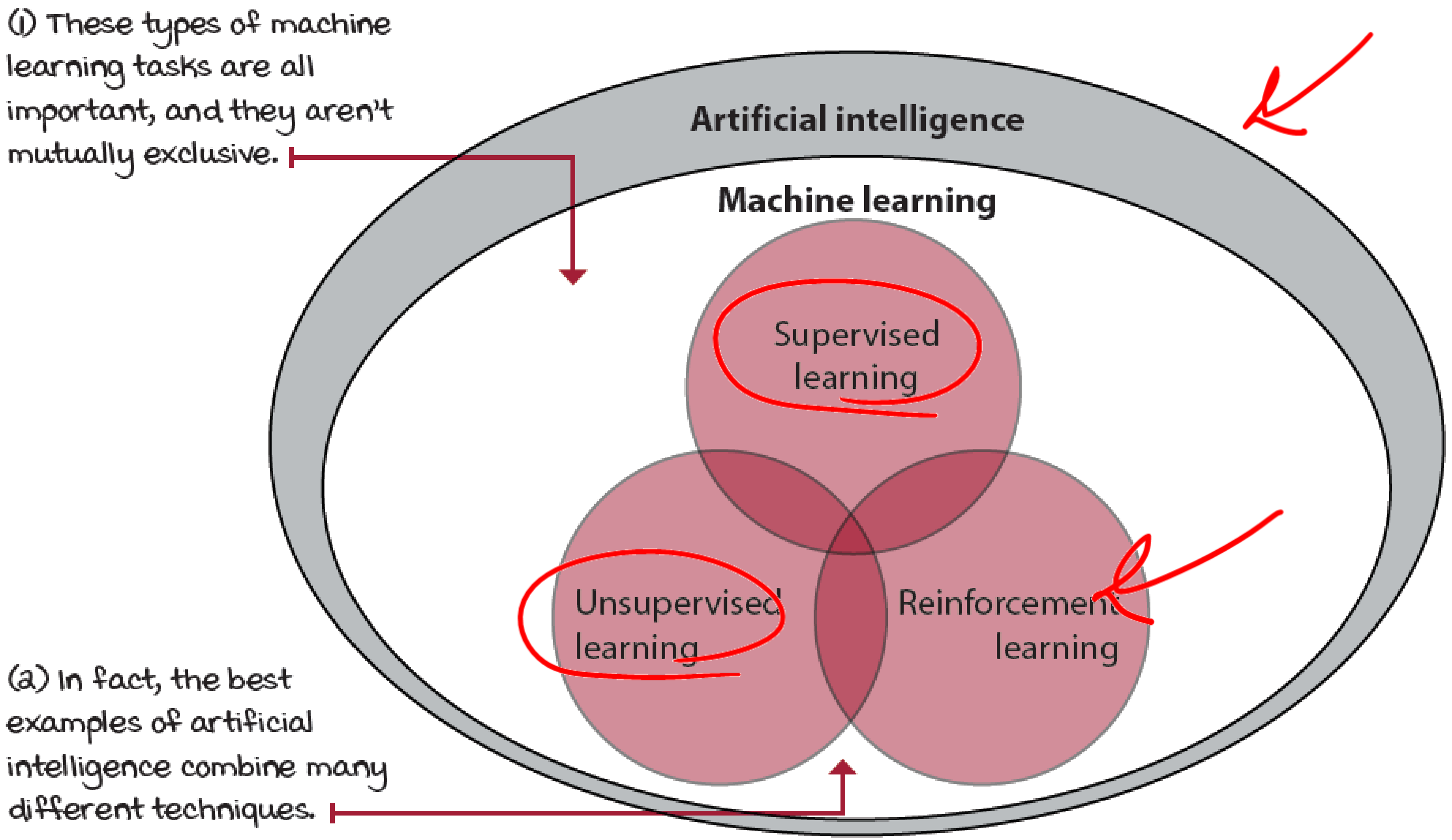
Reinforcement Learning Framework

Main branches of machine learning



MAIN BRANCHES OF MACHINE LEARNING

$y = f(x)$

Supervised learning (SL) is the task of learning from labeled data. In SL, a human decides which data to collect and how to label it. The goal in SL is to generalize.

Unsupervised learning (UL) is the task of learning from unlabeled data. Even though data no longer needs labeling, the methods used by the computer to gather data still need to be designed by a human. The goal in UL is to compress.

$f(x)$

Reinforcement learning (RL) is the task of learning through trial and error. In this type of task, no human labels data, and no human collects or explicitly designs the collection of data. The goal in RL is to act.

$y = f(x)$

z

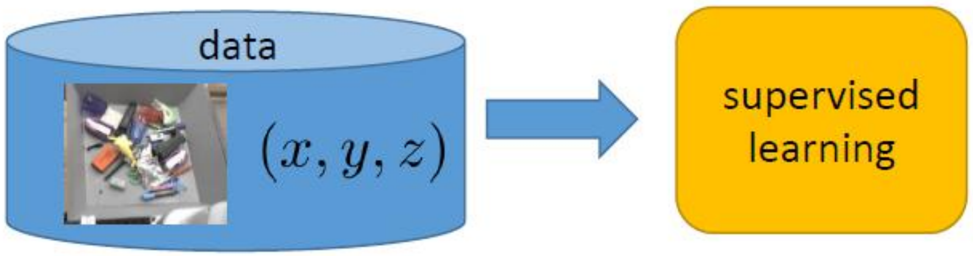
Standard (supervised) machine learning:

given $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$

learn to predict y from $\mathbf{x}$ $\quad\quad f(\mathbf{x}) \approx y$

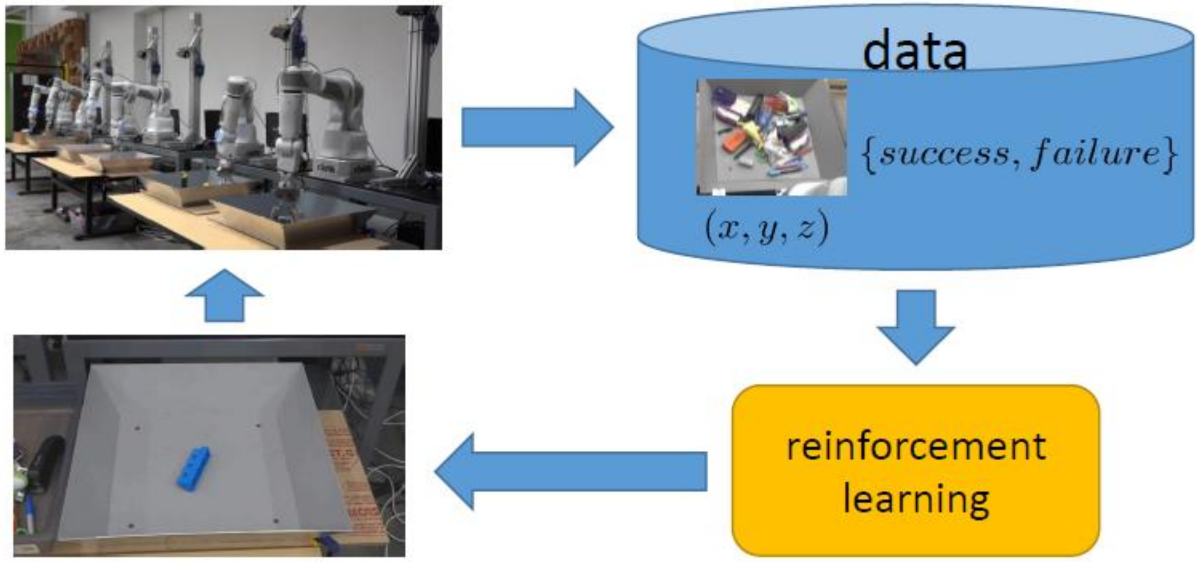
Usually assumes:

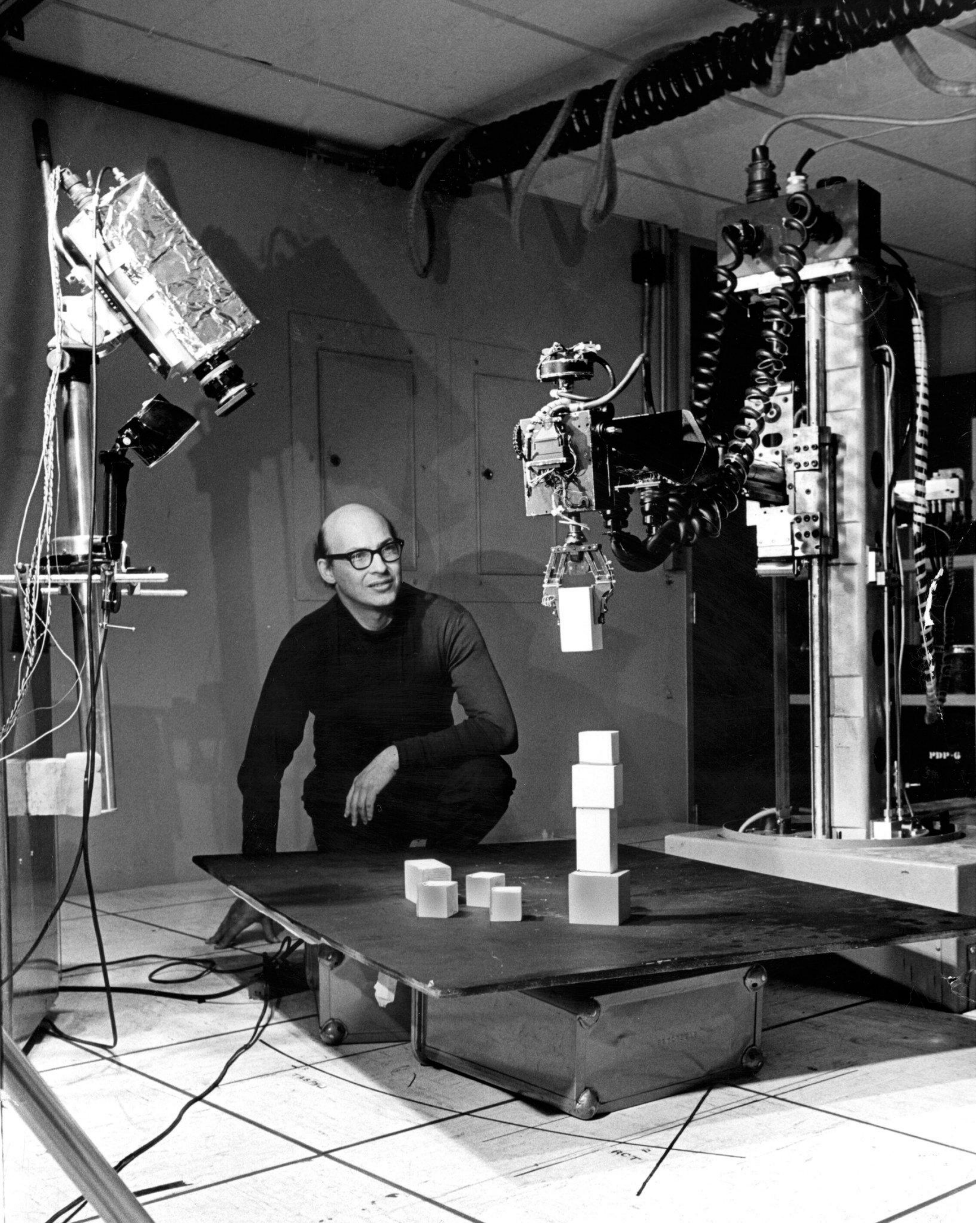
- i.i.d. data
- known ground truth outputs in training



Reinforcement learning:

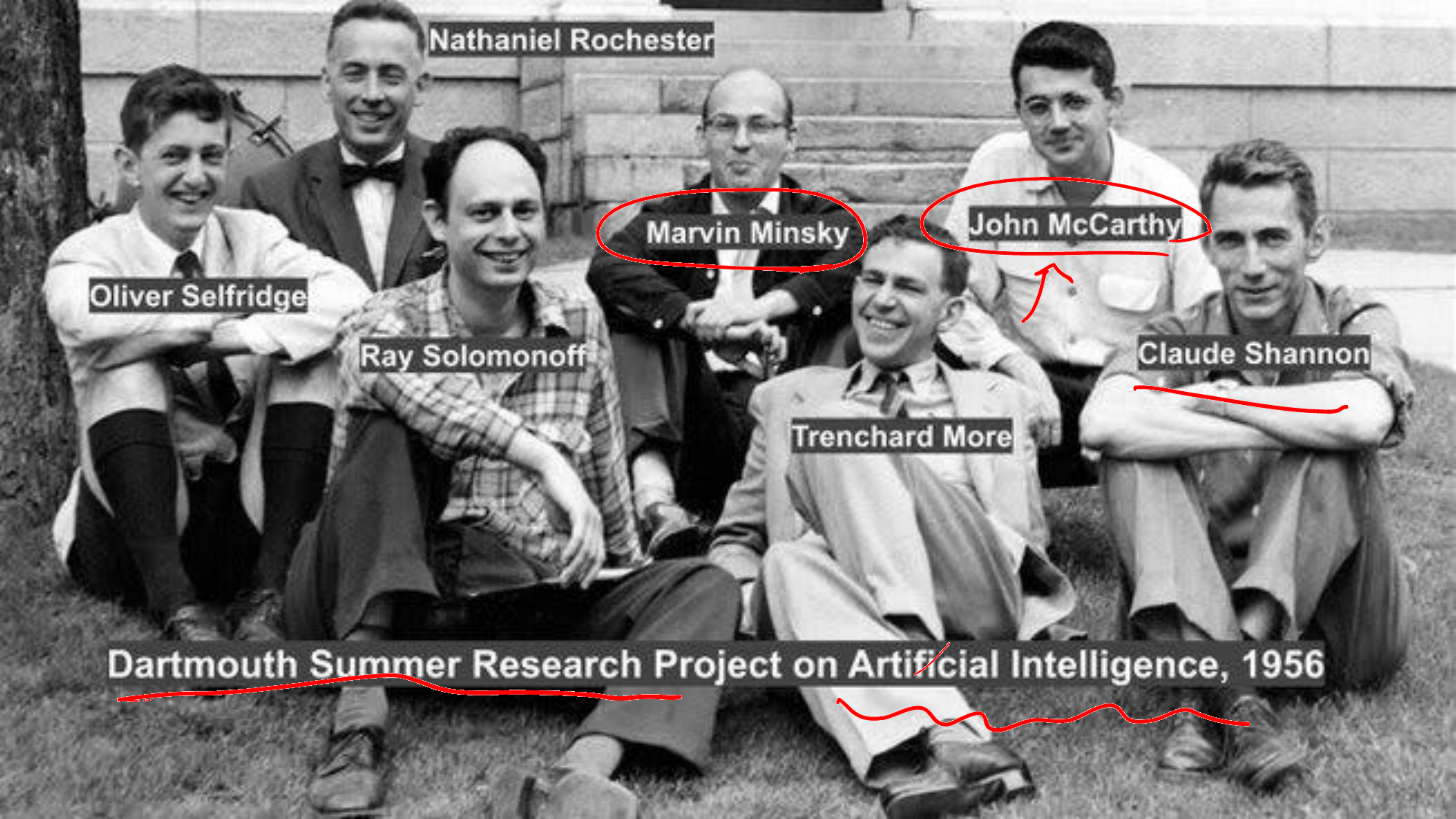
- Data is **not** i.i.d.: previous outputs influence future inputs!
- Ground truth answer is not known, only know if we succeeded or failed
  - more generally, we know the reward



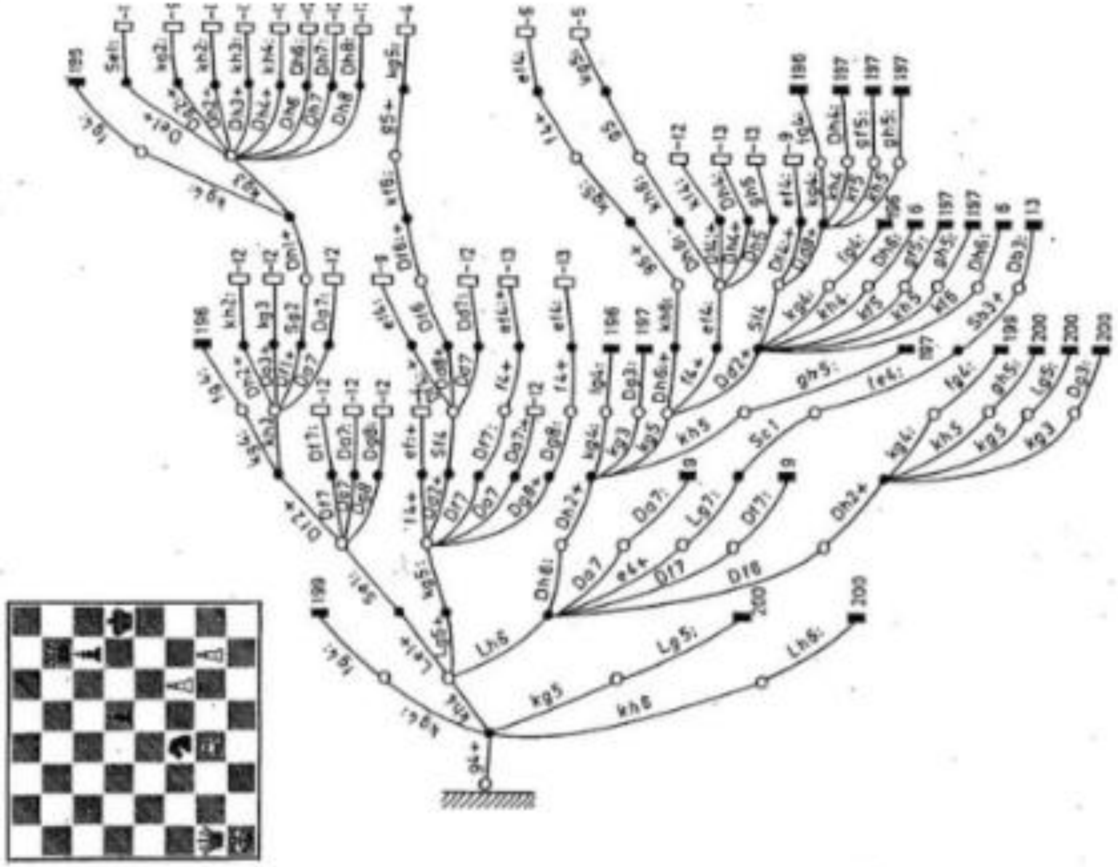


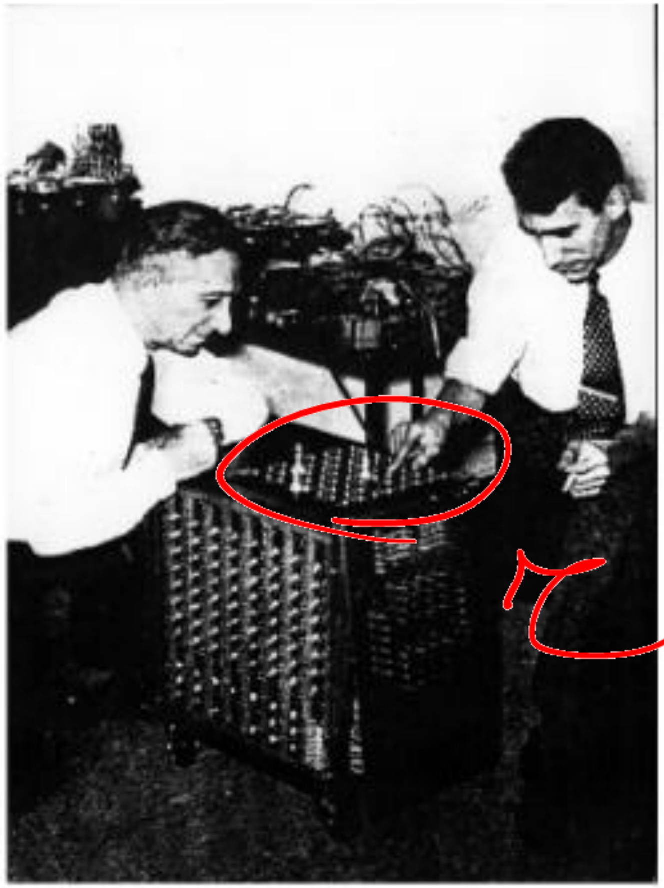
“Almost all young people working on Artificial Intelligence look around and say - What's popular? Statistical learning. So, I'll do that. That's exactly the way to kill yourself scientifically!”

Marvin Minsky during his course called Society of Mind at MIT in 2011

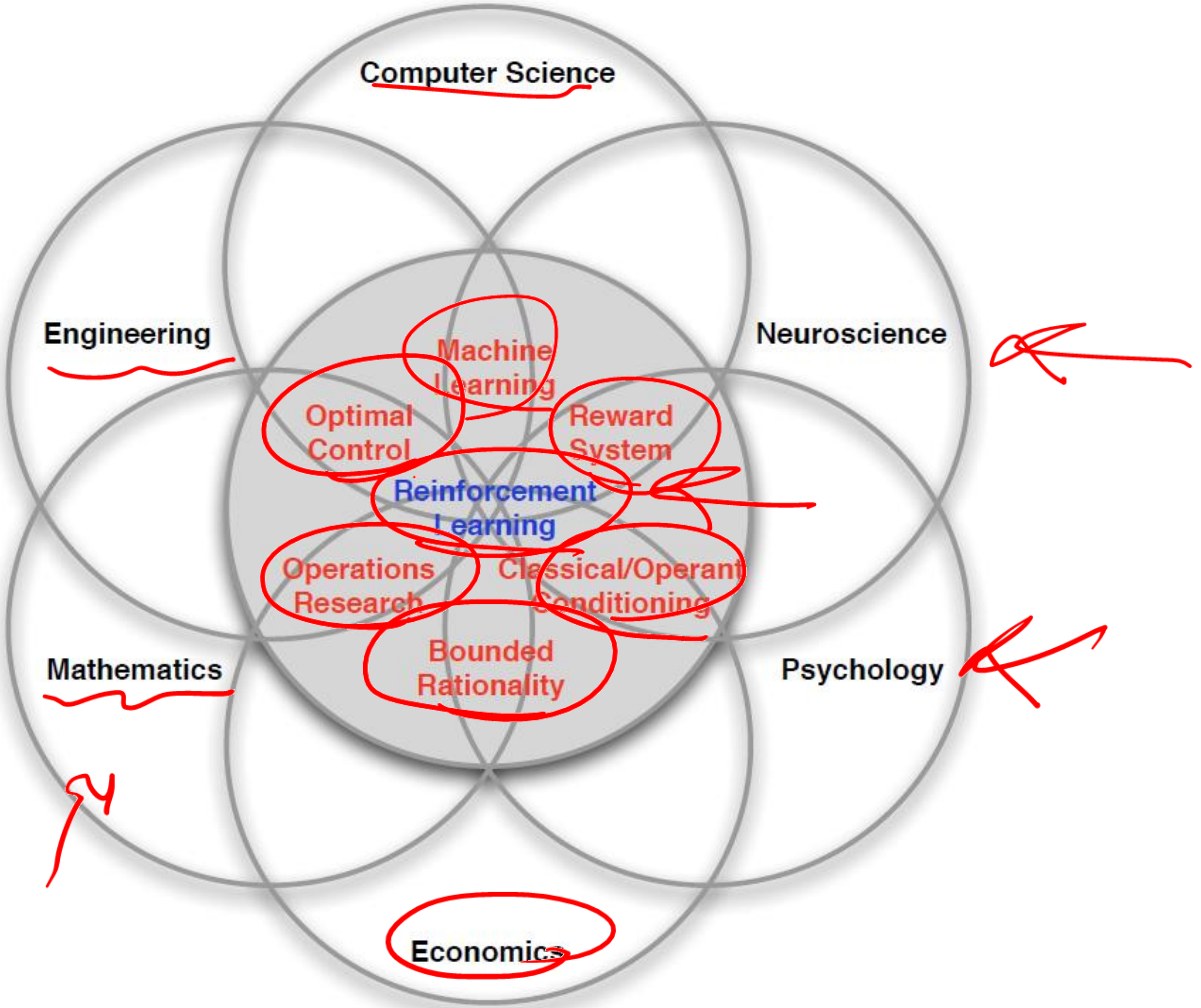

Nathaniel Rochester
Marvin Minsky
John McCarthy
Oliver Selfridge
Ray Solomonoff
Claude Shannon
Trenchard More
Dartmouth Summer Research Project on Artificial Intelligence, 1956

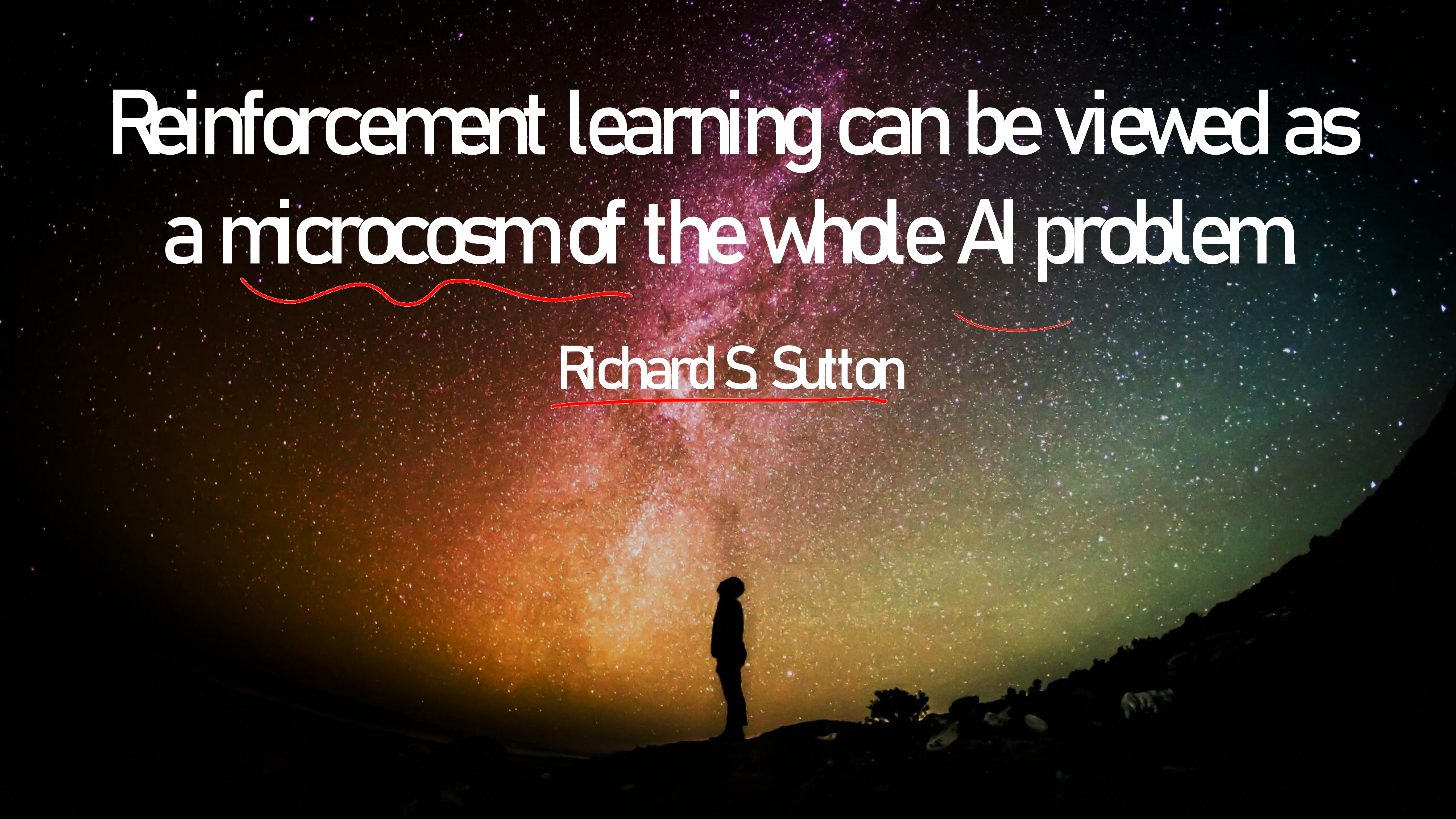
https://www.technologyreview.com/2018/12/19/138508/mighty-mouse





https://www.chessprogramming.org/Claude_Shannon


Computer Science
Engineering
Neuroscience
Machine Learning
Optimal Control
Reward System
Reinforcement Learning
Operations Research
Classical/Operant Conditioning
Bounded Rationality
Mathematics
Psychology
Economics


Reinforcement learning can be viewed as a microcosm of the whole AI problem
Richard S. Sutton

Dataste

RL

	AI Planning	SL	UL	RL	IL
Optimization	X			X	X
Learns from experience		X	X	X	X
Generalization	X	X	X	X	X
Delayed Consequences	X			X	X
Exploration				X	

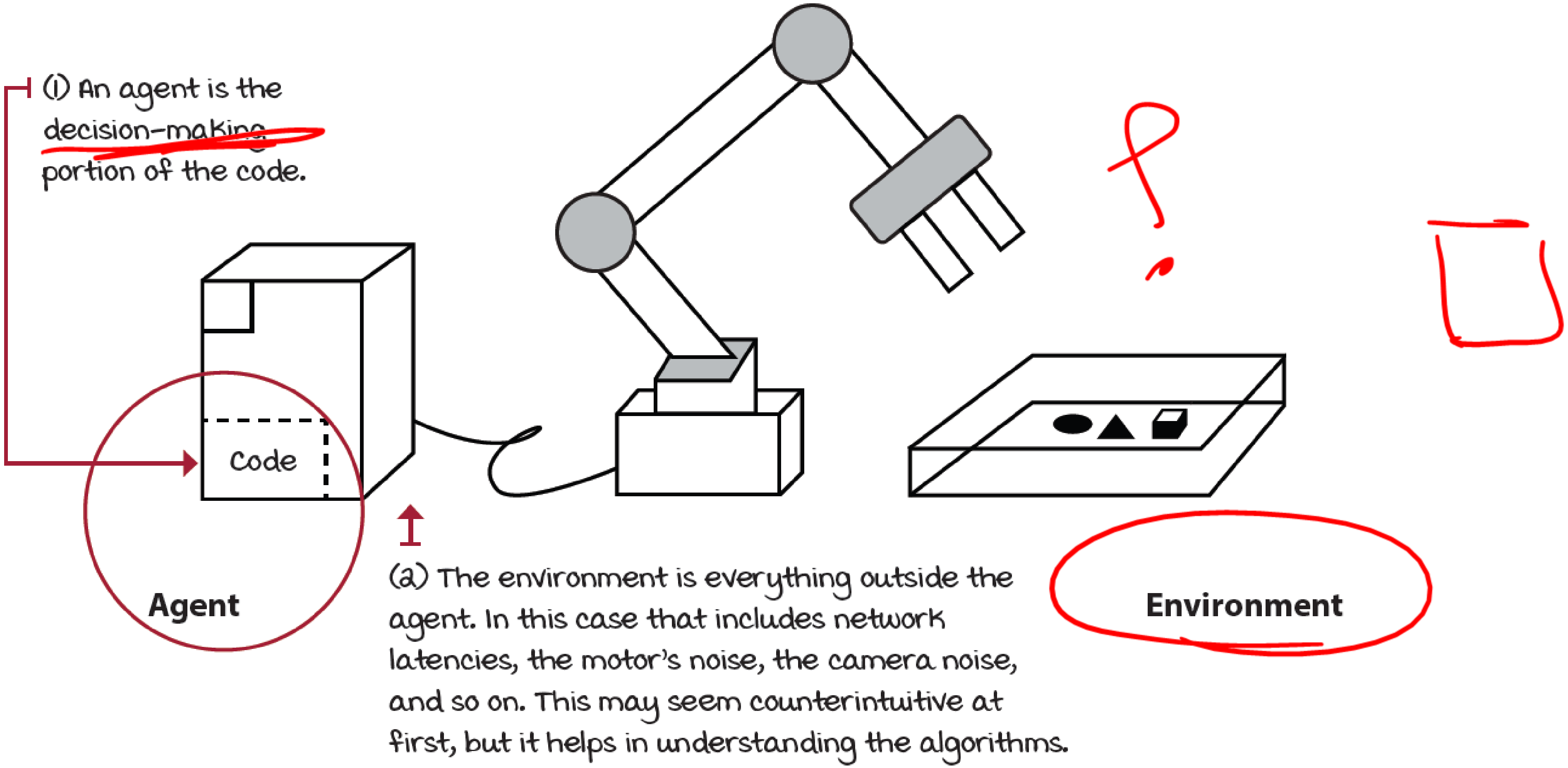
y x

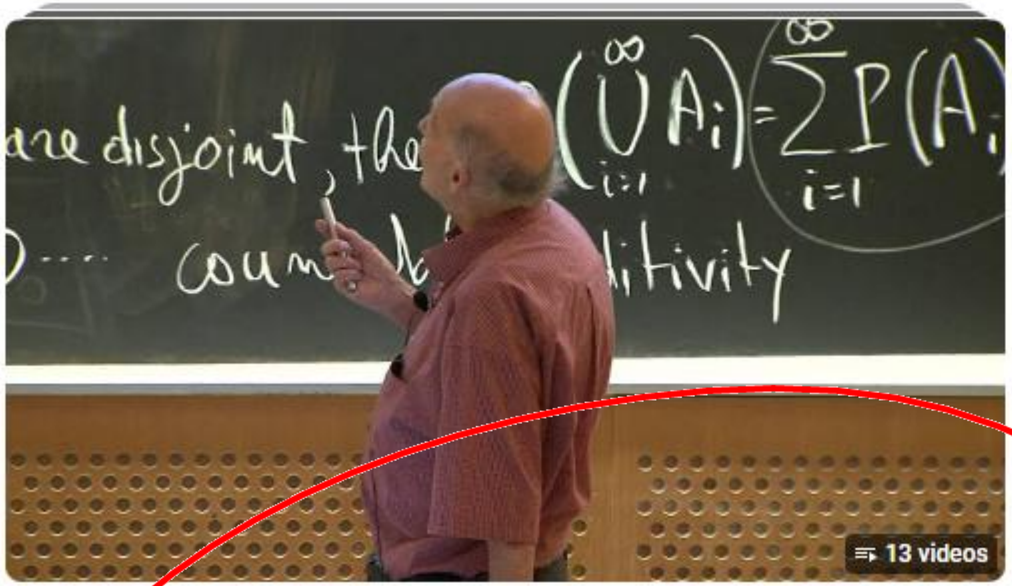
x_1 … x_n


Agent
state
S_t
reward
R_t
action
A_t
R_{t+1}
S_{t+1}
Environment

How should we define the boundary between agent and environment?

ENVIRONMENT AND AGENT





MIT 6.868J The Society of Mind, Fall 2011

MIT OpenCourseWare · Playlist

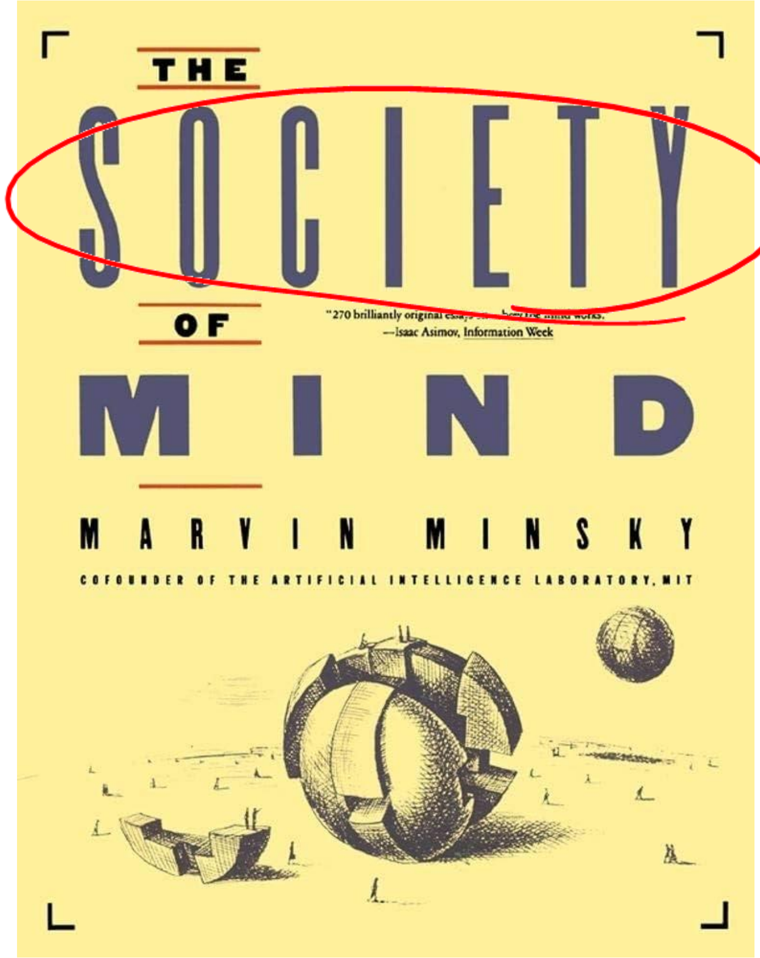
1. Introduction to 'The Society of Mind' · 2:05:54

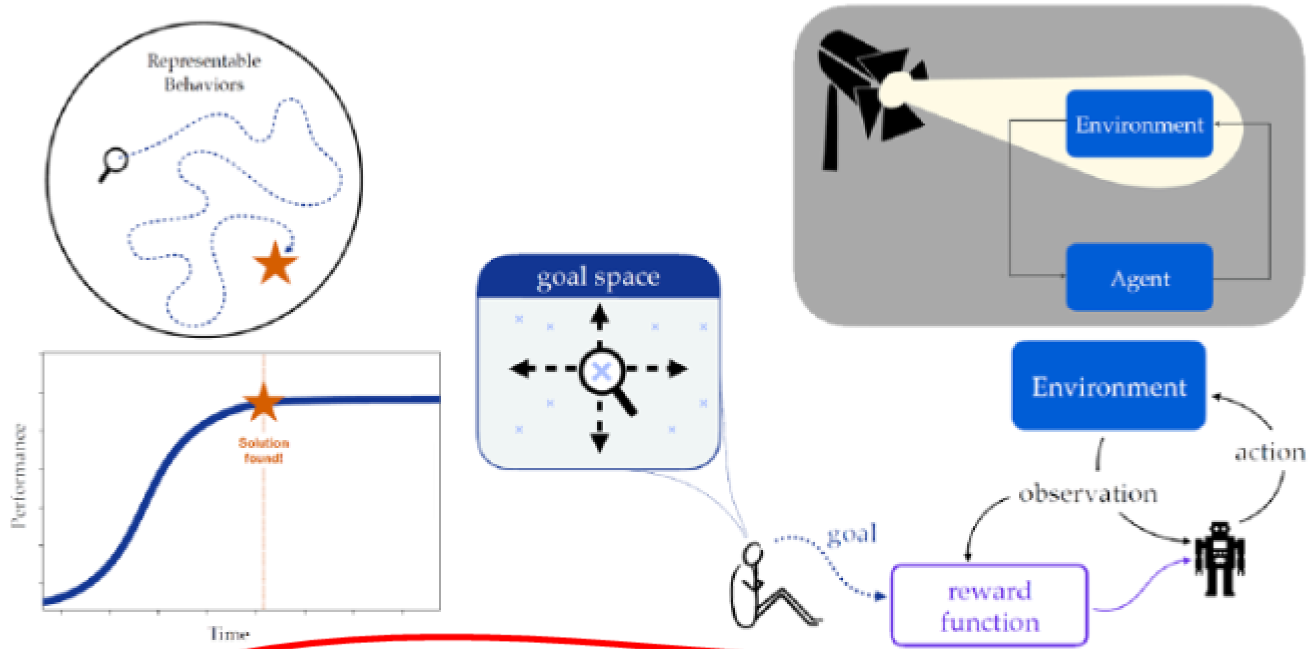
2. Falling In Love · 1:45:55

View full playlist

https://www.youtube.com/playlist?list=PLUl4u3cNGP61E-vNcDV0w5xpsIBYNJDkU

1997





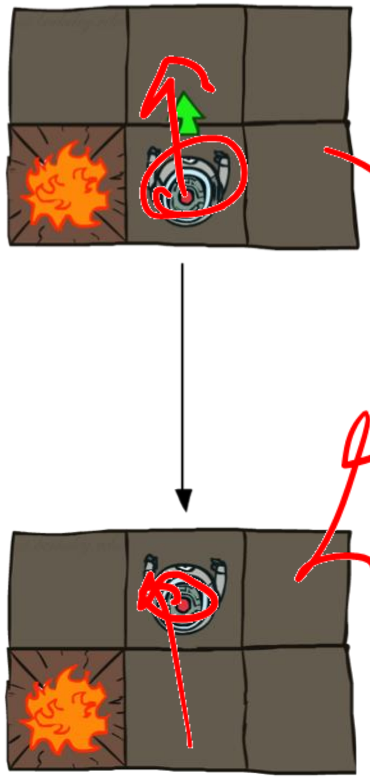
Three Dogmas of Reinforcement Learning

August 6, 2024 · Arash Alikhani

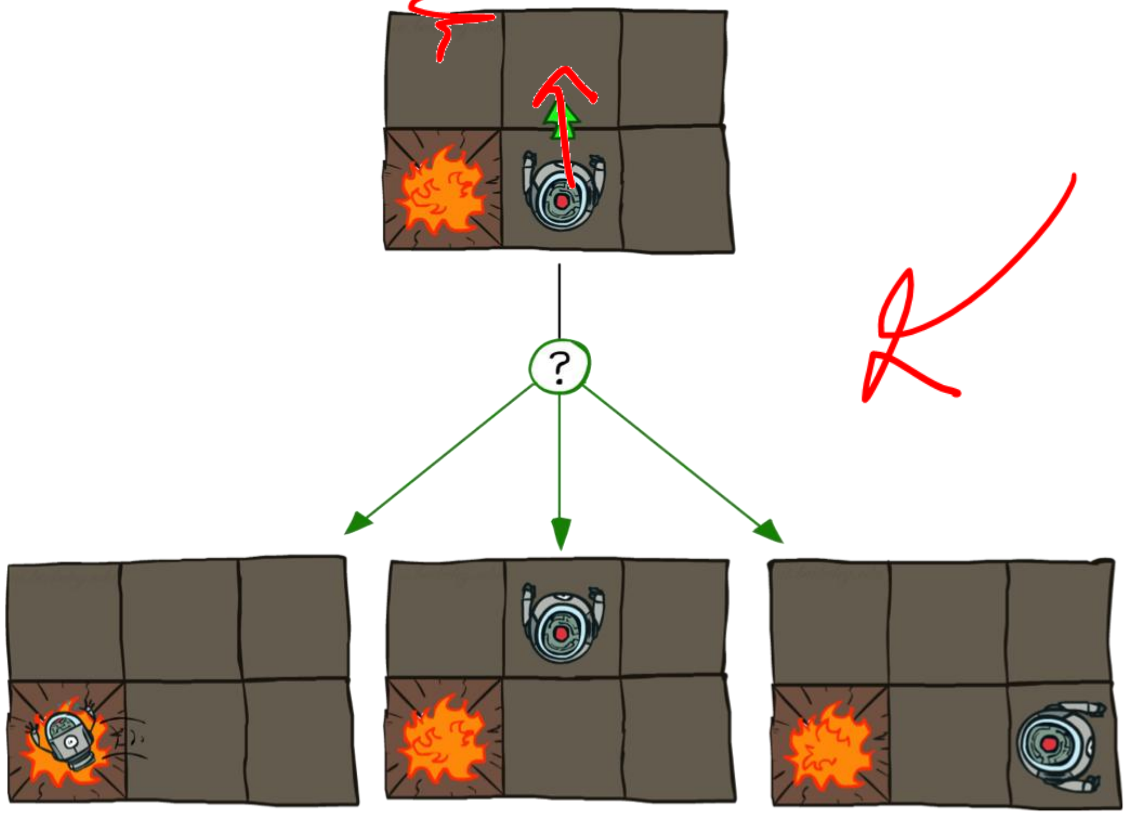
https://rljclub.github.io/posts/three-dogmas-of-reinforcement-learning

ENVIRONMENT AND AGENT

Deterministic Grid World

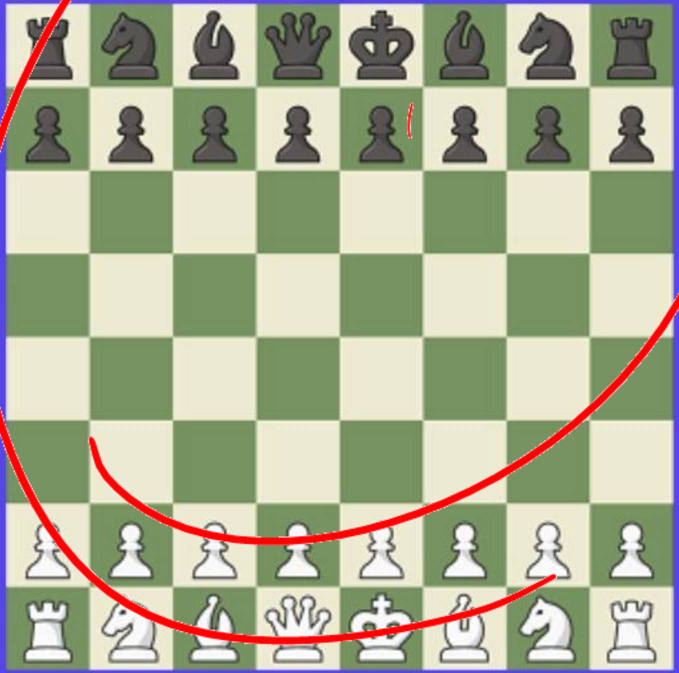


Stochastic Grid World



Observation Space

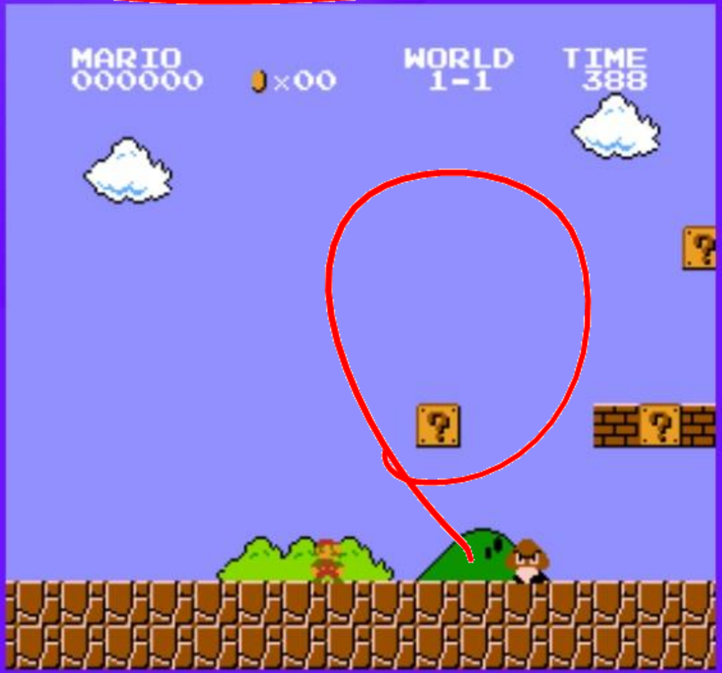
State: **complete description** of the state of the world (no hidden information).



Observation: **partial description** of the state of the world.

Action Space

Discrete: **finite number of possible actions**



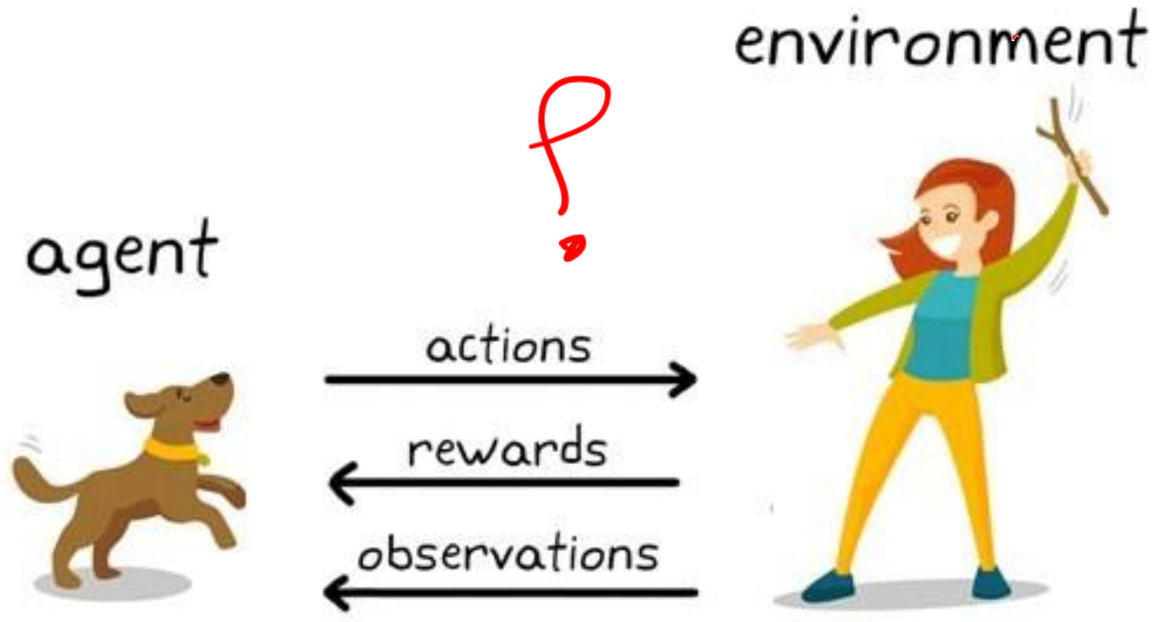
Continuous: **infinite number of possible actions**

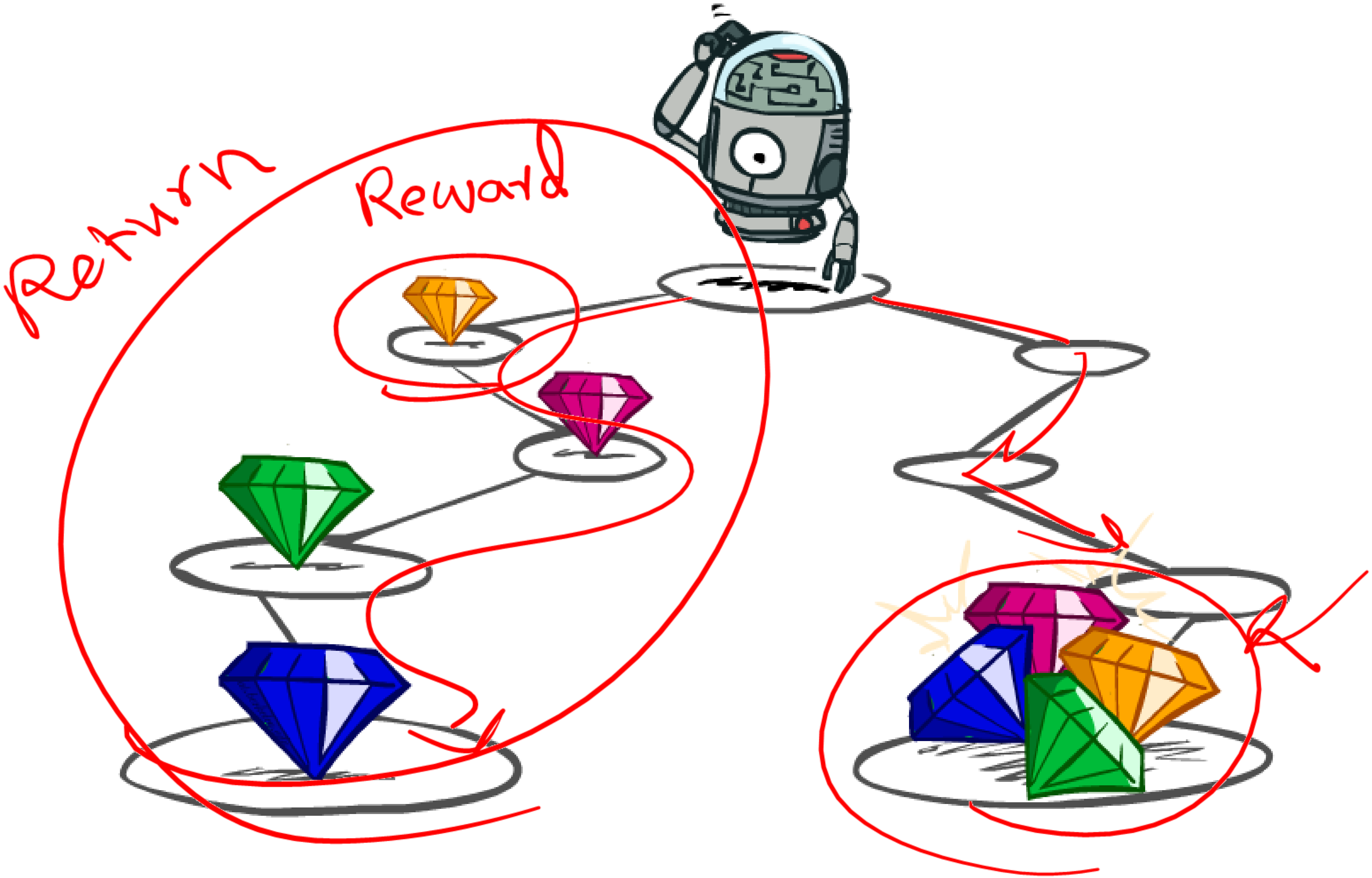
REWARD HYPOTHESIS

The Reward Hypothesis

"...all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward)"

-- Sutton (2004)




Return
Reward

$$R(\tau) = \sum_{k=0}^{\infty} r_{t+k+1}$$

trajectry

$$R(\tau) = r_{t+1} + r_{t+2} + r_{t+3} + r_{t+4} + \dots$$

Return: cumulative reward

Trajectory (read Tau)
Sequence of states and actions

1

Worth Now

γ

Worth Next Step

γ^2

Worth In Two Steps

$$R(\boxed{\tau}) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \dots$$

Return: cumulative reward

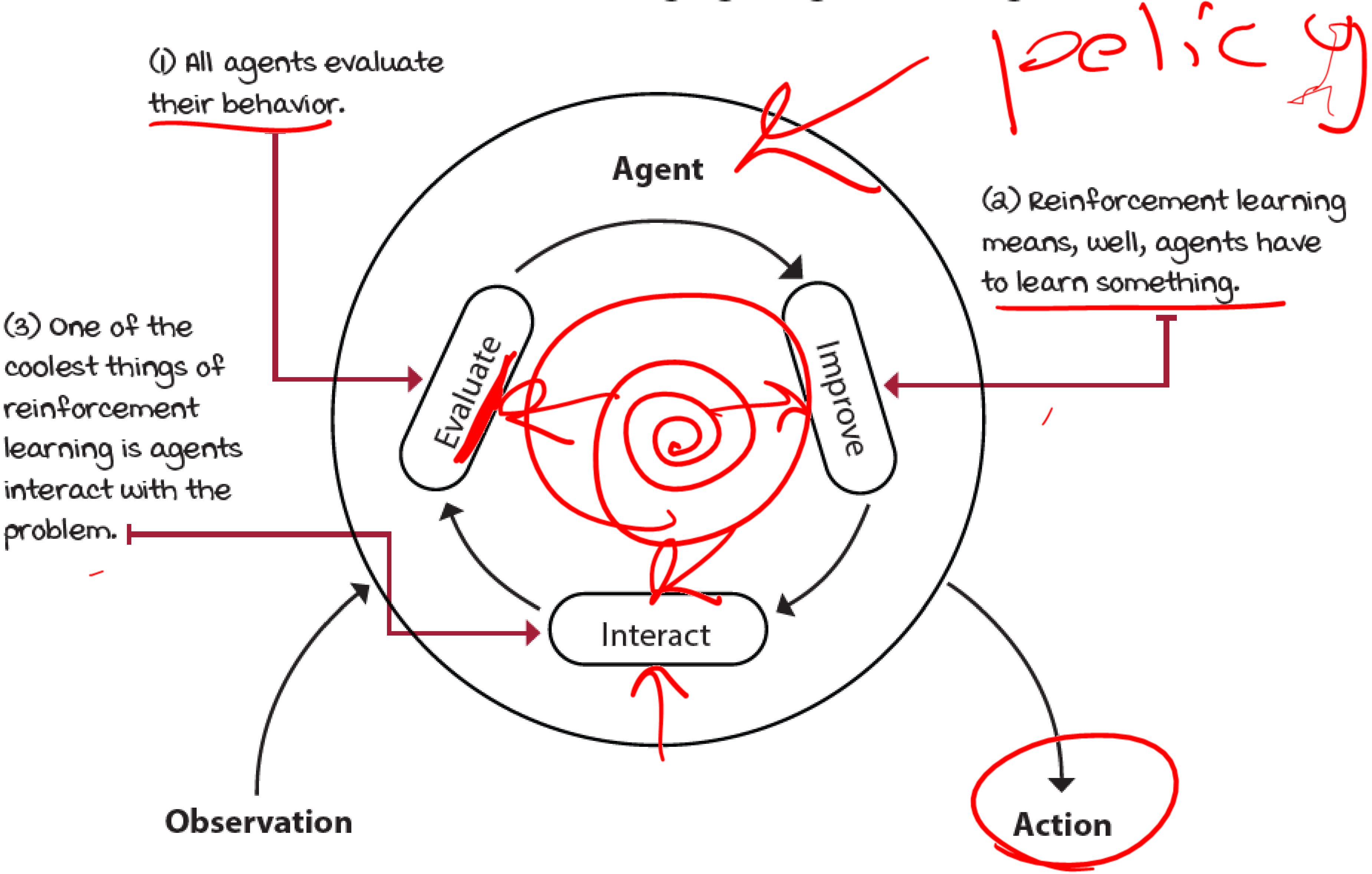
Gamma: discount rate

Trajectory (read Tau)
Sequence of states and actions

$$R(\tau) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

$\gamma < 1$

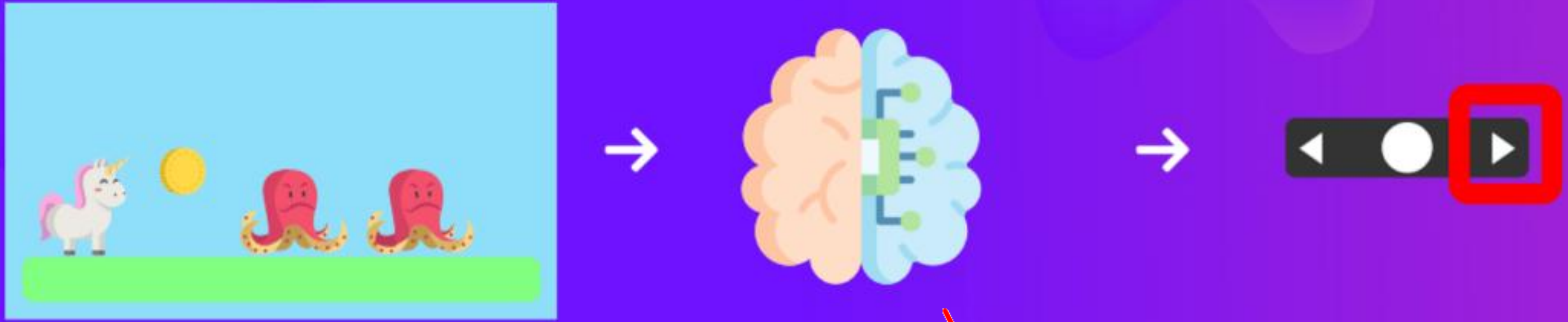
The three internal steps that every reinforcement learning agent goes through



The Policy π: the agent's brain

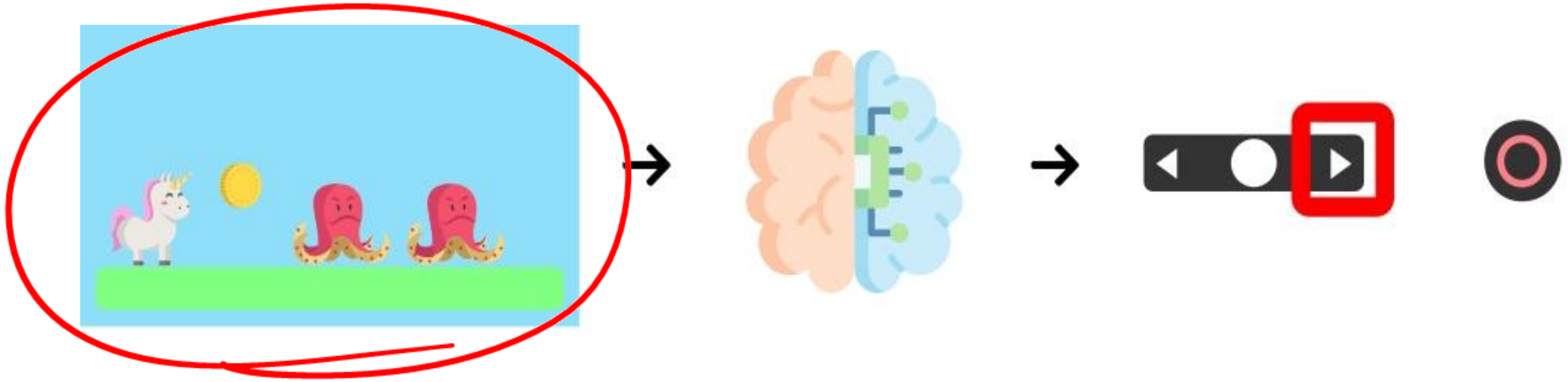
Policy π: is the **brain of our Agent,** it's the function that tell us what **action to take given the state we are.**

→ So it **defines the agent behavior at a given time.**



State → π(State) → Action

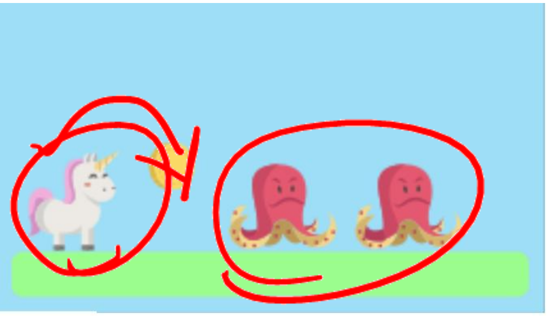
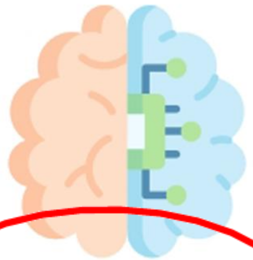
$$a = \pi(s)$$



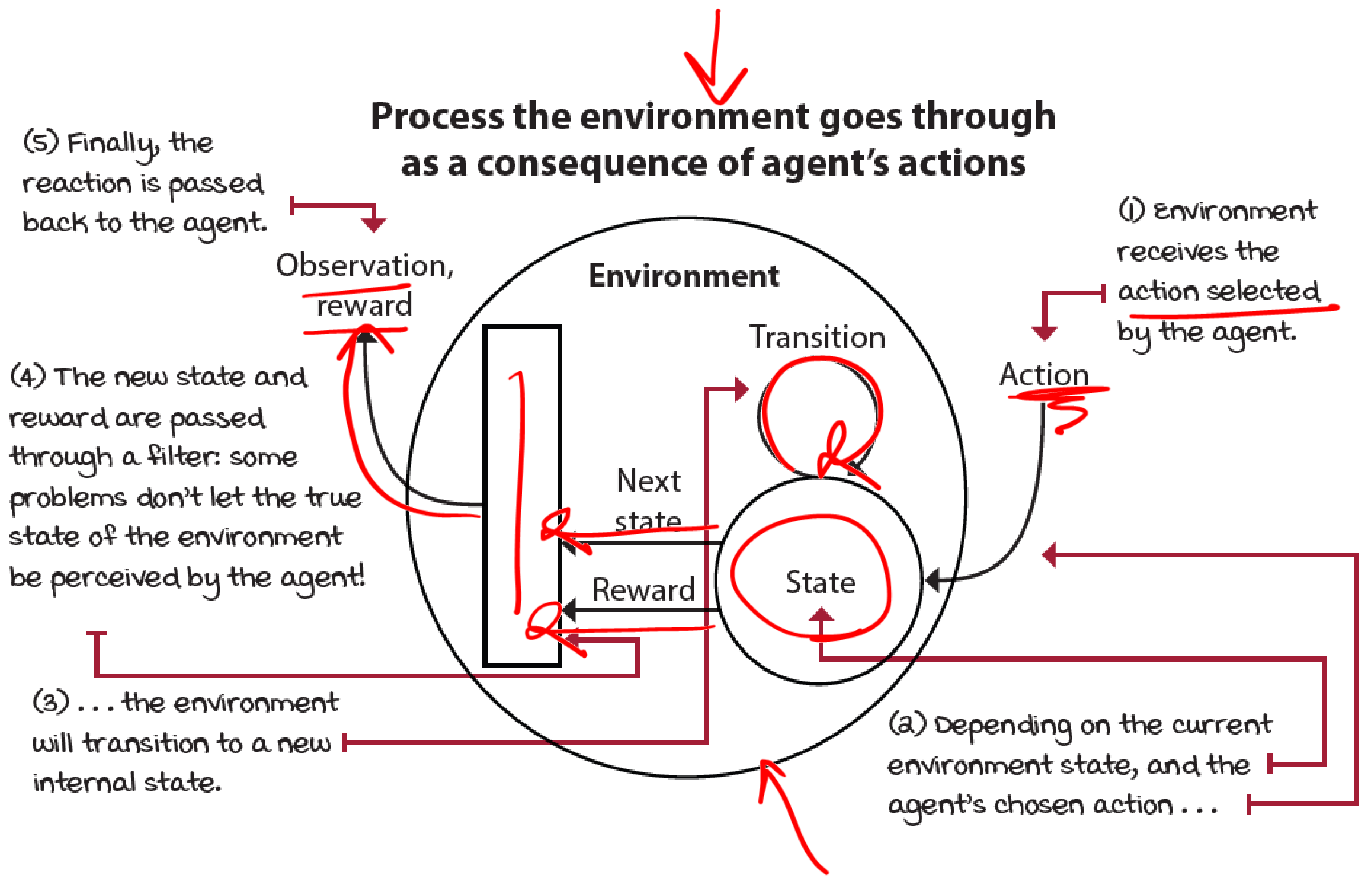
State s_0 → $\pi(s_0)$ → a_0 = Right

$$\pi(a|s) = \boxed{P[A|s]}$$

Probability Distribution over the set of actions given the state

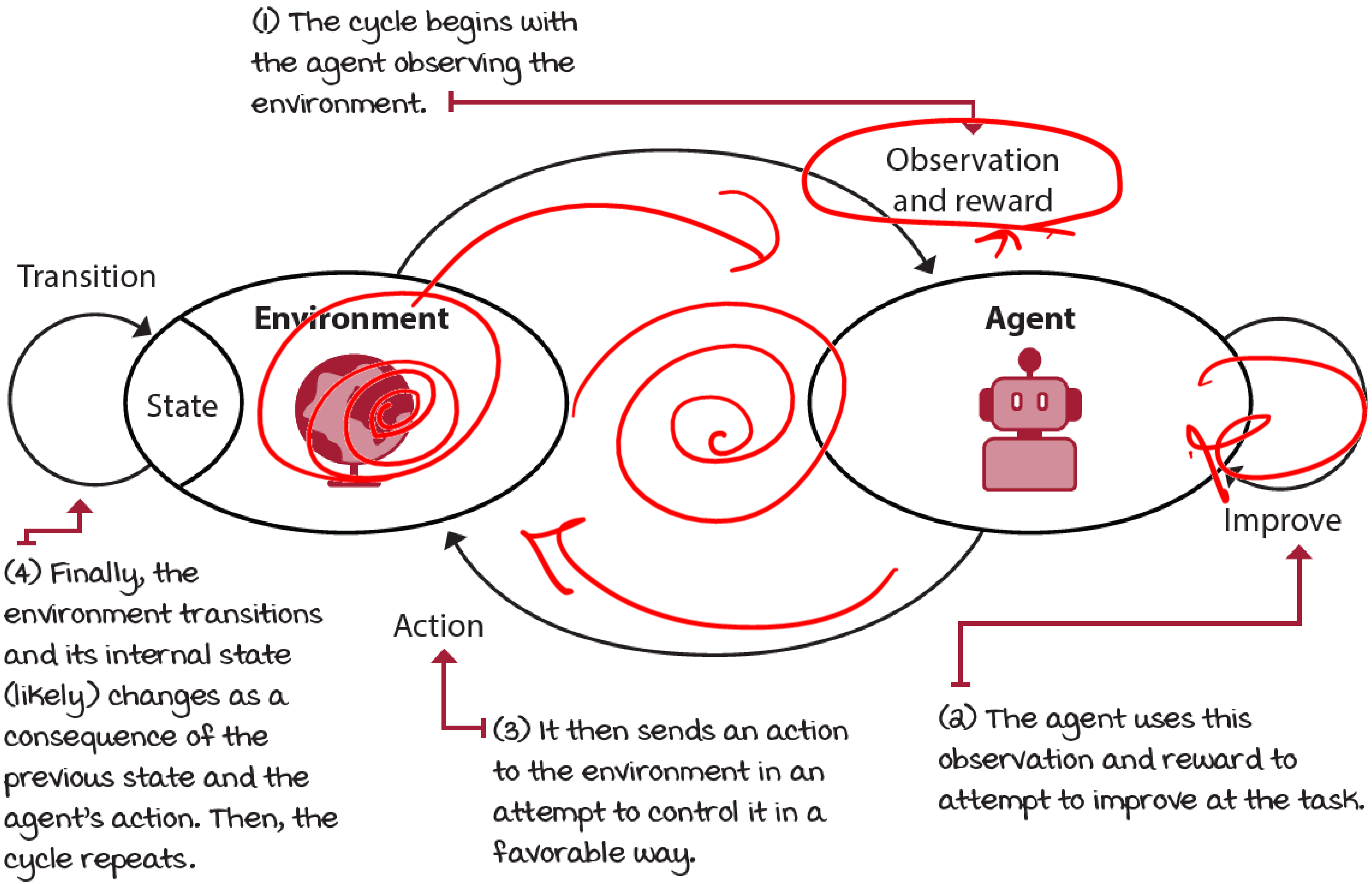
 →  → 

State s_0 → $\pi(A|s_0)$ → [Left: 0.1, Right: 0.7, Jump: 0.2]

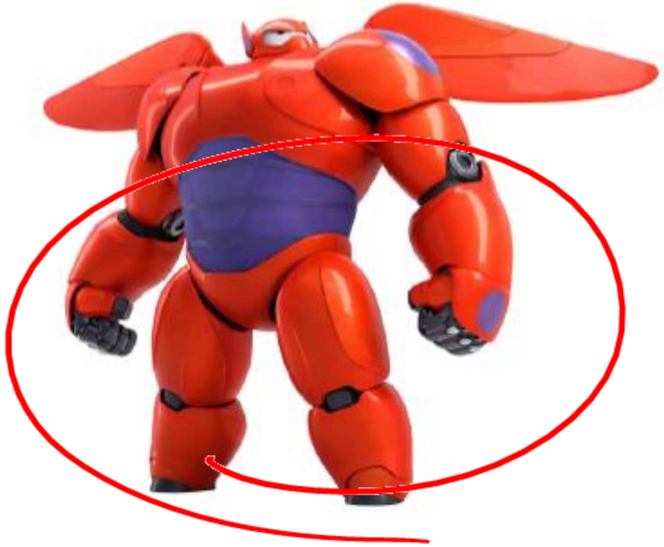


Process the environment goes through
as a consequence of agent's actions
Environment
Transition
Next state
Reward
State
Action
Observation, reward
(1) Environment receives the action selected by the agent.
(2) Depending on the current environment state, and the agent's chosen action . . .
(3) . . . the environment will transition to a new internal state.
(4) The new state and reward are passed through a filter: some problems don't let the true state of the environment be perceived by the agent!
(5) Finally, the reaction is passed back to the agent.

The reinforcement learning cycle



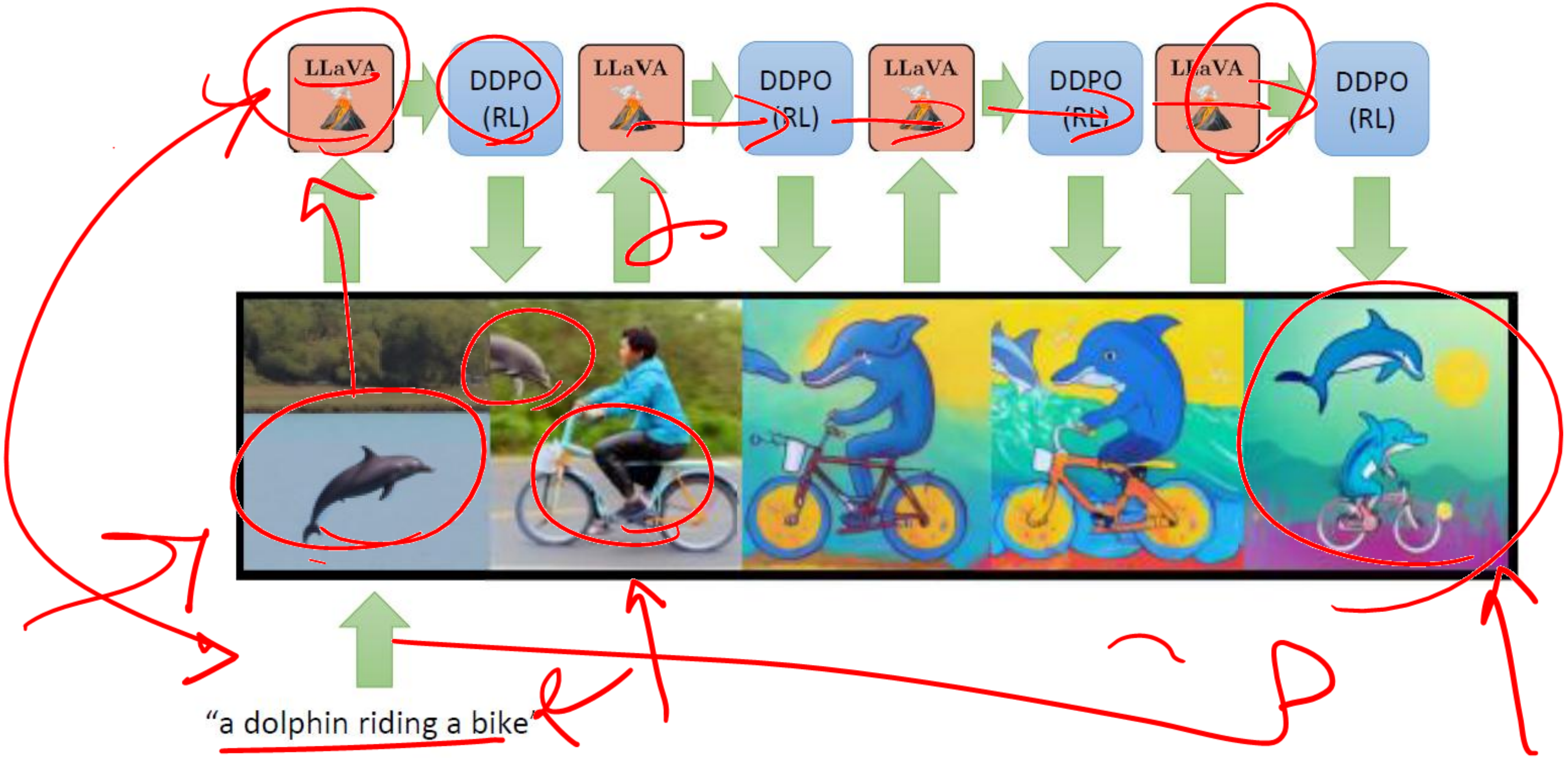
Actions: muscle contractions
Observations: sight, smell
Rewards: food



Actions: motor current or torque
Observations: camera images
Rewards: task success measure (e.g., running speed)

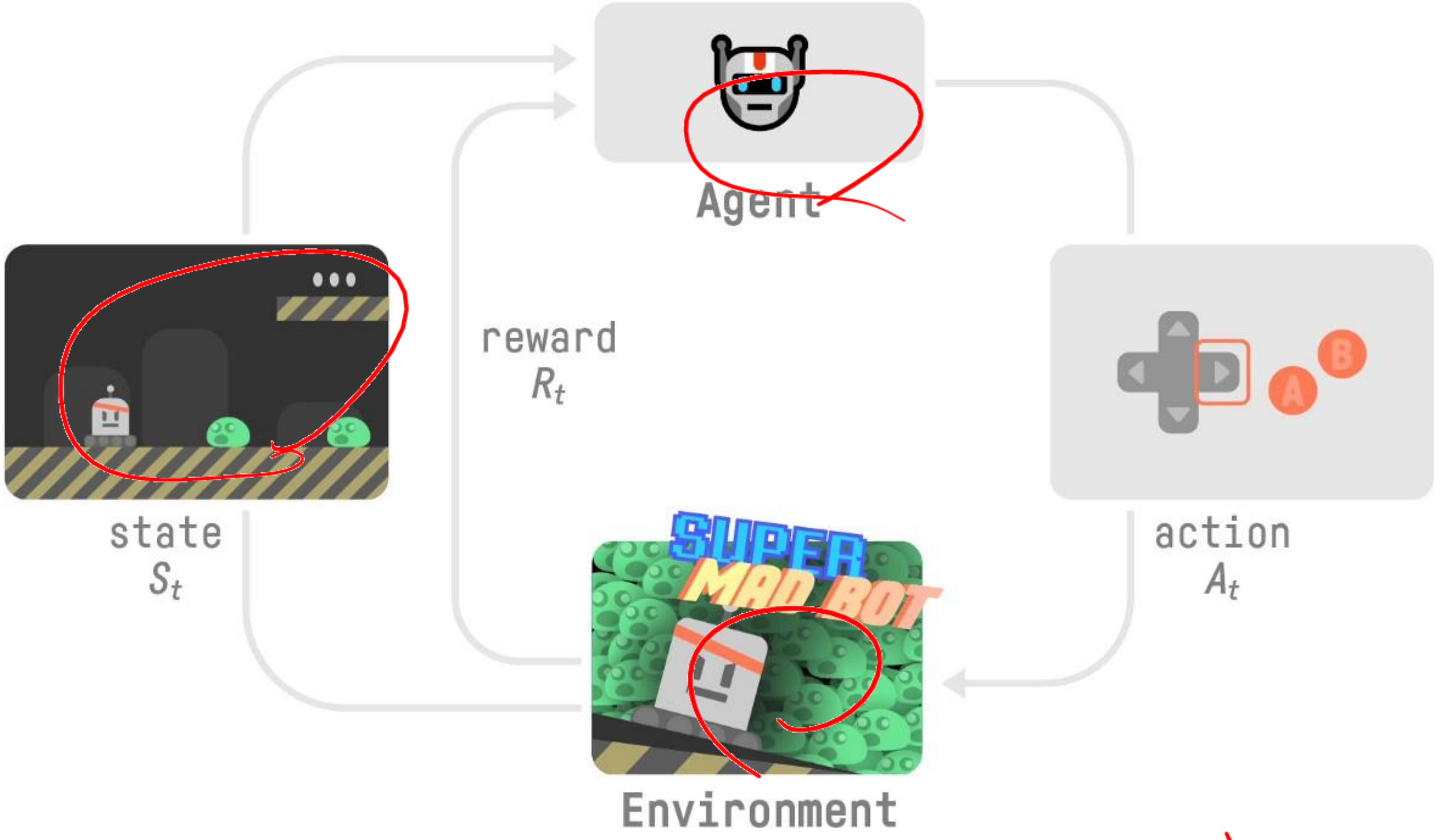
Actions: what to purchase
Observations: inventory levels
Rewards: profit

Reinforcement learning with image generation



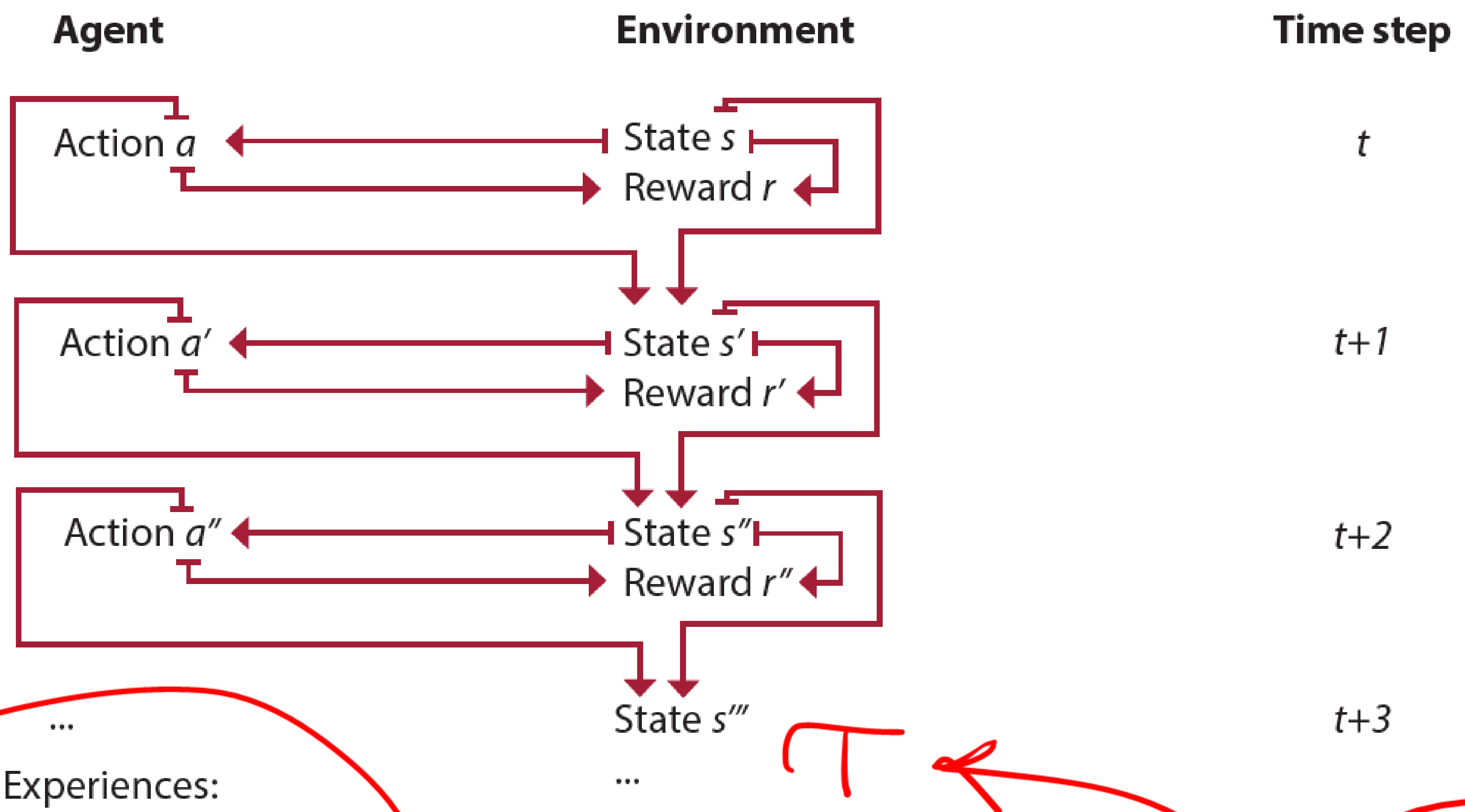
Kevin Black*, Michael Janner*, Yilun Du, Ilya Kostrikov, Sergey Levine. **Training Diffusion Models with Reinforcement Learning**. 2023.


C O N = C 1 C C C 1 [EOS]
Taiga
Decoder Block #n
Feed Forward
Masked Attention
Attention Head #1
Attention Head #2
Attention Head #3
Attention Head #n-1
Attention Head #n
Decoder Block #1
Feed Forward
Linear
ReLU
Linear
Masked Attention
[BOS] C O N = C 1 C C C 1
Stage 1
State
Agent
Action
Next state
Reward
TAIGA
0.07 C
0.03 #
0.9 (
0 [BOS]
0 [EOS]
0
a: Molecule generation process
0 (
0.9 [EOS]
CON=C1CCC(O)CC1
0.674
b: Last generation step and reward calculation
Stage 2



$$\left(\underbrace{S_0}_{\text{State}}, \underbrace{A_0}_{\text{Action}}, \underbrace{R_1}_{\text{Reward}}, \underbrace{S_1}_{\text{Next State}}\right)$$

Experience tuples



...

Experiences:

t, (s, a, r', s')

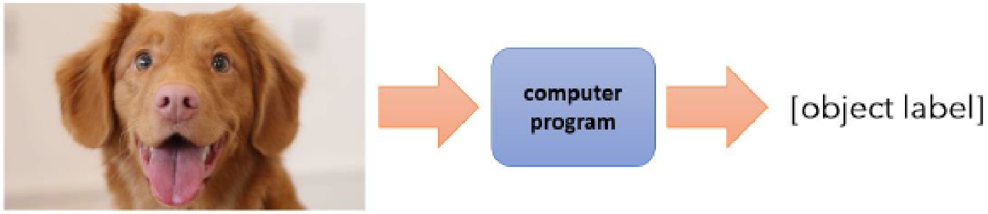
t+1, (s', a', r'', s'')

t+2, (s'', a'', r''', s''')

...

... t_{109}

supervised learning



input: $\mathbf{x}$

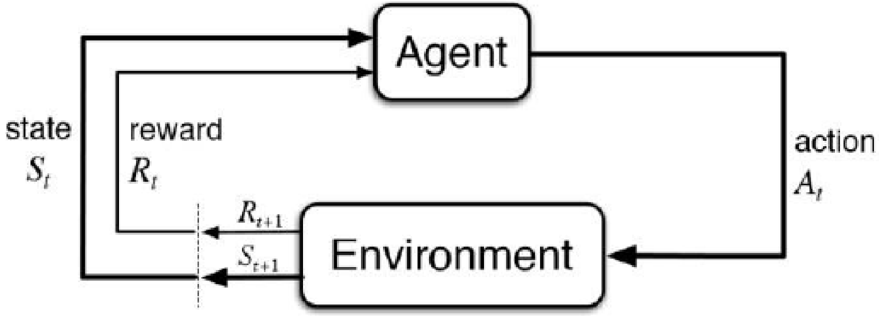
output: $\mathbf{y}$

data: $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$

goal: $f_\theta(\mathbf{x}_i) \approx \mathbf{y}_i$

someone gives this to you

reinforcement learning



pick your own actions

input: $\mathbf{s}_t$ *at each time step*

output: $\mathbf{a}_t$ *at each time step*

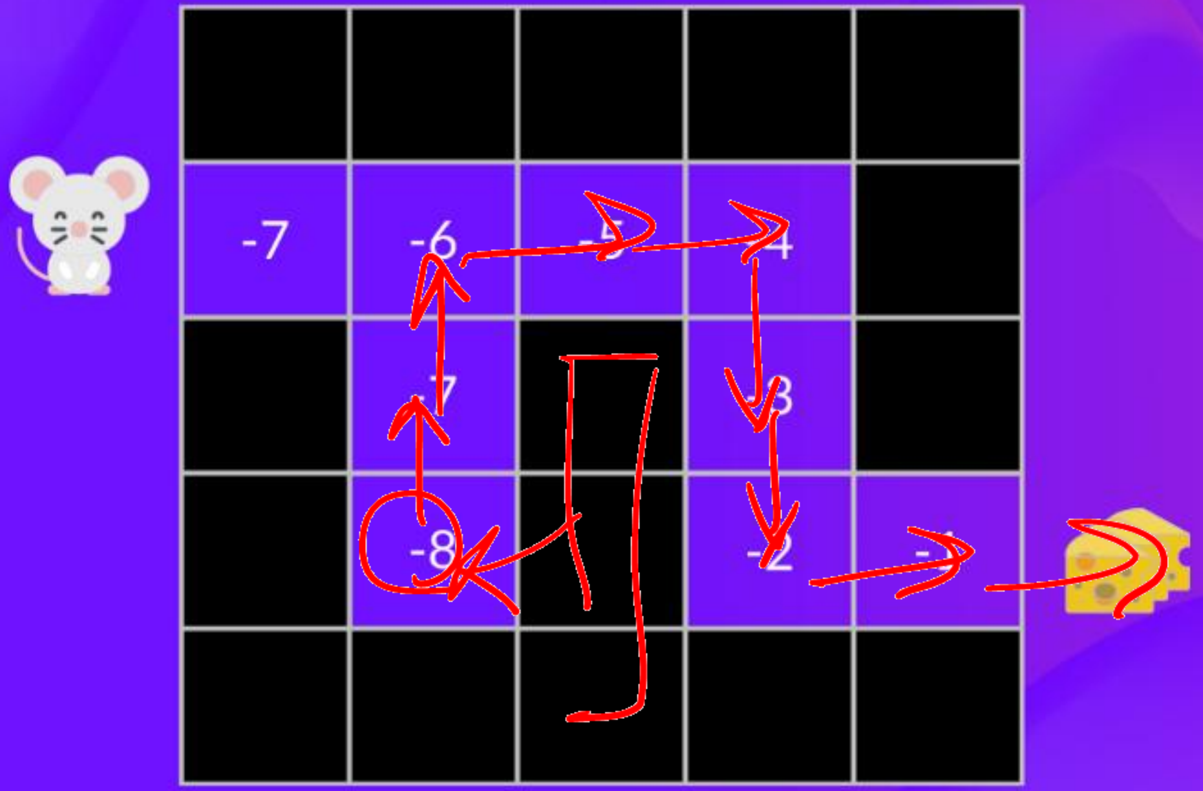
data: $(\mathbf{s}_1, \mathbf{a}_1, r_1, \ldots, \mathbf{s}_T, \mathbf{a}_T, r_T)$

goal: learn $\pi_\theta : \mathbf{s}_t \to \mathbf{a}_t$

to maximize $\sum_t r_t$

The State Value Function

State Value Function: calculate the **value of a state.**



Return

$$\underbrace{V_\pi(s)}_{\text{Value of state s}} = \underbrace{\mathbf{E}_{\boxed{\pi}}[G_t}_{\text{Expected return}} | \underbrace{S_t = s}_{\text{If the agent starts at state s}}]$$

And uses the policy to choose its actions for all time steps

For each state,

the state-value function outputs

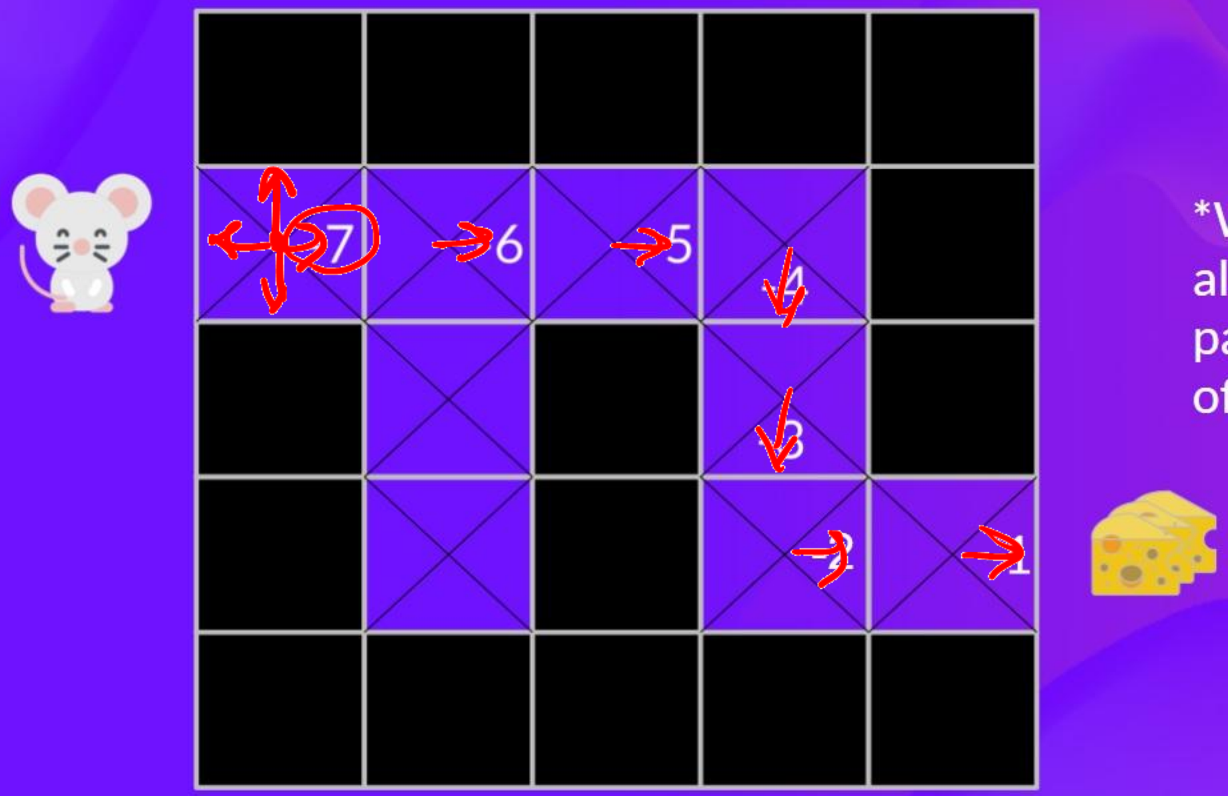
the expected return

if the agent starts in that state

and then follows the policy forever after.

The Action Value Function

Action Value Function: calculate the **value of state-action pair.**



*We didn't fill
all the state-actions
pair for the example
of Action-value function

$$Q_\pi(s,a) = \mathbf{E}_{\boxed{\pi}}[G_t \mid S_t = s, A_t = a]$$

$\underbrace{Q_\pi(s,a)}$ Value of state-action pair s,a

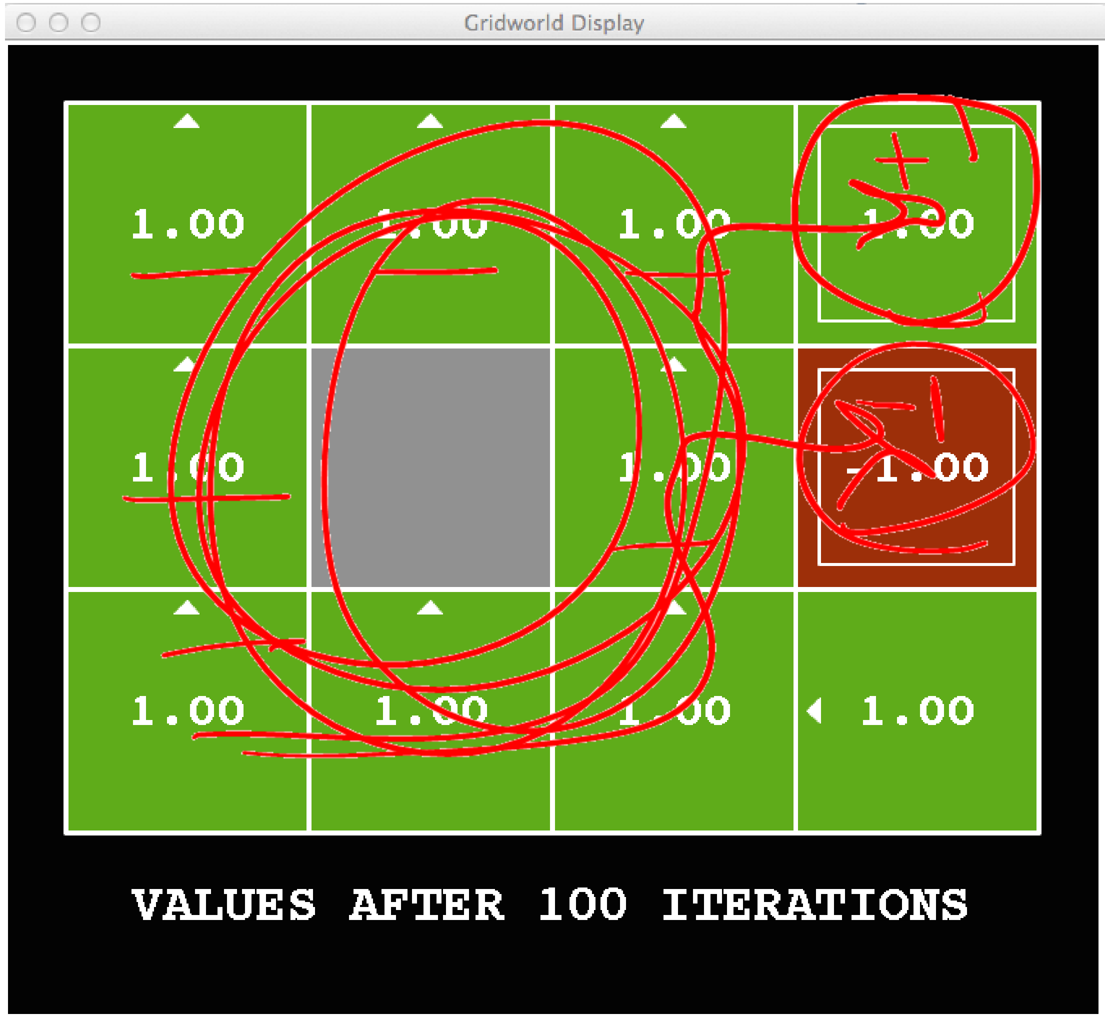
$\underbrace{\mathbf{E}_\pi[G_t}$ Expected return

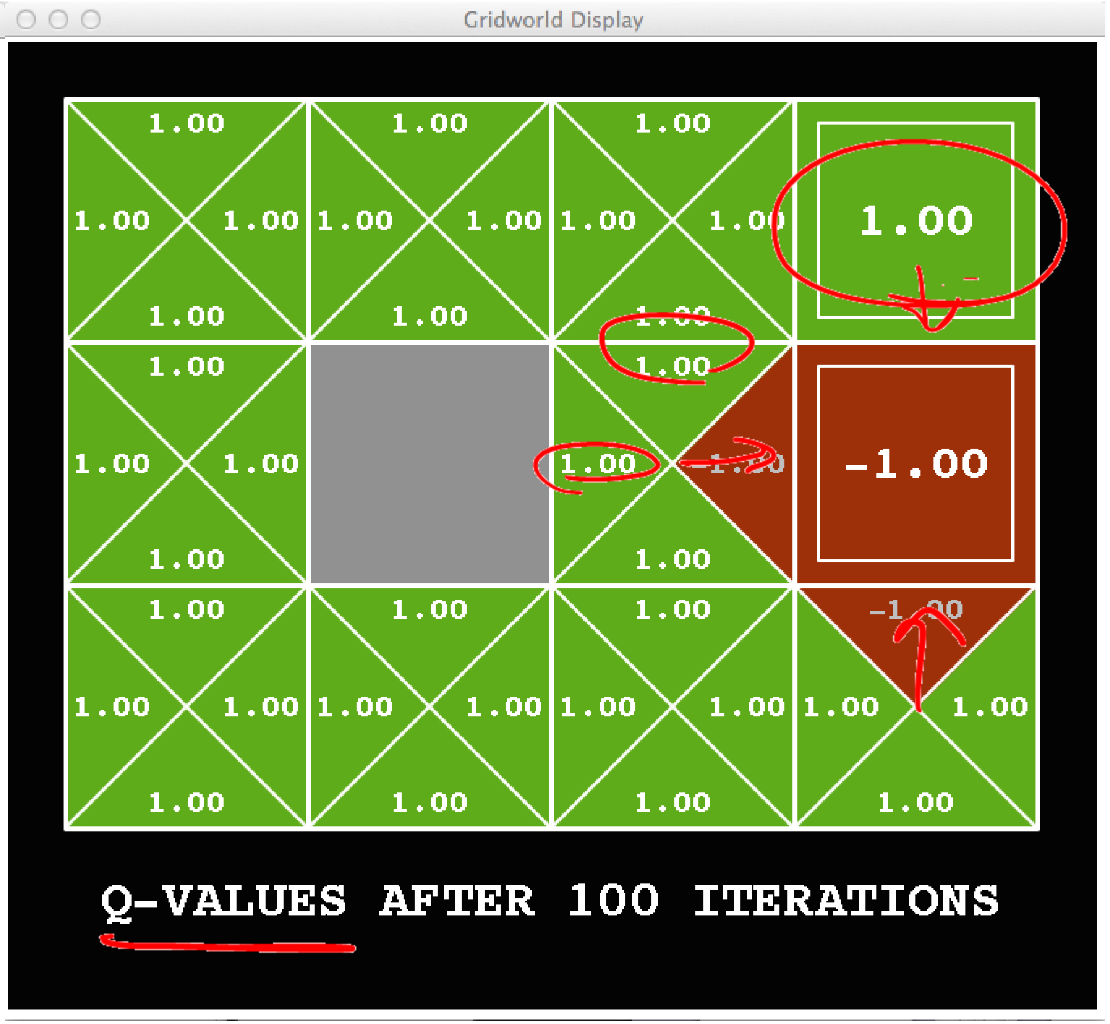
$\underbrace{S_t = s,}$ If the agent starts at state s

$\underbrace{A_t = a]}$ and chooses action a

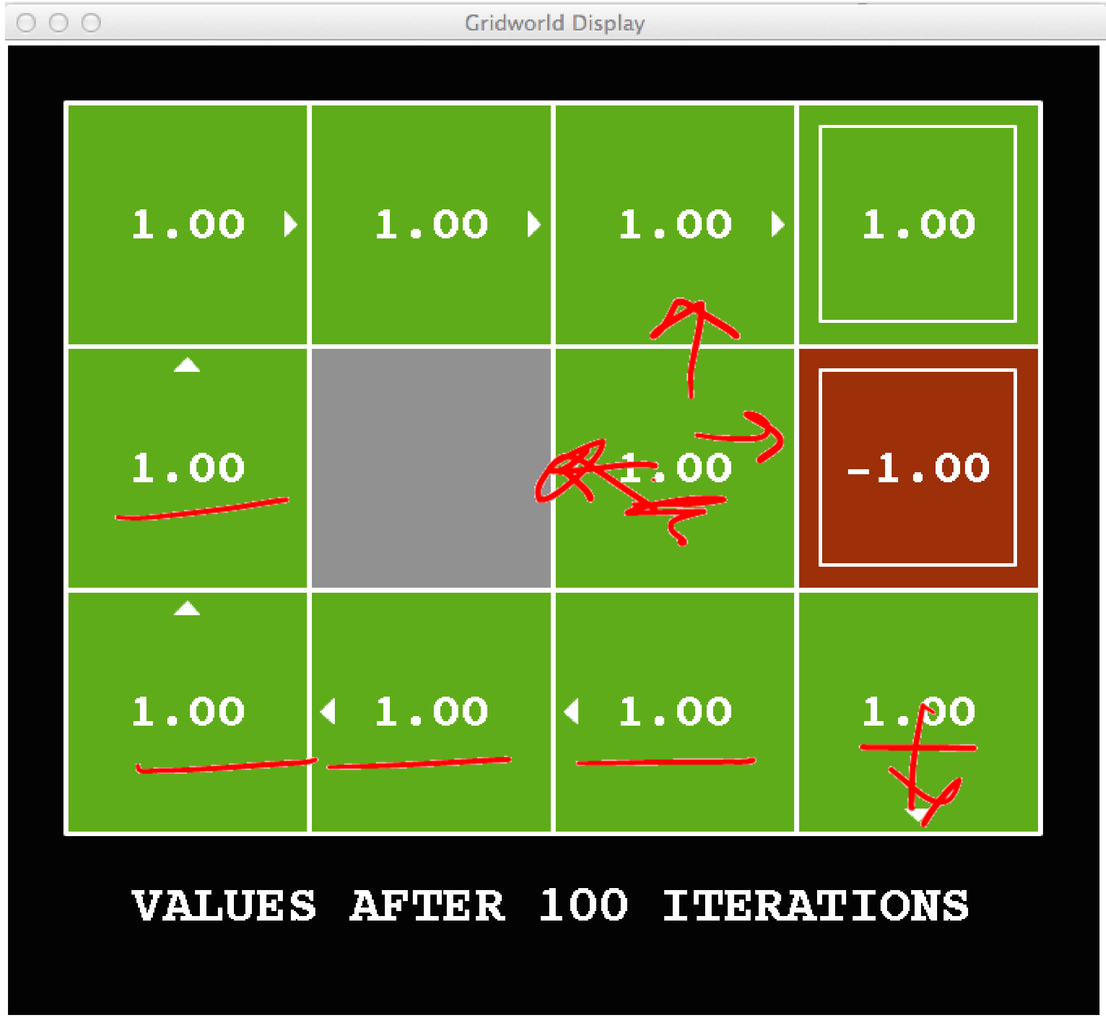
$\boxed{\pi}$ And then uses the policy to choose its actions for all time steps

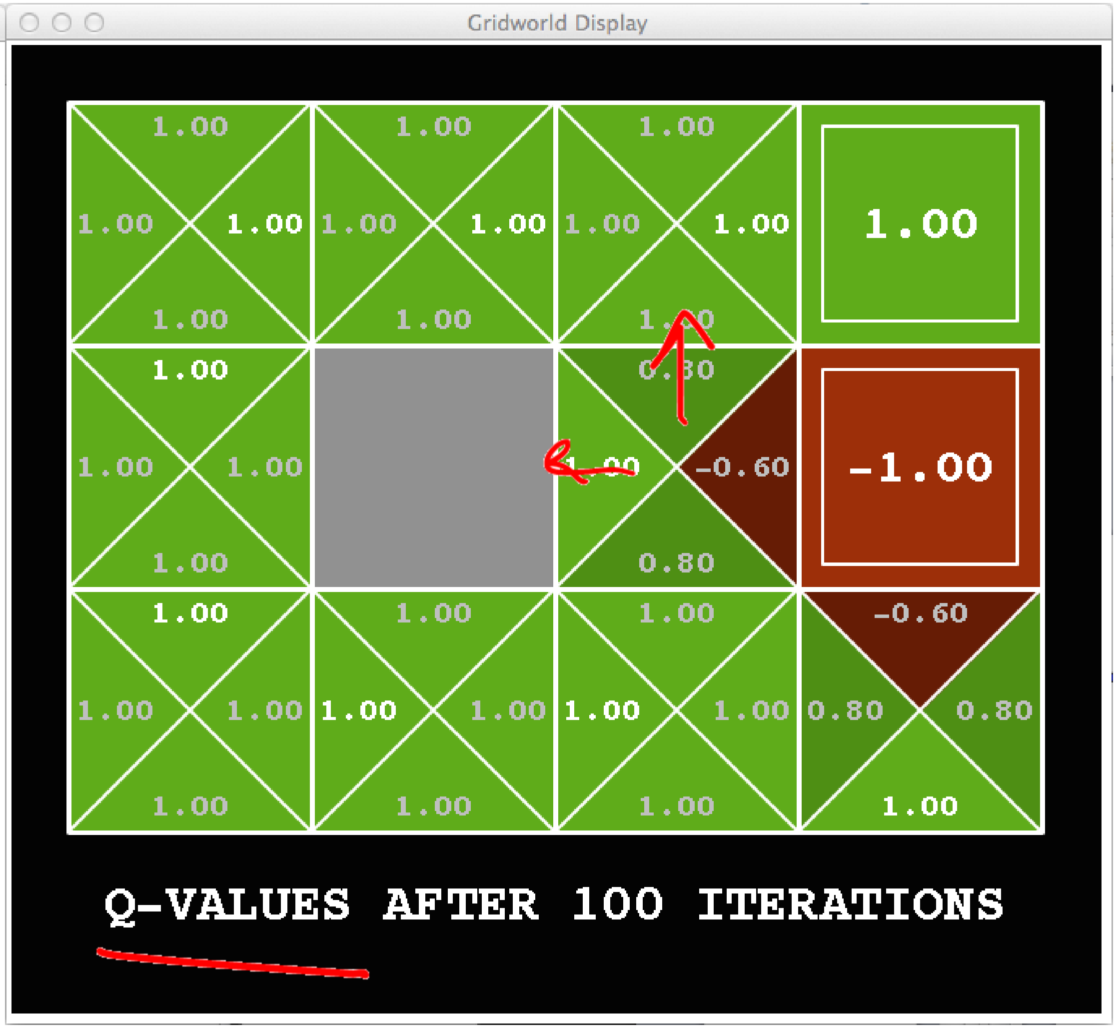
For each state and action,
the action-value function outputs
the expected return
if the agent starts in that state
and takes the action
and then follows the
policy forever after.



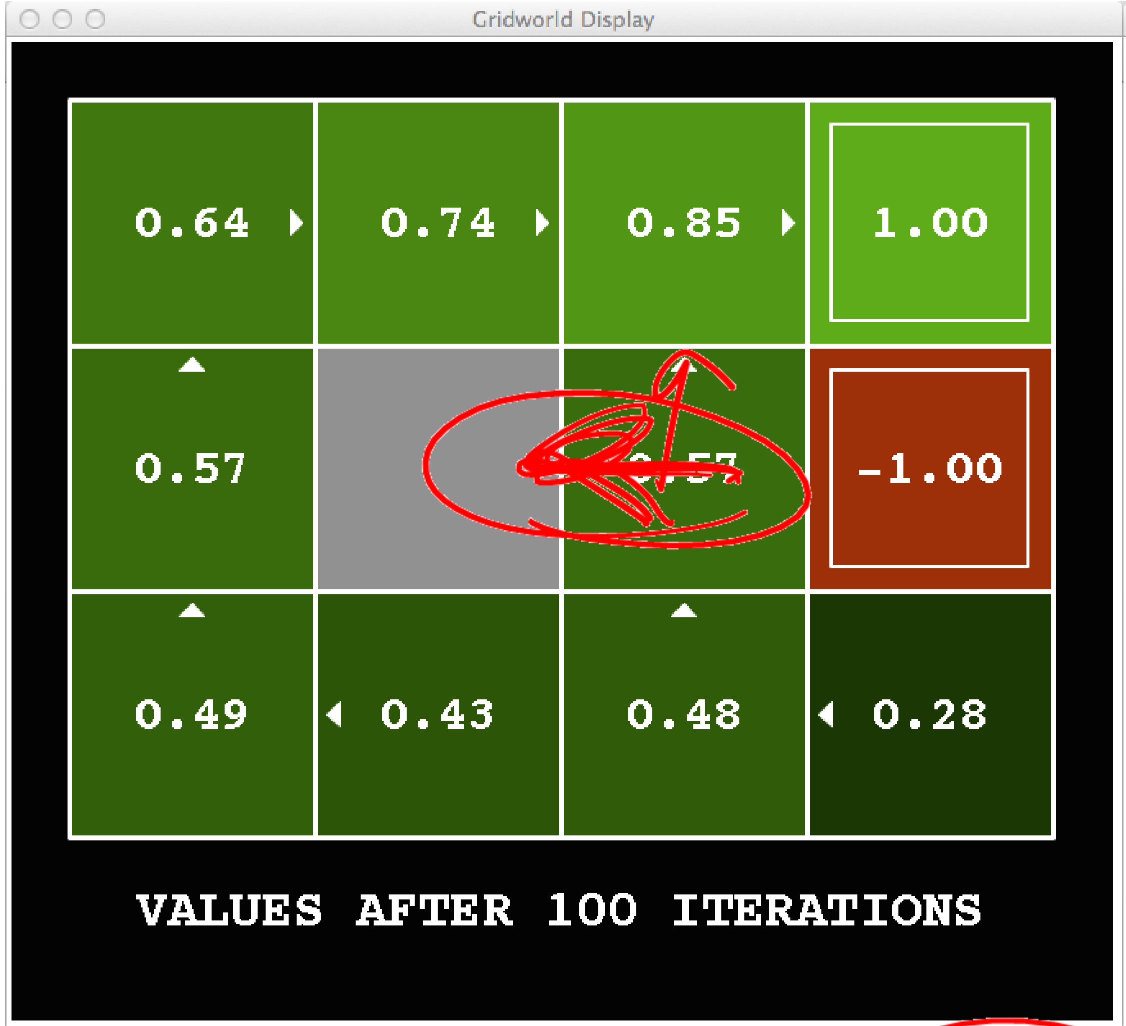


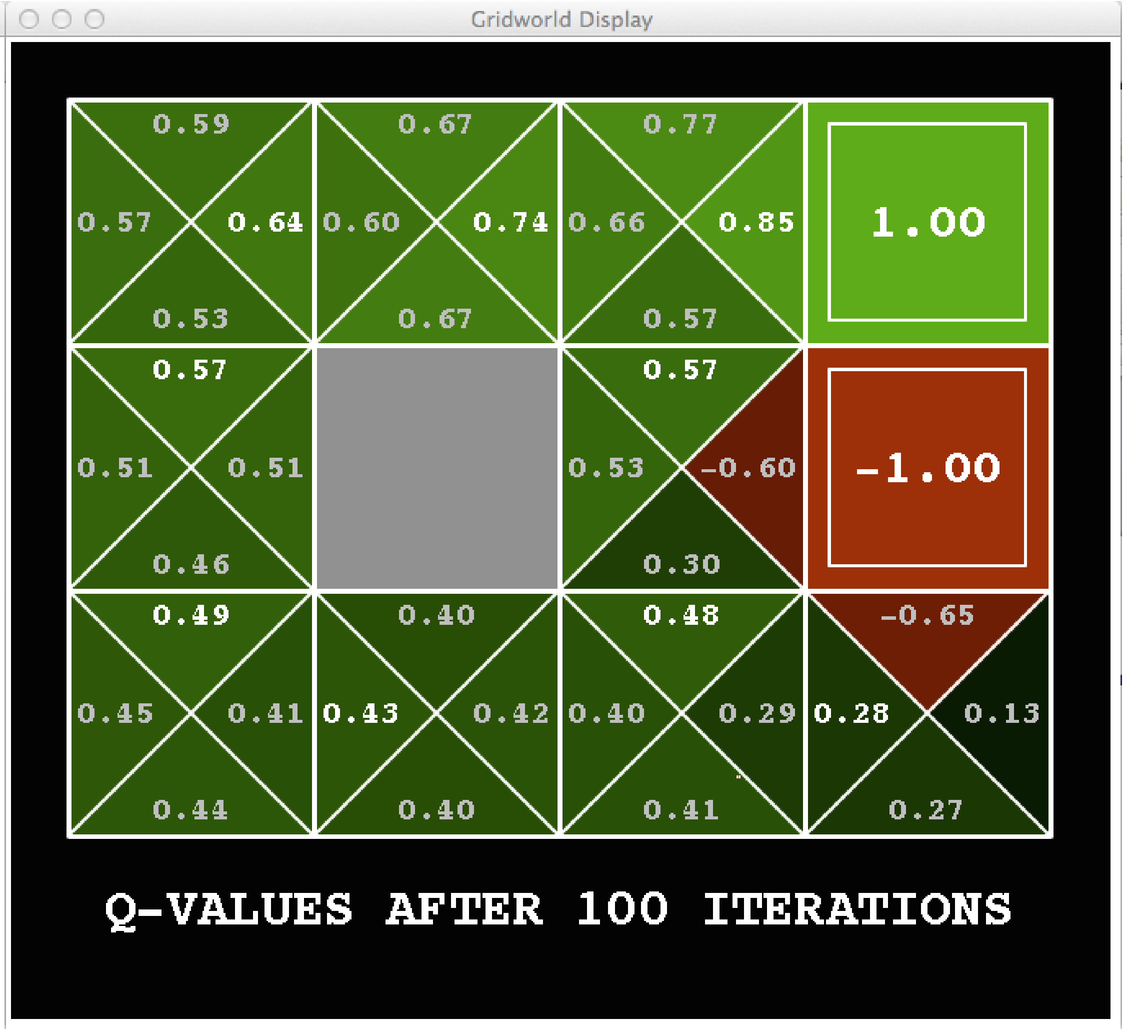
Noise = 0
Discount = 1
Living reward = 0





Noise = 0.2
Discount = 1
Living reward = 0





Noise = 0.2
Discount = 0.9
Living reward = 0

Noise = 0.2
Discount = 0.9
Living reward = -0.1

WHAT WE HAVE LEARNED SO FAR?

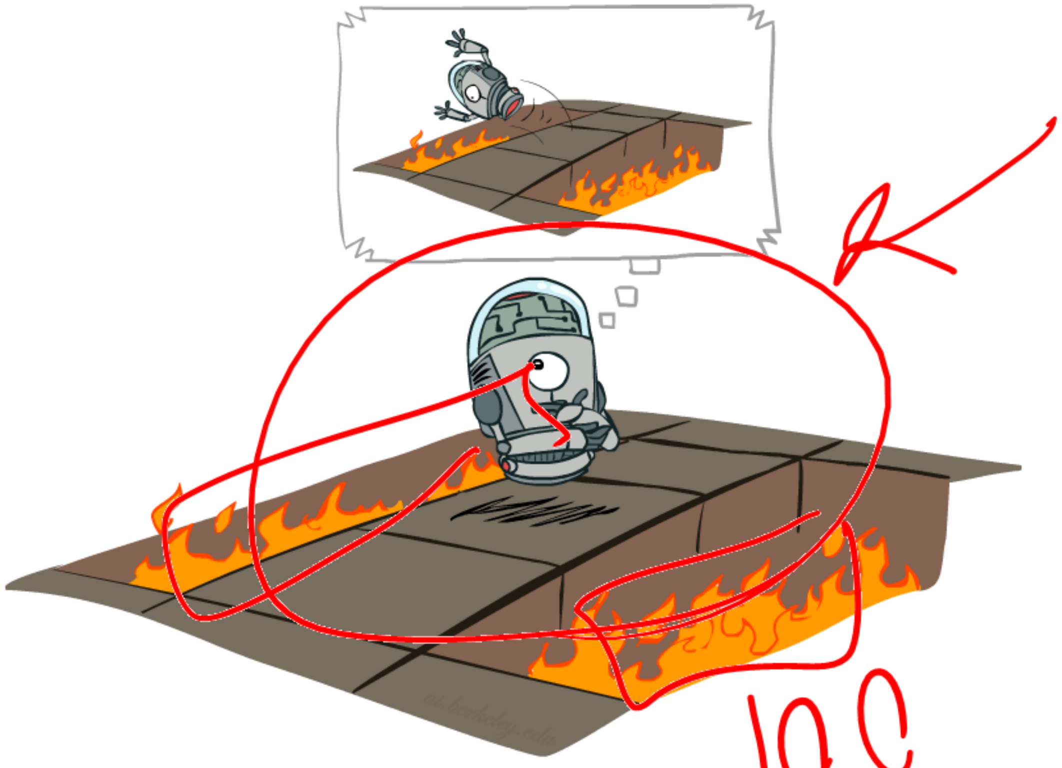
- what is reinforcement learning and its actual place & significance
- reinforcement learning framework & basic concepts
  - agent
  - environment
  - state/observation
  - action
  - reward
  - policy
  - model
  - experience/trajectory/horizon
  - discount factor
  - state value function
  - action value function

Challenges of Reinforcement Learning

Type of tasks

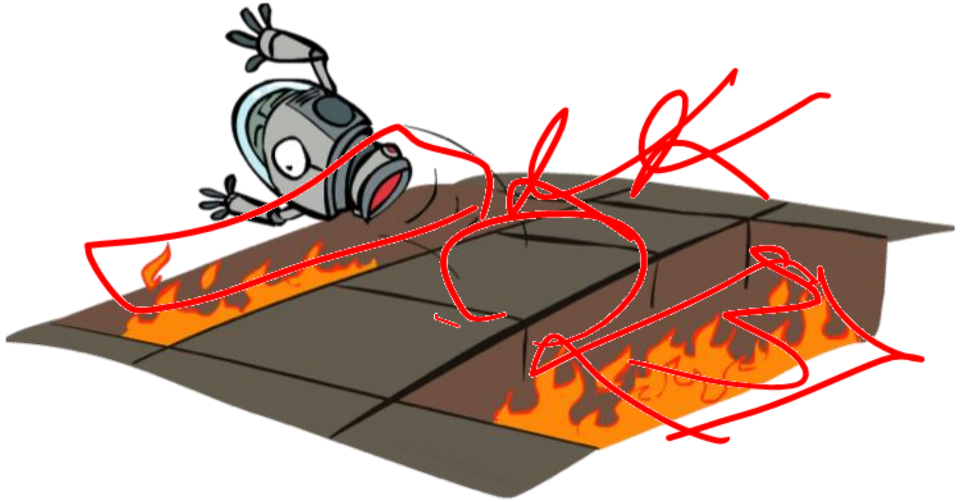
Episodic: **starting point** and an **ending point** (a terminal state)

Continuing: task that **continue forever** (no terminal state)



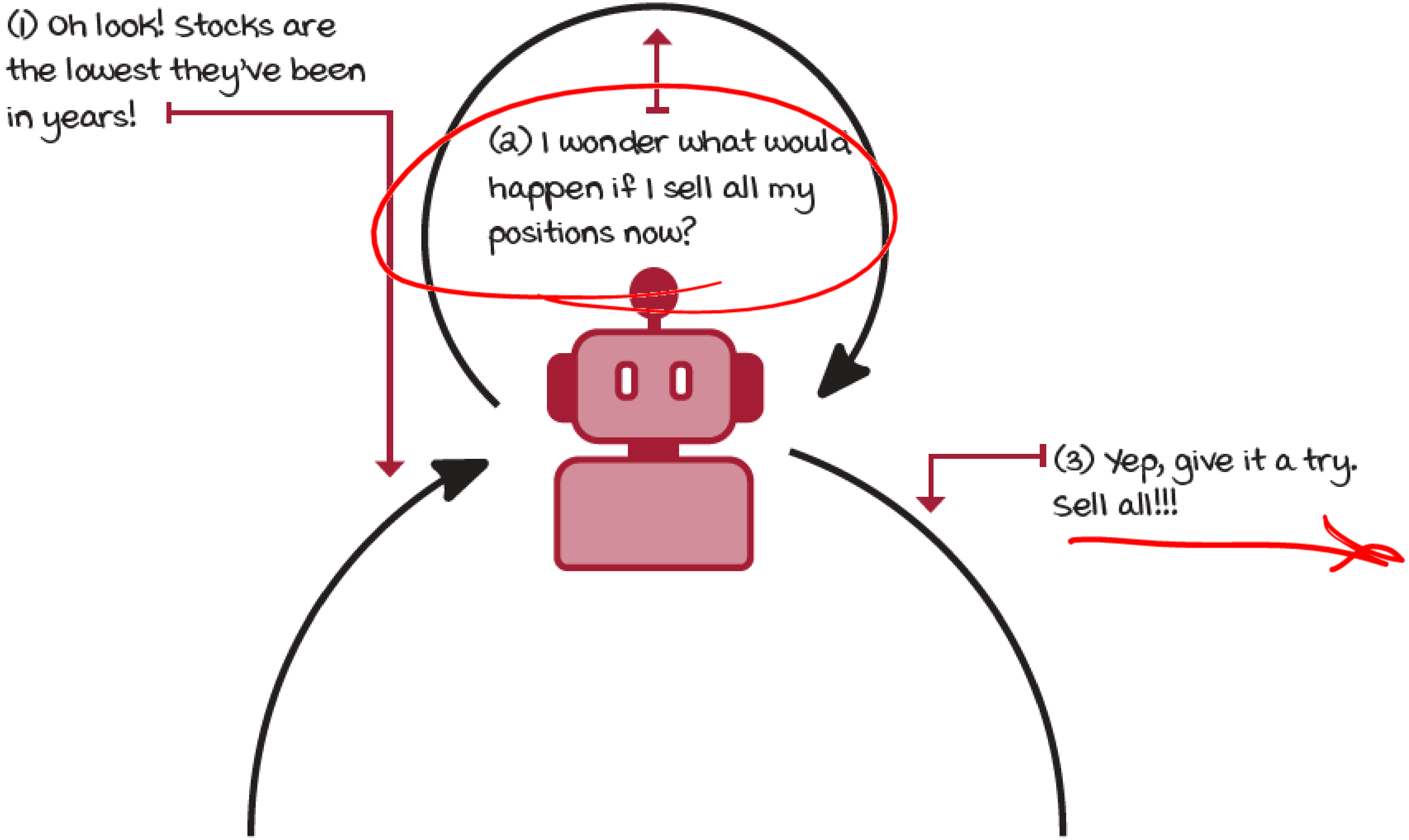
−100

Offline Solution

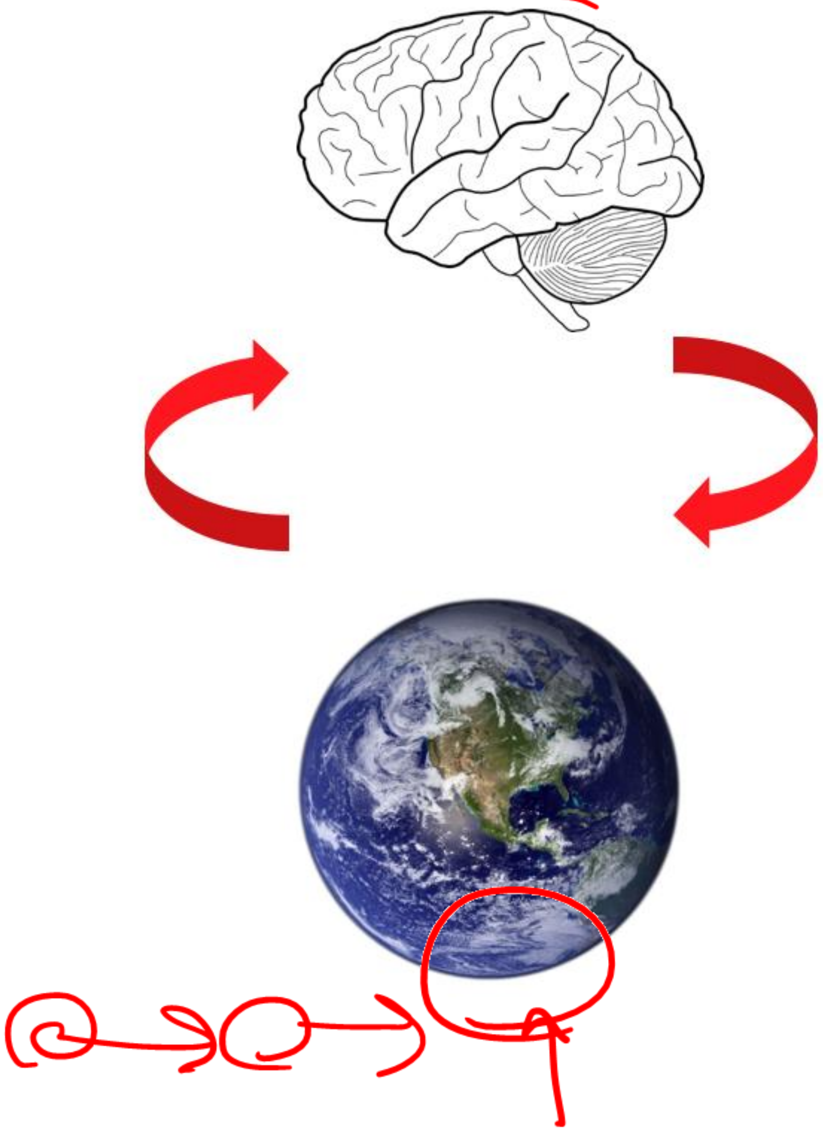


Online Learning

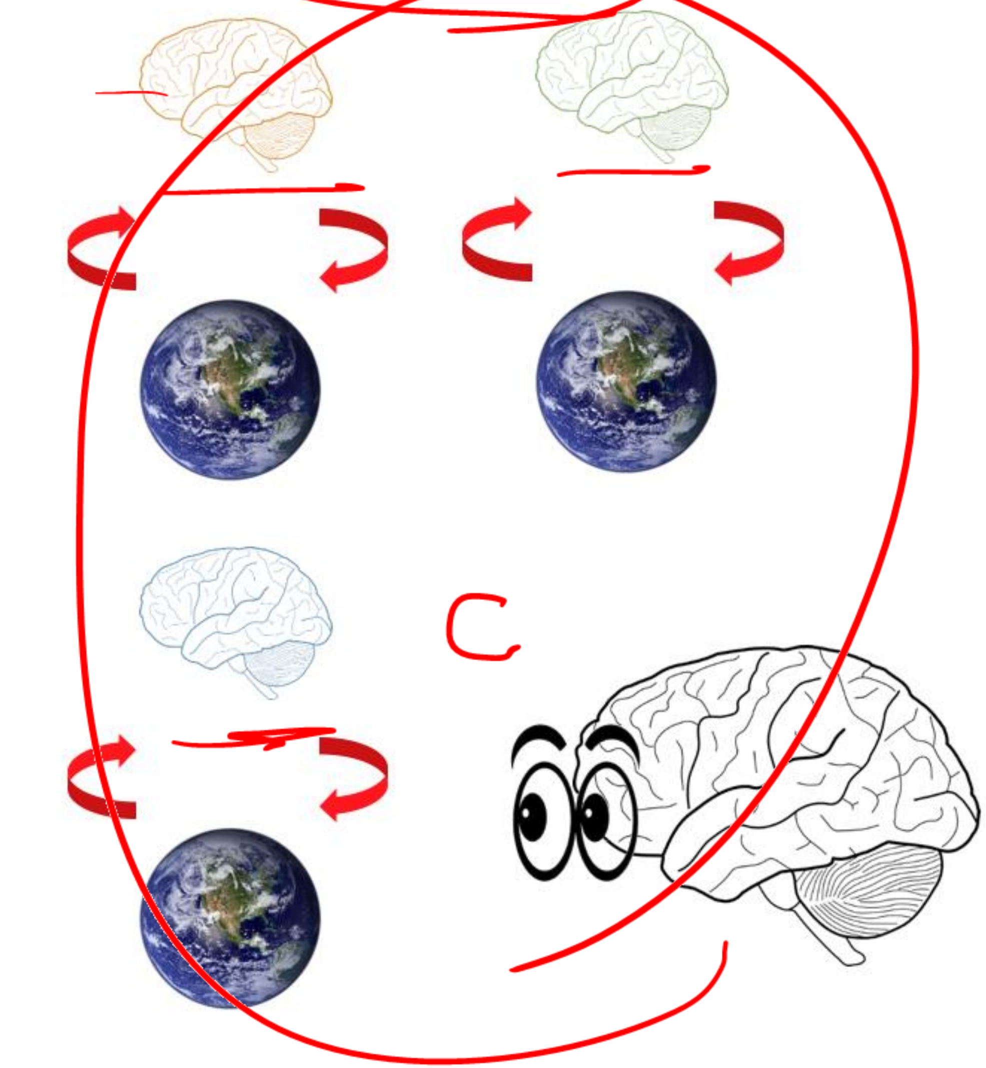
Deep reinforcement learning agents will explore!
Can you afford mistakes?

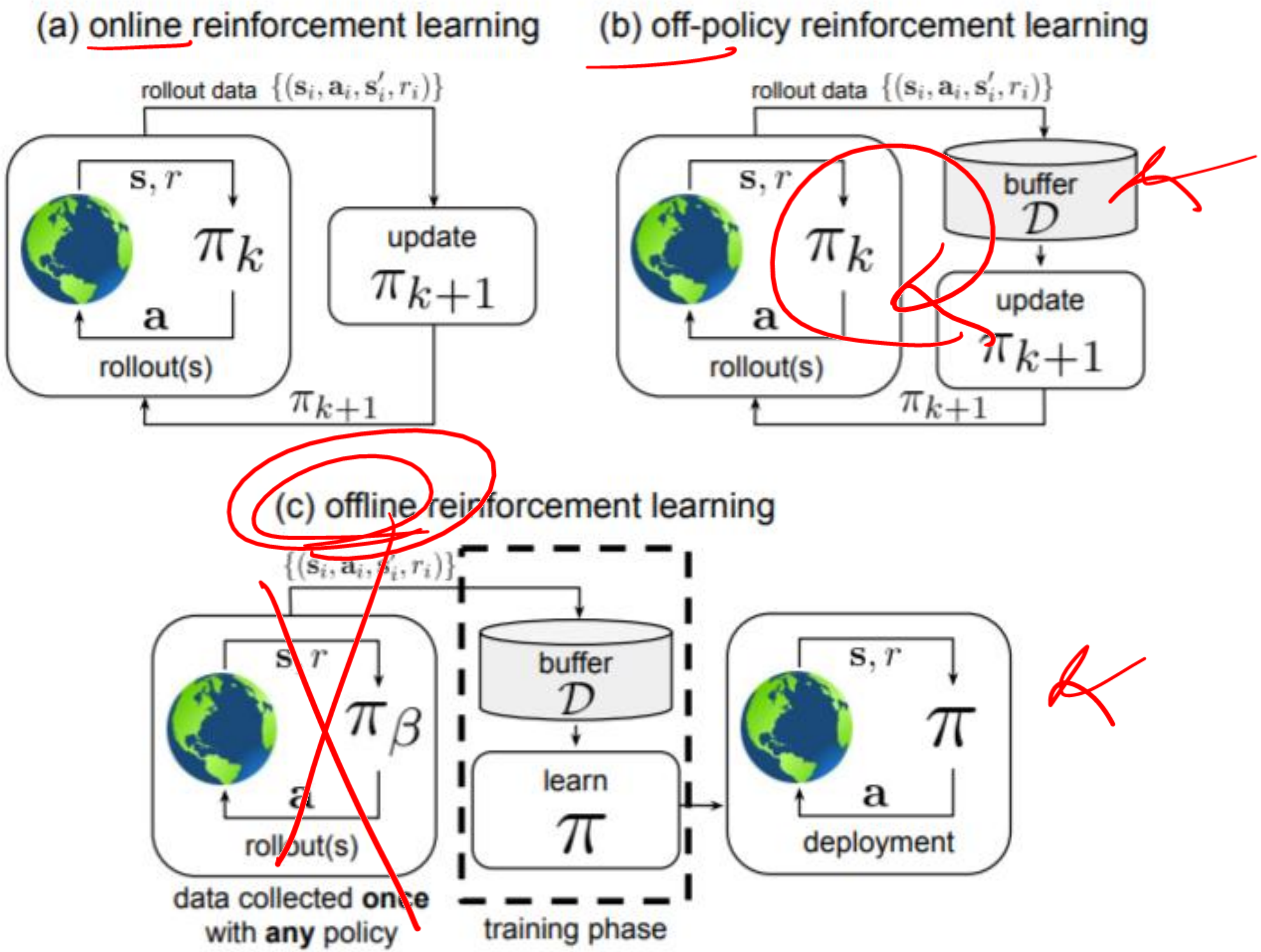


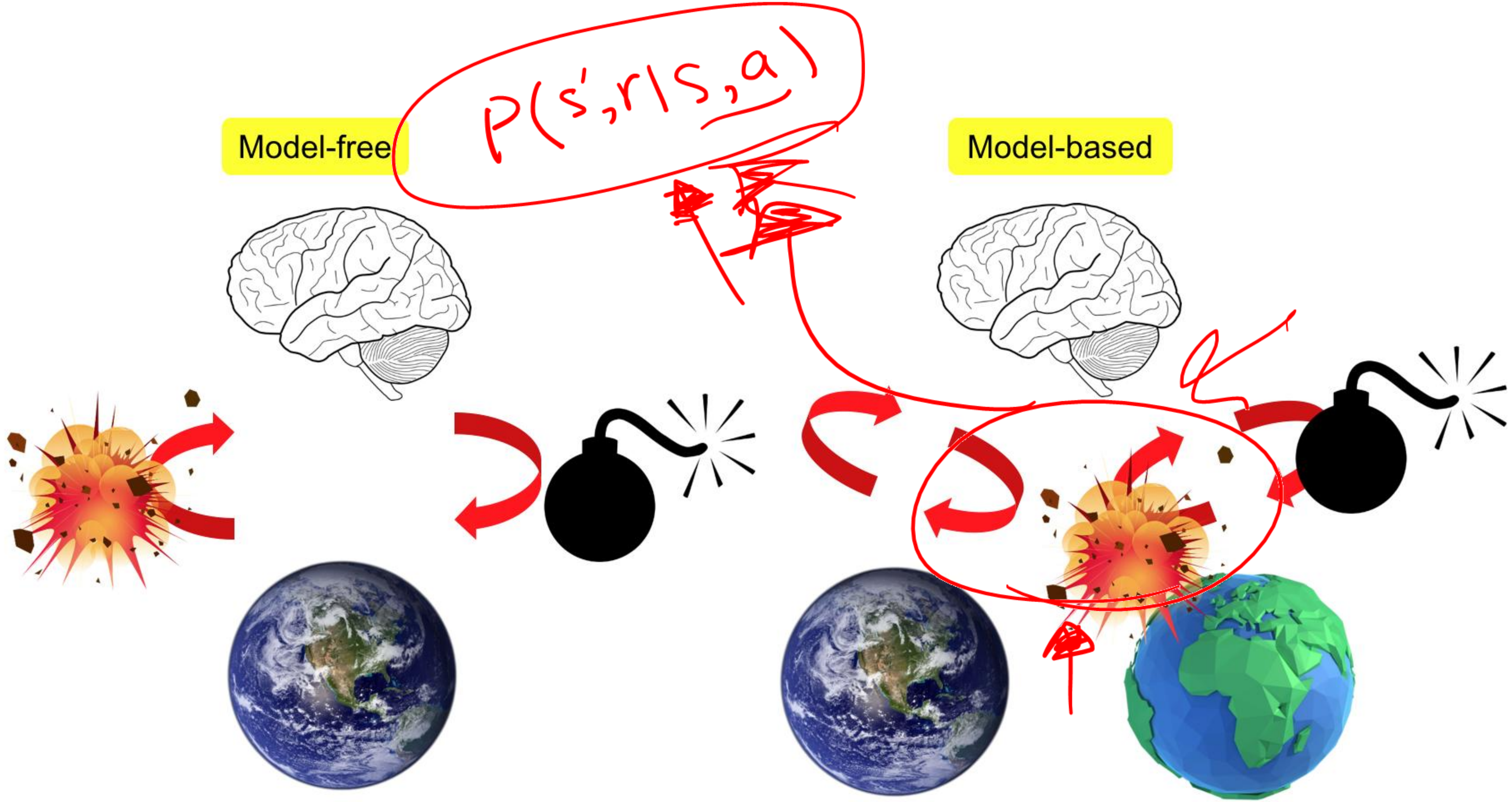
On-policy



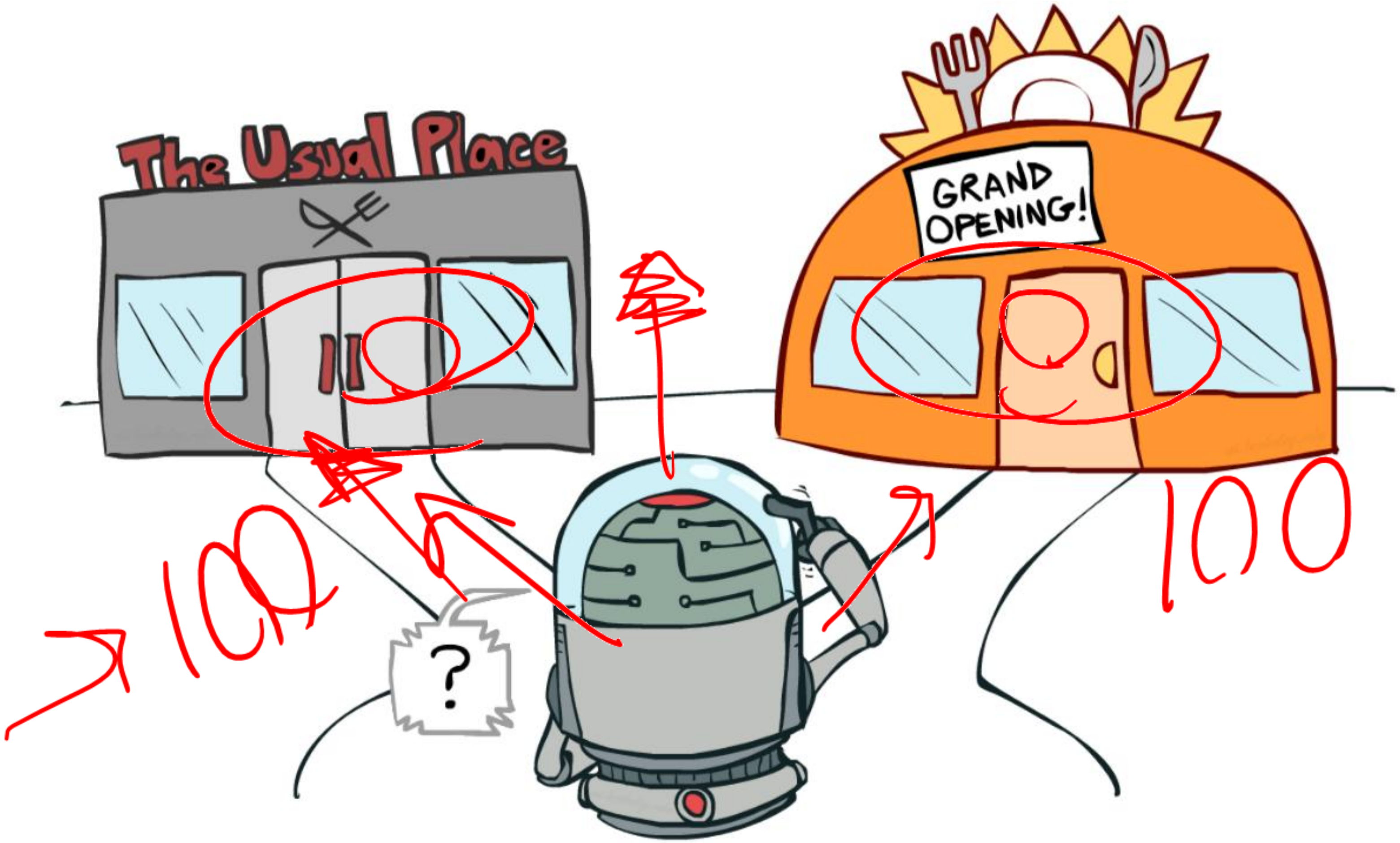
Off-policy




(a) online reinforcement learning
rollout data $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$
$\mathbf{s}, r$
π_k
$\mathbf{a}$
rollout(s)
update
π_{k+1}
π_{k+1}
(b) off-policy reinforcement learning
rollout data $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$
$\mathbf{s}, r$
π_k
$\mathbf{a}$
rollout(s)
buffer
$\mathcal{D}$
update
π_{k+1}
π_{k+1}
(c) offline reinforcement learning
$\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$
$\mathbf{s}, r$
π_β
$\mathbf{a}$
rollout(s)
data collected **once**
with **any** policy
buffer
$\mathcal{D}$
learn
π
training phase
$\mathbf{s}, r$
π
$\mathbf{a}$
deployment

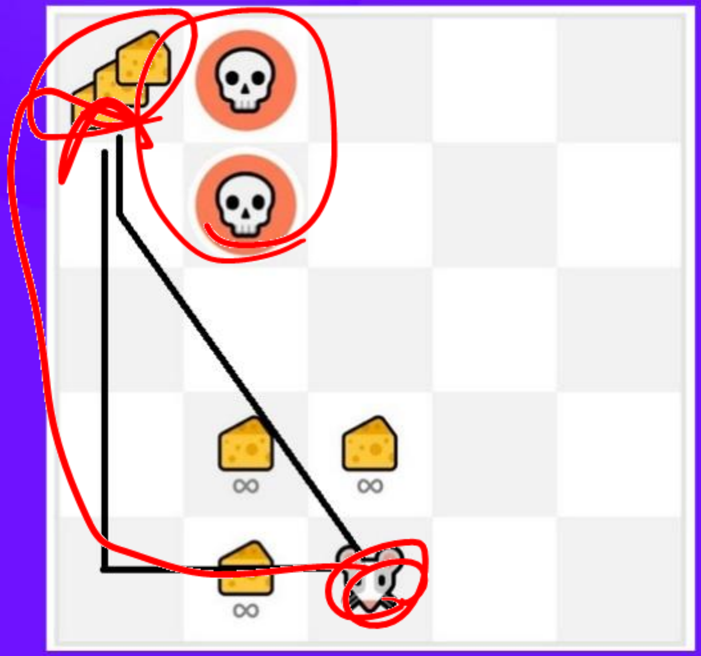

Model-free
Model-based
P(s',r|s,a)

EXPLORATION VS. EXPLOITATION DILEMMA

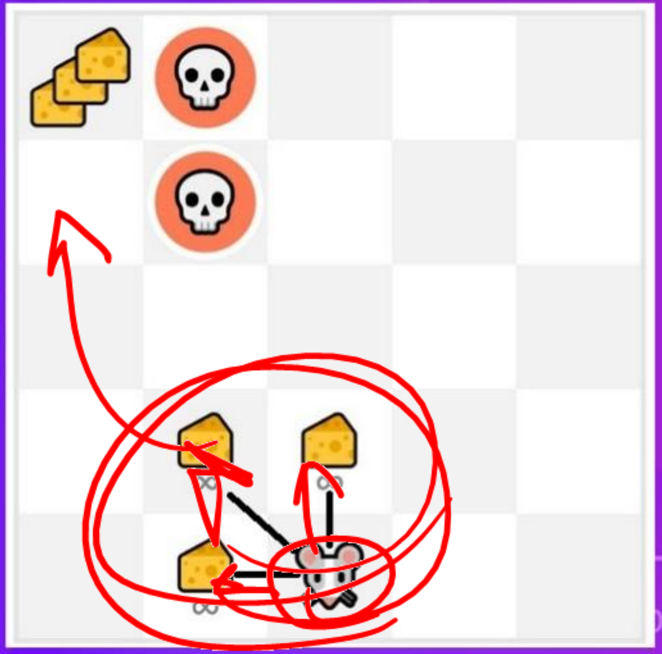


Exploration/ Exploitation tradeoff

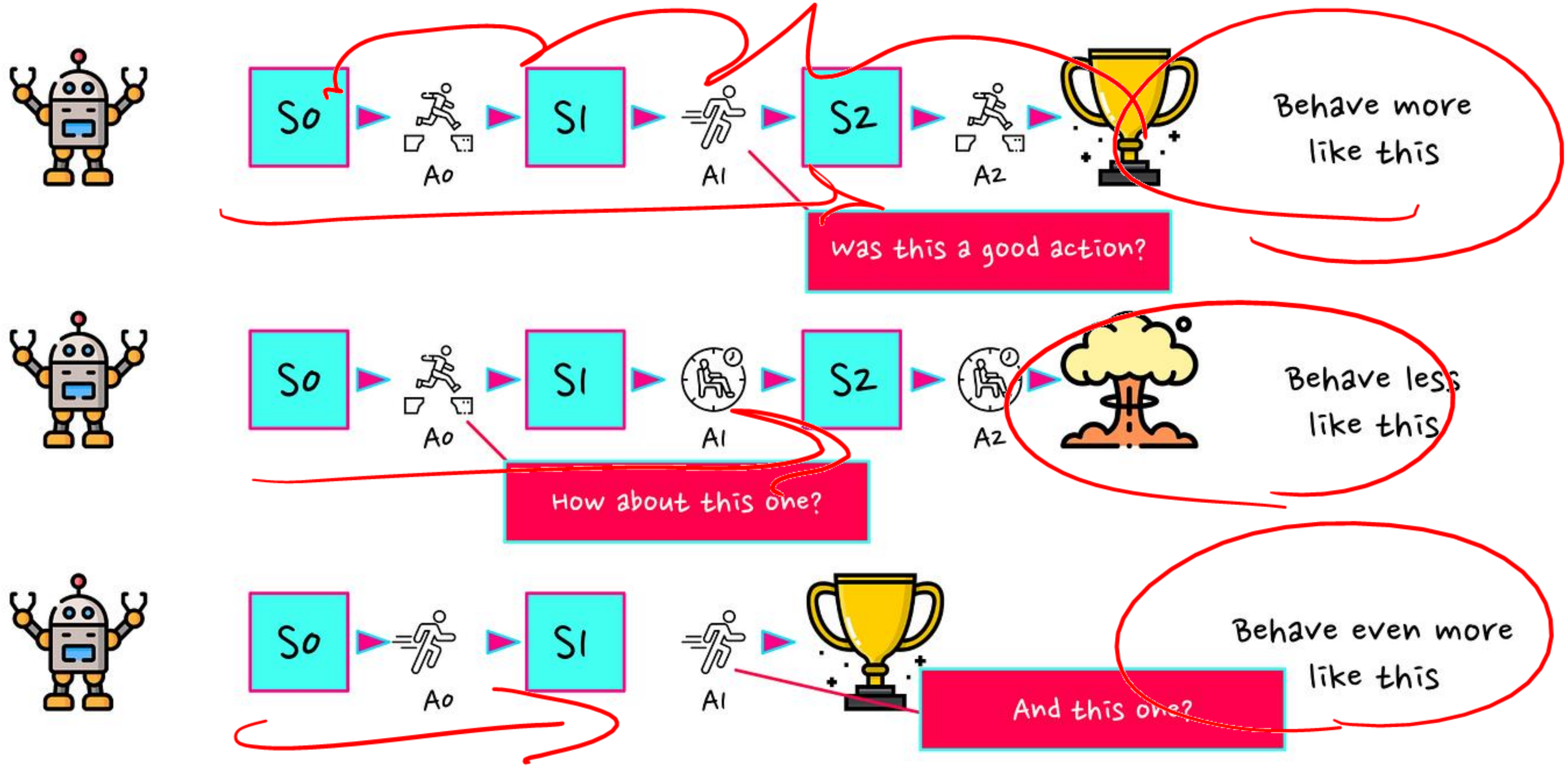
Exploration: trying **random actions** in order to find **more information about the environment.**



Exploitation: using known information to **maximize the reward.**

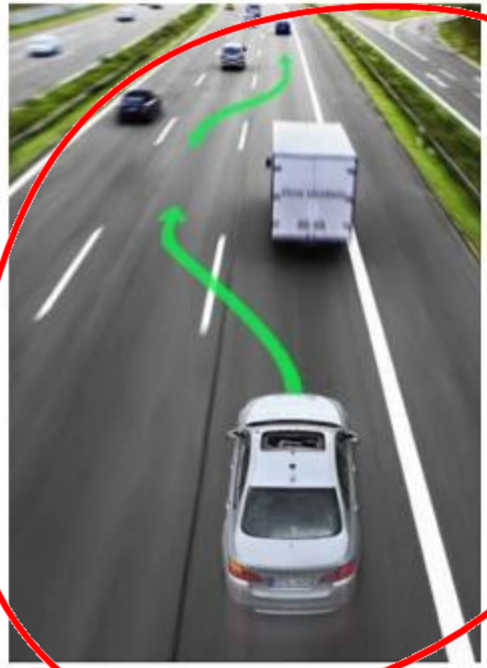


CREDIT ASSIGNMENT PROBLEM



REWARD ENGINEERING PROBLEM

IRL



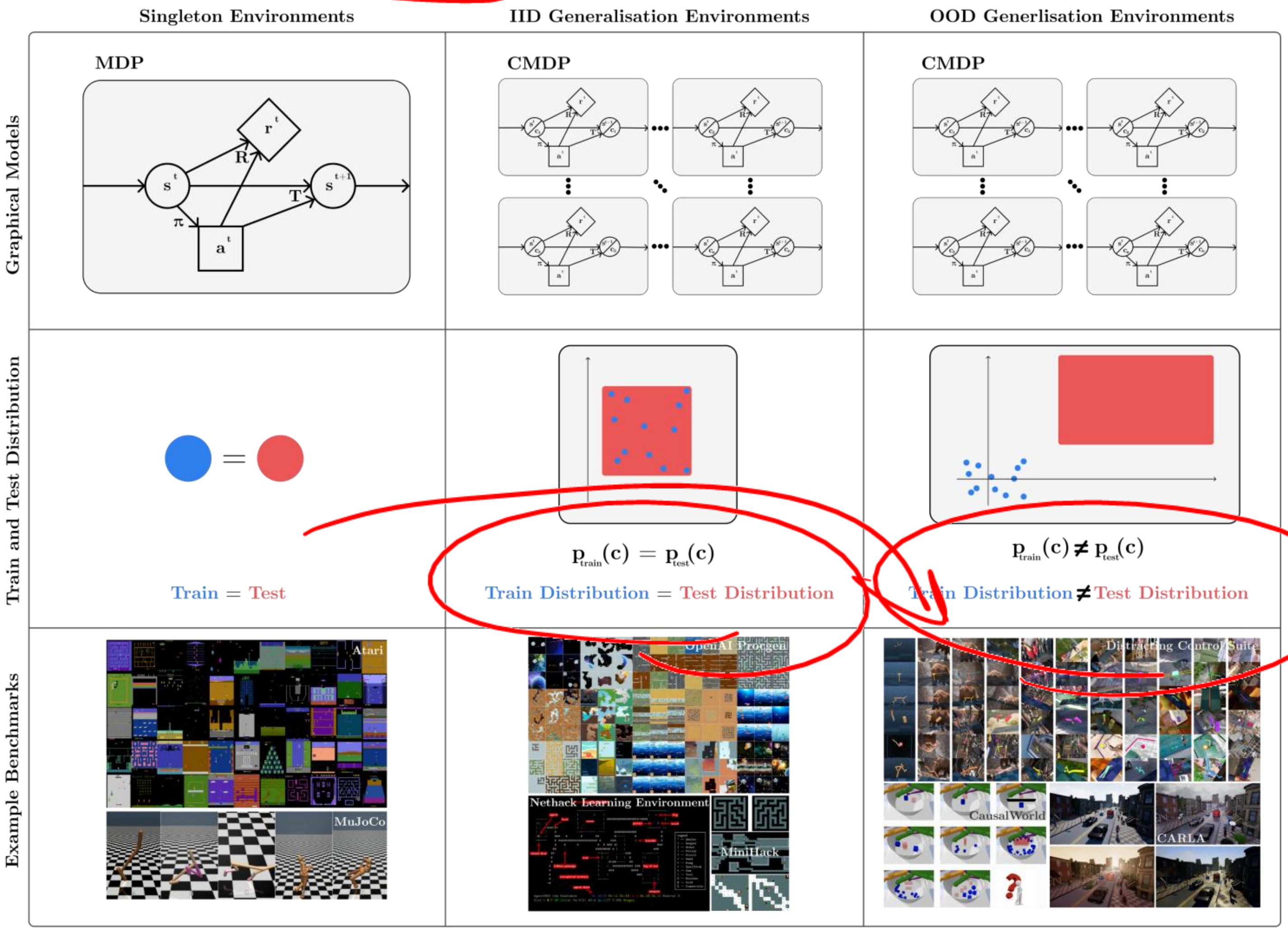
what is the reward?

- ~~Fly a helicopter~~ → **Reward**: air time, inverse distance, ...
- Manage an investment portfolio → **Reward**: gains, gains minus risk, ...
- Control a power station → **Reward**: efficiency, ...
- Make a robot walk → **Reward**: distance, speed, ...
- Play video or board games → **Reward**: win, maximise score, ...

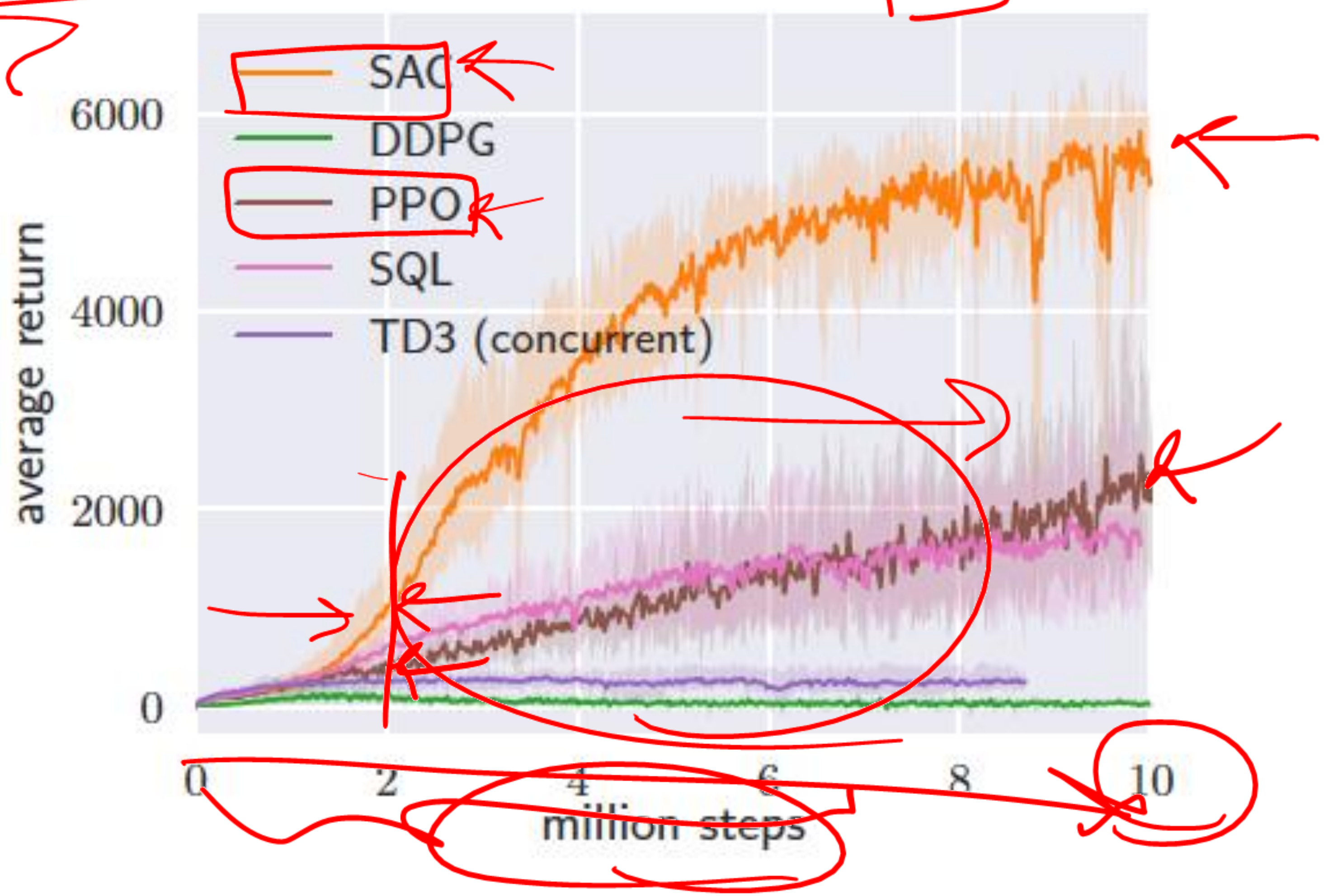
RL

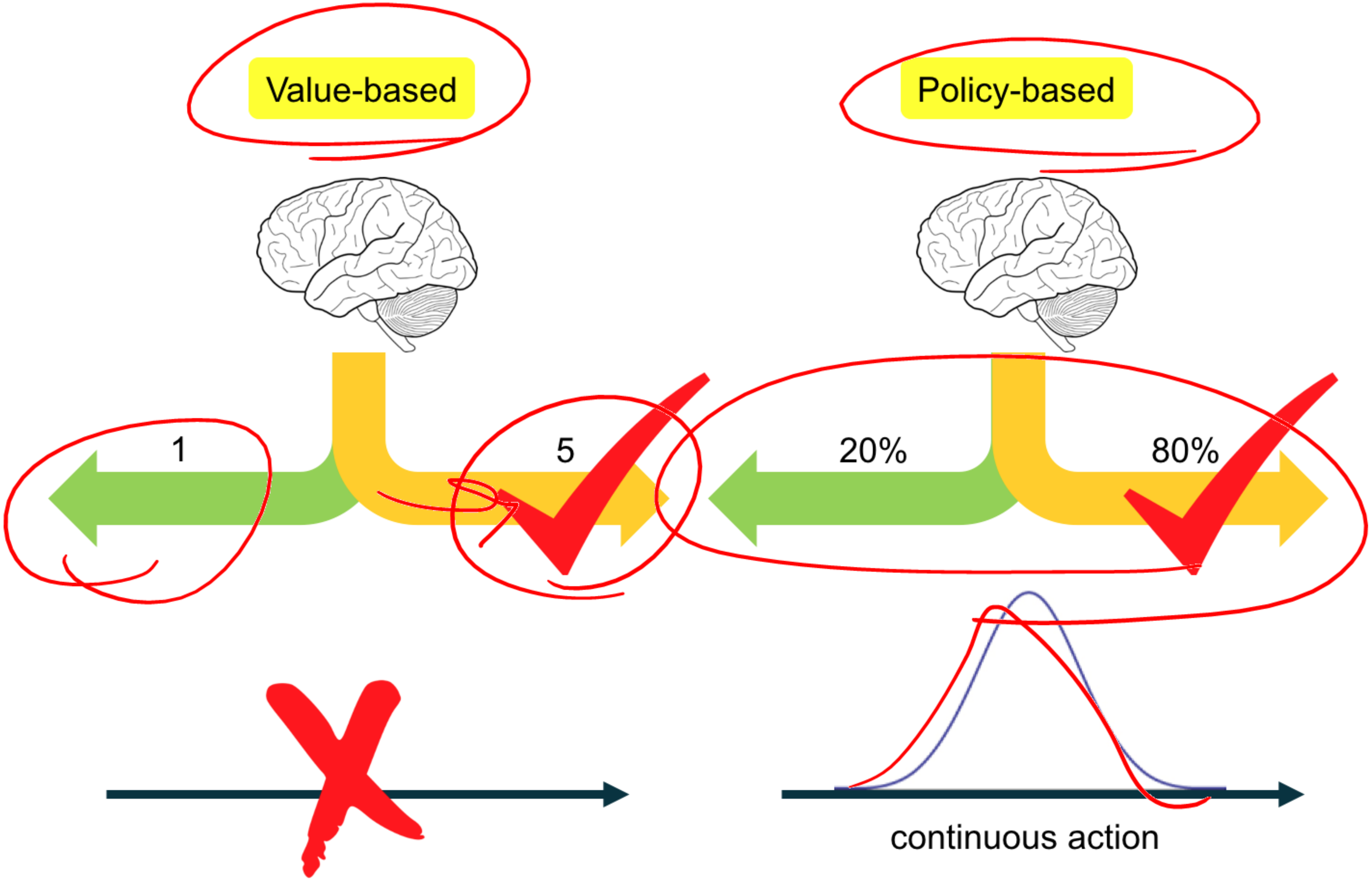
If the goal is to learn via interaction, these are all reinforcement learning problems
(Irrespective of which solution you use)

GENERALIZATION PROBLEM



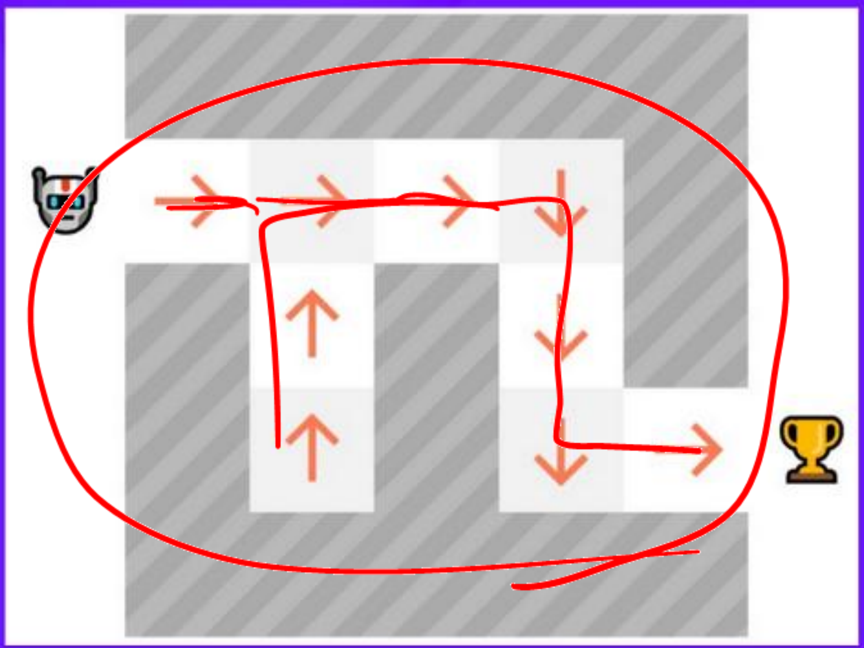
SAMPLE EFFICIENCY PROBLEM




Value-based
Policy-based
1
5
20%
80%
continuous action

Two approaches to find optimal policy π^*:

Policy-Based methods: train the agent to learn which **action to take,** given a state.

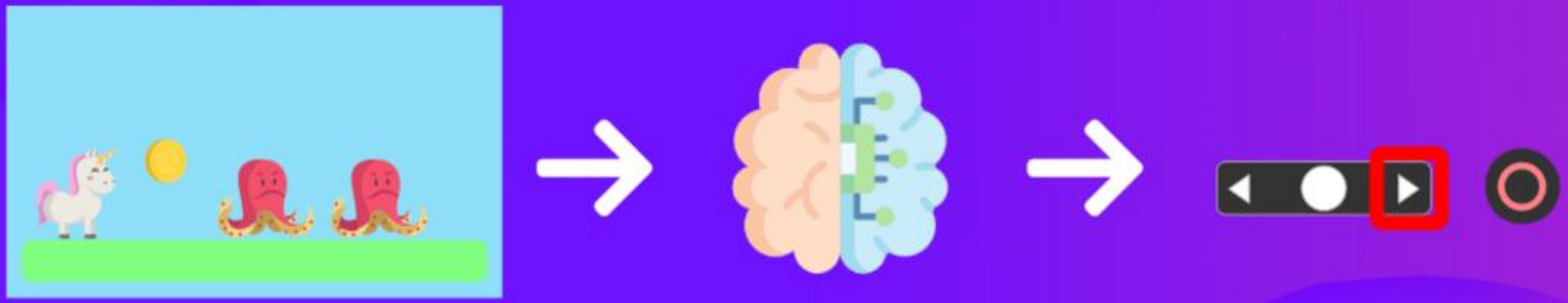


Value-Based methods: train the agent to learn which state **is more valuable** and take the action that **leads to it.**

Two approaches to find optimal policy π^*:

Policy-Based methods:

- Train **directly the policy.**
- Our policy **is a Neural Network.**
- **No value function.**

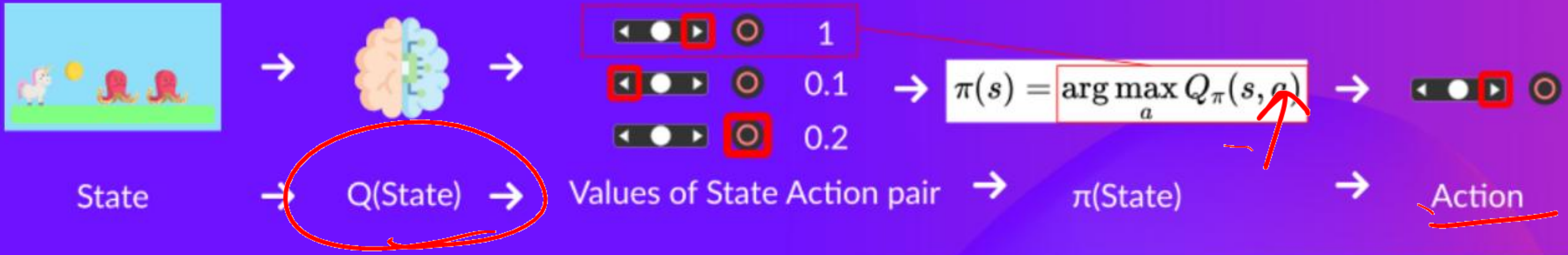


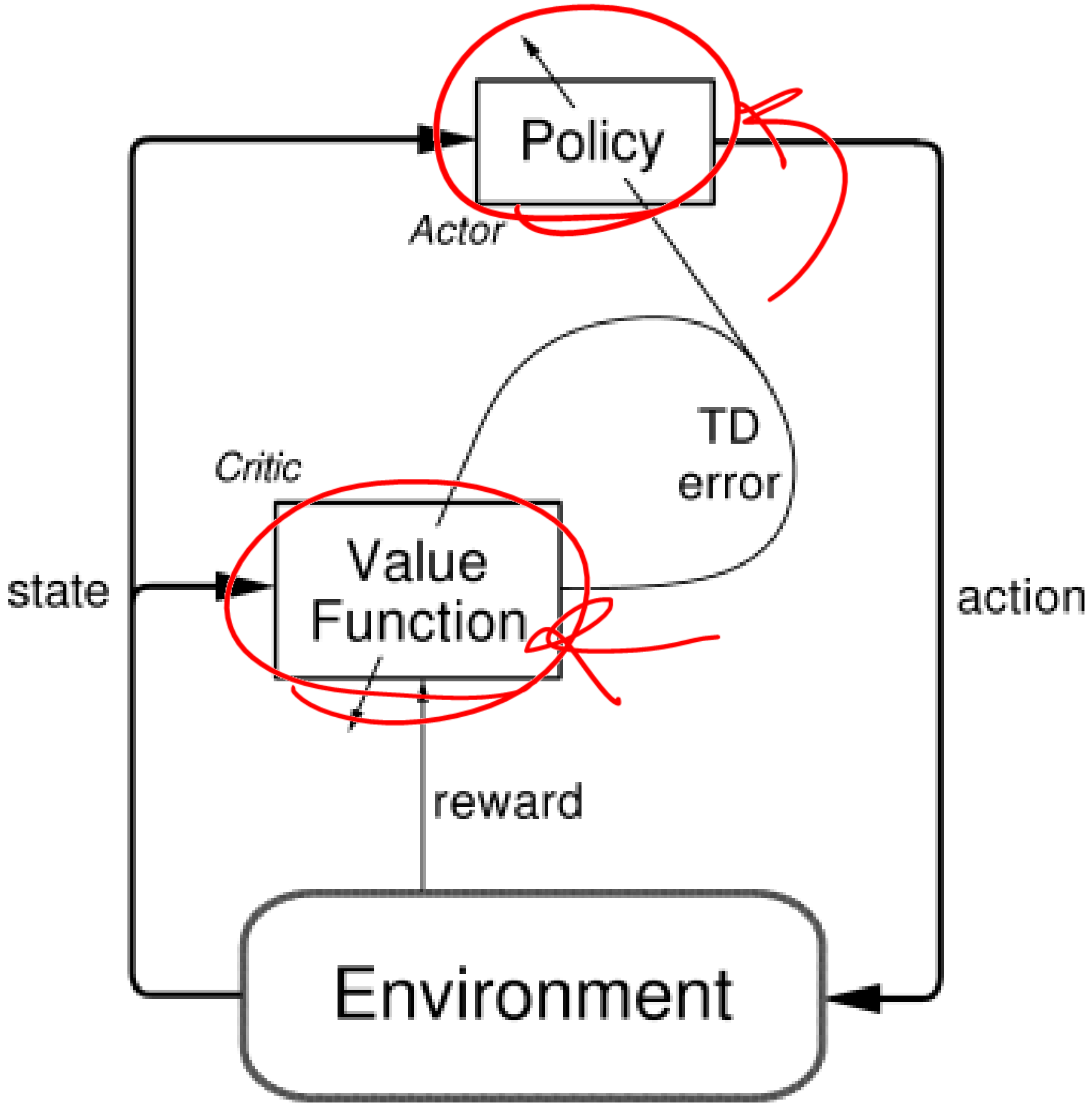
State → π(State) → Action

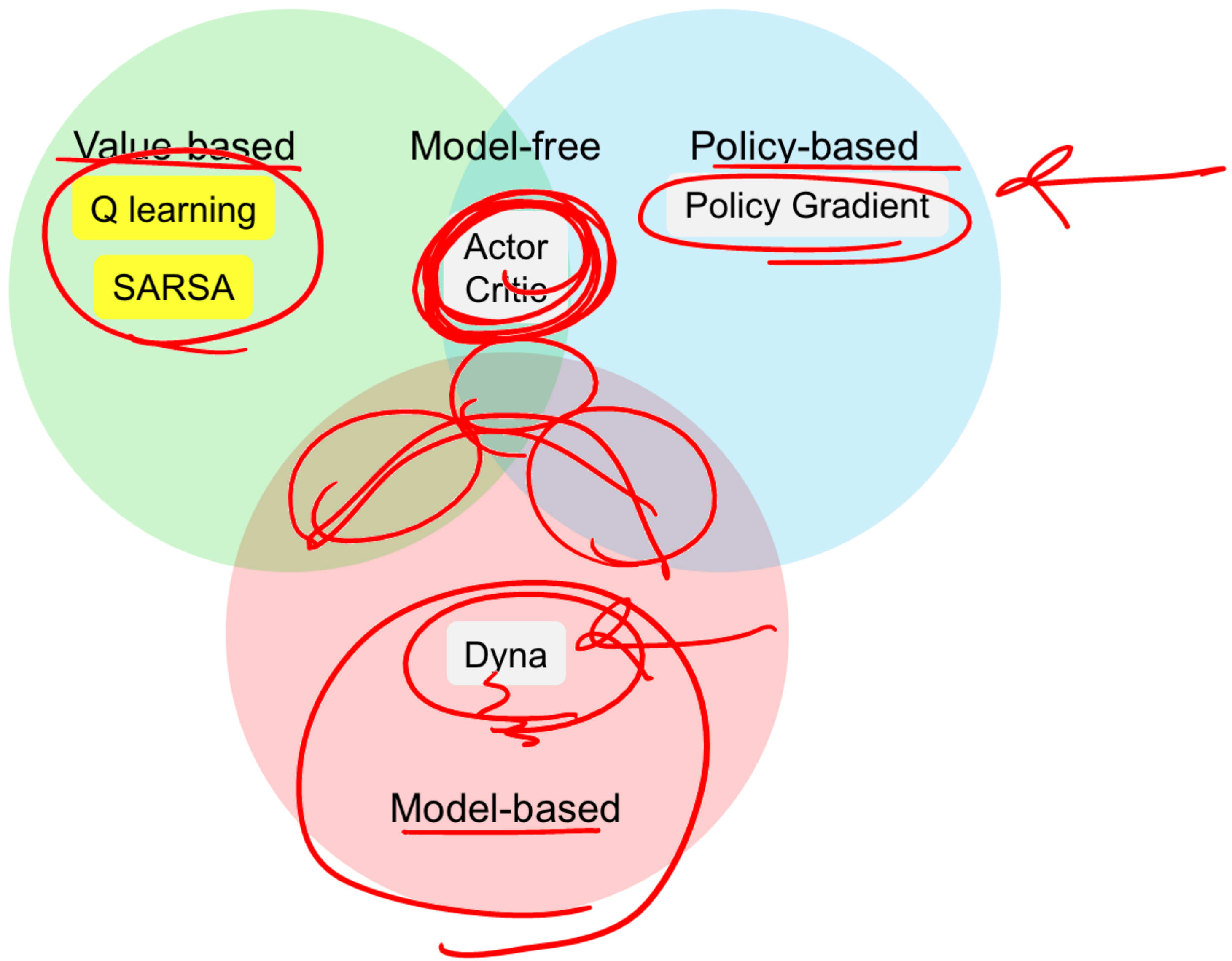
Two approaches to find optimal policy π^*:

Value-Based methods:

- Don't train the policy.
- **Our policy is a function defined by hand.**
- Instead **train a value-function that is a Neural Network.**



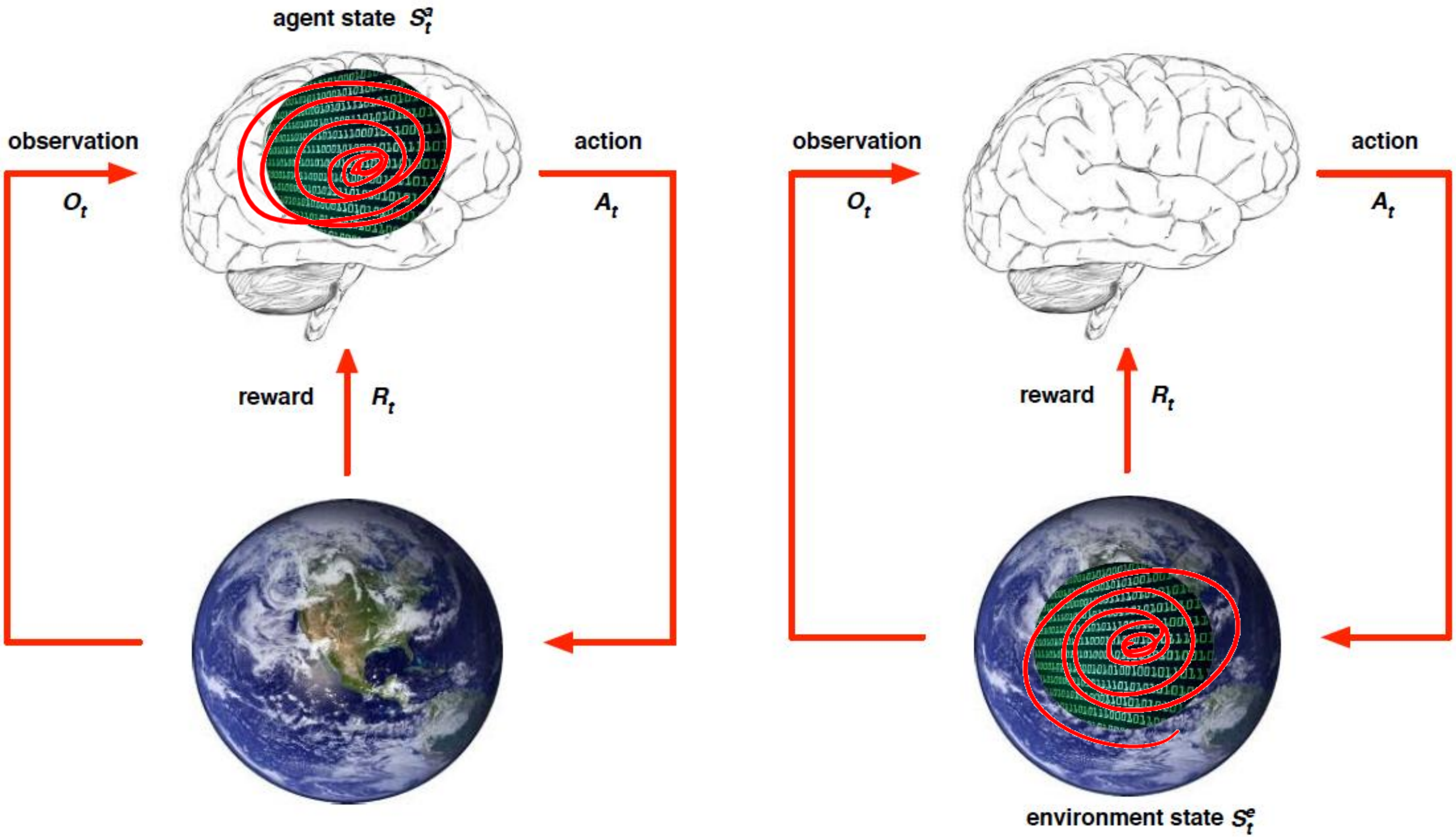

Policy
Actor
TD
error
Critic
Value
Function
state
action
reward
Environment


Value-based
Model-free
Policy-based
Q learning
SARSA
Actor Critic
Policy Gradient
Dyna
Model-based

WHAT WE HAVE LEARNED SO FAR?

- episodic vs continuing reinforcement learning
- offline vs online learning
- safe reinforcement learning
- on-policy vs off-policy vs offline reinforcement learning
- model-free vs model-base reinforcement learning
- exploration vs. exploitation dilemma
- credit assignment problem
- reward engineering problem
- generalization problem
- sample efficiency problem
- value-base vs policy-base vs actor-critic methods

MP, MRP, MDP



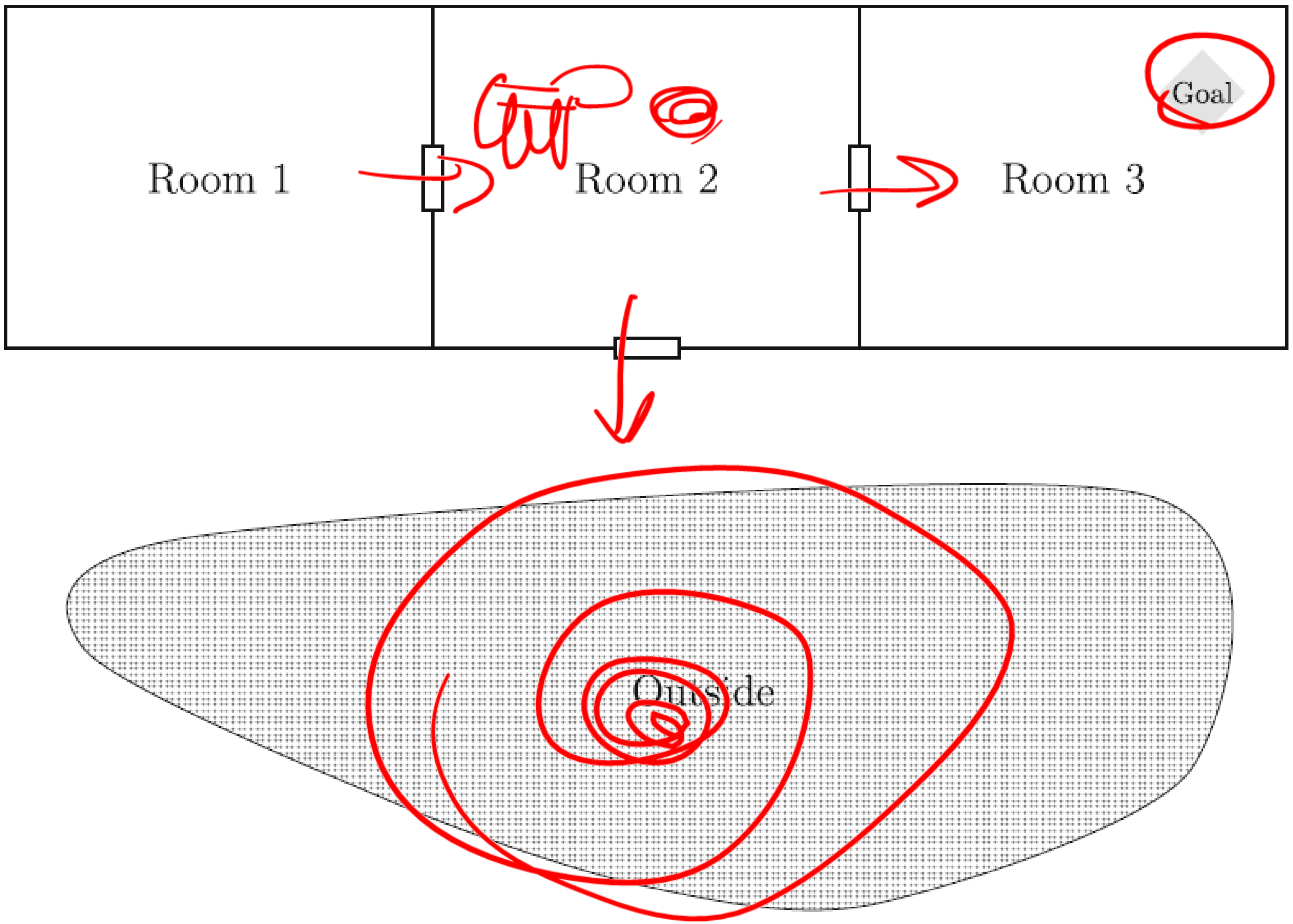
agent state S_t^a
observation
O_t
action
A_t
reward
R_t
observation
O_t
action
A_t
reward
R_t
environment state S_t^e

An information state (a.k.a. Markov state) contains all useful information from the history.

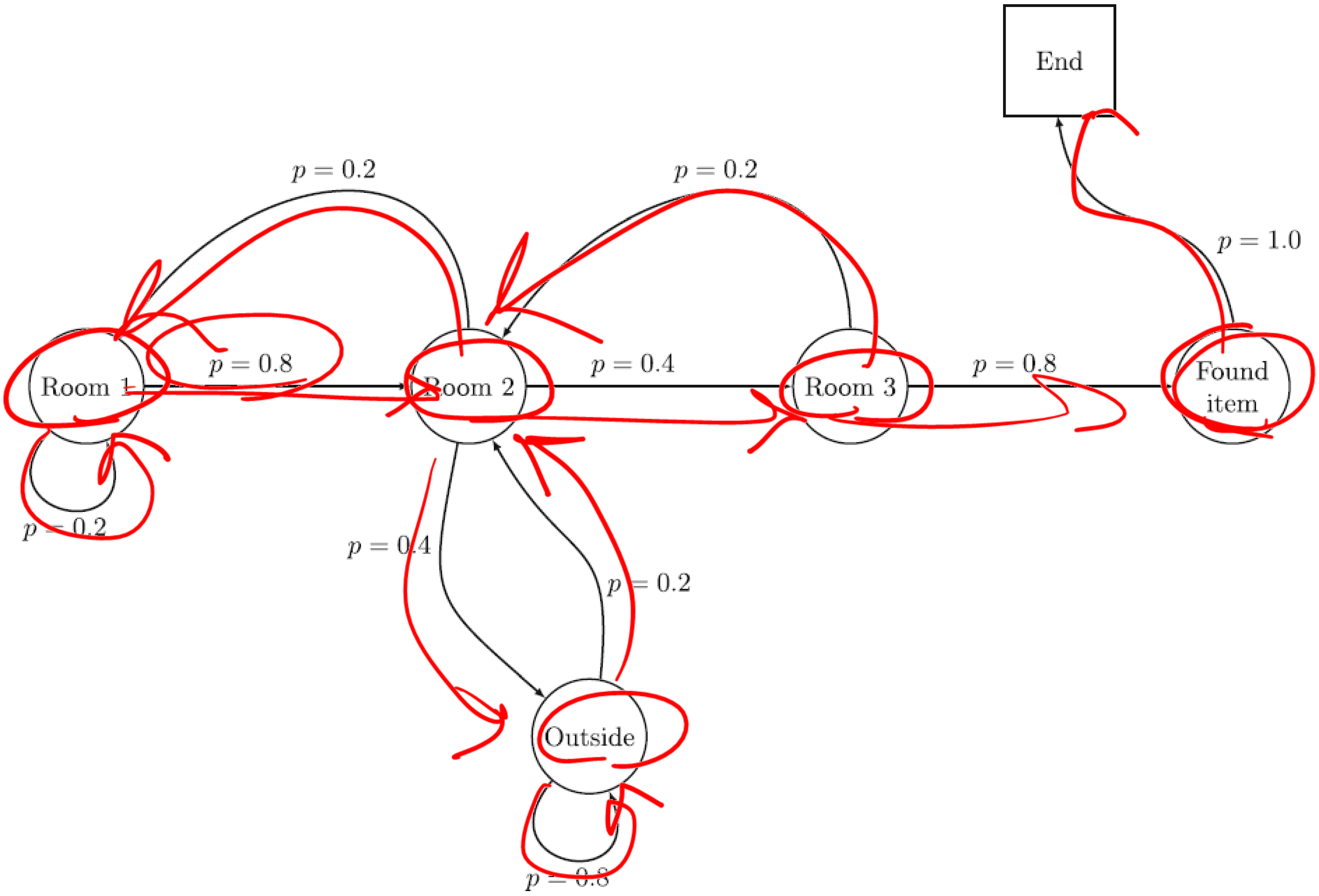
Definition

A state S_t is Markov if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, ..., S_t]$$



Room 1
Room 2
Room 3
Goal
Outside


End
p = 0.2
p = 0.2
p = 1.0
Room 1
p = 0.8
Room 2
p = 0.4
Room 3
p = 0.8
Found item
p = 0.2
p = 0.4
p = 0.2
Outside
p = 0.8

A Markov chain can be defined as a tuple of $(\mathcal{S}, \mathcal{P})$

- $\mathcal{S}$ is a finite set of states called the state space.

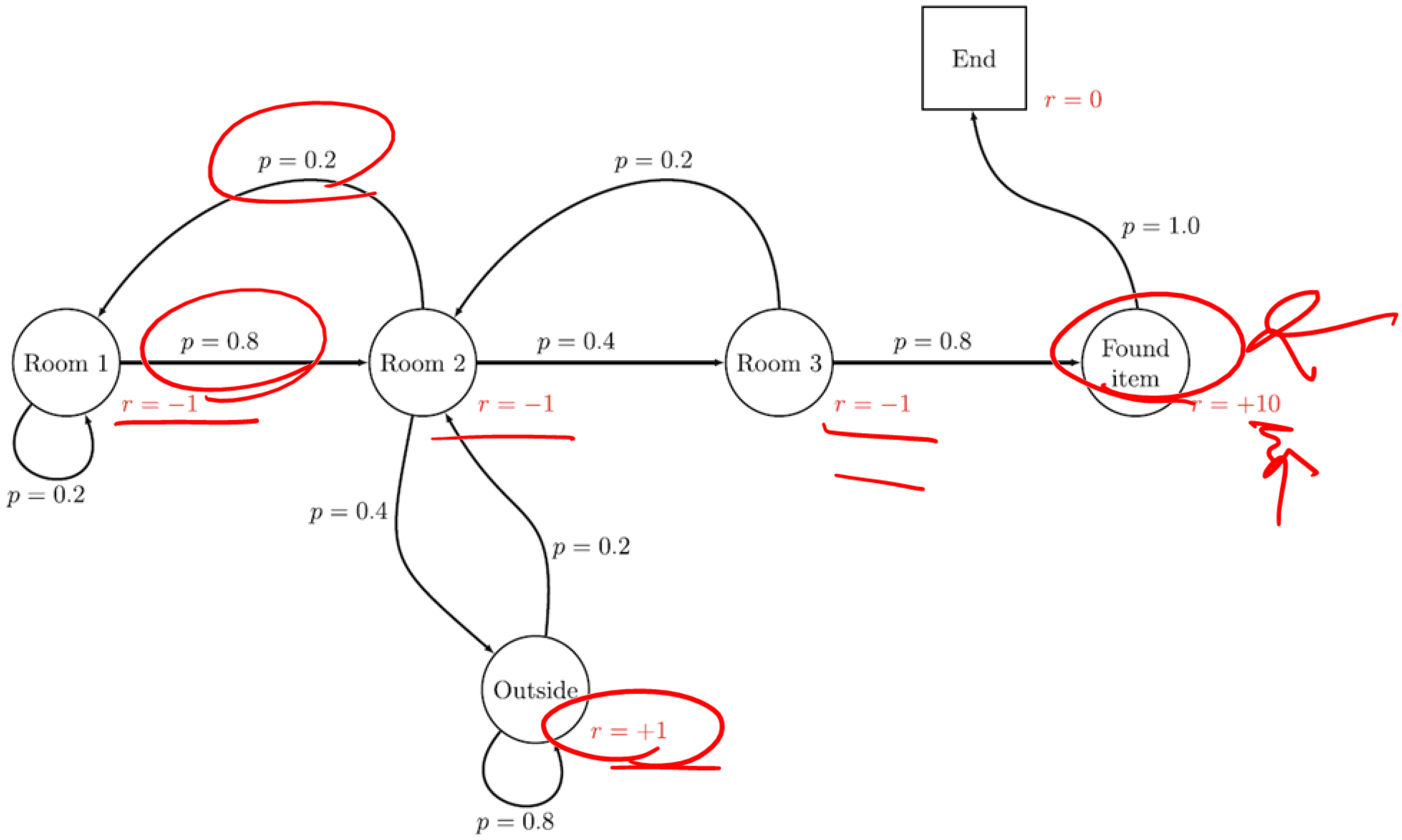
$$
\mathcal{P} = \begin{array}{r} \\ \text{Room 1} \\ \text{Room 2} \\ \text{Room 3} \\ \text{Outside} \\ \text{Found item} \\ \text{End} \end{array}
\begin{array}{c}
\begin{array}{cccccc} \text{Room 1} & \text{Room 2} & \text{Room 3} & \text{Outside} & \text{Found item} & \text{End} \end{array} \\
\begin{pmatrix}
0.2 & 0.8 & 0 & 0 & 0 & 0 \\
0.2 & 0 & 0.4 & 0.4 & 0 & 0 \\
0 & 0.2 & 0 & 0 & 0.8 & 0 \\
0 & 0.2 & 0 & 0.8 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.0 \\
0 & 0 & 0 & 0 & 0 & 1.0
\end{pmatrix}
\end{array}
$$

1

- Episode 1: (Room 1, Room 2, Room 3, Found item, End)
- Episode 2: (Room 3, Found item, End)
- Episode 3: (Room 2, Outside, Room 2, Room 3, Found item, End)
- Episode 4: (Outside, Outside, Outside, . . .)

We can define the Markov reward process as a tuple $(\mathcal{S}, \mathcal{P}, \mathcal{R})$

- $\mathcal{S}$ is a finite set of states called the state space.
- $\mathcal{P}$ is the dynamics function (or transition model) of the environment, where $P(s'|s) = P\left[S_{t+1} = s' \,\middle|\, S_t = s\right]$ specify the probability of environment transition into successor state s' when in current state s.
- $\mathcal{R}$ is a reward function of the environment. $R(s) = \mathbb{E}\left[R_t \,\middle|\, S_t = s\right]$ is the reward signal provided by the environment when the agent is in state s.

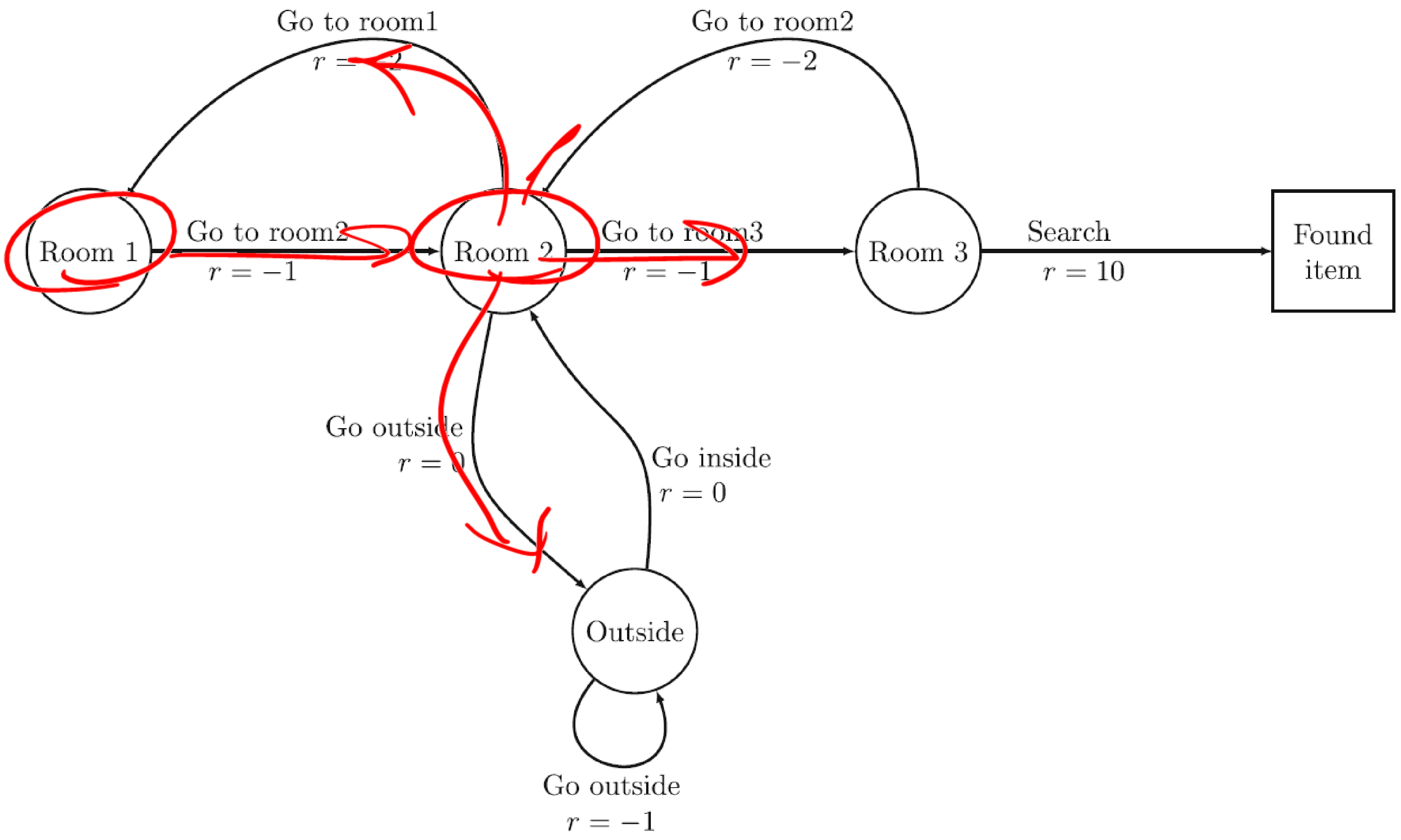


End
$r = 0$
$p = 0.2$
$p = 0.2$
$p = 1.0$
Room 1
$p = 0.8$
Room 2
$p = 0.4$
Room 3
$p = 0.8$
Found item
$r = -1$
$r = -1$
$r = -1$
$r = +10$
$p = 0.2$
$p = 0.4$
$p = 0.2$
Outside
$r = +1$
$p = 0.8$

- Episode 1: (Room 1, Room 2, Room 3, Found item, End)
  Total rewards $= -1 - 1 - 1 + 10 + 0 = 7.0$
- Episode 2: (Room 3, Found item, End)
  Total rewards $= -1 + 10 = 9.0$
- Episode 3: (Room 2, Outside, Room 2, Room 3, Found item, End)
  Total rewards $= -1 + 1 - 1 - 1 + 10 + 0 = 8.0$
- Episode 4: (Outside, Outside, Outside . . .)
  Total rewards $= 1 + 1 + \cdots = \infty$

We can define the MDP as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$:

- $\mathcal{S}$ is a finite set of states called the state space.
- $\mathcal{A}$ is a finite set of actions called the action space.
- $\mathcal{P}$ is the dynamics function (or transition model) of the environment, where $P(s'|s,a) = P\left[S_{t+1} = s' \,\middle|\, S_t = s, A_t = a\right]$ specify the probability of environment transition into successor state s' when in current state s and take action a.
- $\mathcal{R}$ is a reward function of the environment; $R(s,a) = \mathbb{E}\left[R_t \,\middle|\, S_t = s, A_t = a\right]$ is the reward signal provided by the environment when the agent is in state s and taking action a.



- $\mathcal{S}$ = {Room 1, Room 2, Room 3, Outside, Found item}
- $\mathcal{A}$ = {Go to room1, Go to room2, Go to room3, Go outside, Go ~~inside, Search~~}
- $\mathcal{R}$ = $\{-1, -2, +1, 0, +10\}$

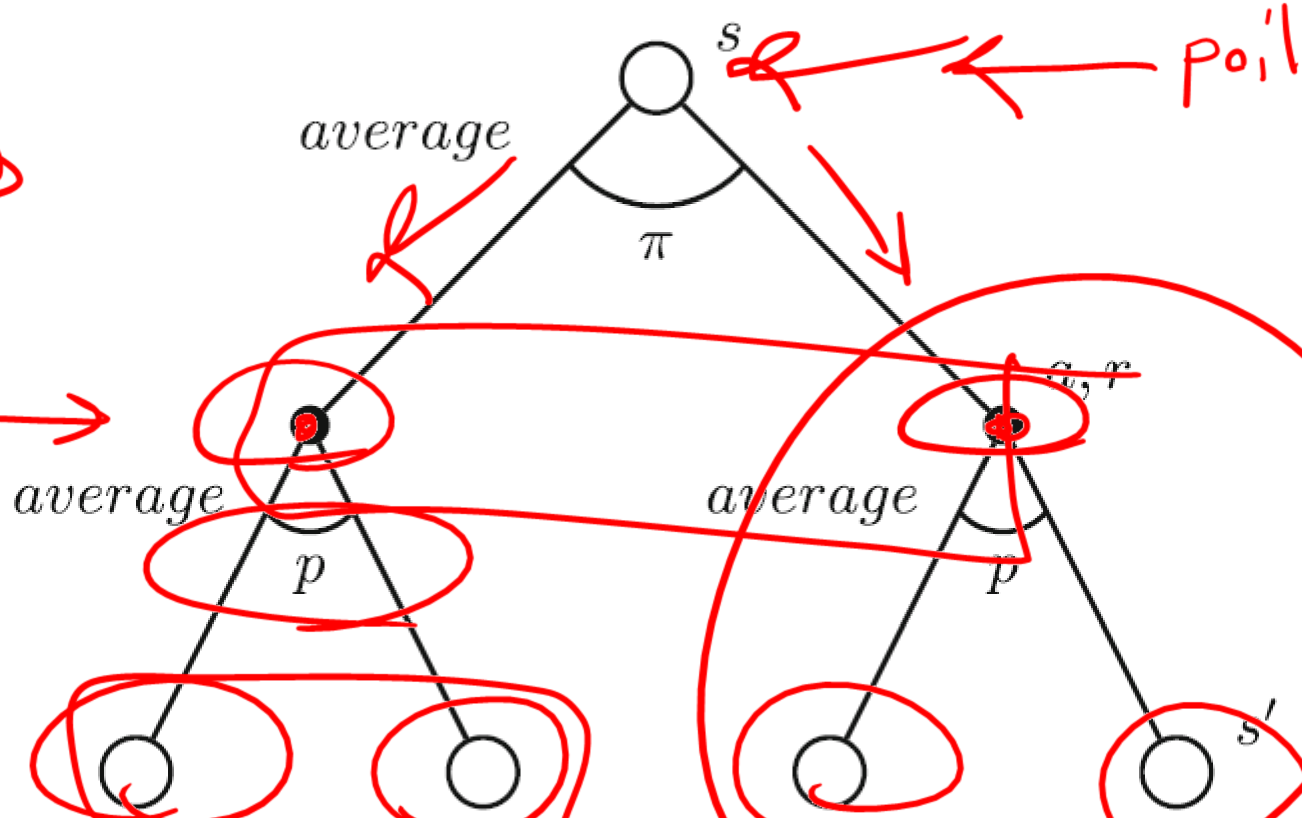
$$
\mathcal{P} = \begin{array}{r c}
 & \begin{array}{ccccc} \text{Room 1} & \text{Room 2} & \text{Room 3} & \text{Outside} & \text{Found item} \end{array} \\
\begin{array}{l} \text{Go to room1} \\ \text{Go to room2} \\ \text{Go to room3} \\ \text{Go outside} \\ \text{Go inside} \\ \text{Search} \end{array} &
\left(\begin{array}{ccccc}
1.0 & 0 & 0 & 0 & 0 \\
0 & 1.0 & 0 & 0 & 0 \\
0 & 0 & 1.0 & 0 & 0 \\
0 & 0 & 0 & 1.0 & 0 \\
0 & 1.0 & 0 & 0 & 0 \\
0 & 1.0 & 0 & 0 & 0
\end{array} \right)
\end{array}
$$

$$
\mathcal{P} = \begin{array}{r c}
 & \begin{array}{ccccc} \text{Room 1} & \text{Room 2} & \text{Room 3} & \text{Outside} & \text{Found item} \end{array} \\
\begin{array}{l} \text{Go to room1} \\ \text{Go to room2} \\ \text{Go to room3} \\ \text{Go outside} \\ \text{Go inside} \\ \text{Search} \end{array} &
\left(\begin{array}{ccccc}
0.6 & 0 & 0 & 0.4 & 0 \\
0 & 1.0 & 0 & 0 & 0 \\
0 & 0 & 0.2 & 0.8 & 0 \\
0 & 0 & 0 & 1.0 & 0 \\
0 & 1.0 & 0 & 0 & 0 \\
0 & 1.0 & 0 & 0.0 & 0
\end{array} \right)
\end{array}
$$

$$V_\pi(s) = \mathbb{E}_\pi\Big[G_t \;\Big|\; S_t = s\Big], \quad \text{for all } s \in \mathcal{S}$$

$$Q_\pi(s, a) = \mathbb{E}_\pi\Big[G_t \;\Big|\; S_t = s, A_t = a\Big], \quad \text{for all } s \in \mathcal{S}, a \in A$$

Backup Diagram

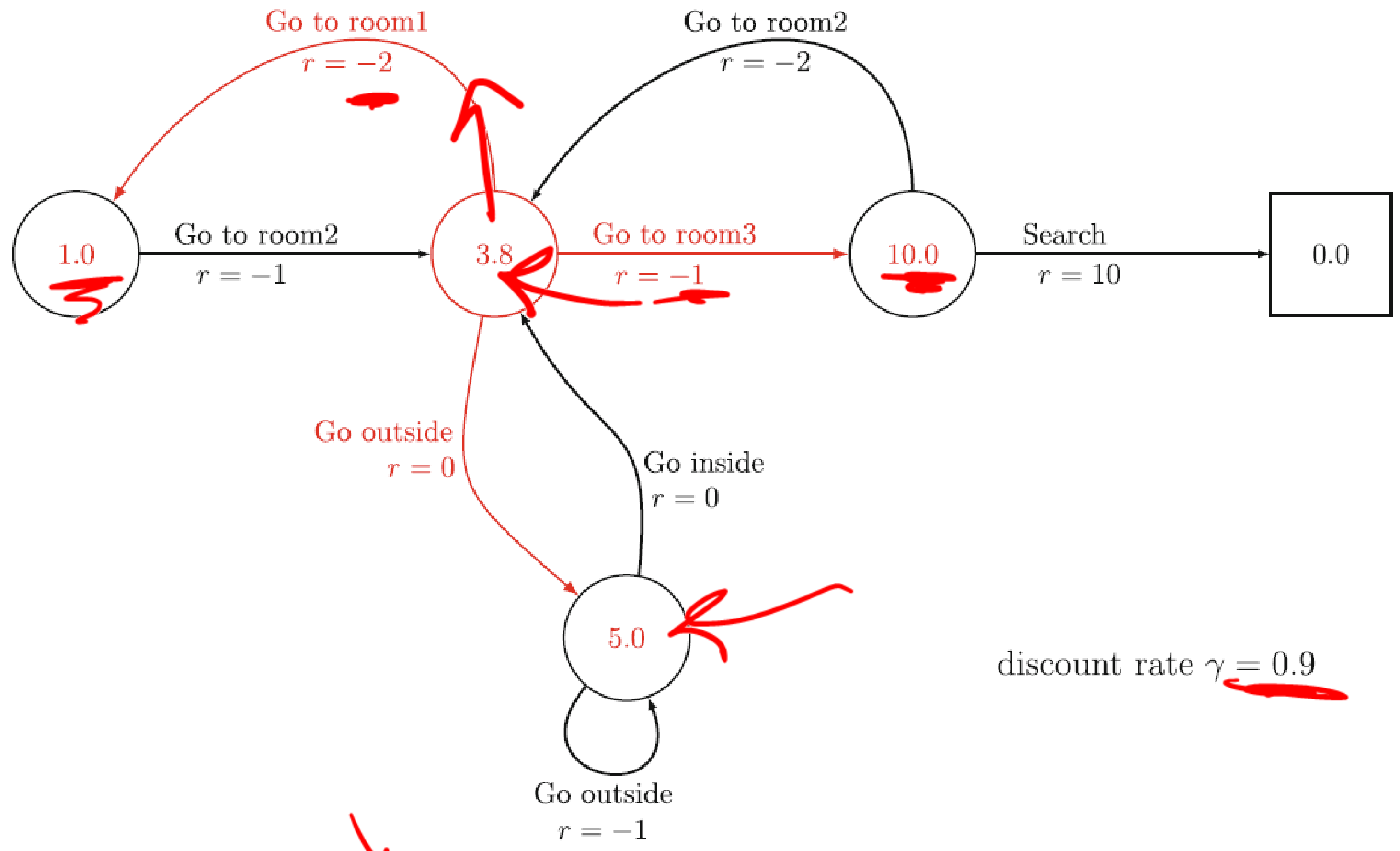


policy

env

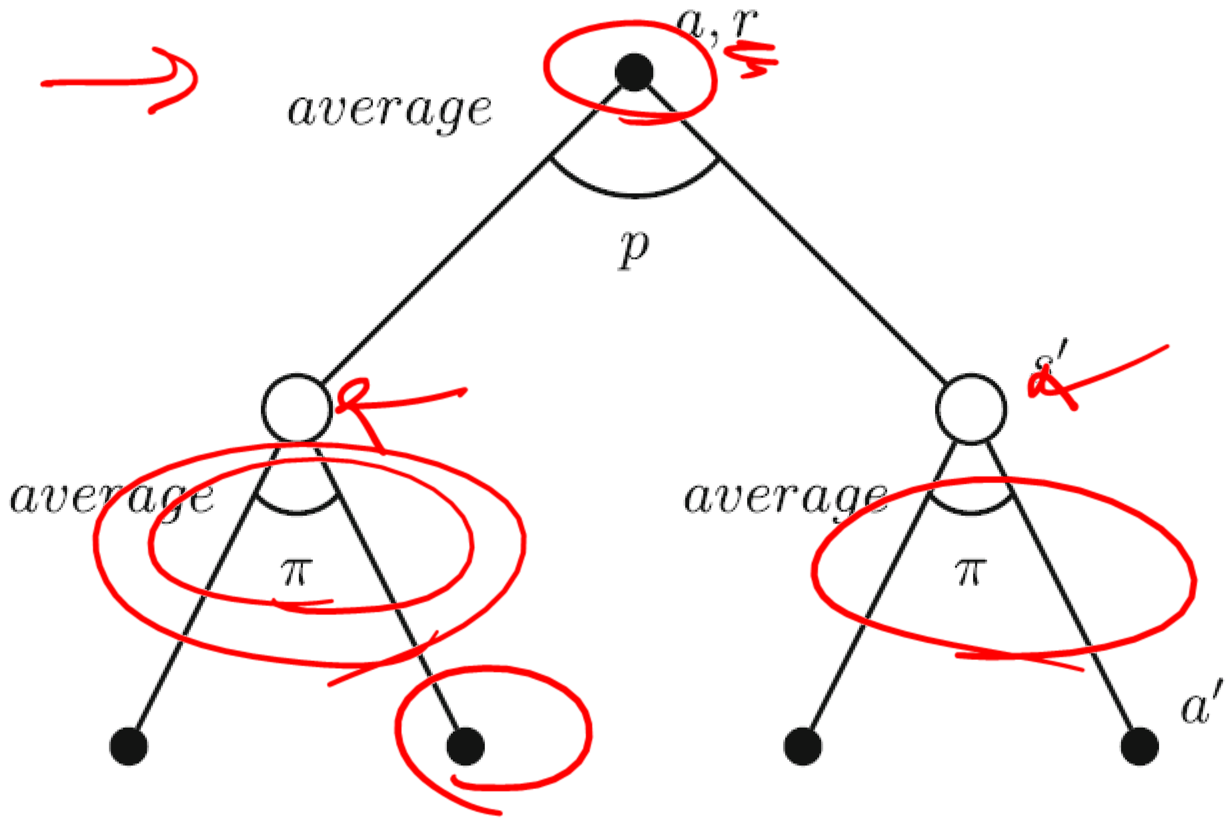
$$V_\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right]$$

$$= \sum_{a \in A} \pi(a|s)\left[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V_\pi(s')\right], \quad \text{for all } s \in \mathcal{S}$$

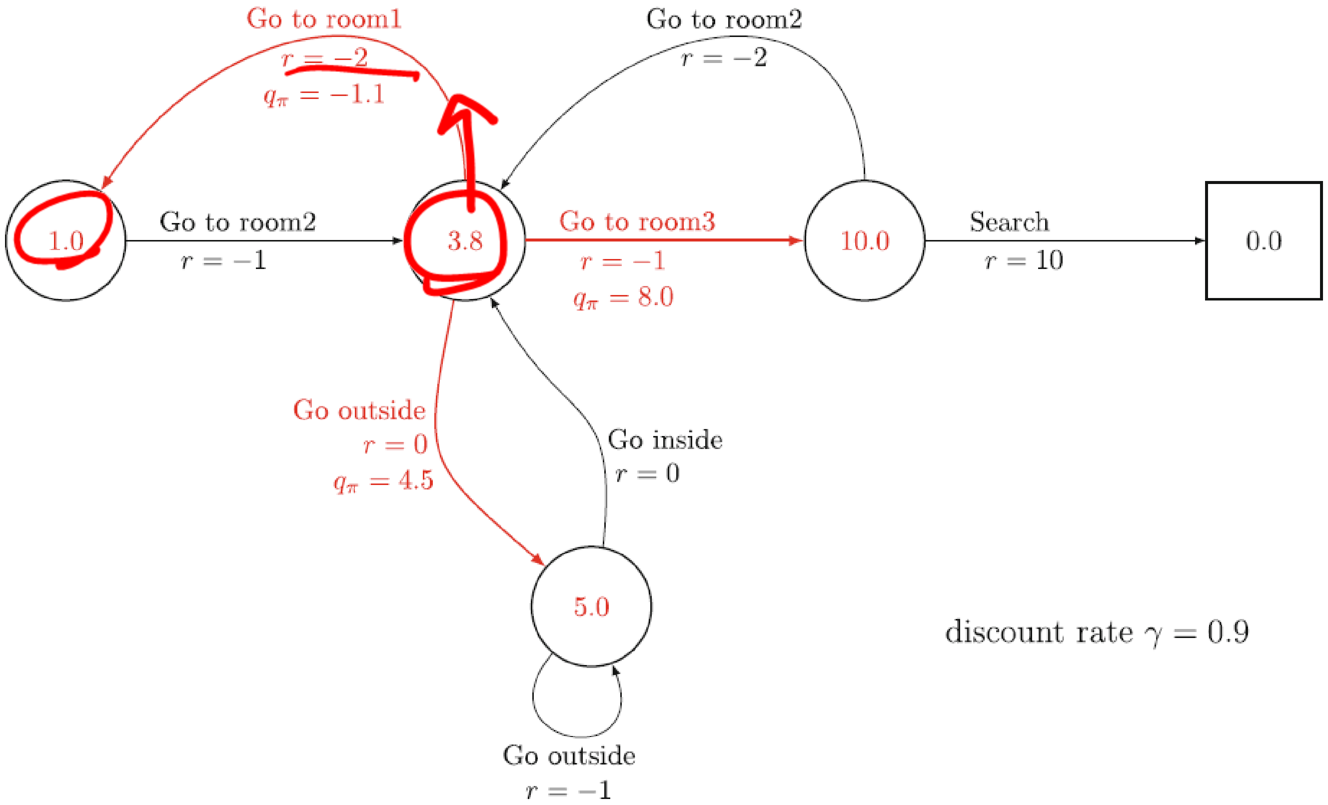


$$V_\pi(Room\ 2) = 0.33 * (-2 \boxed{+ 0.9 * 1.0)} + 0.33 * (-1 + 0.9 * 10.0)$$
$$+ 0.33 * (0 + 0.9 * 5.0)$$
$$= 0.33 * -1.1 + 0.33 * 8 + 0.33 * 4.5$$
$$= 3.76$$

env



$$
\begin{aligned}
Q_\pi(s,a) &= \mathbb{E}_\pi\Big[G_t \;\Big|\; S_t = s, A_t = a\Big] \\
&= R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \sum_{a' \in A} \pi(a'|s') Q_\pi(s',a'), \quad \text{for all } s \in \mathcal{S}, a \in A
\end{aligned}
$$



discount rate $\gamma = 0.9$

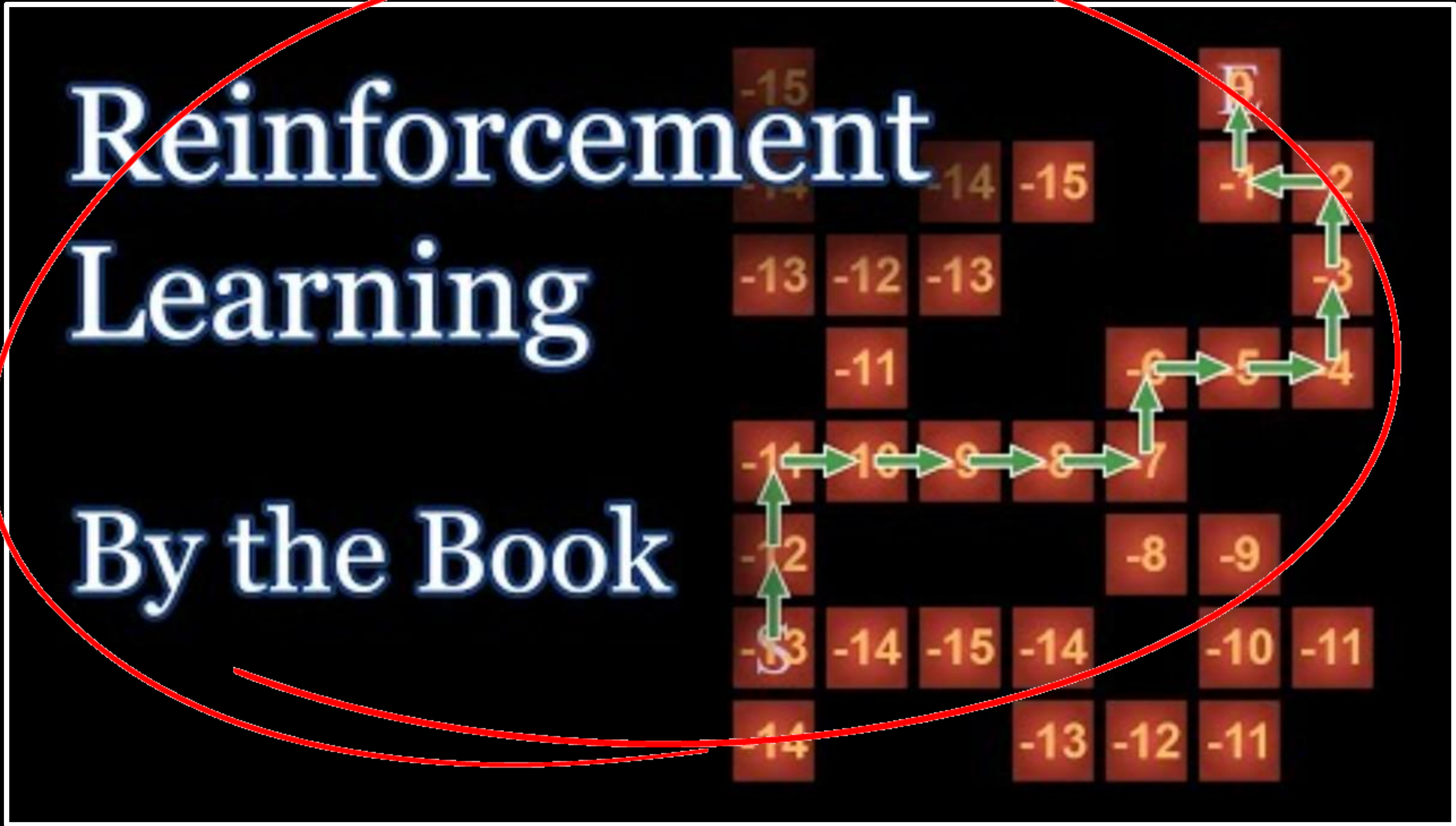
$$Q_\pi(Room\ 2,\ Go\ to\ room1) = -2 + 0.9 * 1.0$$
$$= -1.1$$

$$Q_\pi(Room\ 2,\ Go\ to\ room3) = -1 + 0.9 * 10.0$$
$$= 8.0$$

$$Q_\pi(Room\ 2,\ Go\ outside) = 0 + 0.9 * 5.0$$
$$= 4.5$$

$$V_\pi(Room\ 2) = 0.33 * -1.1 + 0.33 * 8 + 0.33 * 4.5$$
$$= 3.76$$

WATCH THE FOLLOWING VIDEO



https://www.youtube.com/watch?v=NFo9v_yKQXA

How to solve full RL problem?

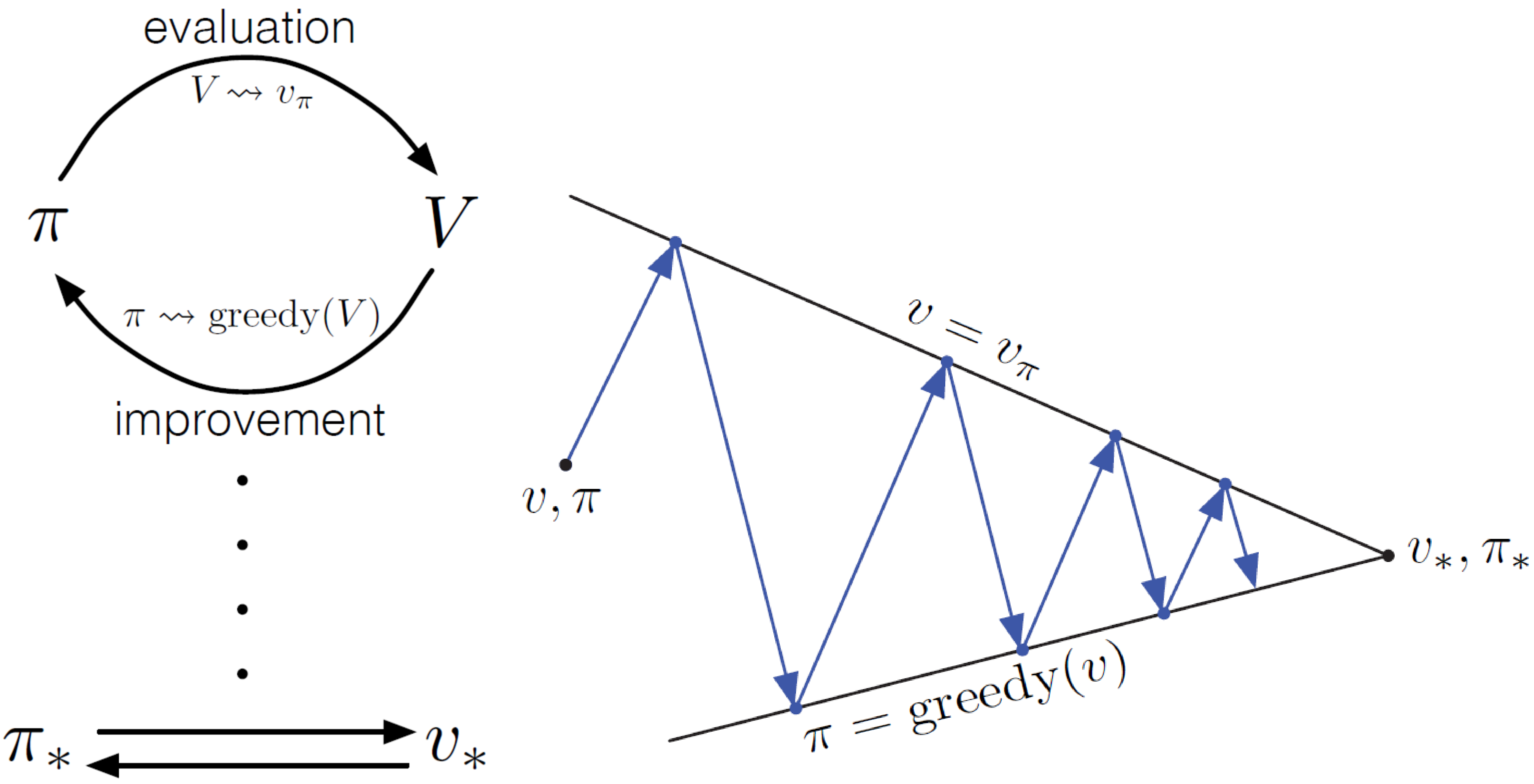
When we have:

$$P(s', r|s, a) = \mathbb{P}[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a]$$

OPTIMAL VALUE AND POLICY

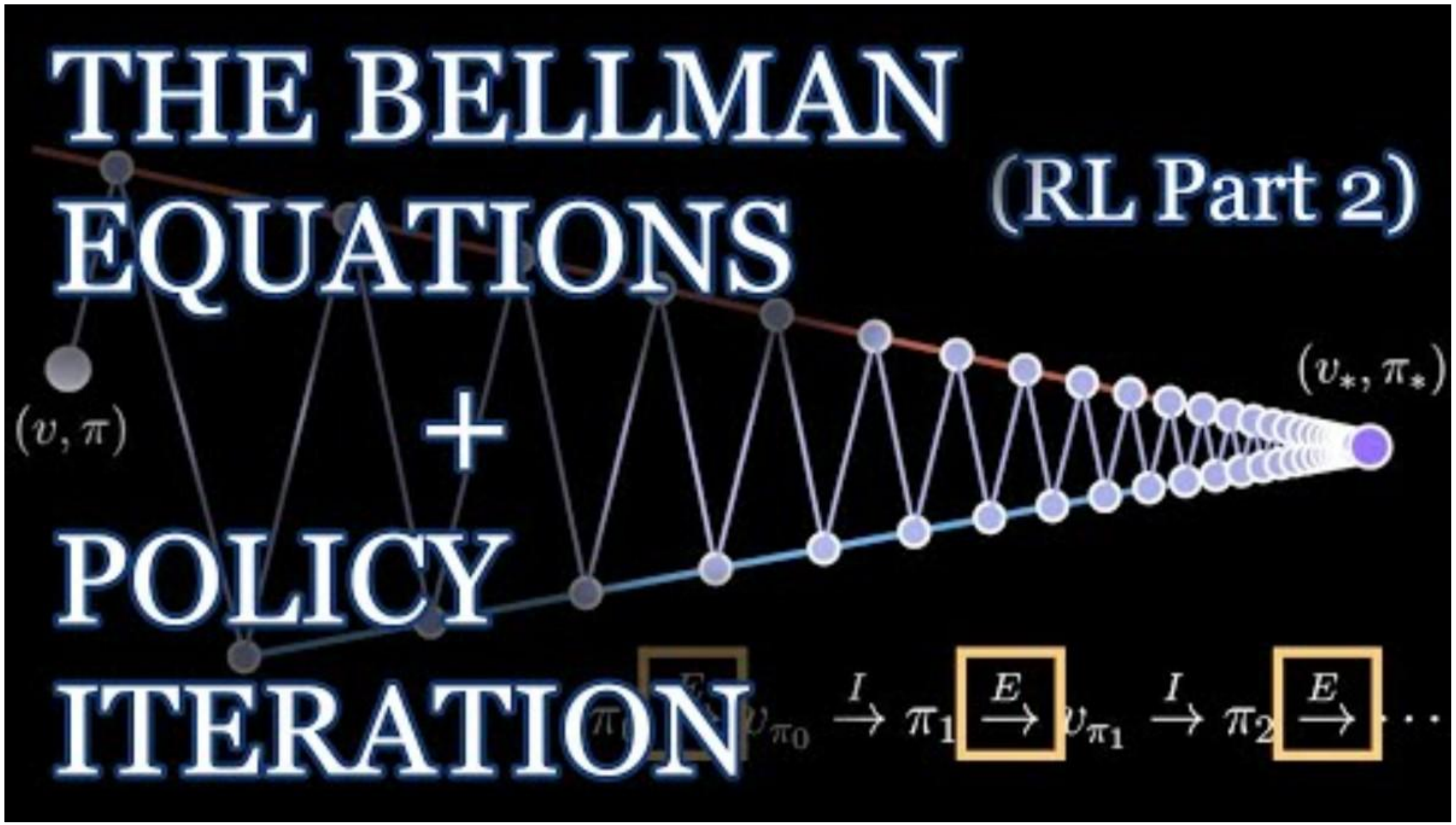
$$Q_*(s,a) = \max_{\pi} Q_\pi(s,a), \quad \text{for all } s \in \mathcal{S}, a \in \mathcal{A}$$

$$\pi_*(a|s) = \begin{cases} 1, & \text{if} \quad a = \arg\max_{a \in A} Q_*(s,a) \\ 0, & \text{otherwise} \end{cases}$$



evaluation
$V \rightsquigarrow v_\pi$
π
V
$\pi \rightsquigarrow \mathrm{greedy}(V)$
improvement
π_*
v_*
$v = v_\pi$
v, π
v_*, π_*
$\pi = \mathrm{greedy}(v)$

WATCH THE FOLLOWING VIDEO



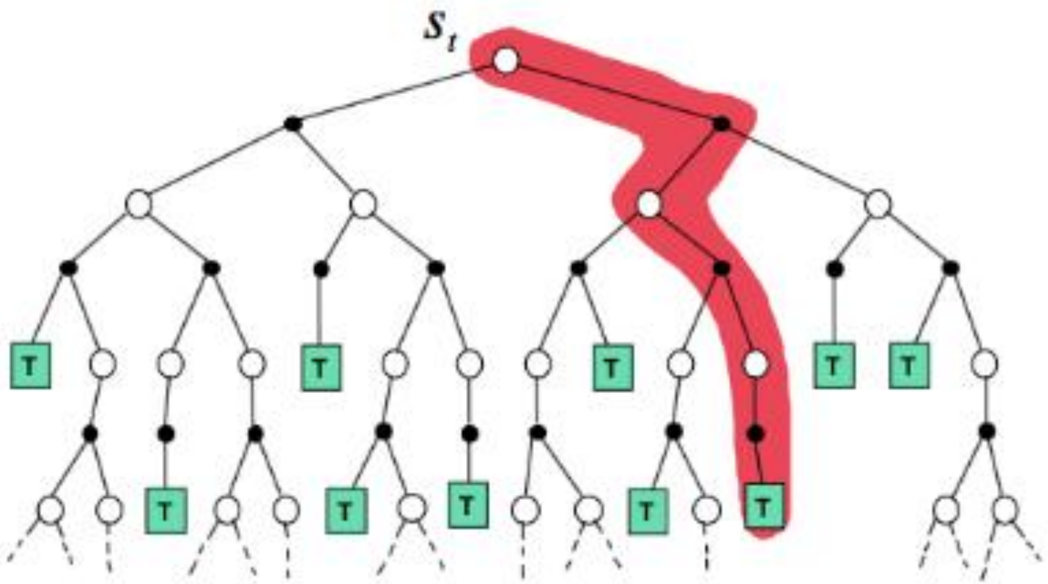
https://www.youtube.com/watch?v=_j6pvGEchWU

When we don't have:

$$\cancel{P(s', r|s, a) = \mathbb{P}[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a]}$$

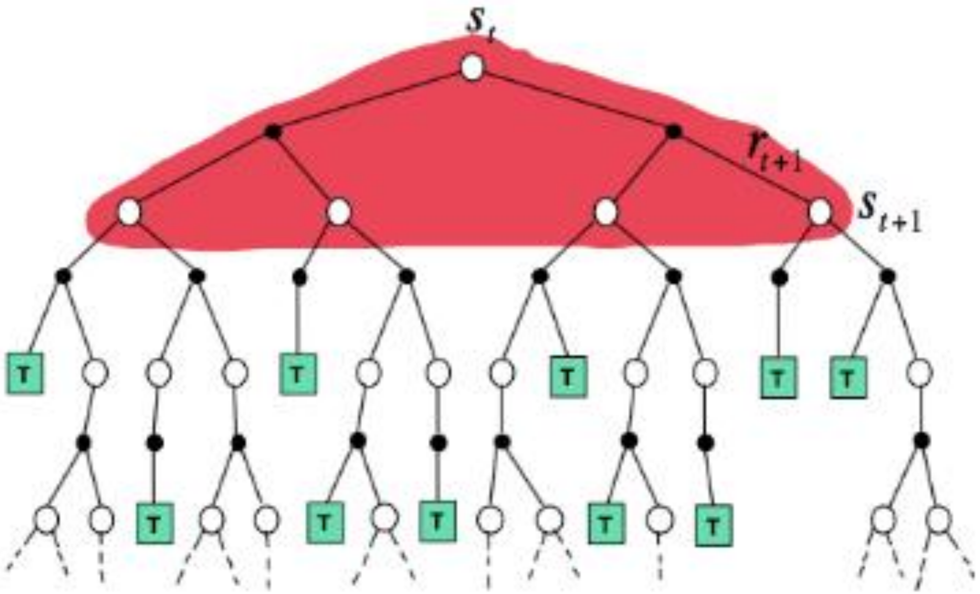
Monte-Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

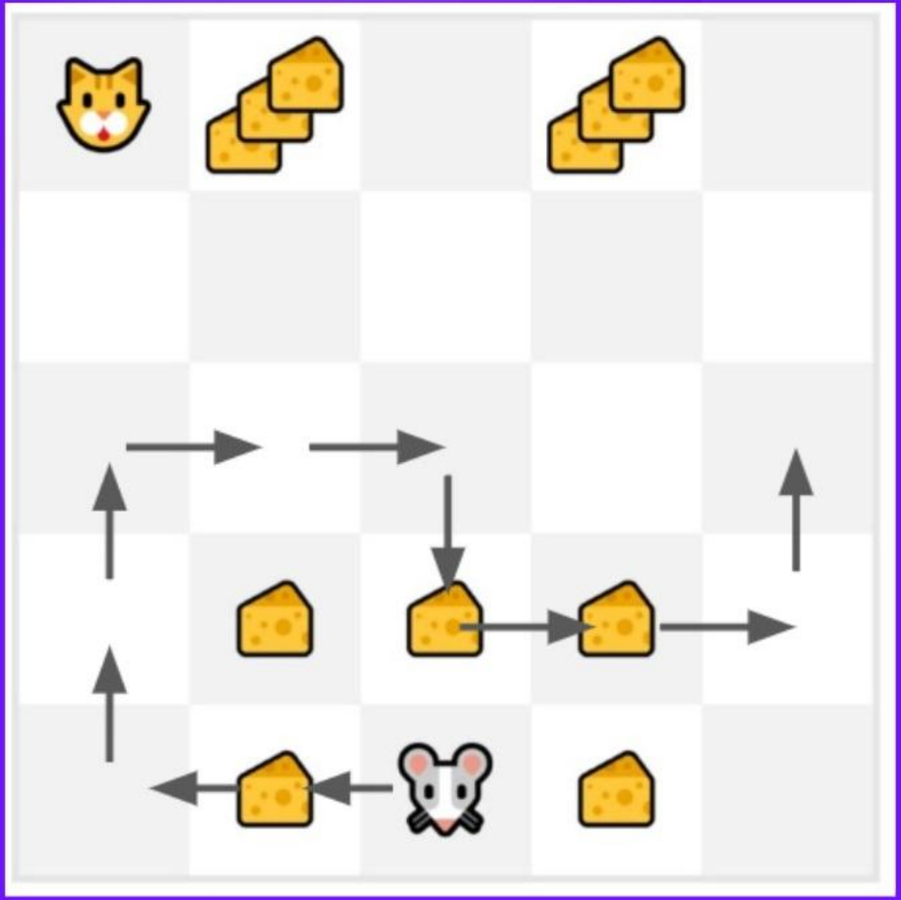


Dynamic Programming

$$V(S_t) \leftarrow \mathbb{E}_\pi\left[R_{t+1} + \gamma V(S_{t+1})\right]$$



Monte Carlo Approach:



- **Calculate the return Gt.**

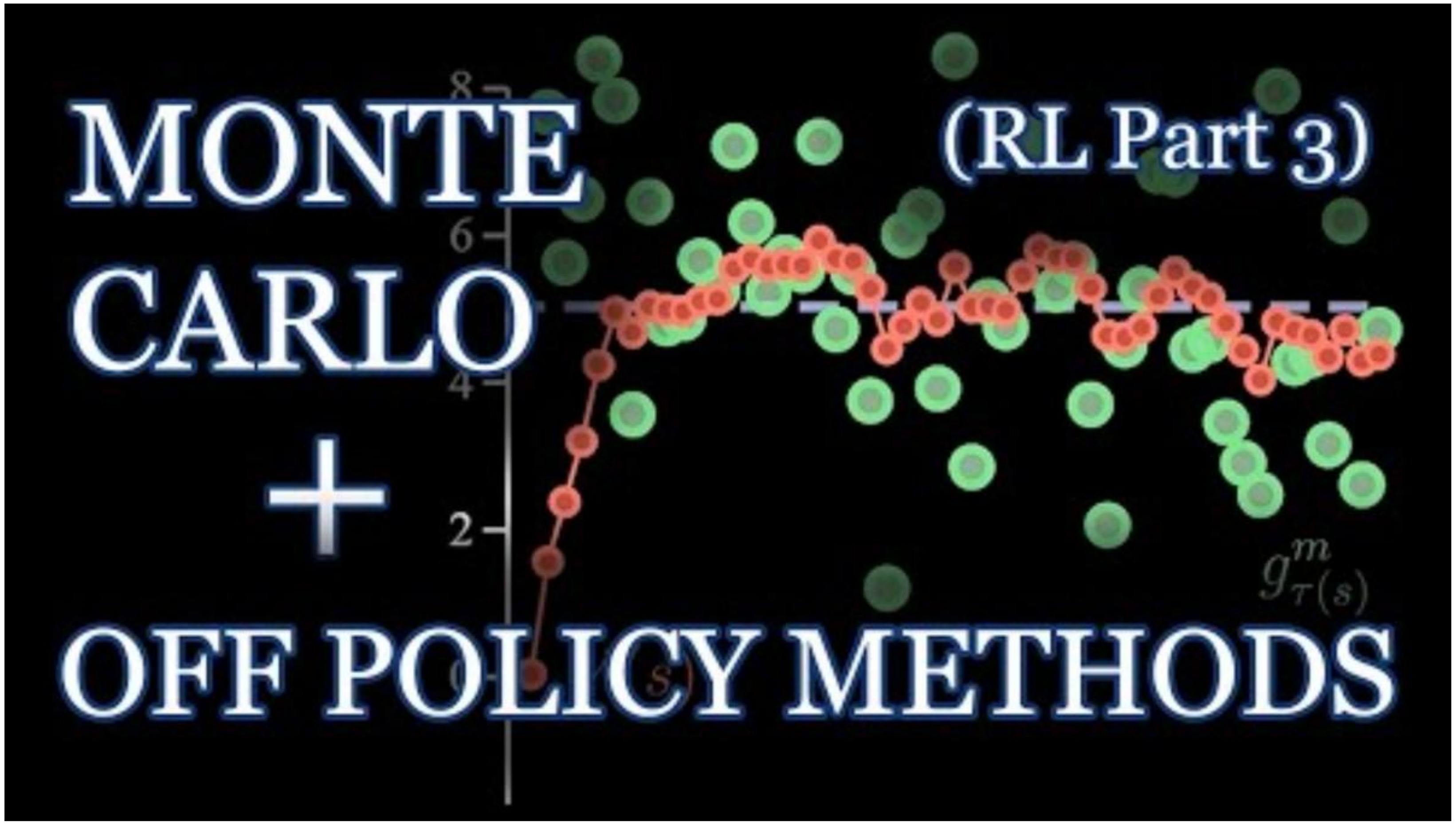
Gt = Rt+1 + Rt+2 + Rt+3…
Gt = 1 + 0 + 0 + 0+ 0 + 0 + 1 + 1+ 0 + 0
Gt= 3

- **We can now update V(S0).**

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

New V(S0) = V(S0) + lr * [Gt-V(S0)]
New V(S0) = 0 + 0.1 * [3 –0]
New V(S0) = 0.3

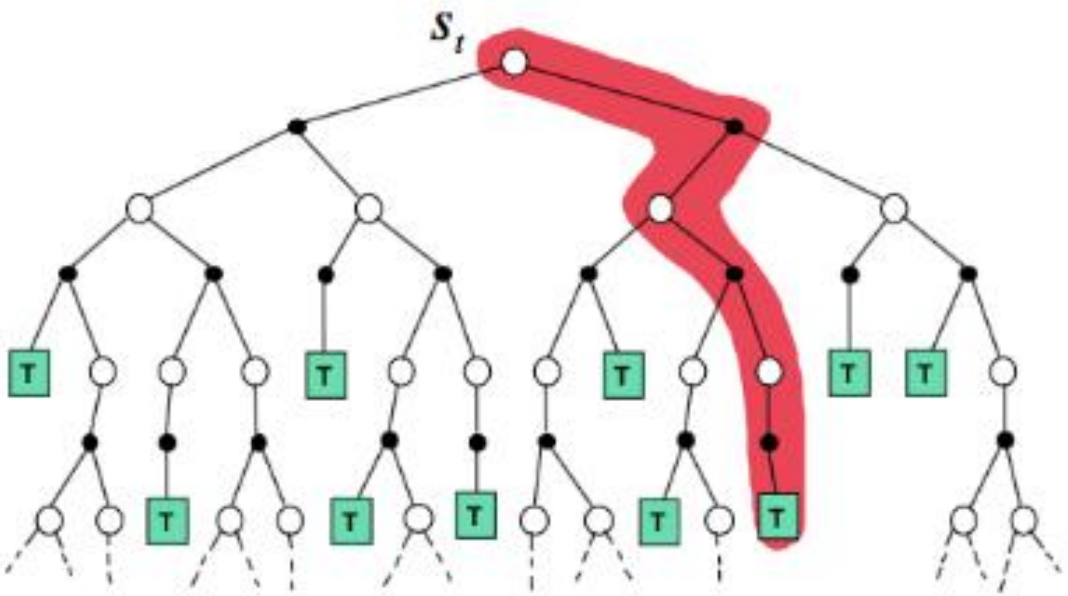
WATCH THE FOLLOWING VIDEO



https://www.youtube.com/watch?v=bpUszPiWM7o

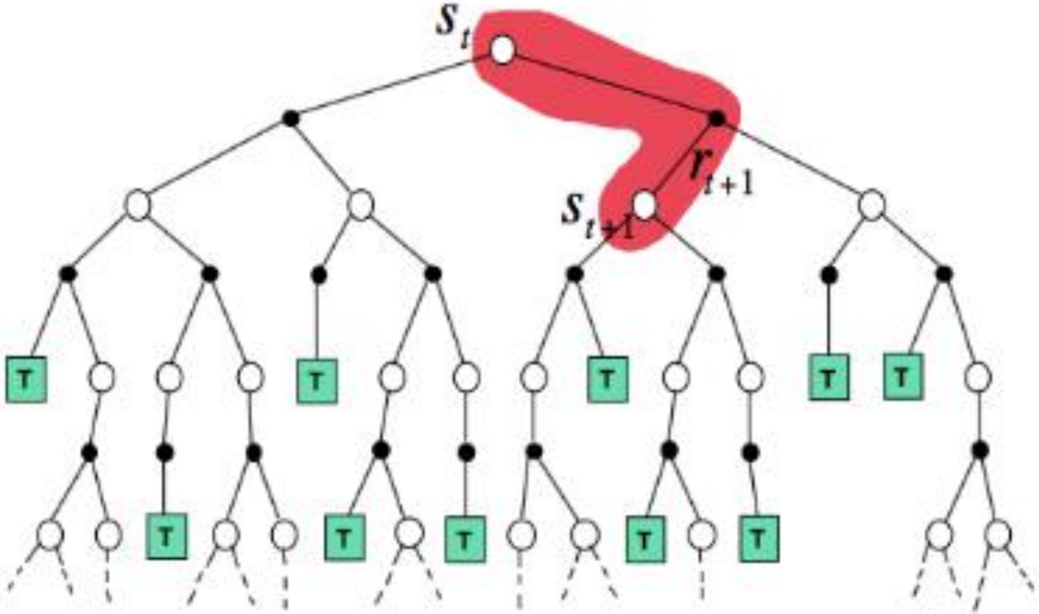
Monte-Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$$



Temporal-Difference

$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$



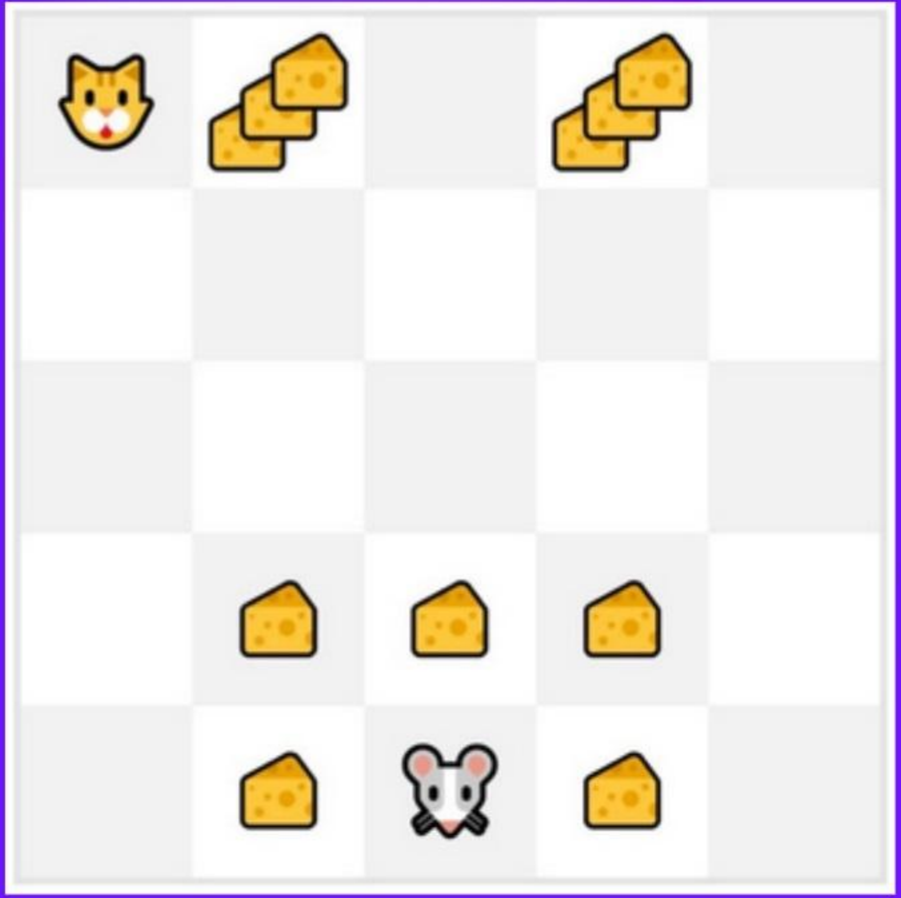
https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_td.html

TD Learning Approach:

***Temporal Difference Learning*:** learning at **each time step.**

$$\underbrace{V(S_t)}_{\text{New value of state t}} \leftarrow \underbrace{V(S_t)}_{\text{Former estimation of value of state t}} + \underbrace{\alpha}_{\text{Learning Rate}}[\underbrace{\underbrace{R_{t+1}}_{\text{Reward}} + \underbrace{\gamma V(S_{t+1})}_{\text{Discounted value of next state}}}_{\text{TD Target}} - V(S_t)]$$

TD Approach:



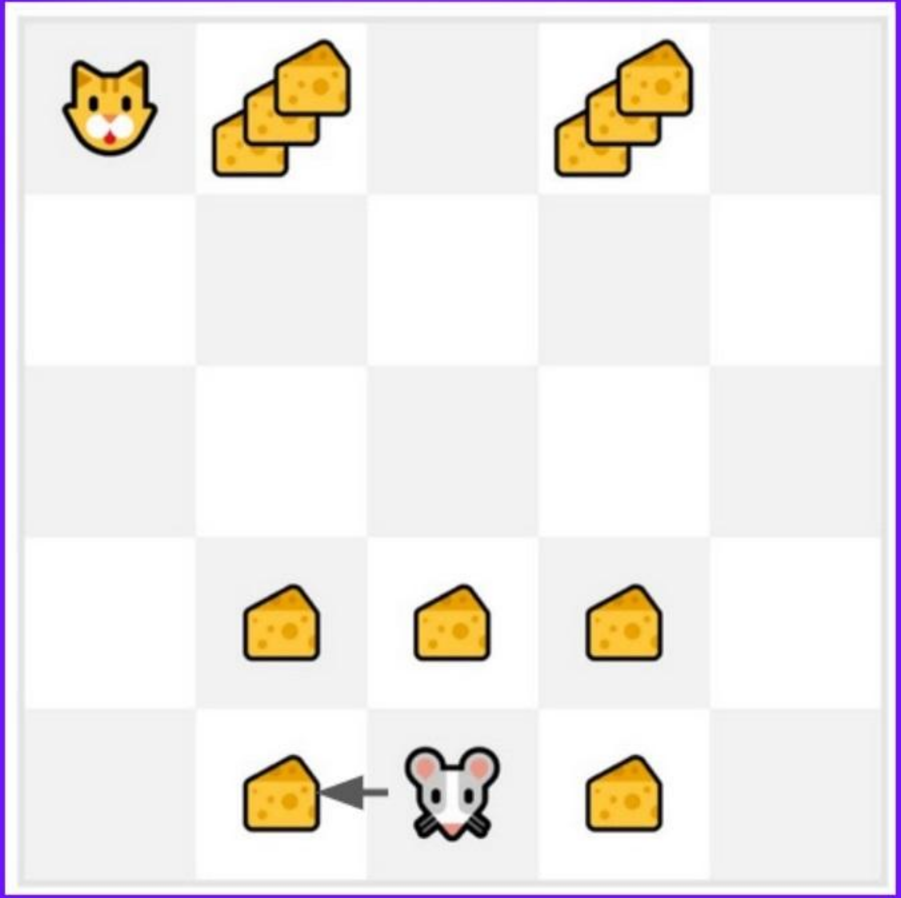
At the end of one step (State, Action, Reward, Next State):

- We have Rt+1 and St+1
- We update V(St):
    - **We estimate Gt** by adding Rt+1 and the discounted value of next state. **TD target** : Rt+1 + gamma * V(St+1)

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

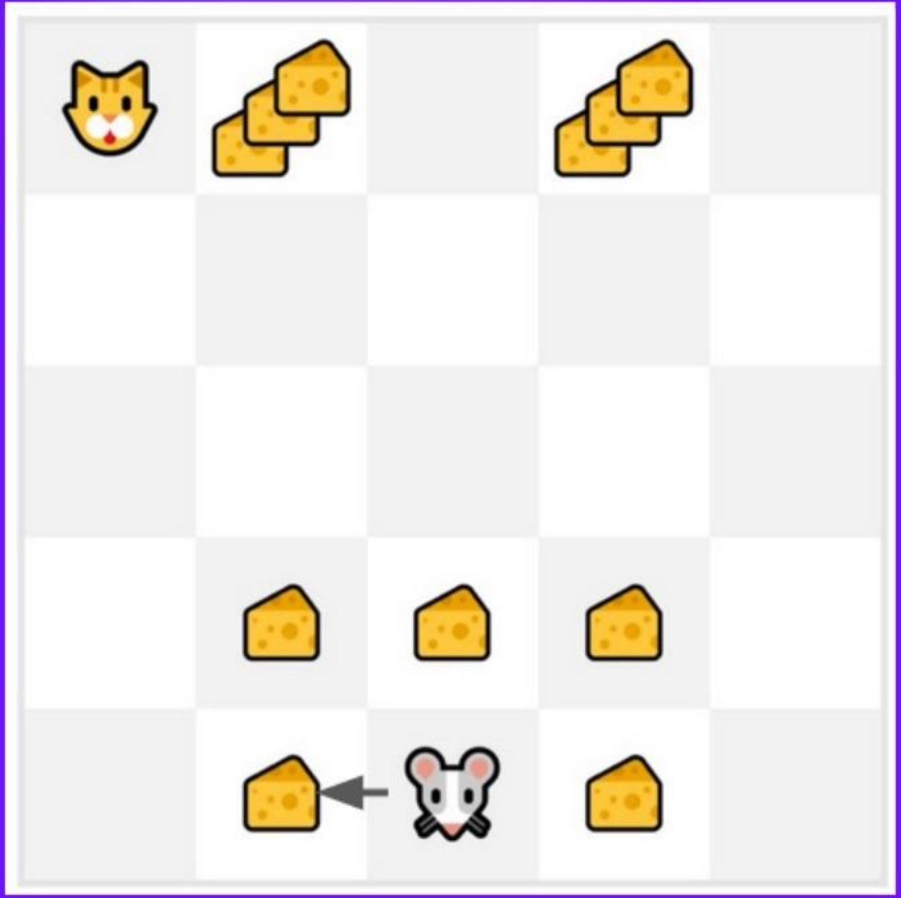
Now we **continue to interact with this environment** with our updated value function. By running more and more steps, **the agent will learn to play better and better.**

TD Approach:



- We just started to train our Value function **so it returns 0 value for each state.**
- Learning rate (lr) is 0.1 and our discount rate is 1 **(no discount)**
- Our mouse, **explore the environment** and take a random action: going left.
- It gets **a +1 reward (cheese).**

TD Approach:



- **We can now update V(S0):**

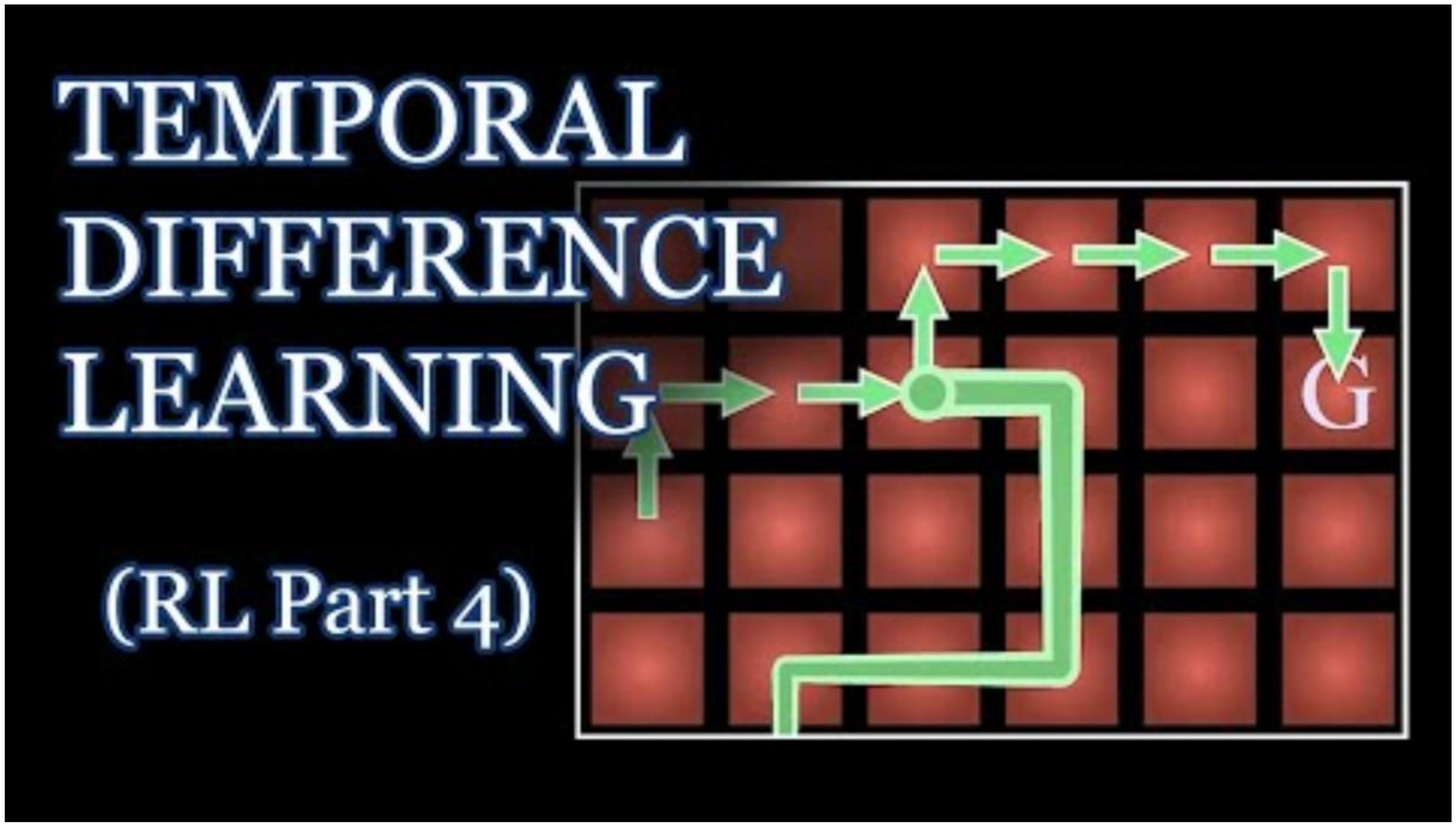
$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

New V(S0) = 0 + 0.1 * [1 + 1 * 0–0]
The new V(S0) = 0.1

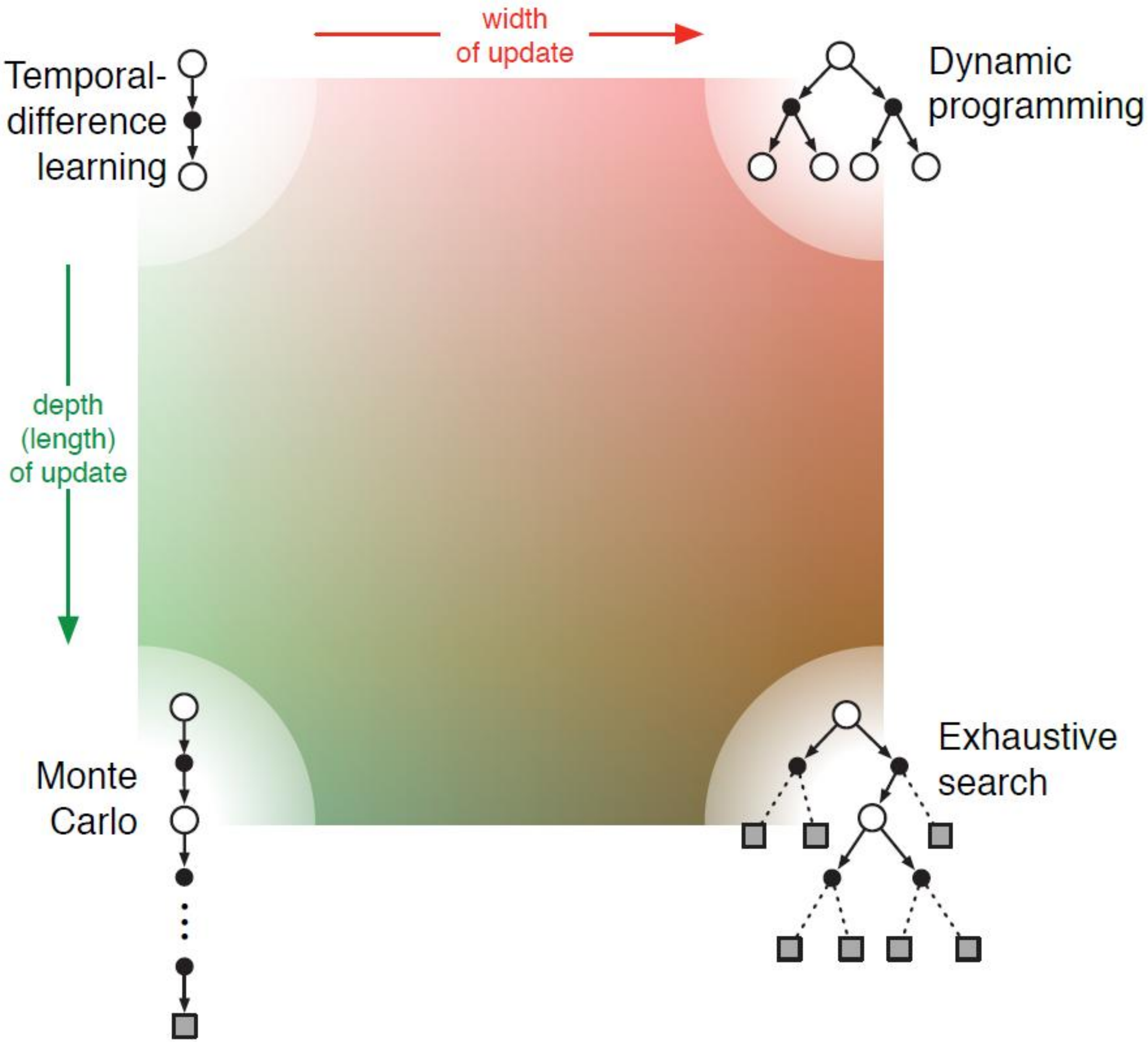
So we just updated our **value function for State 0.**

Now **we continue to interact with this environment with our updated value function.**

WATCH THE FOLLOWING VIDEO



https://www.youtube.com/watch?v=AJiG3ykOxmY


width
of update
Temporal-
difference
learning
Dynamic
programming
depth
(length)
of update
Monte
Carlo
Exhaustive
search