action, policy, reward, markovian property
- Main problems

Let’s estimate policies.

Basics

s ~ S; a ~ A - state/action spaces (can be infinite)

$p(s_{t+1} | s_t, a_t)$ - dynamics of transitions in the environment (Markovian)
$r(s, a)$ - reward for action a in state s (can be random or depends on other variables)
$\pi(a \mid s)$ - agent policy

now consider is known, but in practice - NO!

$p(\tau \mid \pi) = p(s_0) \prod^{\infty}_{t=0} \pi(a_t \mid s_t)\, p(s_{t+1} \mid a_t, s_t)$ - agent policy

where $\tau = (s_0, a_0, s_1, a_1, \dots)$ - agent trajectory

$R_t = \sum_{k=0}^{\infty} \gamma^k\, r(s_{t+k}, a_{t+k})$

$R_t = r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \dots$ - reward to go **or** return

Time-independence

$\tau_t = (s_t, a_t, s_{t+1}, a_{t+1}, \ldots)$ – trajectory from time t

Statement 1. $\forall t \ \ P(\tau_t \mid s_t = s) = P(\tau \mid s_0 = s)$

Statement 2. $\forall f \ \forall t \ \ E_{\tau\_t \mid s\_t = s} f(\tau_t) = E_{\tau \mid s\_0 = s} f(\tau)$

All properties of reward-to-go are defined by the start state s

Rate policy

How good is the policy π, if we start in state s?

$$V_t^{\pi}(s) = E_{\tau \sim \pi}\left[R_t \mid s_t = s\right] = V^{\pi}(s)$$

Policy $\forall s$: $\longrightarrow$

Terminal state

+1 for fish

V - value function:

γ^4	γ^3	γ^2	γ	1
s_0 s_4	s_1	s_2		s_3

Rate policy

How good is the policy π, if we start in state s?

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

Policy $\forall s$: $\longrightarrow$

Terminal state

+1 for fish

V - value function:

γ^4	γ^3	γ^2	γ	1

s_0 s_1 s_2 s_3 s_4

What if we "force" to choose the action a in s, and only then follow the policy π?

Q - value function:

$\longrightarrow$	γ^4	γ^3	γ^2	γ	1
$\longleftarrow$	γ^5	γ^5	γ^4	γ^3	1

s_0 s_1 s_2 s_3 s_4

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

In complex environments, it is inconvenient to count!

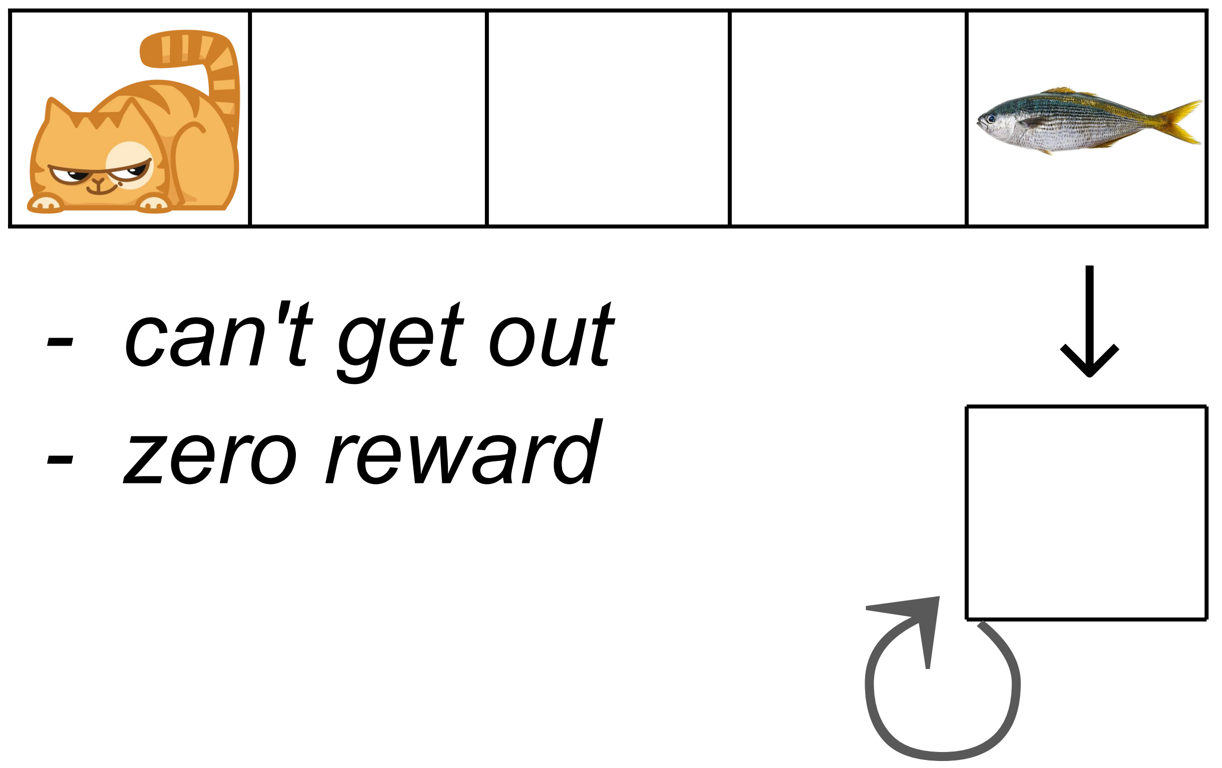
Finite and infinite time

In practice, the interaction between agent and environment can be completed in a finite number of steps

At each step, agent receives predicate *done(s)* $\in \{0, 1\}$, whether the state is terminal or not

If the agent reached terminal state, then we can reset the environment in s_0

That’s why we assume that $T = \infty$



Dynamic programming

Reformulation of a complex problem as a recursive sequence of simpler problems.

Get the recursive ratio for the cumulative reward R_t:

$$R_t = r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \dots = r(s_t, a_t) + \gamma R_{t+1}$$

<u>For V - function:</u>

$$V^{\pi}(s) = \mathrm{E}\,[R_t | s_t = s] = \mathrm{E}\,[r(s_t, a_t) + \gamma R_{t+1} | s_t = s] = \mathrm{E}_{a \sim \pi(\cdot|s)}\,[r(s, a) + \gamma \mathrm{E}_{s' \sim p(s'|s, a)}\,\mathrm{E}[R_{t+1} | s_{t+1} = s']] =$$

$$\mathrm{E}_{a \sim \pi(\cdot|s)}\,[r(s, a) + \gamma \mathrm{E}_{s' \sim p(s'|s, a)}\,V^{\pi}(s')]$$

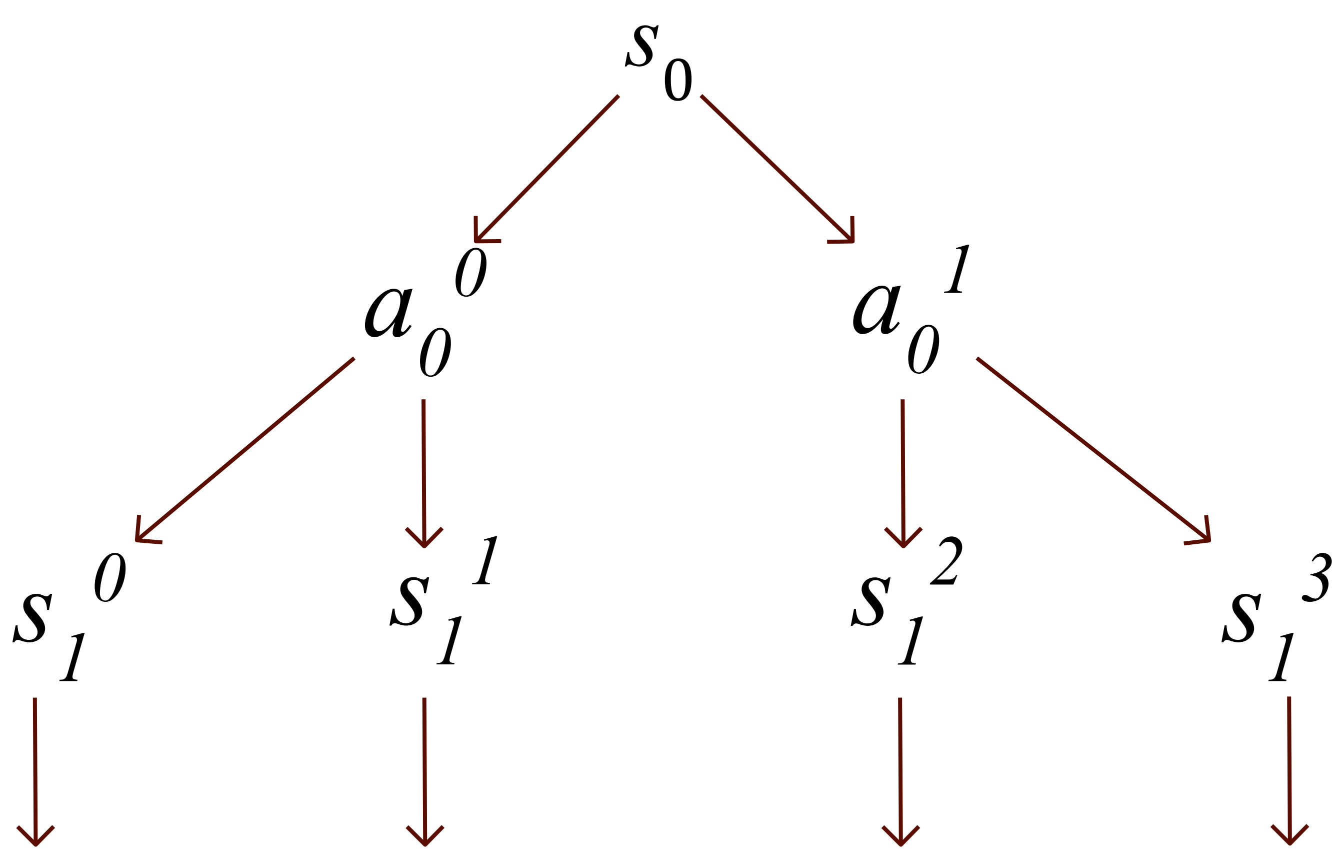
<u>For Q - function:</u>

$$Q^{\pi}(s, a) = r(s, a) + \gamma\,\mathrm{E}_{s' \sim p(\cdot|s,a)}\,\mathrm{E}_{a' \sim \pi(\cdot|s')}\,Q^{\pi}(s', a')$$

Dynamic programming

If states never repeat in the environment, the graph of this MDP will be a tree

$$V^{\pi}(s_T) = E_{a\sim\pi(\cdot\,|s\_T)}\, r(s_T, a)$$



$$V^{\pi}(s) = E_{a\sim\pi(\cdot\,|s)}\,[r(s, a) + \gamma E_{s'\sim p(s'\,|s,a)}\, V^{\pi}(s')]$$

Bellman's Equations tell you how to calculate value "backwards".

Relationship of Q and V functions

Expressing V in terms of Q:

$$V^{\pi}(s) = \mathrm{E}_{a \sim \pi(\cdot|s)} Q^{\pi}(s, a)$$

V - this is Q, in which the action from the policy was substituted

Expressing Q in terms of V:

$$Q^{\pi}(s, a) = r(s, a) + \mathrm{E}_{s' \sim p(\cdot|s, a)} V^{\pi}(s')$$

Q - is the instant reward for (s, a) plus future state value

How to solve the Bellman equation?

$V^{\pi}(s) = \mathrm{E}_{a \sim \pi(\cdot|s)} [r(s, a) + \gamma \mathrm{E}_{s' \sim p(s'|s,a)} V^{\pi}(s')] =$

$= \mathrm{E}_{a \sim \pi(\cdot|s)} r(s, a) + \gamma \mathrm{E}_{s' \sim p(s'|s)} V^{\pi}(s') = u(s) + \gamma \mathrm{E}_{s' \sim p(s'|s)} V^{\pi}(s')$

Everything is linear with respect to V

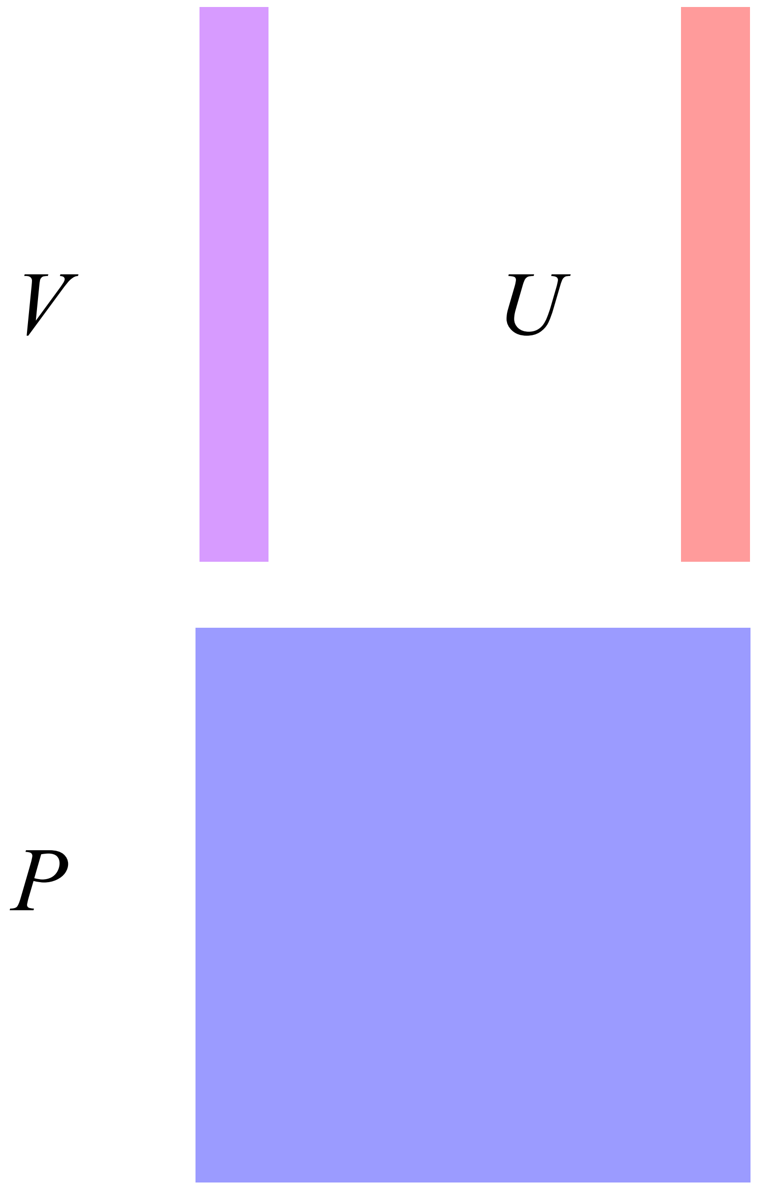
$$V = U + \gamma P V$$

$$(I - \gamma P) V = U$$

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

It will be expensive!

Without taking into account $|A|$ - already $O(|S|^3)$



How to solve the Bellman equation?

Simple iteration method:

$$V_{new} = F(V_{old})$$

$$F(V_s) = \mathrm{E}_{a \sim \pi(\cdot \mid s)} [r(s, a) + \gamma \mathrm{E}_{s' \sim p(s' \mid s, a)} V_{s'}]$$

Will the algorithm converge? *It will be if the mapping F is contractive*

Is the Bellman operator a contraction?

By the infinite norm:

reminder: $||x||_\infty = max_i |x_i|$

$$||F(V) - F(W)||_\infty = ||U + \gamma PV - U - \gamma PW||_\infty =$$

$$= ||\gamma P(V - W)||_\infty \leq \gamma ||P||_\infty ||V - W||_\infty$$

where is the matrix norm:

$$||P||_\infty = \max_{x:||x||_\infty=1} ||Px||_\infty = \max_{x:||x||_\infty=1} \max_i |\textstyle\sum_j P_{ij}x_j|$$

$$= \max_i |\textstyle\sum_j P_{ij}| = 1$$

$x_j = \text{sign}(P_{ij})$

Q.E.D.: $||F(V) - F(W)||_\infty \leq \gamma ||V - W||_\infty$

Algorithm Policy Evaluation

- Initialize $V(s)\ \forall s$
- Repeat:
  - $\Delta = 0$
  - For all s:
    - $v = V(s)$
    - $V(s) = \mathrm{E}_{a \sim \pi(\cdot|s)}[r(s, a) + \gamma \mathrm{E}_{s' \sim p(\cdot|s, a)} V$
    - $(s')]\ \ \Delta = \max(\Delta, |v - V(s)|)$

while $\Delta > \epsilon$

Policy improvement

Optimal Bellman Equations

Def: π^* - optimal policy $\Leftrightarrow$ $\forall \pi$ $\forall s \in S$
$V^{\pi^*}(s) \geq V^{\pi}(s)$

Def: $V^*(s) = \max_\pi V^\pi(s, a)$
$Q^*(s, a) = \max_\pi Q^\pi(s, a)$

Bellman's optimality principle:

A greedy choice of action under the assumption of further optimality is optimal

Th:
π - optimal $\Leftrightarrow$ $\forall s, a$: $\pi(a \mid s) > 0$
$a \in \arg\max_{a'} Q^\pi(s, a')$

Optimal Bellman equations:

$$V^*(s) = \max_a [r(s, a) + \gamma \, E_{s'} V^*(s')]$$

$$Q^*(s, a) = r(s, a) + \gamma \, E_{s'} \max_{a'} Q^*(s', a')$$

Expressing V^* in terms of Q^*:

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

Expressing Q^* in terms of V^*:

$$Q^*(s, a) = r(s, a) + \gamma \, E_{s'} V^*(s')$$

Policy improvement

Def: $\pi' \geq \pi$ if $V^{\pi'}(s) \geq V^{\pi}(s)$ $\forall s \in S$

Our Policy Update Strategy:

- let $\exists$ s be such that: $\exists a : Q^{\pi}(s, a) > V^{\pi}(s)$
- then $\pi'(s) := a$. In all $s^{\sim} \,!= s$ define $\pi'(s^{\sim}) = \pi(s^{\sim})$
- note that $\forall s \in S$ $V^{\pi'}(s) \leq Q^{\pi}(s, \pi'(s))$

In this case, $\pi' \geq \pi$

Policy improvement

Given: $\forall s \in S \;\; V^{\pi'}(s) \leq Q^{\pi}(s, \pi'(s))$

Prove: $\forall s \in S \;\; V^{\pi}(s) \leq V^{\pi'}(s)$

Proof: $V^{\pi}(s) \leq Q^{\pi}(s, \pi'(s)) = r(s, \pi'(s)) + \gamma\, E_{s'} V^{\pi}(s') \leq r(s, \pi'(s)) + \gamma\, E_{s'} Q^{\pi}(s', \pi'(s'))$

$\leq r(s, \pi'(s)) + \gamma\, r(s', \pi'(s')) + \gamma^2 E_{s''} V^{\pi}(s'') \leq \ldots \leq V^{\pi'}(s)$

Algorithm Policy Iteration

- **Initialize** $V(s), \pi(s) \quad \forall s$
- estimate V for policy π by method PE
- stop = $True$
- **For all** s:
  - $a = \pi(s)$
  - $\pi(s) = \arg\max_a [r(s, a) + \mathrm{E}_{s'} V(s')]$
  - **if** a != $\pi(s)$:
    - stop = $False$
- **if not** stop

Algorithm Value Iteration

- **Initialize** $V(s)\ \forall s$
- **Repeat:**
  - $\Delta = 0$
  - **For all** s:
    - $v = V(s)$
    - $V(s) = \max_a [r(s, a) + \gamma \mathrm{E}_{s' \sim p(\cdot \mid s, a)} V(s')]$
    - $\Delta = \max(\Delta, |v - V(s)|)$
- **while** $\Delta > \epsilon$

Recap

- What is RL?
- When do we need it?
- State, action, policy, reward, markovian property, MDP
- Main problems
- V-function
- Q-function
- Value Iteration