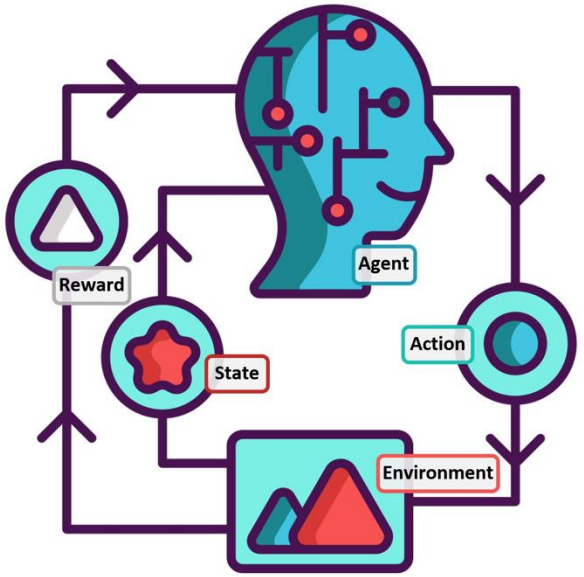
AMCO
ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND CONTROL RESEARCH GROUP

Reinforcement Learning
2025/2026

RL2
Reinforcement Learning Research Lab @ UniPD

Lecture #03
Multi-armed Bandits + Markov Decision Processes

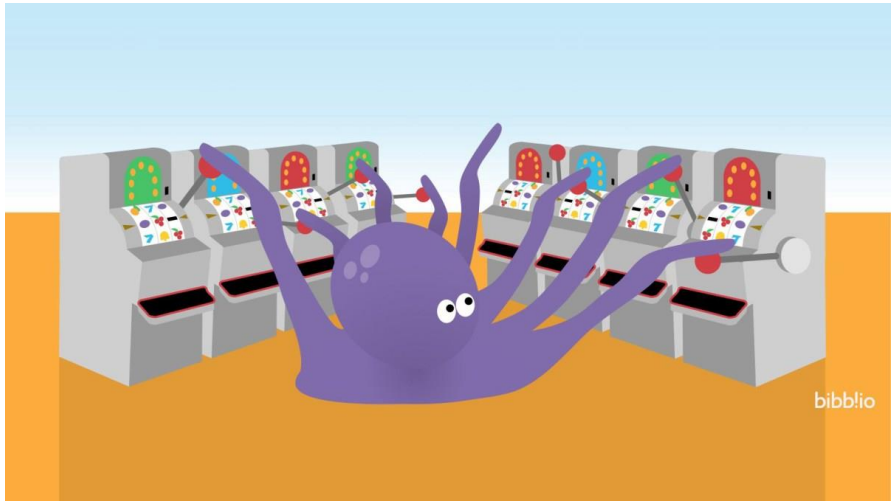
Gian Antonio Susto



Announcements

- If you cannot take the first partial for a ‘serious’ reason, I will allow to take the second partial and, during the first exam on January, just the portion of the exam associated with the contents on the first partial
- Lab 2 ‘k-armed Bandits’ Fri 10 Oct 2025 14:30-16:00 Room De

Recap:
k-armed Bandits



'Multi-armed' (or 'k-armed'):

- many levers (k levers)/actions associated with different machines
- Each machine is associated with a different distribution of rewards (one machine will pay more than the others)
- At each episode (we are dealing with an episodic task), the agent can pull only one lever
- What is the best strategy to maximize cumulative (over different episodes) rewards?
- Simple setting: one state! One episode = one action
- We have seen the concept of action-value function

$$q_*(a) \stackrel{\text{def}}{=} \mathbb{E}[R_t | A_t = a]$$

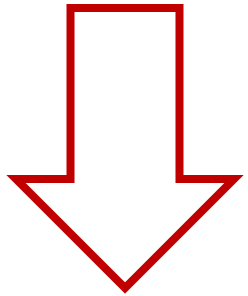
Incremental Implementation of Action Value Estimate

- This implementation requires only Q_n and n to be stored

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\Big(R_n + (n-1)Q_n\Big) \\
&= \frac{1}{n}\Big(R_n + nQ_n - Q_n\Big) \\
&= Q_n + \frac{1}{n}\Big[R_n - Q_n\Big],
\end{aligned}
$$

Incremental Estimations are commonly used in Reinforcement Learning

$$Q_{n+1} = Q_n + \frac{1}{n}\left[R_n - Q_n\right]$$

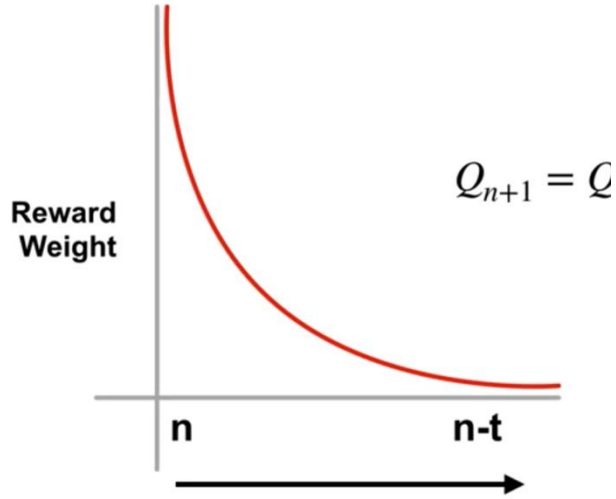


Typically, we will employ fixed step-size $\alpha \in (0,1]$

$$NewEstimate \leftarrow OldEstimate + StepSize\left[Target - OldEstimate\right]$$

Incremental Estimations are commonly used in Reinforcement Learning

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \\
&\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i .
\end{aligned}
$$

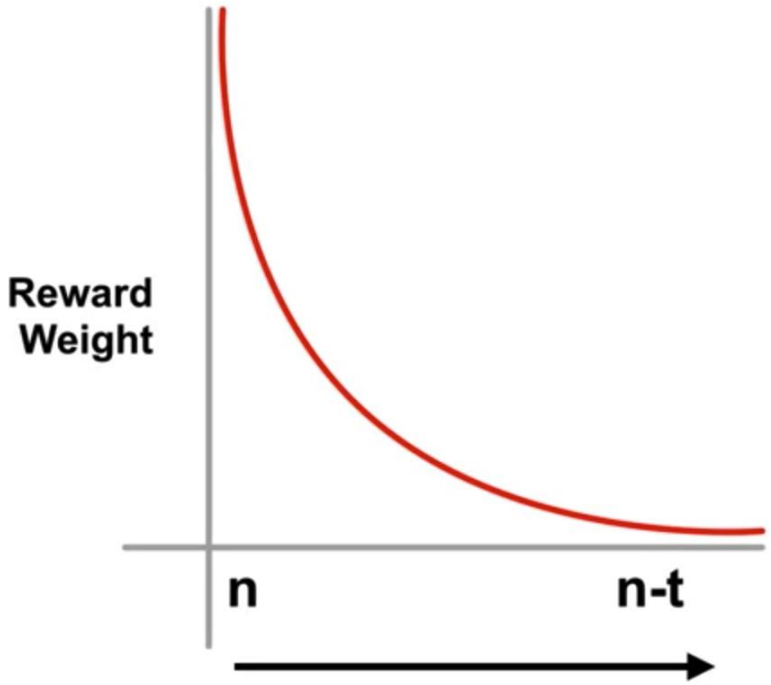


Exponentially weighted moving average (exponential recency-weighted moving average)Typically, we will employ fixed step-size $\alpha \in (0,1]$

Incremental Estimations for dealing with nonstationary problems

- This choice has the advantage of forgetting over time old rewards
- This is extremely convenient in nonstationary problems: problems were the reward distributions of the actions change over time
- Other solutions, with time-varying step size, are possible (see Section 2.5)

$$Q_{n+1} = \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \cdots + (1-\alpha)^{n-1} \alpha R_1 + (1-\alpha)^n Q_1$$



A. White, M. White 'Fundamentals of Reinforcement Learning'

Recap: Exploration vs. Exploitation

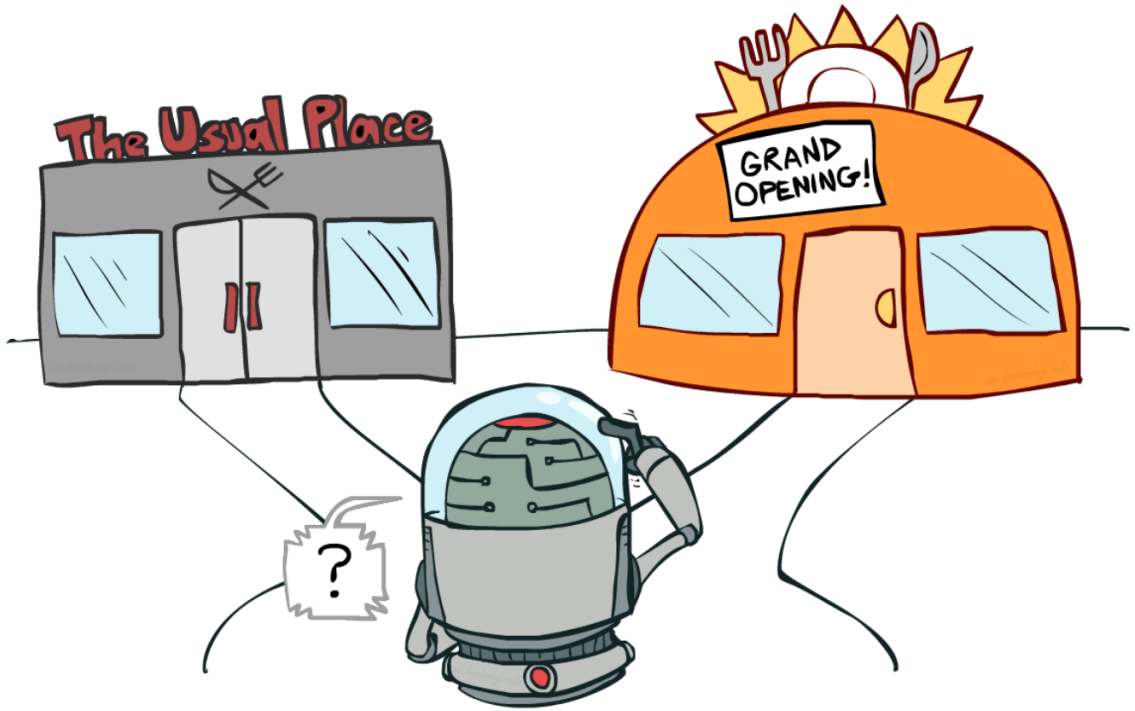
K-Bandits allowed us to introduced the exploration vs. exploitation dilemma

We have seen different approaches to cope with the trade-off:

1. Ɛ-greedy
2. Greedy with optimistic inizialization
3. Upper Confidence Bound (UCB)
4. Gradient Bandit

$$A_t \doteq \arg\max_a Q_t(a)$$

Greedy policy (no exploration)



Recap: Exploration vs. Exploitation

K-Bandits allowed us to introduced the exploration vs. exploitation dilemma

We have seen different approaches to cope with the trade-off:

1. ε-greedy
2. Greedy with optimistic inizialization
3. Upper Confidence Bound (UCB)
4. Gradient Bandit

$$A_t \doteq \arg\max_a Q_t(a)$$

Greedy policy (no exploration)

Recap: Exploration vs. Exploitation

K-Bandits allowed us to introduced the exploration vs. exploitation dilemma

$$A_t \doteq \arg\max_a Q_t(a)$$

Greedy policy (no exploration)

We have seen different approaches to cope with the trade-off:

1. Ɛ-greedy
2. Greedy with optimistic inizialization
3. Upper Confidence Bound (UCB)
4. Gradient Bandit

$$A_t \overset{\text{def}}{=} \begin{cases} \text{greedy action with probability } 1-\varepsilon \\ \text{non greedy action with probability } \varepsilon \end{cases}$$

Recap: Exploration vs. Exploitation

K-Bandits allowed us to introduced the exploration vs. exploitation dilemma

$$A_t \doteq \arg\max_a Q_t(a)$$

Greedy policy (no exploration)

We have seen different approaches to cope with the trade-off:

1. Ɛ-greedy
2. Greedy with optimistic inizialization
3. Upper Confidence Bound (UCB)
4. Gradient Bandit

$$A_t \overset{\text{def}}{=} \begin{cases} \text{greedy action with probability } 1-\varepsilon \\ \text{non greedy action with probability } \varepsilon \end{cases}$$

#2 - Optimistic Initial Values

There are other ways to encourage exploration…

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop forever:

$$A \leftarrow \begin{cases} \text{argmax}_a\, Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$R \leftarrow bandit(A)$

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

#2 - Optimistic Initial Values

There are other ways to encourage exploration…

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Optimism in the face of uncertainty: turns out that choosing optimistic initial values for $Q(a)$ can help exploration!

Loop forever:

$$A \leftarrow \begin{cases} \text{argmax}_a\, Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$R \leftarrow bandit(A)$

$N(A) \leftarrow N(A) + 1$

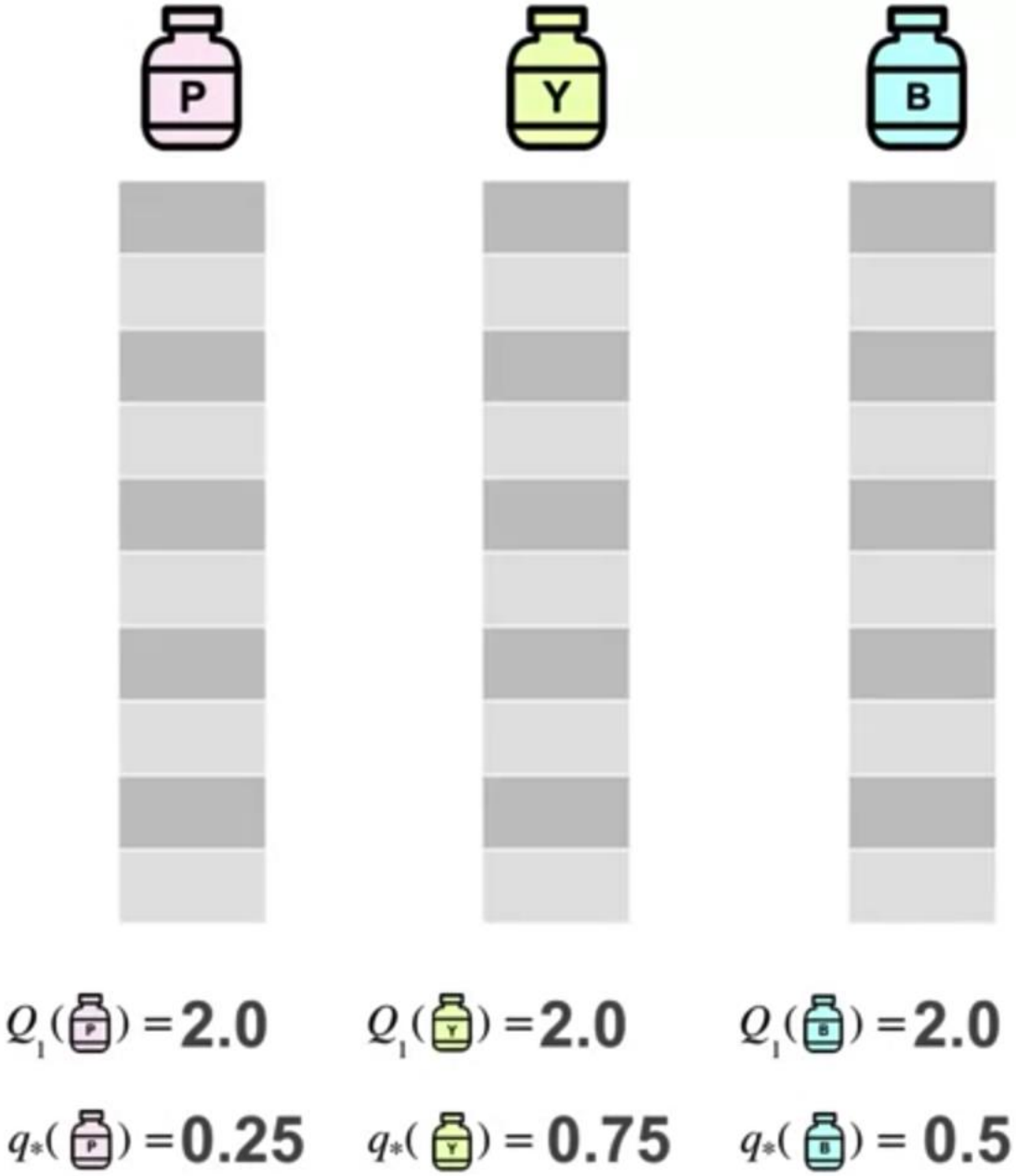
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

Example: clinical trials (reprise)

- A treatment provides +1 reward if it is effective (0 otherwise)
- This time we choose a clearly optimistic choice for the action values, being equal to 2
- We choose a greedy policy with:

$$Q_{n+1} \leftarrow Q_n + \alpha\left(R_n - Q_n\right)$$

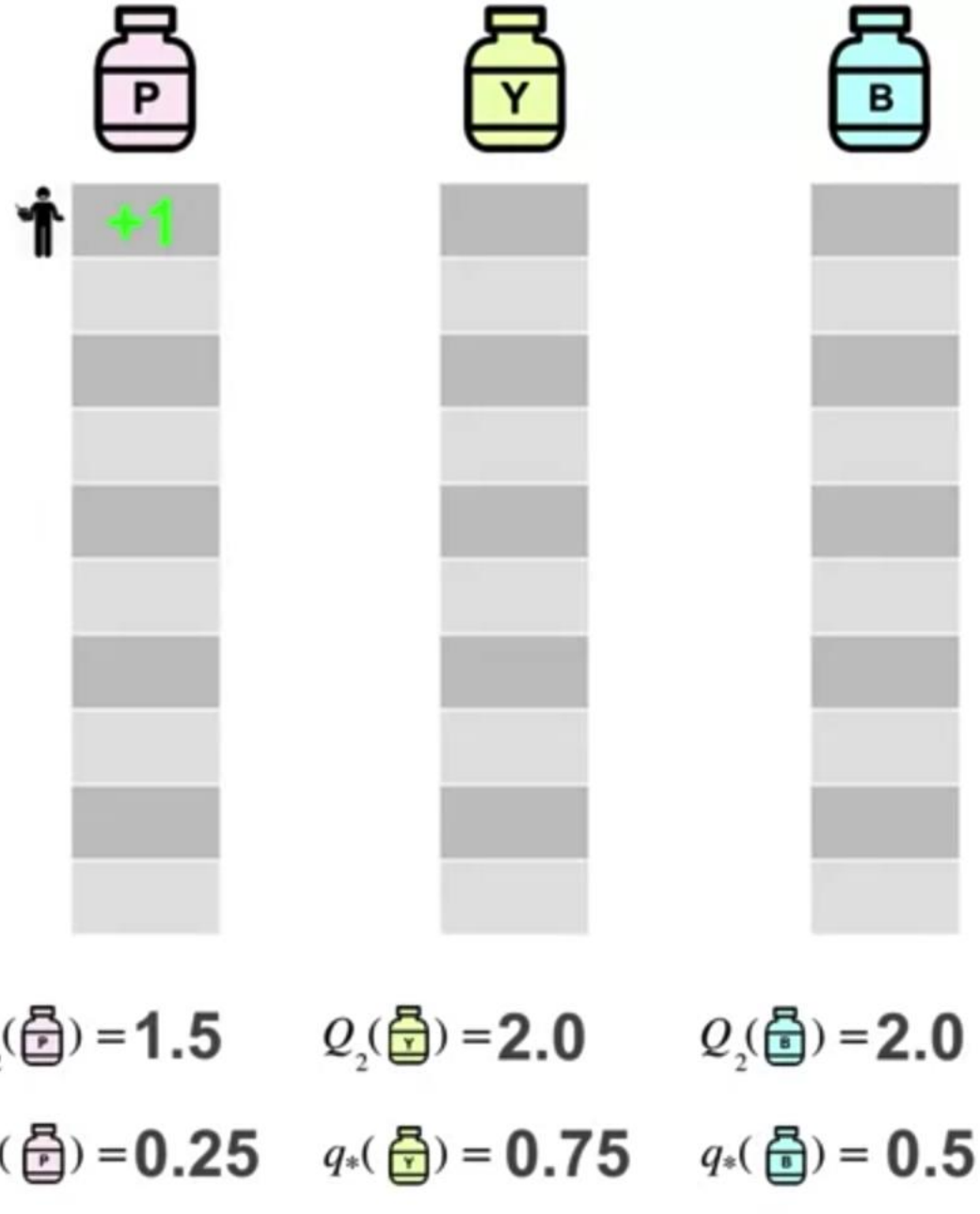
$$\textbf{Let}\ \alpha = 0.5$$



A. White, M. White 'Fundamentals of Reinforcement Learning'

Example: clinical trials (reprise)

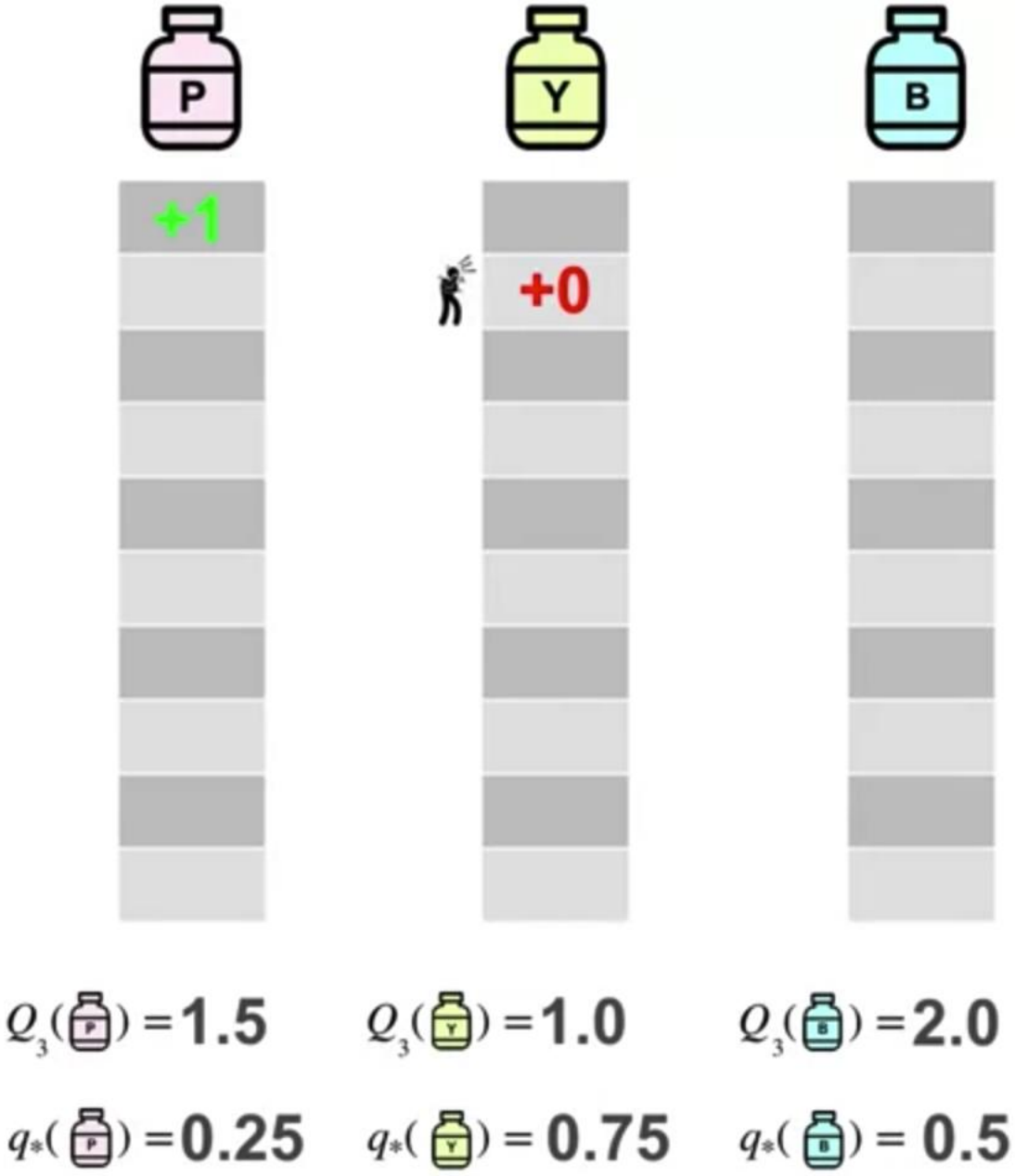
Epoch #1:



A. White, M. White 'Fundamentals of Reinforcement Learning'

Example: clinical trials (reprise)

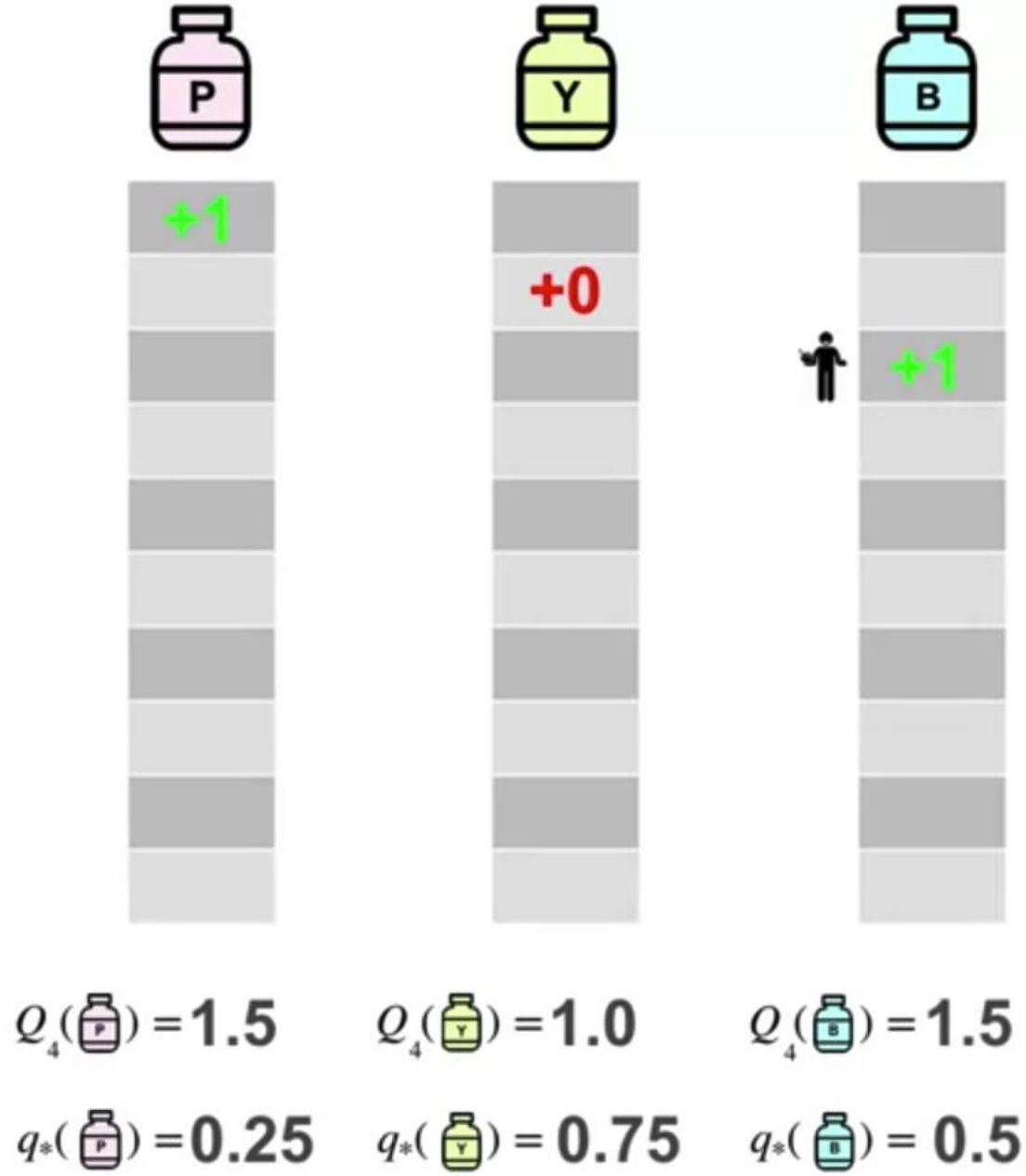
Epoch #2:



A. White, M. White 'Fundamentals of Reinforcement Learning'

Example: clinical trials (reprise)

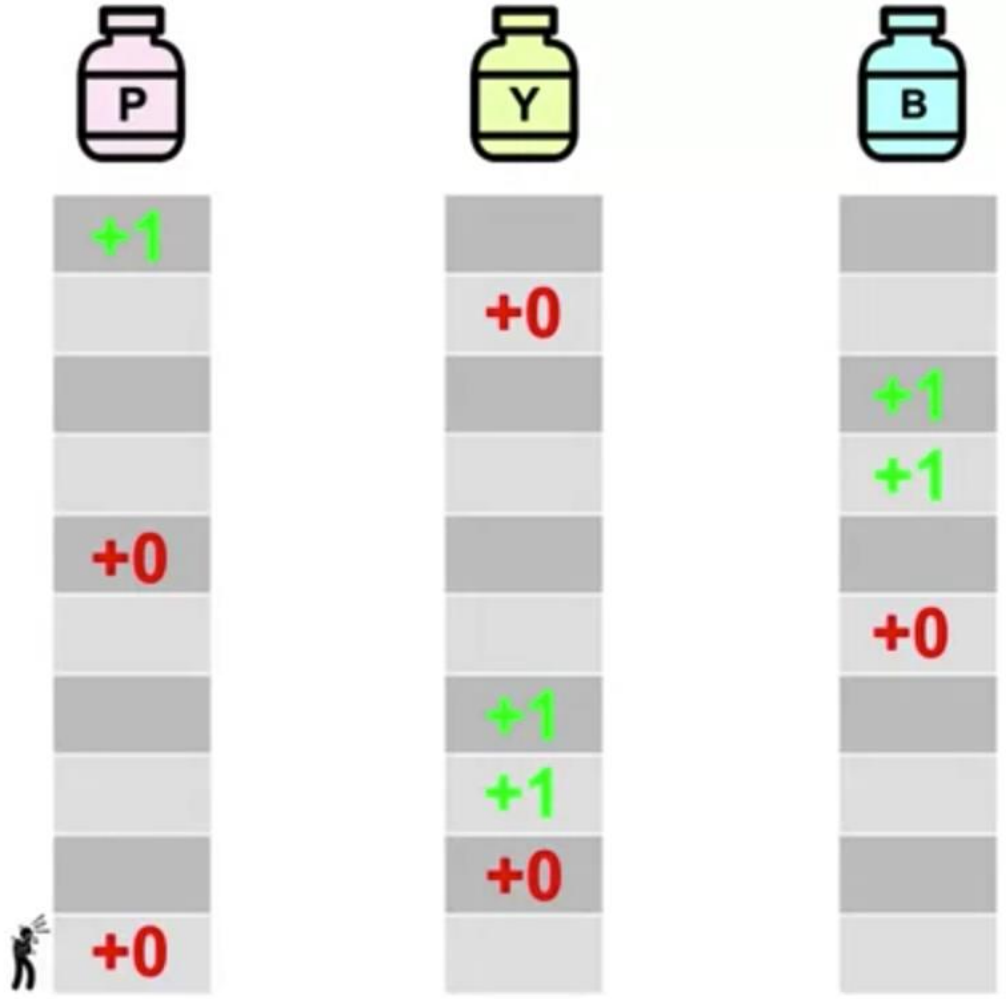
Epoch #3:



A. White, M. White 'Fundamentals of Reinforcement Learning'

Example: clinical trials (reprise)

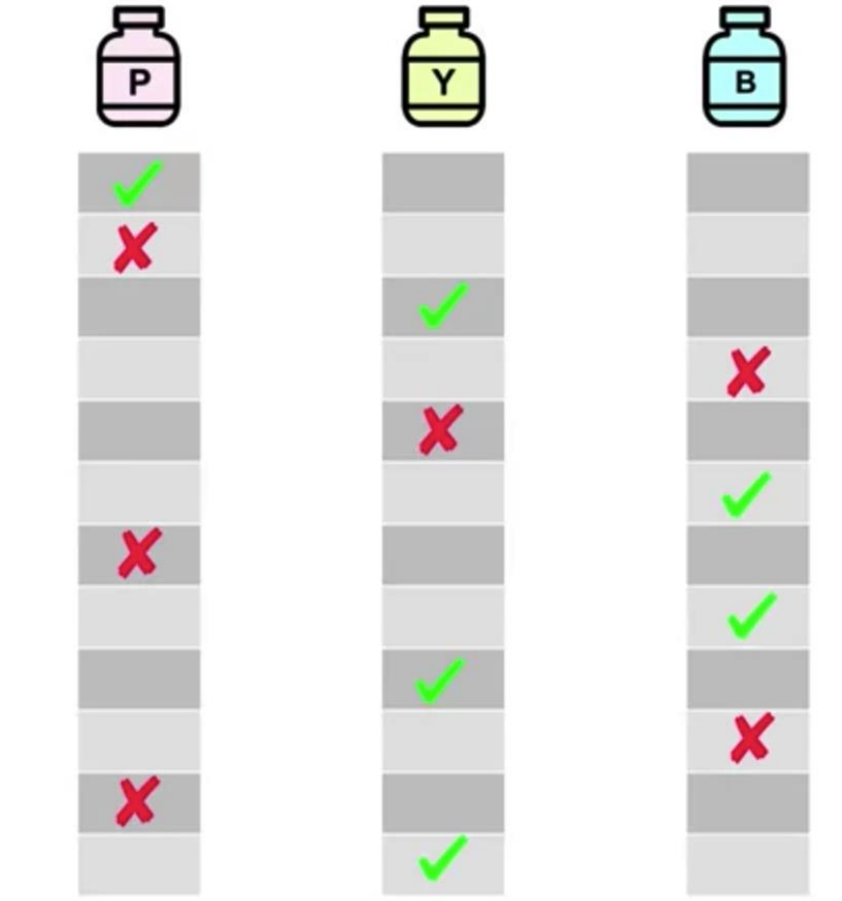
Epoch #10:



A. White, M. White 'Fundamentals of Reinforcement Learning'

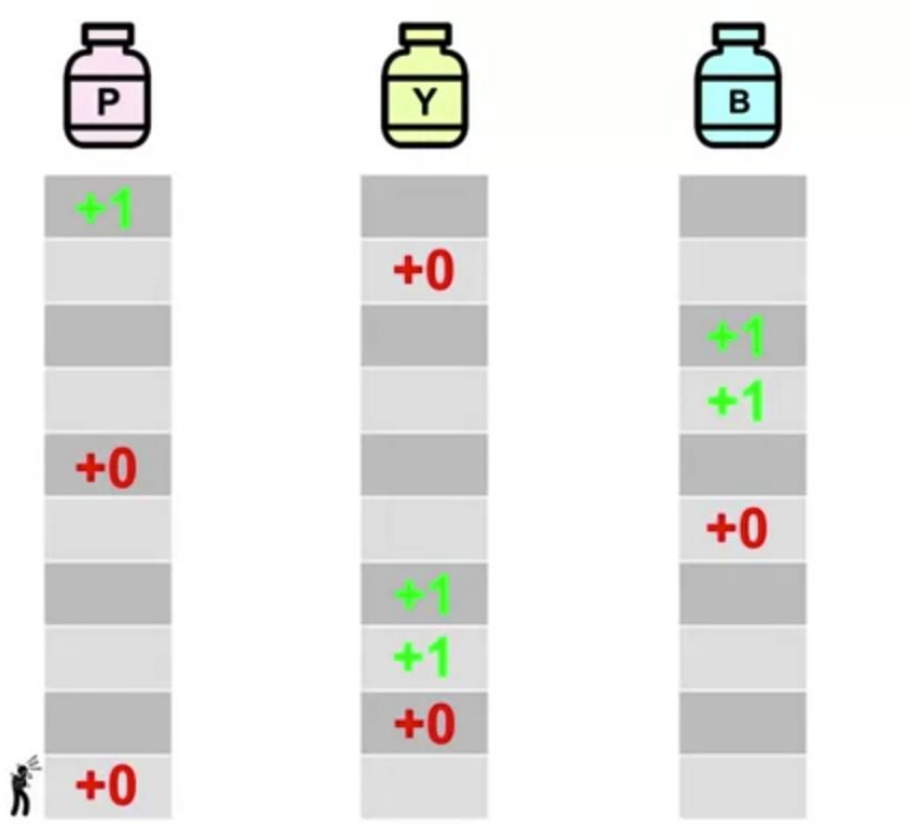
Example: clinical trials (reprise)

Initial Estimation = 0



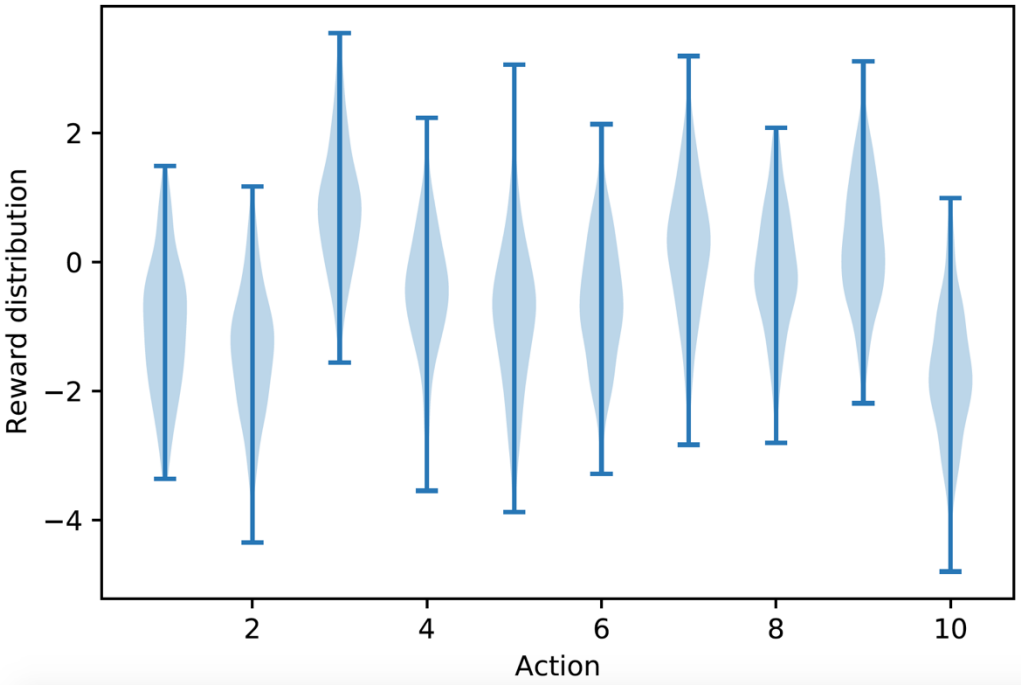
$Q_{12}(P) = 0.25$ $Q_{12}(Y) = 0.75$ $Q_{12}(B) = 0.5$

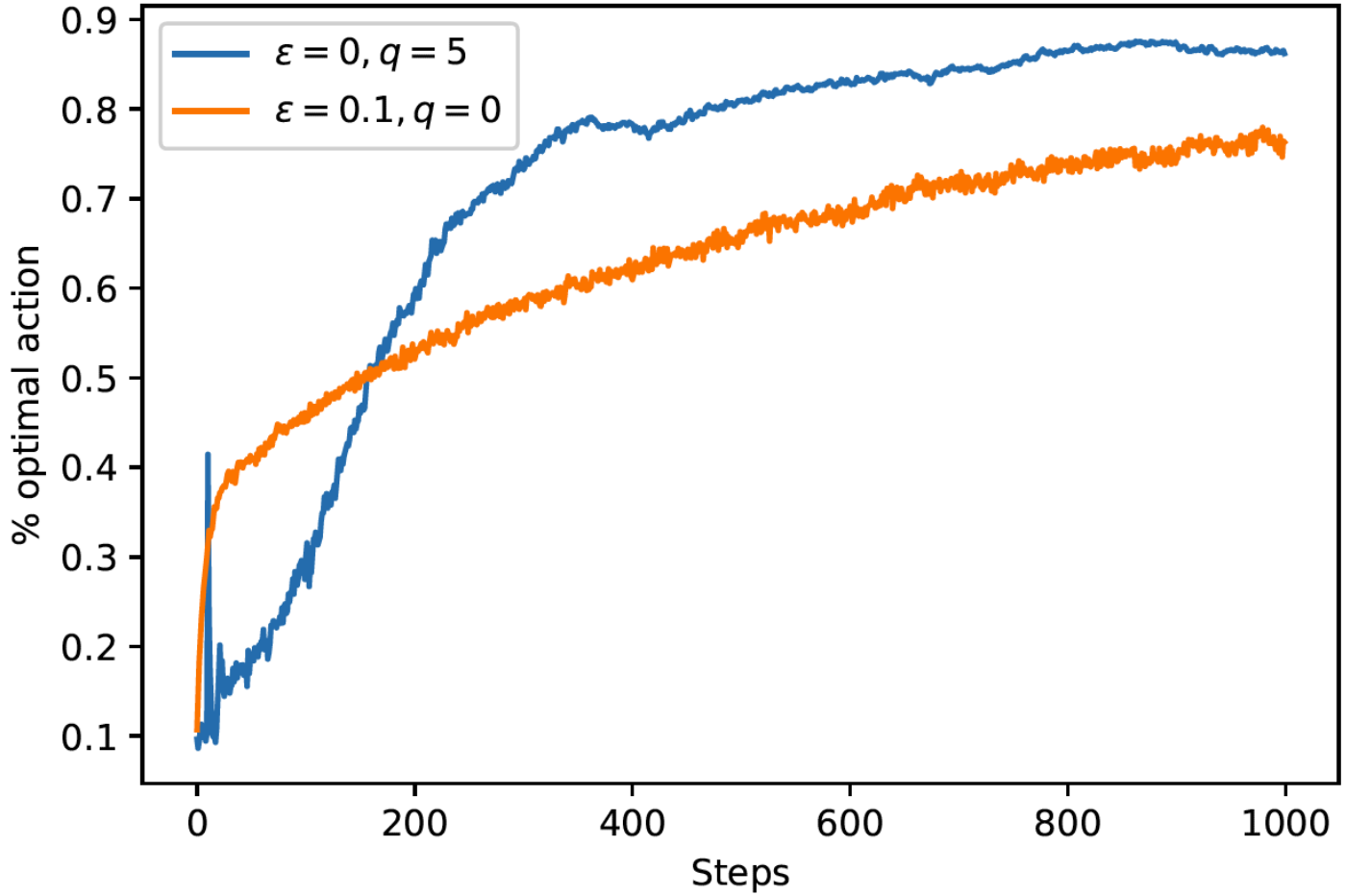
Initial Estimation = 2



$Q_{11}(P) = 0.375$ $Q_{11}(Y) = 0.5$ $Q_{11}(B) = 0.625$

Example: 10-armed bandits (reprise)





Optimistic Initial Values Limitations

Limitations

- Optimistic initial values only drive early exploration
- Such approaches are not well-suited for non-stationary problems
- In some applications we may not know what the optimistic initial value should be

However, it is a convenient heuristic, and it is typically used in combination with other approaches that allowed us to encourage exploitation

#3 - Upper-Confidence-Bound (UCB) Action Selection

- Another approach for dealing with the Exploration-Exploitation dilemma is provided by UCB
- UCB uses uncertainty in the value estimates for governing the trade-off between exploration and exploitation
- Recall that in Ɛ-greedy

$$A_t \leftarrow \begin{cases} \underset{a}{\operatorname{argmax}}\ Q_t(a) & \text{with probability } 1-\epsilon \\ a \sim Uniform(\{a_1 \ldots a_k\}) & \text{with probability } \epsilon \end{cases}$$

A. White, M. White 'Fundamentals of Reinforcement Learning'

#3 - Upper-Confidence-Bound (UCB) Action Selection

- Another approach for dealing with t
  provided by UCB
- UCB uses uncertainty in the value es
  between exploration and exploitatic
- Recall that in Ɛ-greedy

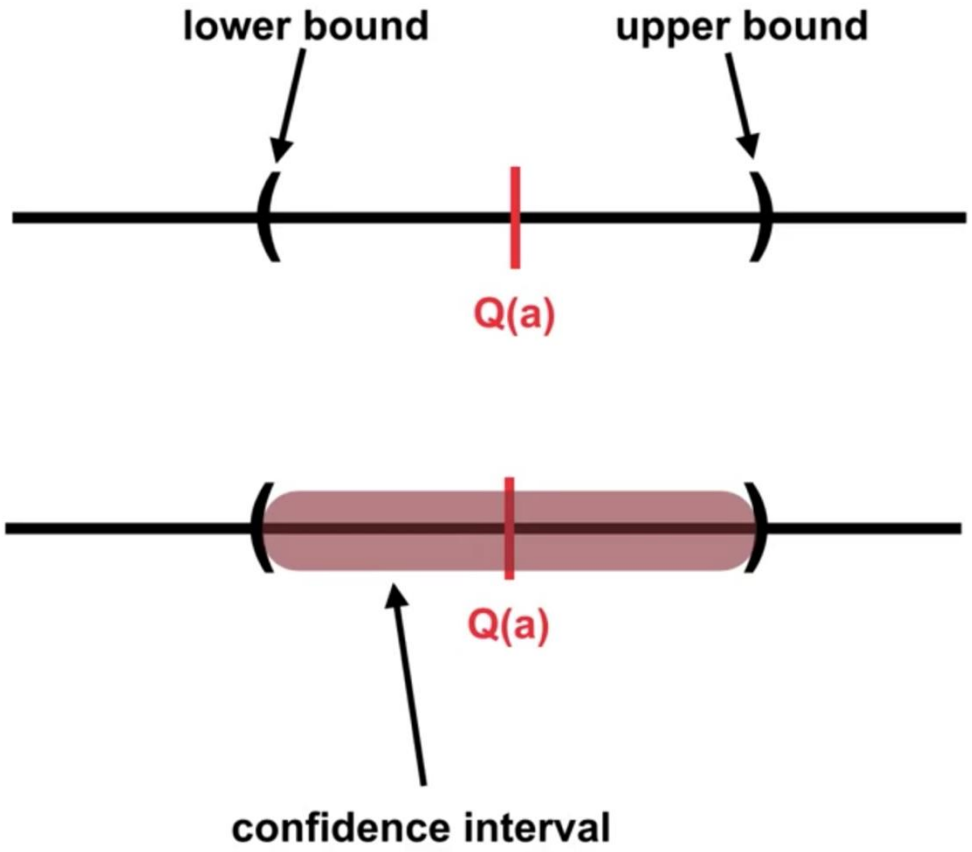
Can we do better? What if we can quantify the uncertainty in our action value estimation?

A simple idea could be to exploit how many times an actions has been taken!

$$A_t \leftarrow \begin{cases} \underset{a}{\operatorname{argmax}}\ Q_t(a) & \text{with probability } 1-\epsilon \\ a \sim Uniform(\{a_1 \dots a_k\}) & \text{with probability } \epsilon \end{cases}$$

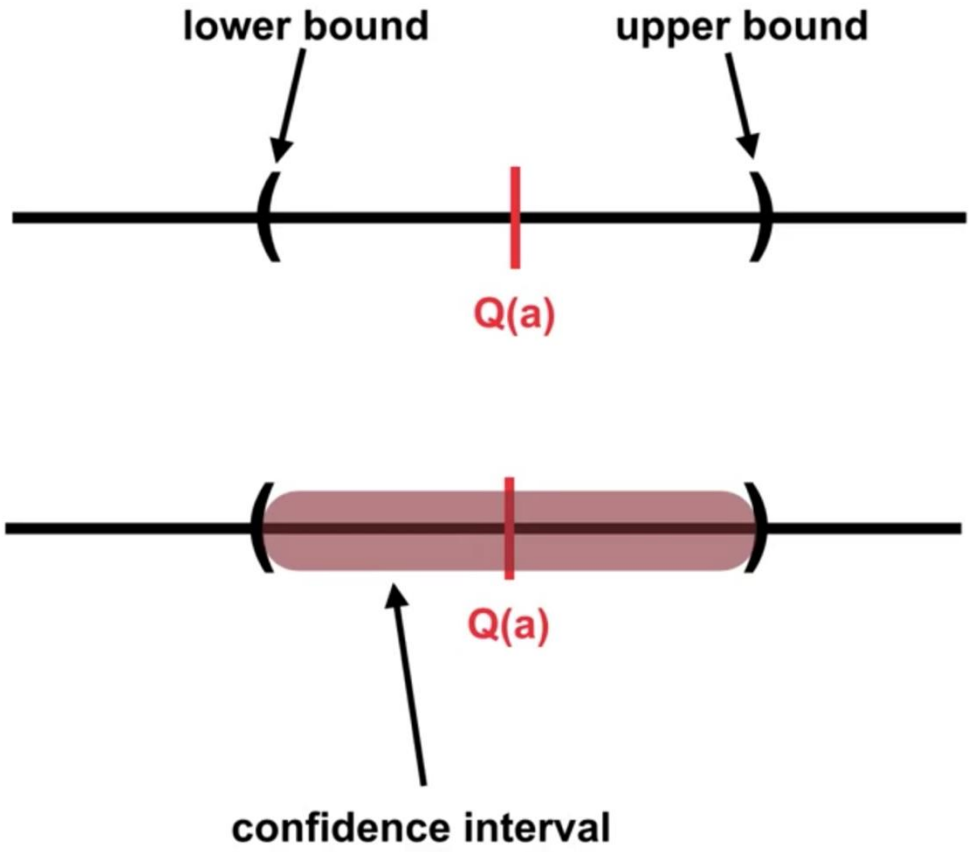
A. White, M. White 'Fundamentals of Reinforcement Learning'

UCB Action Selection



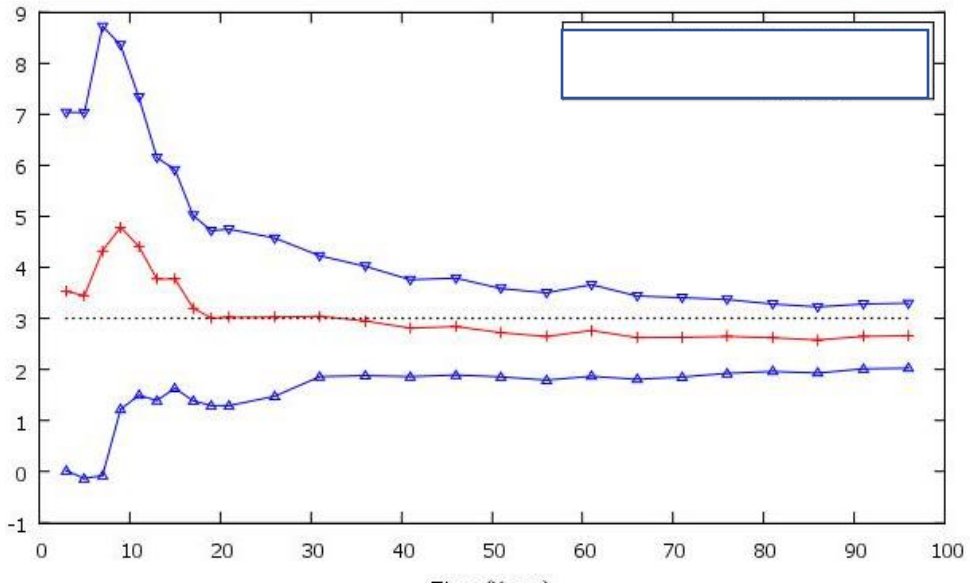
A. White, M. White 'Fundamentals of Reinforcement Learning'

UCB Action Selection



The more we take an action, the lower the confidence interval

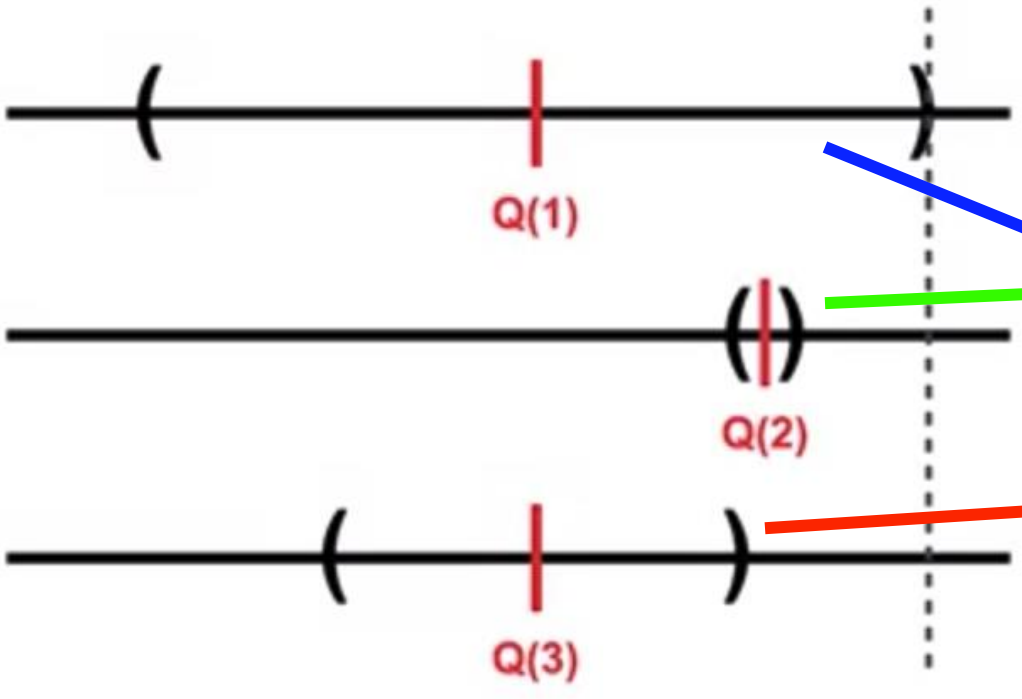
Pay attention: this is not the distribution of the award for action a, but our estimation of Q(a)!

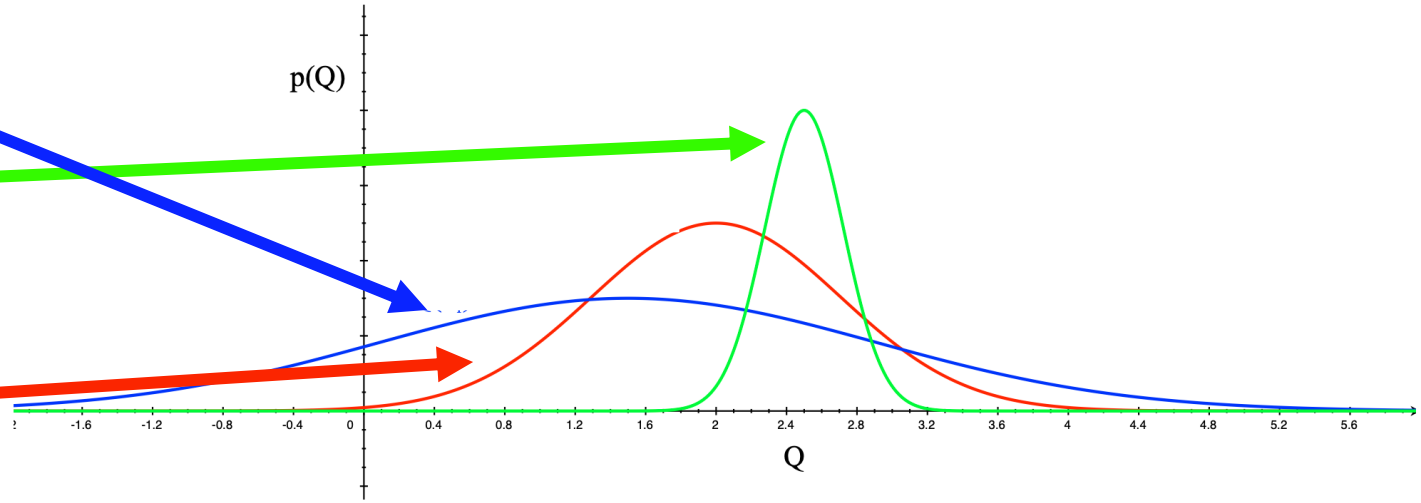


A. White, M. White 'Fundamentals of Reinforcement Learning'

UCB Action Selection

Optimism in the face of uncertainty: we take the action that 'potentially' could be the best!



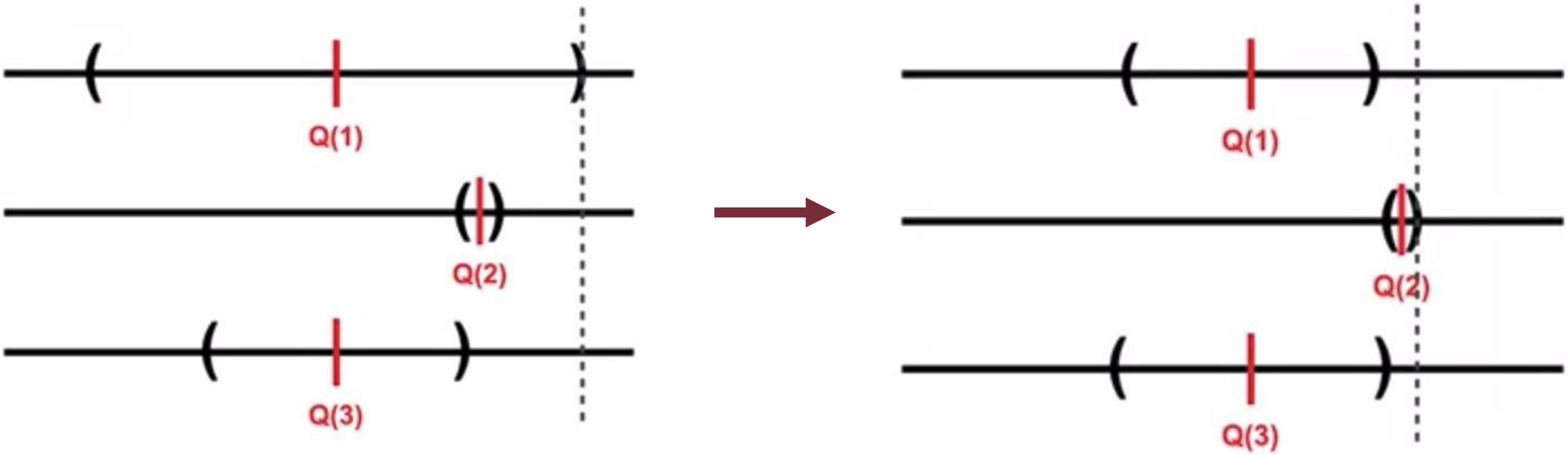


D. Silver 'Reinforcement Learning course' @ UCL

A. White, M. White 'Fundamentals of Reinforcement Learning'

UCB Action Selection

Optimism in the face of uncertainty: we take the action that 'potentially' could be the best!



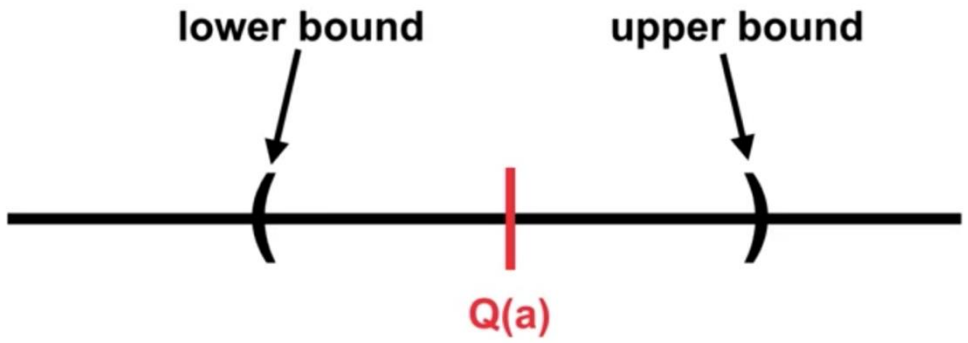
A. White, M. White 'Fundamentals of Reinforcement Learning'

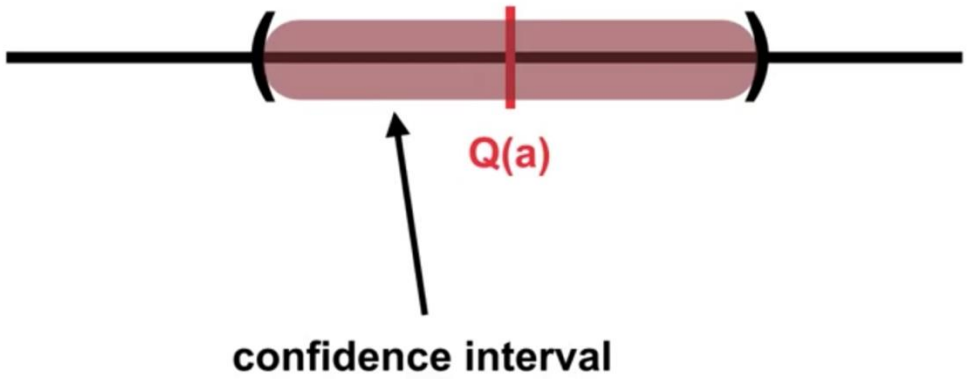
UCB Action Selection

$$A_t \doteq \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

Exploration (Hoeffding's Inequality)

Exploitation





A. White, M. White 'Fundamentals of Reinforcement Learning'

UCB Action Selection

$$A_t \doteq \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

Exploration (Hoeffding's Inequality)

Exploitation

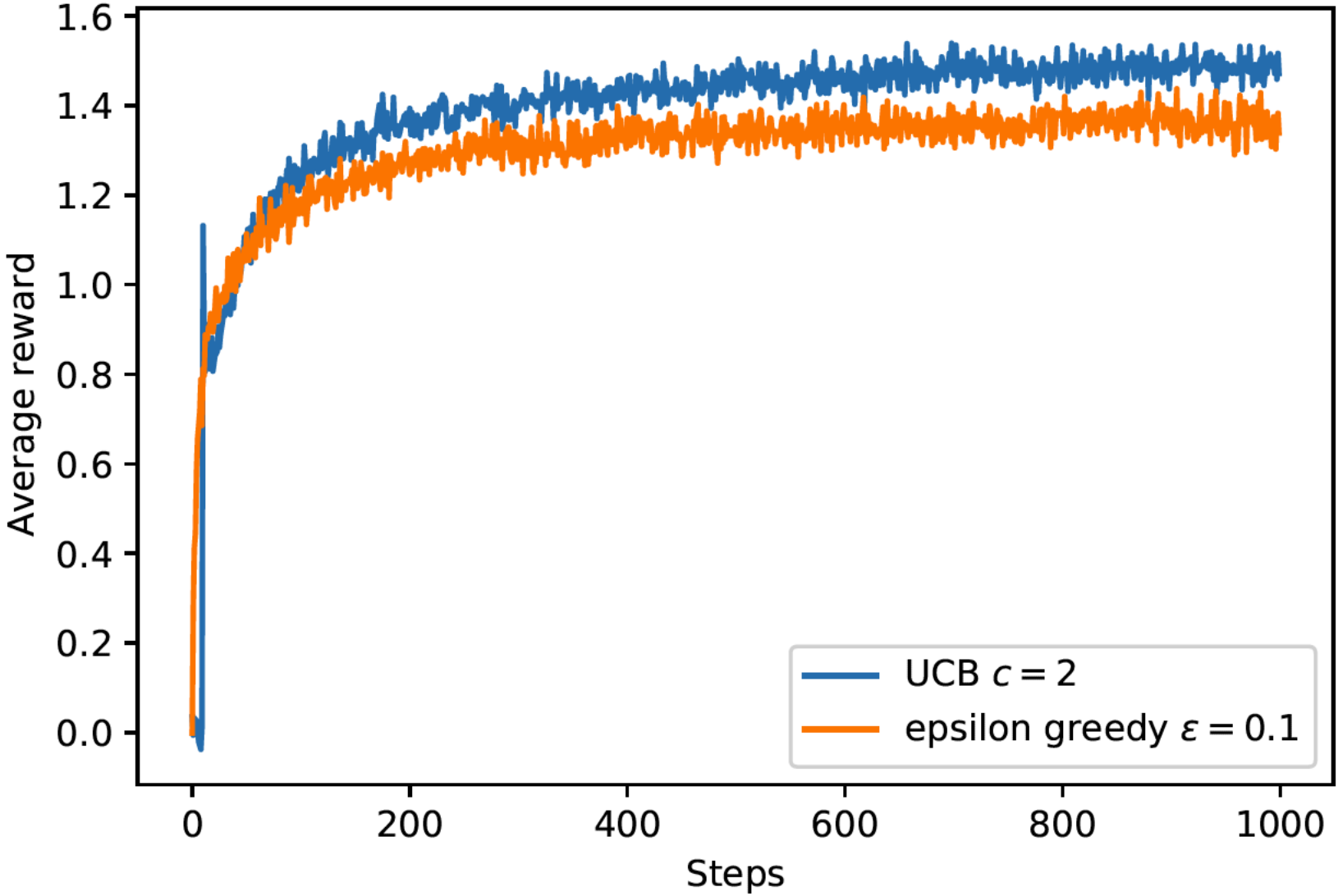
$$c\sqrt{\frac{ln\ t}{N_t(a)}} \rightarrow c\sqrt{\frac{ln\ \text{timesteps}}{\text{times action } a \text{ taken}}}$$

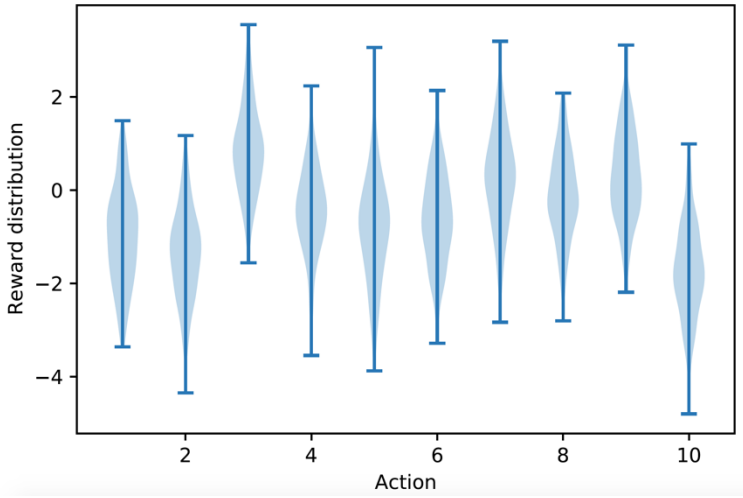
$$c\sqrt{\frac{ln\ 10000}{5000}} \rightarrow 0.043c$$

$$c\sqrt{\frac{ln\ 10000}{100}} \rightarrow 0.303c$$

A. White, M. White 'Fundamentals of Reinforcement Learning'

Example: 10-armed bandits (reprise)





Recap: Exploration vs. Exploitation

K-Bandits allowed us to introduced the exploration vs. exploitation dilemma

$$A_t \doteq \arg\max_a Q_t(a)$$

Greedy policy (no exploration)

We have seen different approaches to cope with the trade-off:

1. Ɛ-greedy
2. Greedy with optimistic inizialization
3. Upper Confidence Bound (UCB)
4. Gradient Bandit

$$A_t \overset{\text{def}}{=} \begin{cases} \text{greedy action with probability } 1-\varepsilon \\ \text{non greedy action with probability } \varepsilon \end{cases}$$

Optimistic choice of initial values for $Q_t(a)$

$$A_t \doteq \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

#4 - Gradient Bandit Algorithms

- Another approach to encourage exploration without choosing randomly the action to explore is represented by Gradient Bandit Algorithms (GBA)
- In GBA a preference for each action is defined $H_t(a) \in \mathbb{R}$: the larger the preference, the more often the action is taken
- Preference has no interpretation in terms of reward: only relative preference of one action over another is important
- For preference we use the soft-max distribution*

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

$\pi_t(a)$ is the probability of taking action a at time t

* It is a probability distribution

#4 - Gradient Bandit Algorithms

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

An example:

$$H = [0, -10, 3, 1]$$

$$p \approx [0.042,\ 0.000002,\ 0.844,\ 0.114]$$

Gradient Bandit Algorithms

How to choose preferences?

Let's exploit gradient ascent!

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

You'll probably have seen gradient descent in Machine Learning…

The derivation of this is provided in the book (pag 38) and it can be considered optional

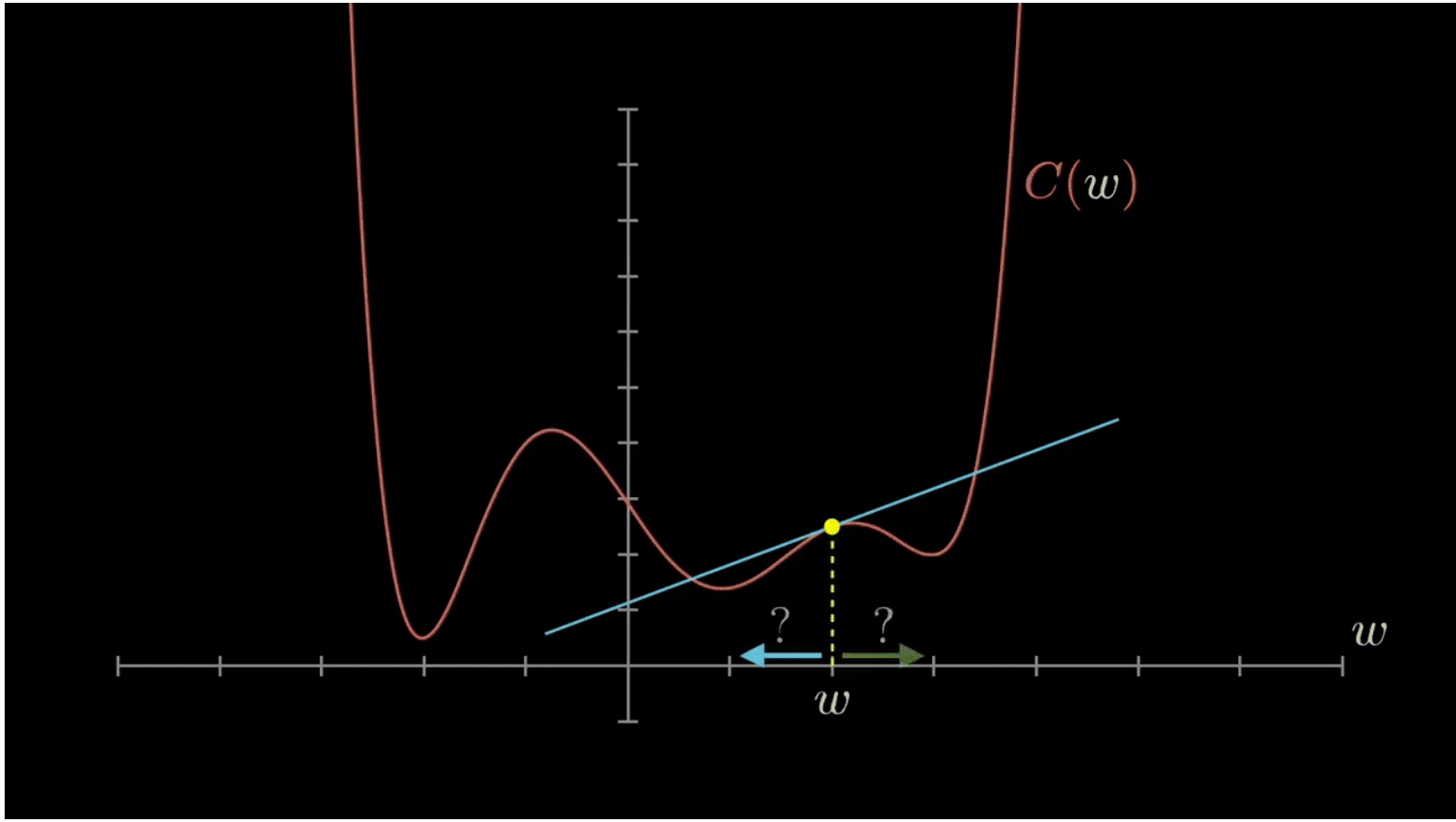
Gradient Bandit Algorithms

How to choose preferences?

Let's exploit gradient ascent!

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

You'll probably have seen gradient descent in Machine Learning…



(Gradient Descent)

We seek for a set of weights to minimize (or maximize) a function *J* (in ML, this happens frequently with loss functions) w.r.t. parameters *W*:

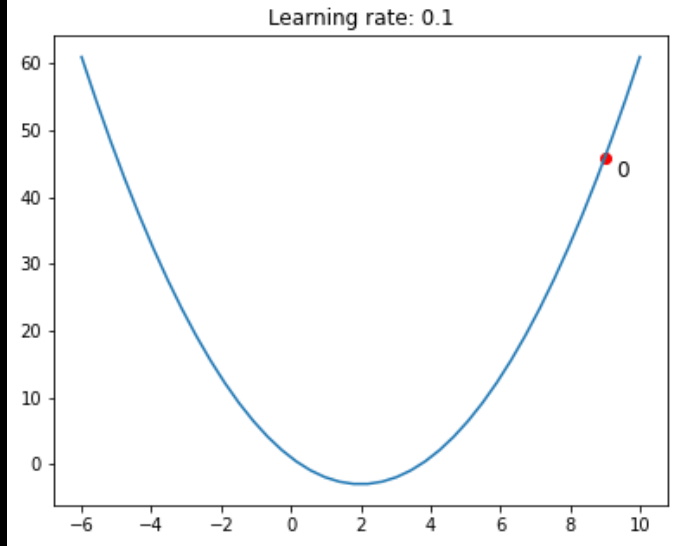
Algorithm (Gradient Descent)

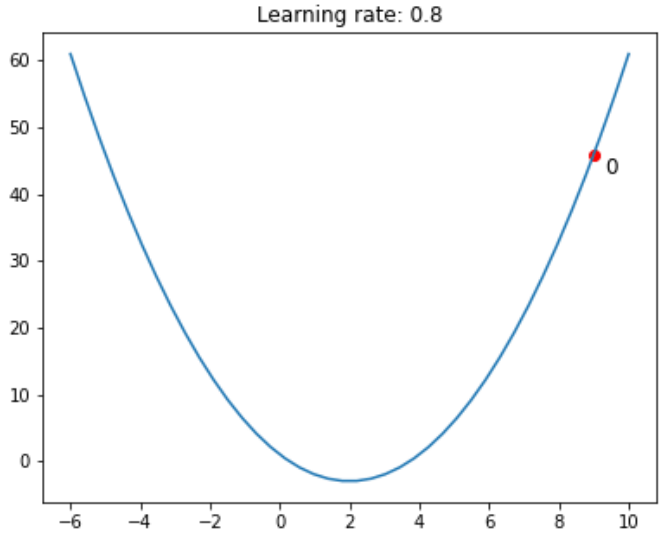
1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
4. Update weights, $\boldsymbol{W} \leftarrow \boldsymbol{W} - \alpha \frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
5. Return weights

Learning Rate

(Gradient Descent)

We seek for a set of weights to minin
frequently with loss functions) w.r.t.





Algorithm (Gradient Descent)

1. Initialize weights randomly $\sim \mathcal{N}(0,$
2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
4. Update weights, $\boldsymbol{W} \leftarrow \boldsymbol{W} - \alpha \frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$
5. Return weights

Learning Rate

Gradient Bandit Algorithms

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

- Let's start with all preferences equal to each other: (ie. $H_1(a) = 0 \; \forall \; a$)

- At each step, after selecting action A_t we receive reward R_t and we update our action preferences as follows:

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$

$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$

with:

α the step-size parameter

$\bar{R}_t$ the average of the rewards up to but not including time t; this terms serves as a baseline, if the A_t provides better results than the average then we should prefer this actions over the others!

Gradient Bandit Algorithms

By having preferences, I can define a ranking of actions: this may be handful in real world applications when some actions are not available

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

rences

$H_1(a) = 0 \; \forall \; a$)

cting action A_t we receive reward R_t and we update our action preferences as follows:

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$

$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$

with:

α the step-size parameter

$\bar{R}_t$ the average of the rewards up to but not including time t; this terms serves as a baseline, if the A_t provides better results than the average then we should prefer this actions over the others!

#4 - Gradient Bandit Algorithms

An example:

$$H = [0, -10, 3, 1]$$

$$p \approx [0.042,\ 0.000002,\ 0.844,\ 0.114]$$

I took action 4 with positive reward!

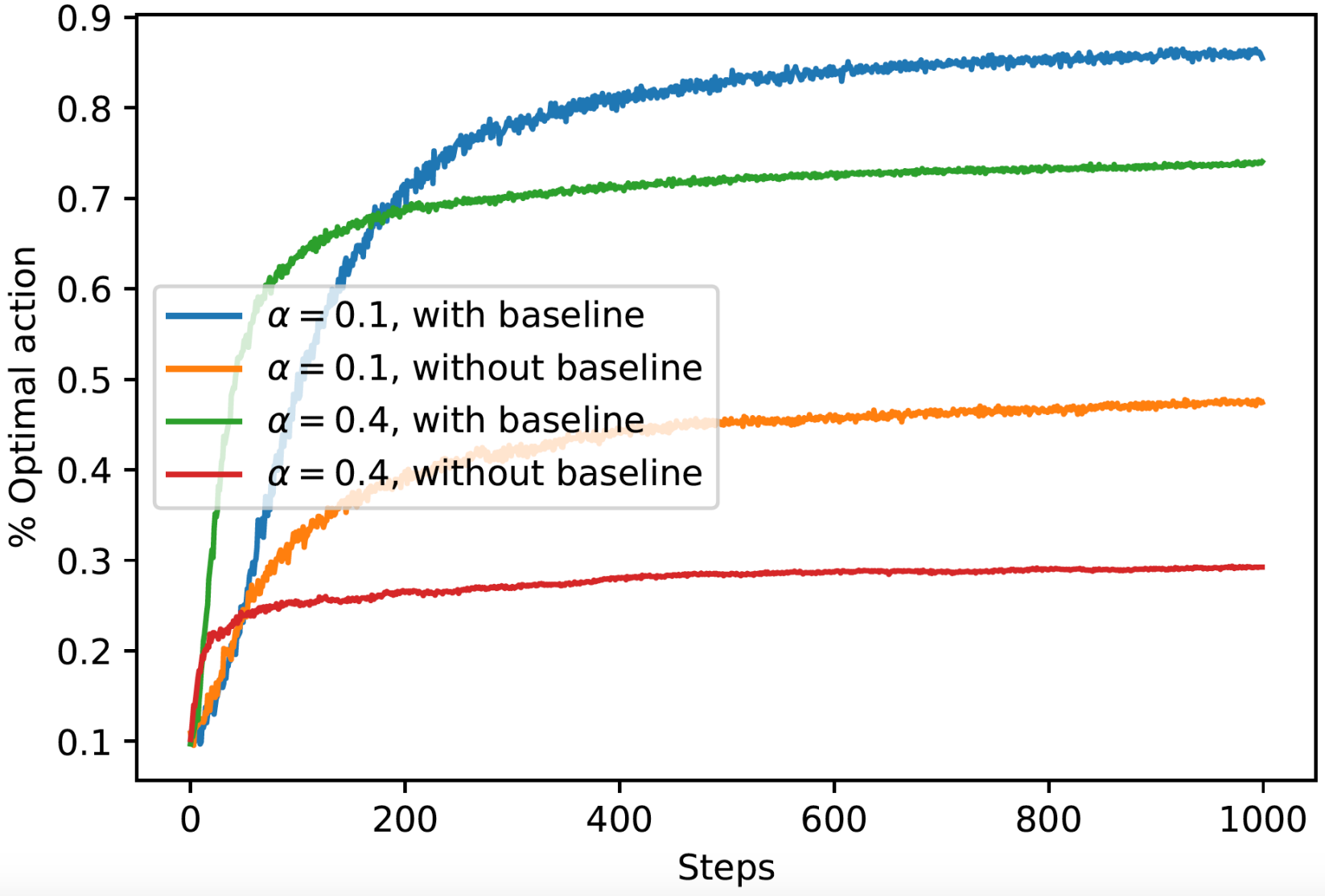
$$H = [-1, -11, 2, 4]$$

$$p \approx [0.0059,\ 0.0000003,\ 0.119,\ 0.876]$$

Example: 10-armed bandits (reprise)

With baseline: with $\bar{R}_t$

Without baseline:
with $\bar{R}_t = 0$

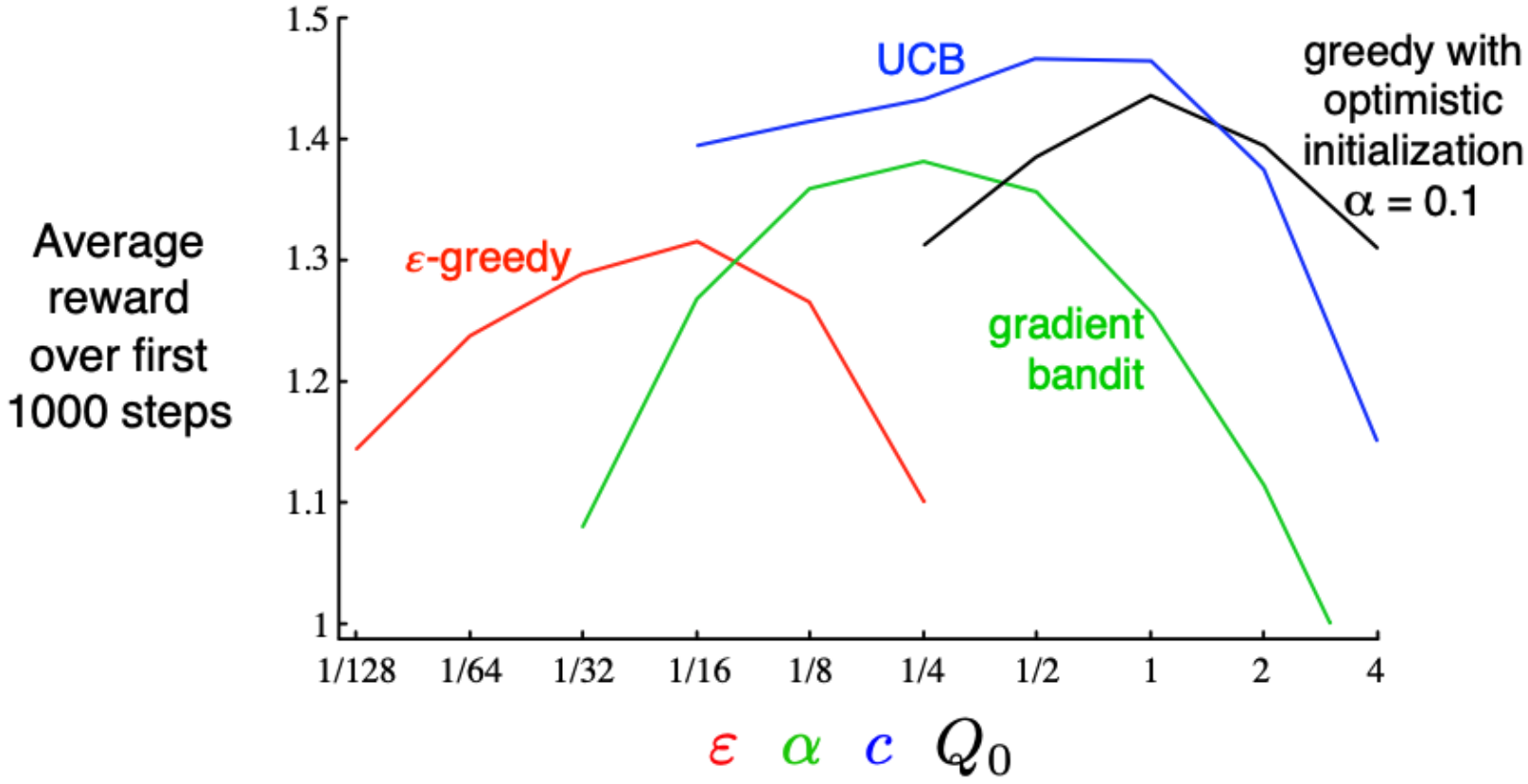


Summarizing

- We have seen the multi-armed bandits problem (ABP)
- We have seen several ways to handle the trade-off between exploration and exploitation:

1. Ɛ-greedy
2. Greedy with optimistic inizialization
3. Upper Confidence Bound (UCB)
4. Gradient Bandit

Parameter study with the 10 ABP->

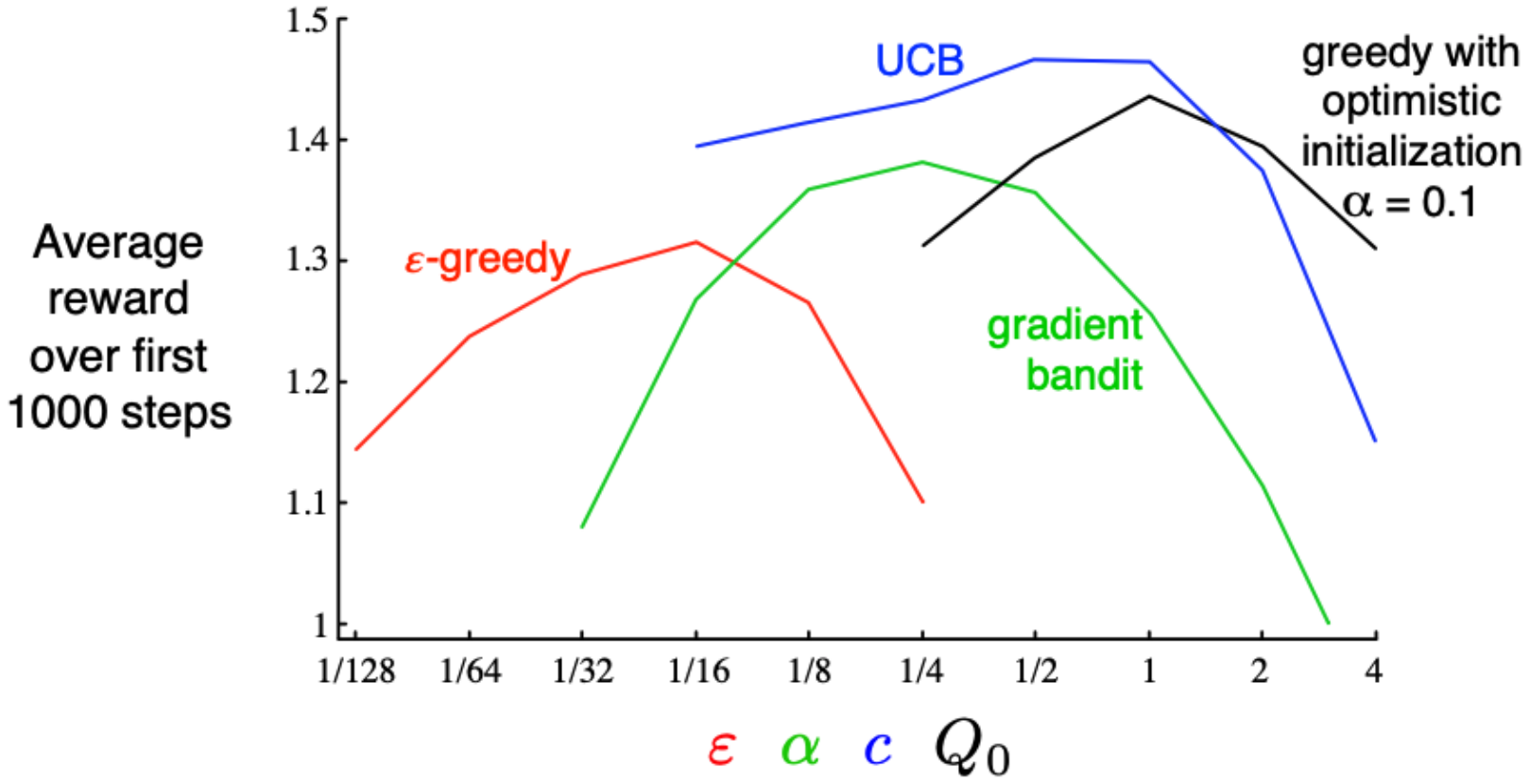


Summarizing

The parameter study showed us that:

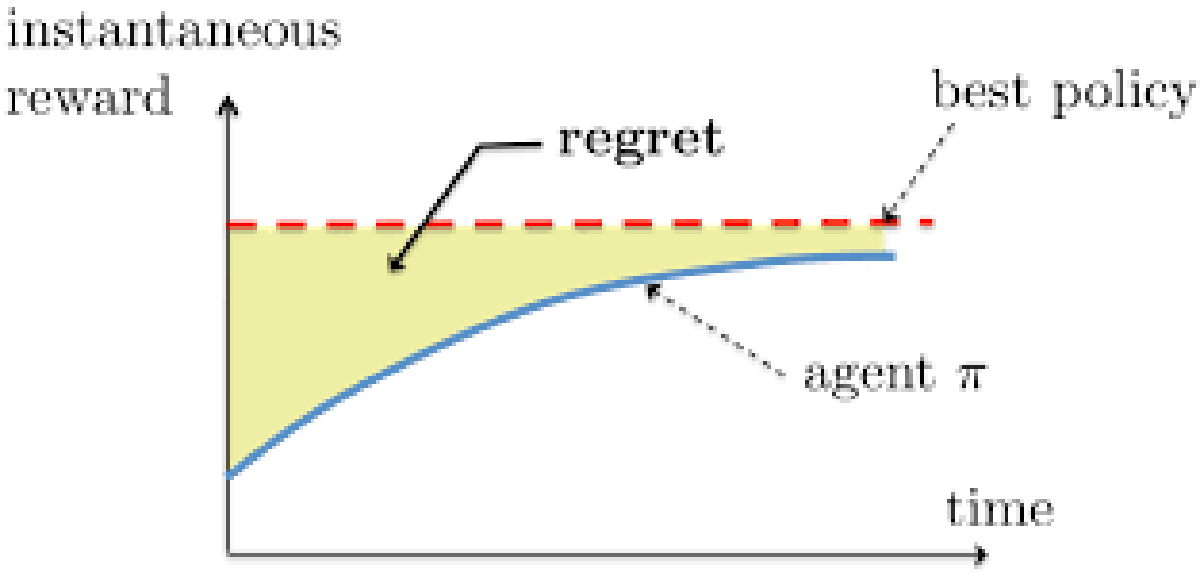
- The is always a trade-off in the choice of the parameter that can be associated with the trade-off between exploration and exploitation

- The number of episodes we have available may change a lot the efficiency of a strategy



We haven’t covered (1/2):

- The concept of regret, ie. the missed opportunity which is really helpful when studying problems taking into account the perspective of the available episodes



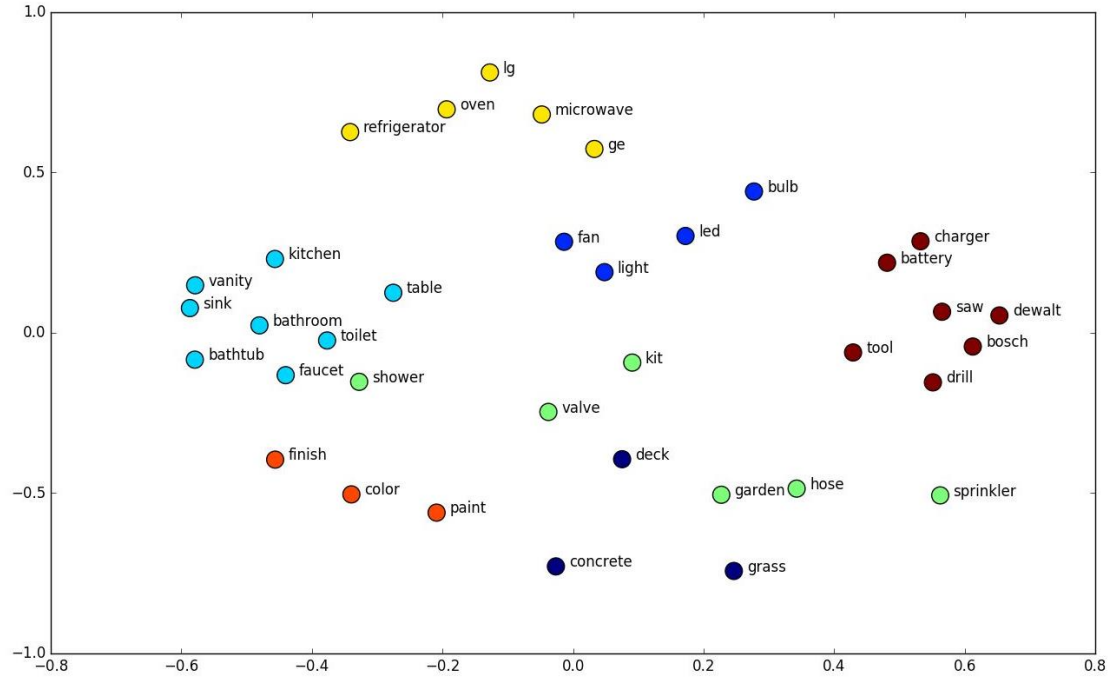
We haven’t covered (2/2):

Contextual bandits: we consider states! For example, in advertising suggestion we want to take into account information on the users actual search (or past search)

Contextual bandits can be seen as a combination of supervised learning (SL) & reinforcement learning (RL):

- RL optimize part of the suggestions based on previous decisions;

- SL optimize part of the suggestions based on the state (the current search or user)



(For handling text in SL you need tools like word-embedding…)

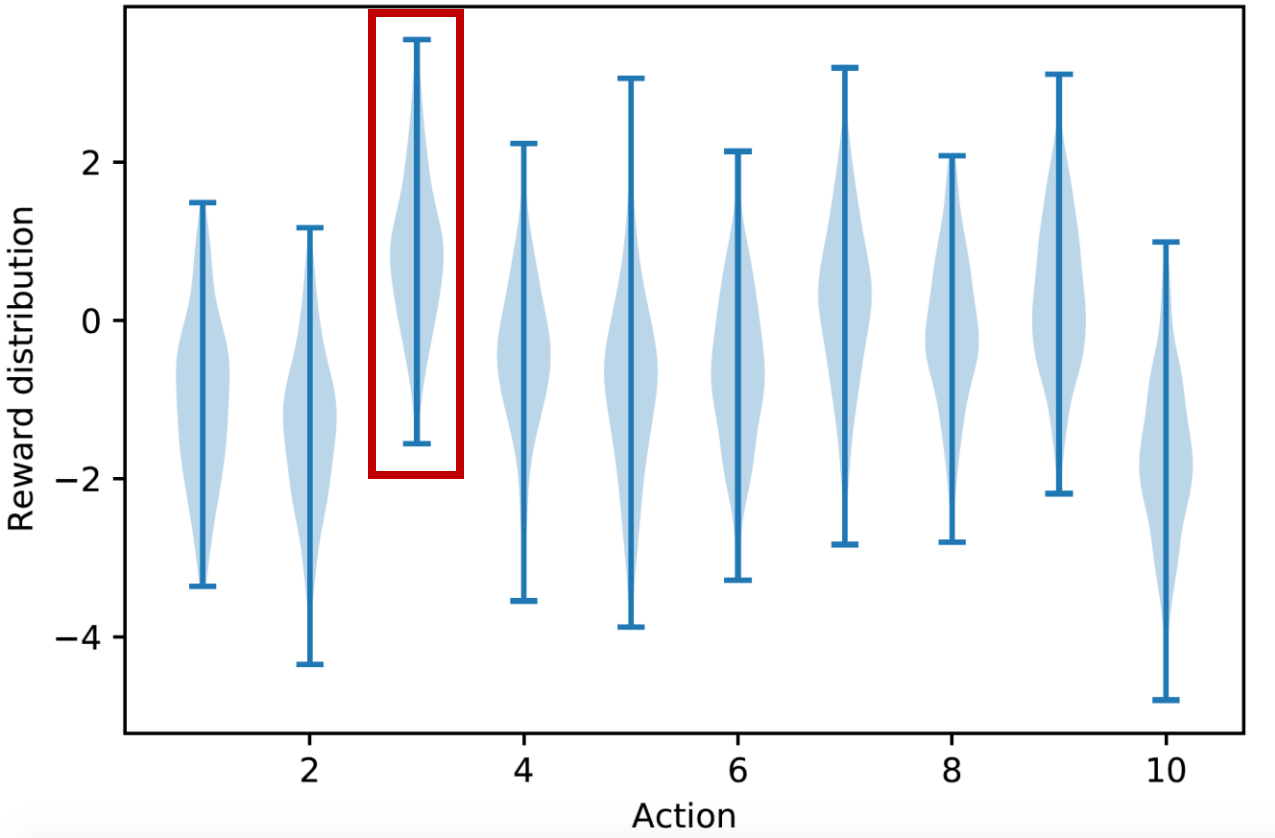
K-armed Bandits: Comments

1. Why don’t we just estimate. rewards distributions from collected data?

-> It is too data expensive, and we don’t really care, we just want to maximize our goal!

G. Box ‘All models are wrong, but some are useful’

2. We have already seen key RL concepts (q-value function, incremental computations)



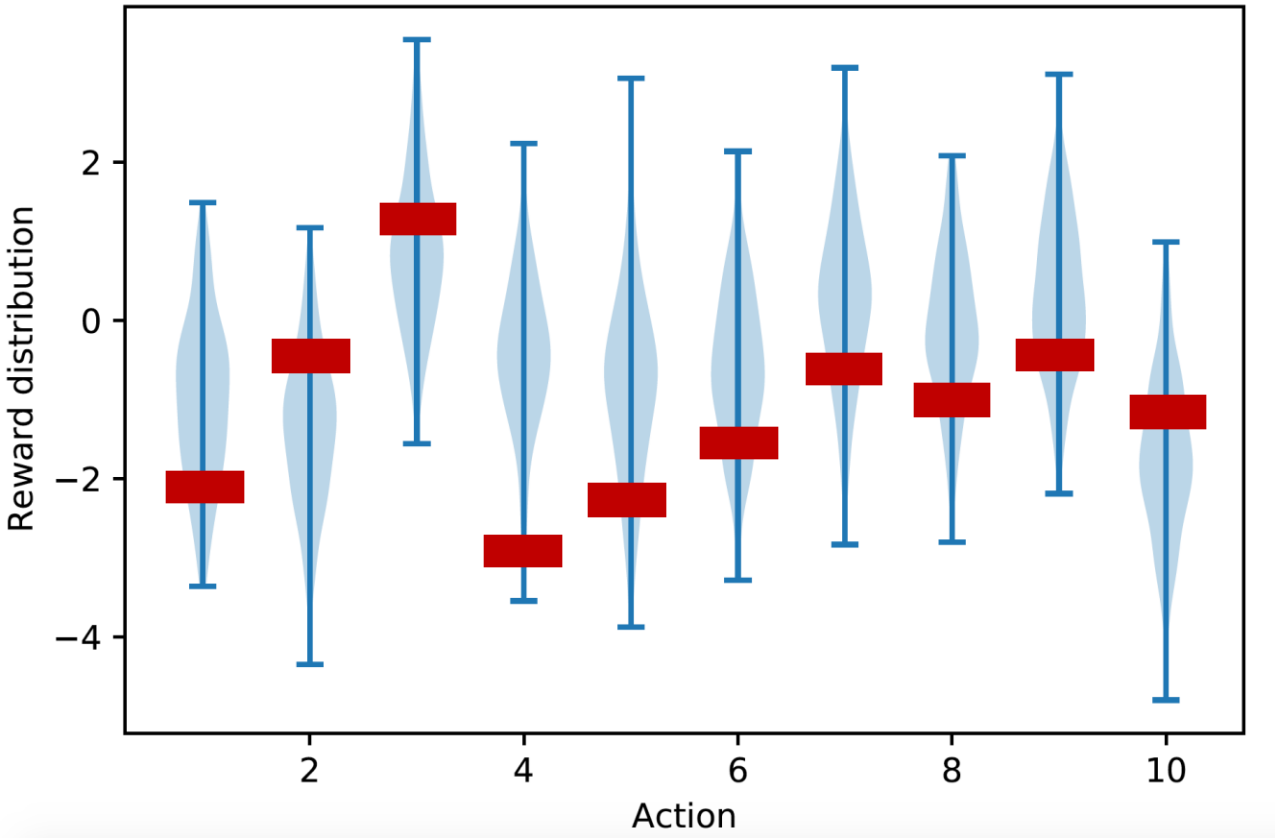
K-armed Bandits: Comments

1. Why don’t we just estimate. rewards distributions from collected data?

-> It is too data expensive, and we don’t really care, we just want to maximize our goal!

G. Box ‘All models are wrong, but some are useful’

2. We have already seen key RL concepts (q-value function, incremental computations)



Great q on a RL perspective! We don’t care about knowing the distributions!!!!

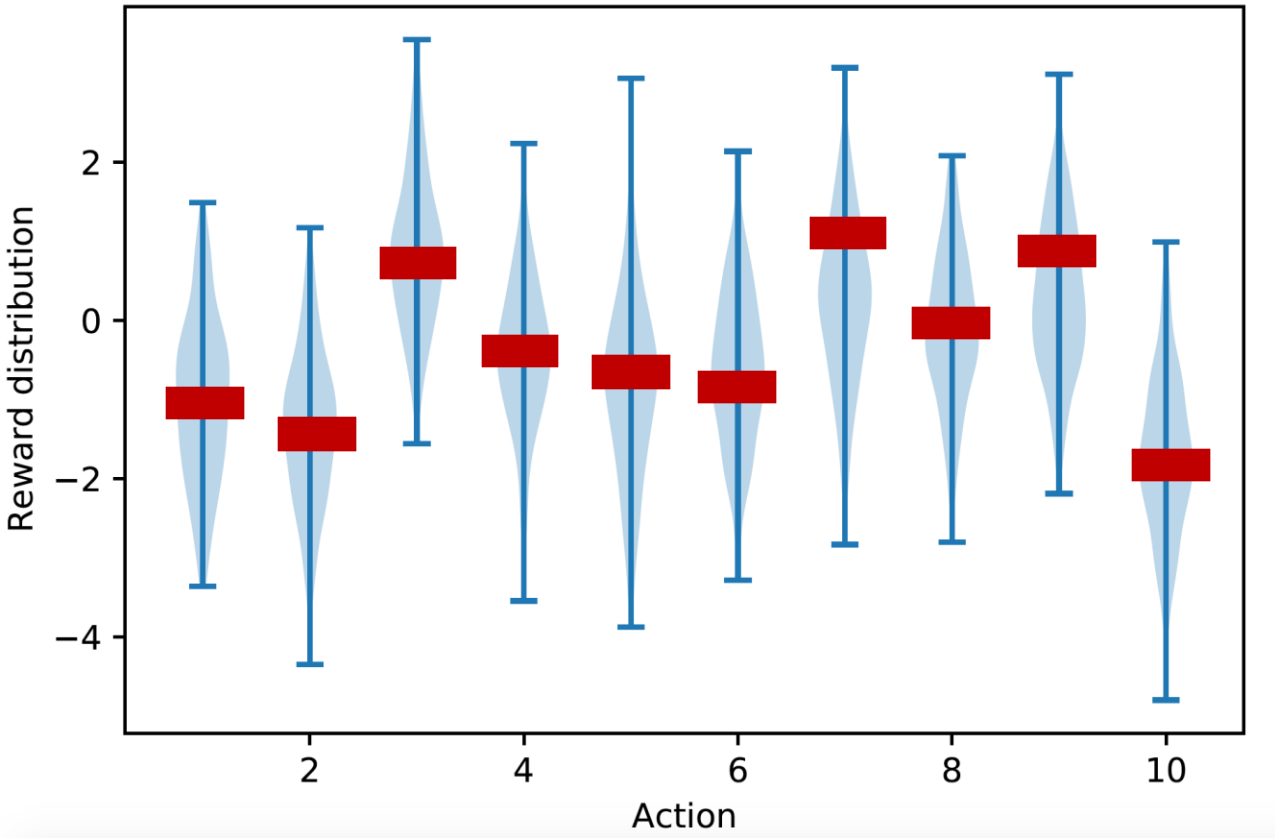
K-armed Bandits: Comments

1. Why don’t we just estimate. rewards distributions from collected data?

-> It is too data expensive, and we don’t really care, we just want to maximize our goal!

G. Box ‘All models are wrong, but some are useful’

2. We have already seen key RL concepts (q-value function, incremental computations)



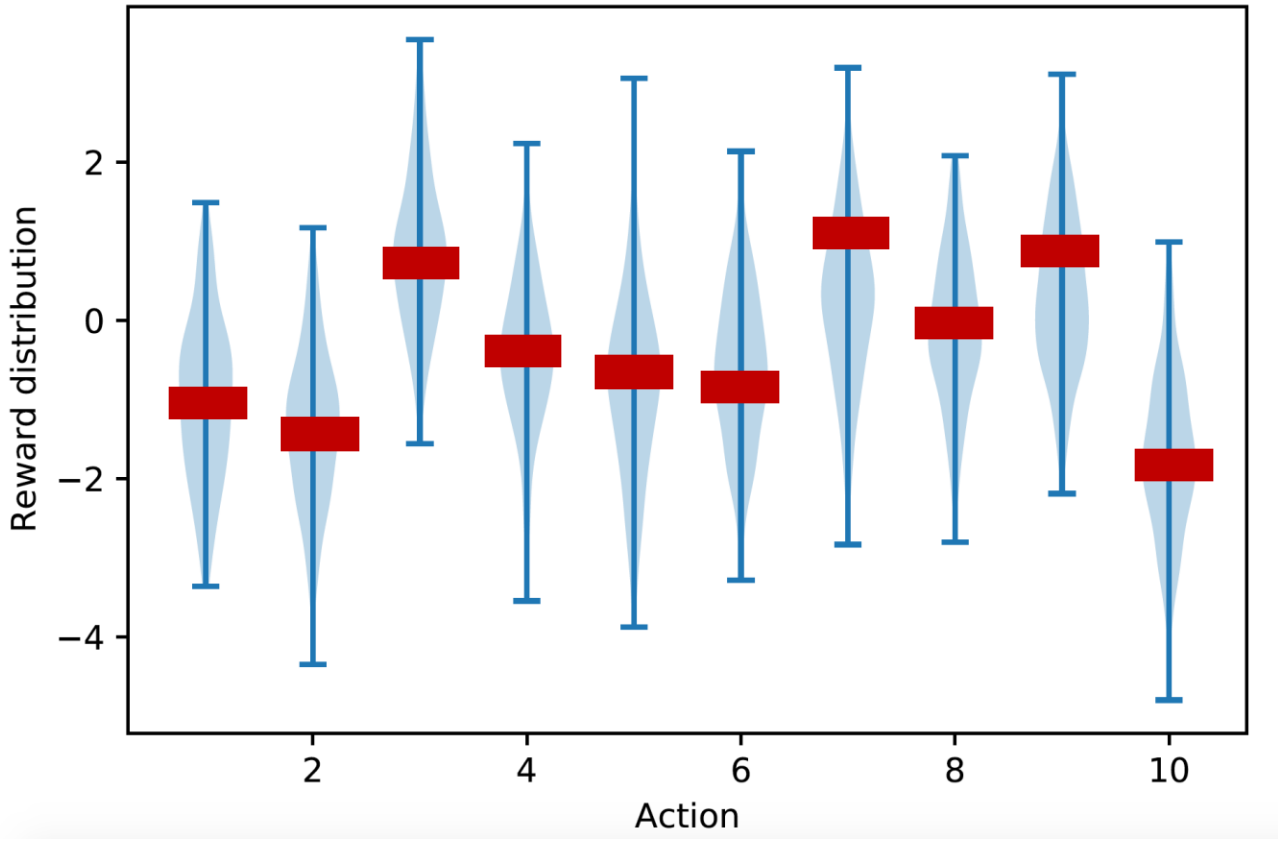
K-armed Bandits: Comments

1. Why don’t we just estimate. rewards distributions from collected data?

-> It is too data expensive, and we don’t really care, we just want to maximize our goal!

G. Box ‘All models are wrong, but some are useful’

2. We have already seen key RL concepts (q-value function, incremental computations)



Even if more accurate, this estimation of q is ‘bad’ on a RL perspective! We don’t measure the goodness of our policy on MSE on q, but on the rewards that we are getting!

K-armed Bandits: Exam

- All the content of Chapter 2 of the book may be exam material beside:

1. ‘The Bandit Gradient Algorithm as Stochastic Gradient Ascent’ from page 38 is not exam material
2. Section 2.9 is not exam material

Credits & Additional Readings

- Several examples were taken from the Fundamental of Reinforcement Learning course by A. White and M. White
- Some contents were taken from ‘Reinforcement Learning @ UCL’ by D. Silver
- Additional Readings (inspired by D. Bacciu):

1. Seminal UCB paper https://nlp.jbnu.ac.kr/AML/papers/auer-ml-02.pdf
2. Randomized UCB algorithm for contextual bandits https://arxiv.org/pdf/1106.2369.pdf
3. Efficient learning of contextual bandit with an oracle http://proceedings.mlr.press/v32/agarwalb14.pdf
4. A Deep Learning based approach to generate exploration bonuses via model-based https://arxiv.org/pdf/1507.00814.pdf

The story so far…

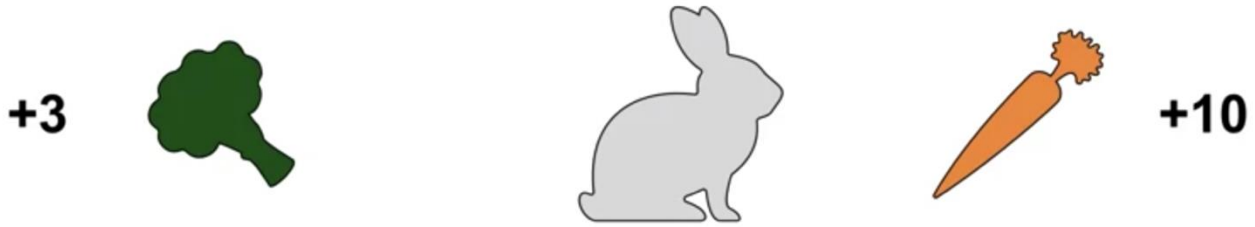
Multi-armed bandits were helpful to introduce some elements (like the exploration-exploitation dilemma), however we have reached a ‘real’ RL scenario:

- The state is always the same, the same actions is always optimal
- While the agent learns from his experience, the decision making is quite simple: a single action do not influence future rewards (and states)

The story so far…

Multi-armed bandits were helpful to introduce some elements (like the exploration-exploitation dilemma), however we have reached a ‘real’ RL scenario:

- The state is always the same, the same actions is always optimal
- While the agent learns from his experience, the decision making is quite simple: a single action do not influence future rewards (and states)

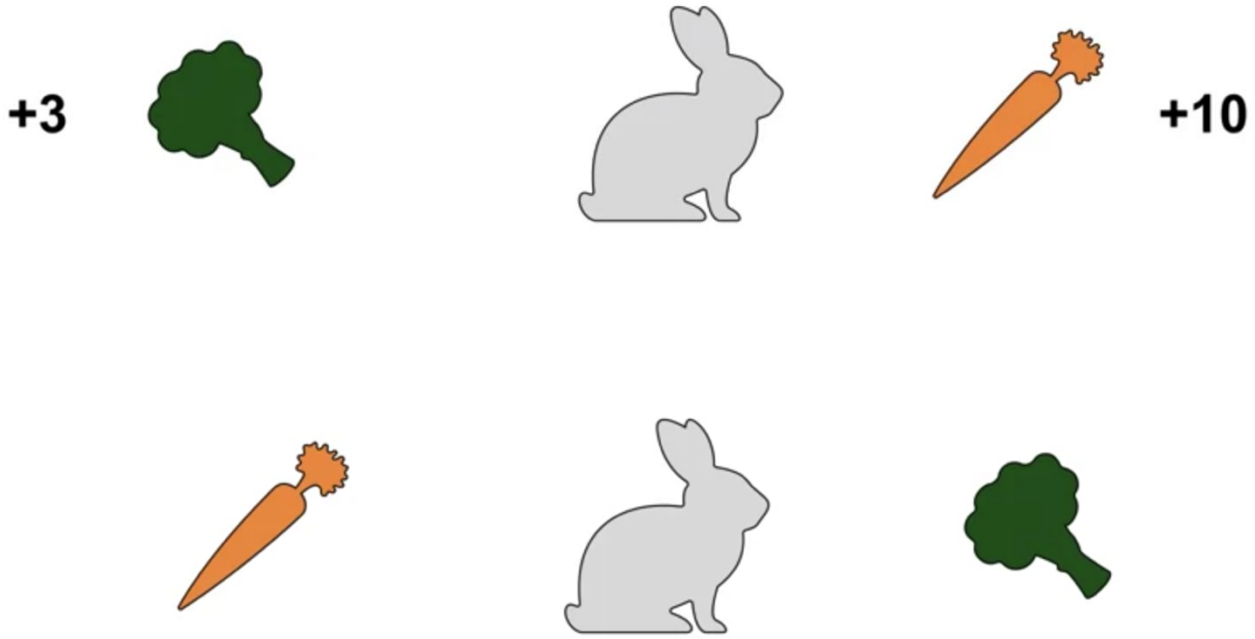


A. White, M. White ‘Fundamentals of Reinforcement Learning’

The story so far…

Multi-armed bandits were helpful to introduce some elements (like the exploration-exploitation dilemma), however we have reached a ‘real’ RL scenario:

- The state is always the same, the same actions is always optimal
- While the agent learns from his experience, the decision making is quite simple: a single action do not influence future rewards (and states)

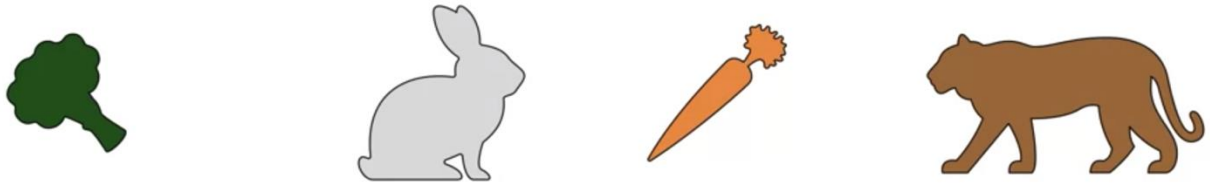


A. White, M. White ‘Fundamentals of Reinforcement Learning’

The story so far…

Multi-armed bandits were helpful to introduce some elements (like the exploration-exploitation dilemma), however we have reached a ‘real’ RL scenario:

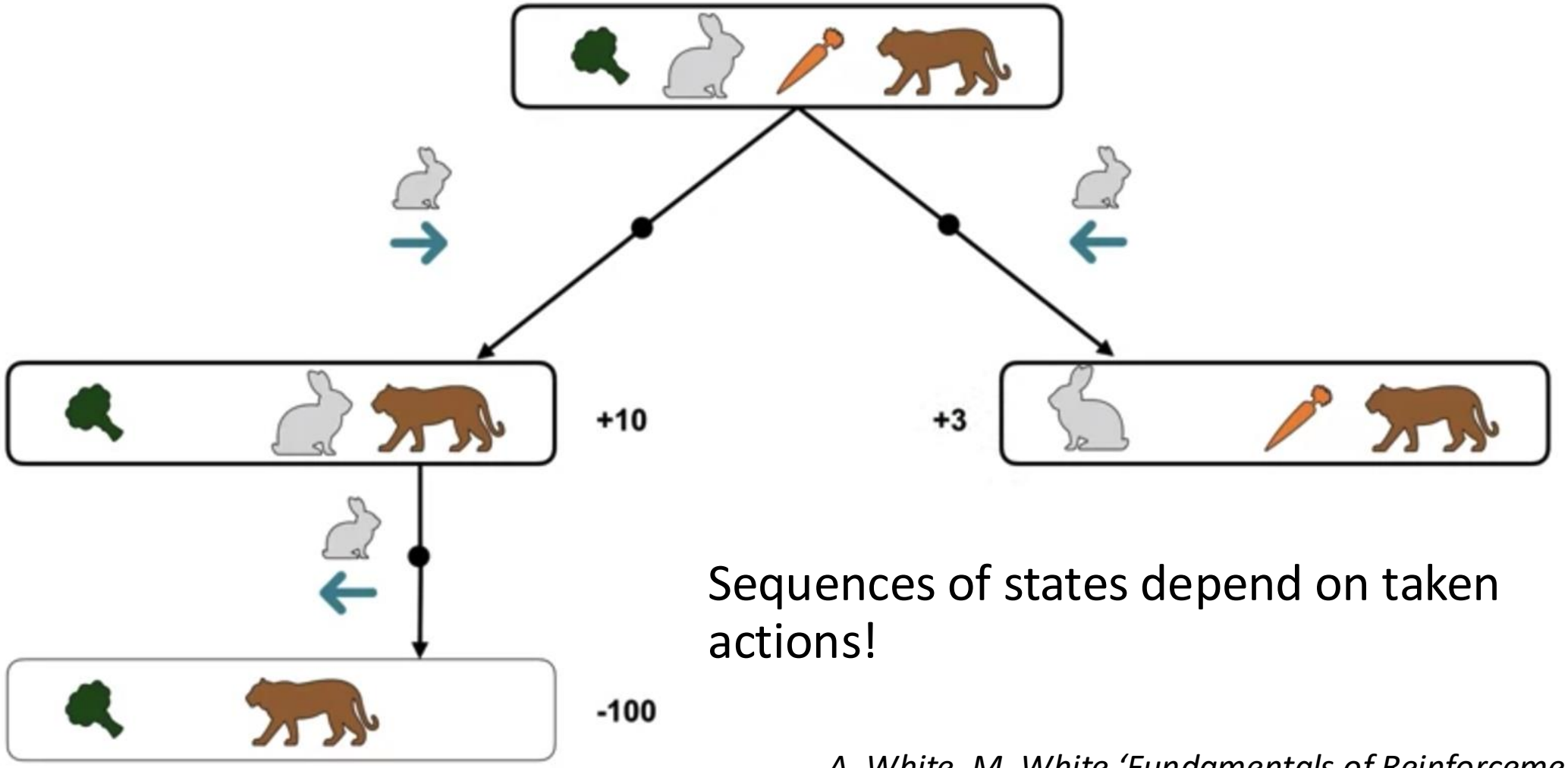
- The state is always the same, the same actions is always optimal
- While the agent learns from his experience, the decision making is quite simple: a single action do not influence future rewards (and states)



Consequences of actions must be taken into account!

A. White, M. White ‘Fundamentals of Reinforcement Learning’

State and long-term planning



Sequences of states depend on taken actions!

A. White, M. White 'Fundamentals of Reinforcement Learning'

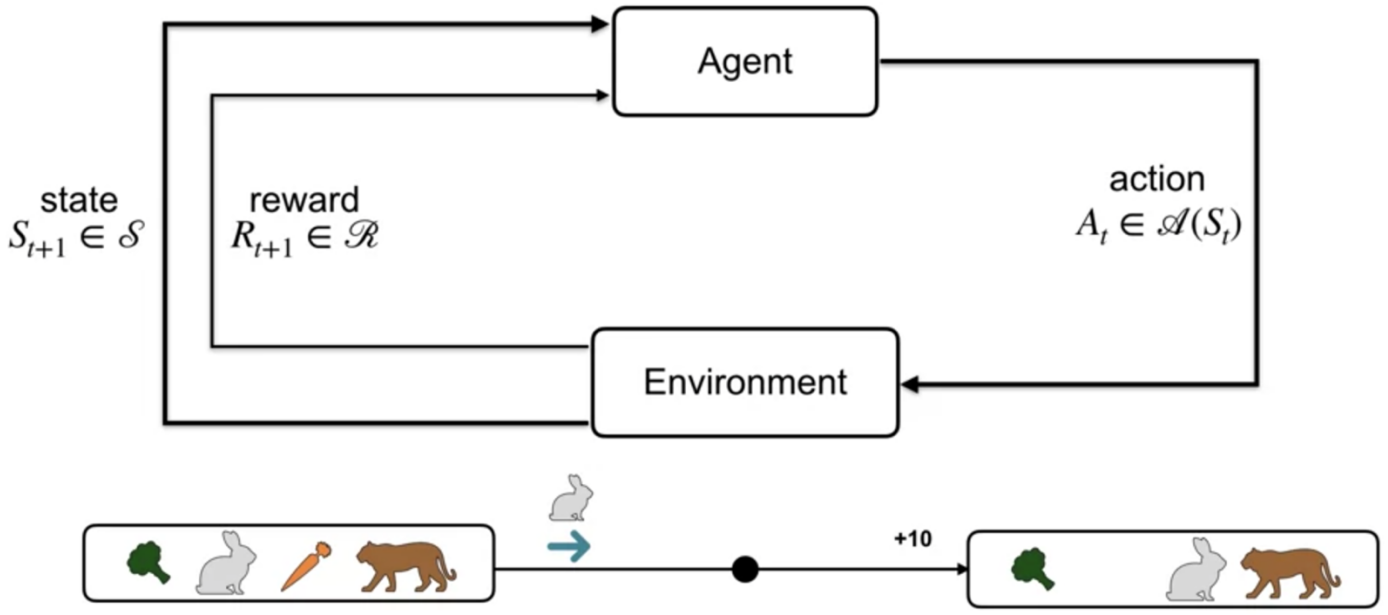
Agent-Environment Interaction (reprise)

At each step t the agent (something that we can program/control):

- Is in the state S_t, an exhaustive description of the system (agent and environment) at time t
- Execute action A_t (that is a valid action from state S_t)

The environment, based of the state/action pair (S_t, A_t):

- provides a reward R_{t+1}
- 'move' the agent in state S_{t+1}



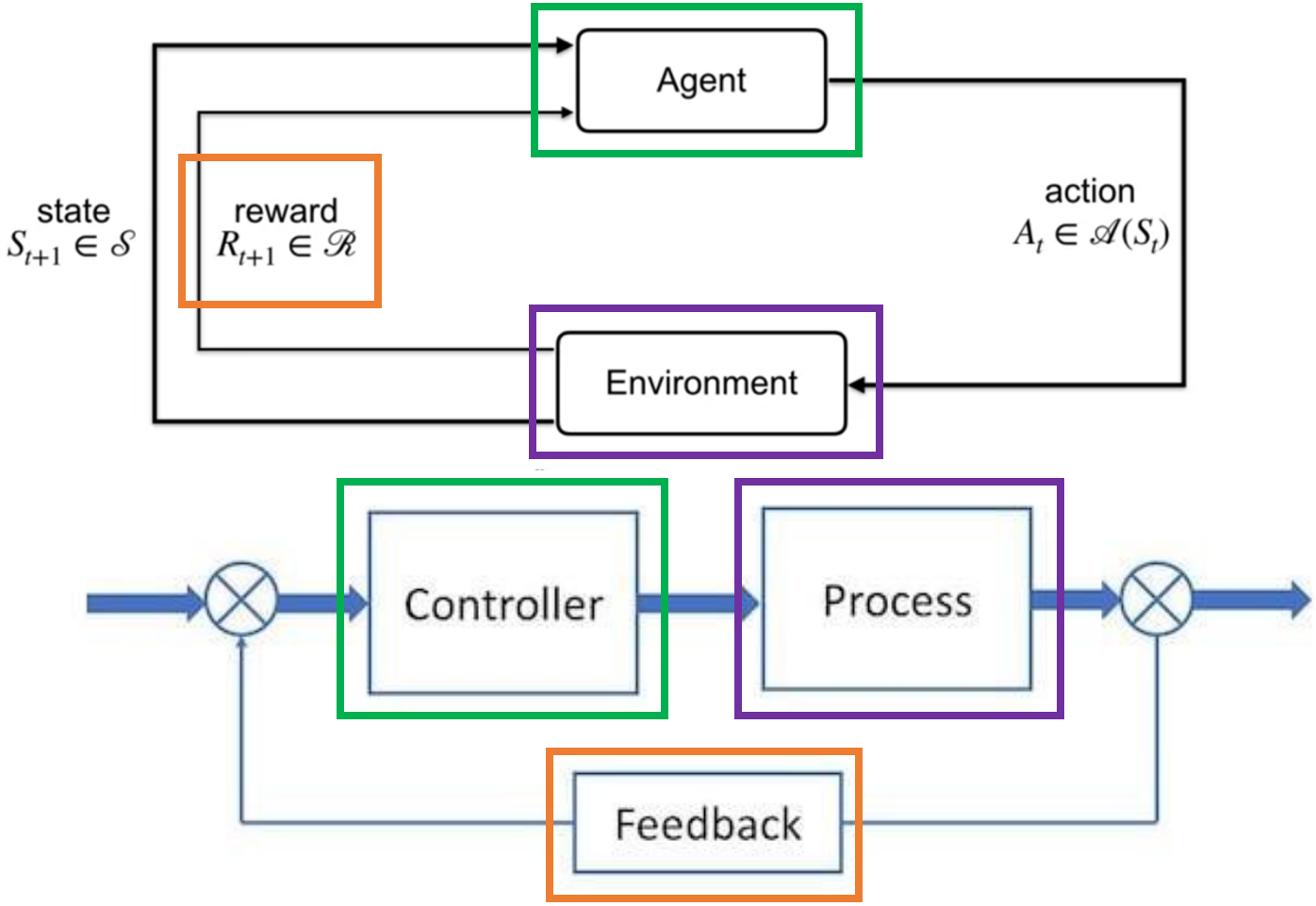
A. White, M. White 'Fundamentals of Reinforcement Learning'

Agent-Environment Interaction (reprise)

We have control over the agent

We have to learn how to ‘beat’ the environment: understanding the long-term behaviour and expected rewards, even if stochasticity is in place

Markov Decision Processes (MDPs) formally describe an environment for Reinforcement Learning



Agent-Environment Interaction (reprise)

We have control over the agent

We have to learn how to ‘beat’ the environment: understanding the long-term behaviour and expected rewards, even if stochasticity is in place

Markov Decision Processes (MDPs) formally describe an environment for Reinforcement Learning

In RL problems the MDP is not typically given!
Some information (for example the transition probability from one state to another) is not known by the agent!

Nevertheless, a MDP is always ‘hidden’ when dealing with a RL problem and it is the basic formalism that we need: MDP-based problems are the ones we want to solve with RL algorithms

Markov Decision Processes (MDPs): Outline

i. Markov Processes
ii. Markov Reward Processes
iii. Markov Decision Processes

We will start considering the fully observable case: we know everything about the environment – ie. the current state completely characterize the process

MDPS are a general framework for almost all RL problems:

- Optimal control deals with continuous MDPs (extensions of what we see today)
- Bandits are 1-state MPDs
- If the environment is not observable, there are extension to MDPs to deal with that

i. Markov Processes: Markov State

- We will always consider the case in which a state is a Markov state, ie. contains all the useful information from the history.

A state s_t is Markov if and only if

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

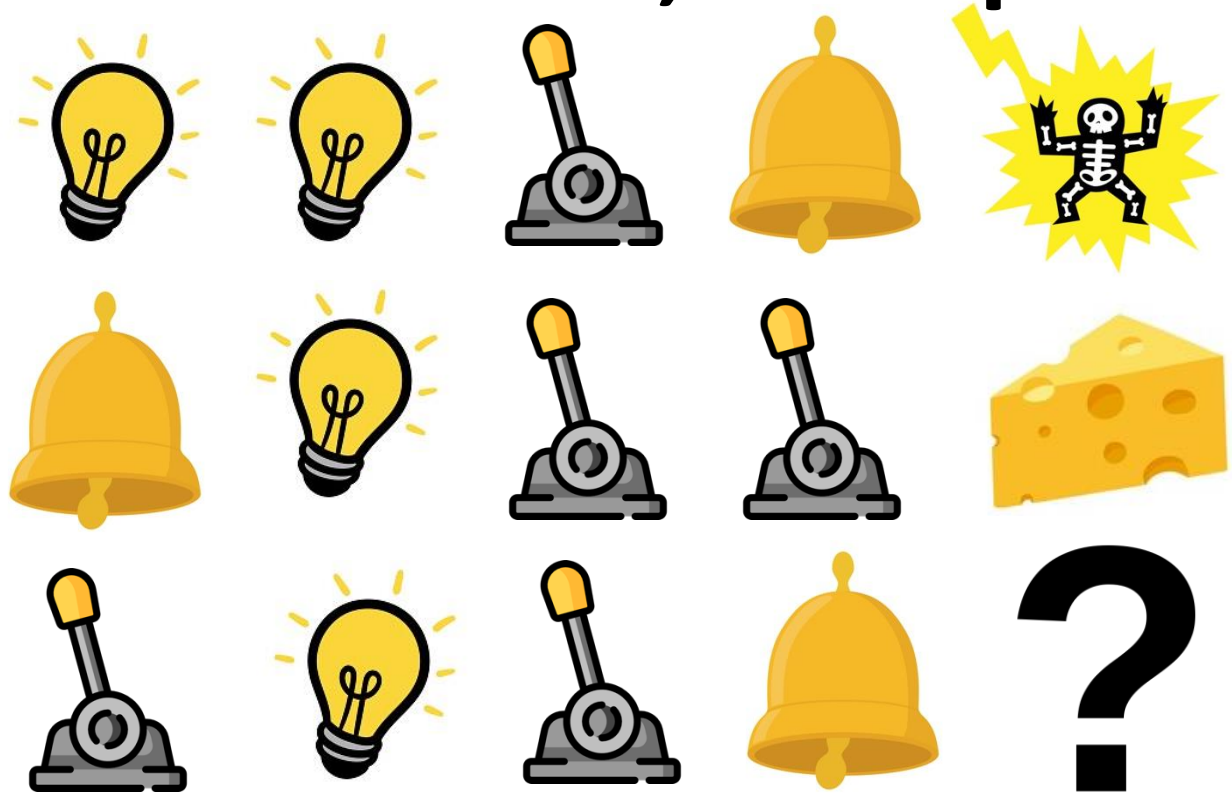
- The state is a sufficient statistics for the present/the past: we can throw away the rest of the history
- Defining a good state is fundamental to develop effective RL solutions: it typically require good domain expertise
- Previous examples definition of state?

i. Markov Processes: State Definition, example

Episode A:

Episode B:

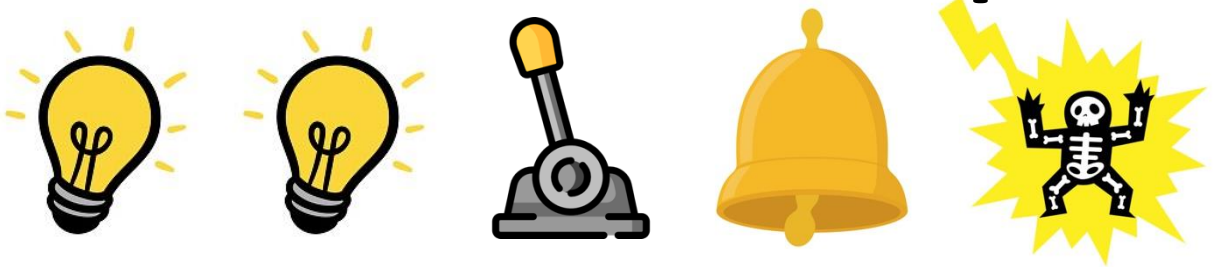
Episode C:



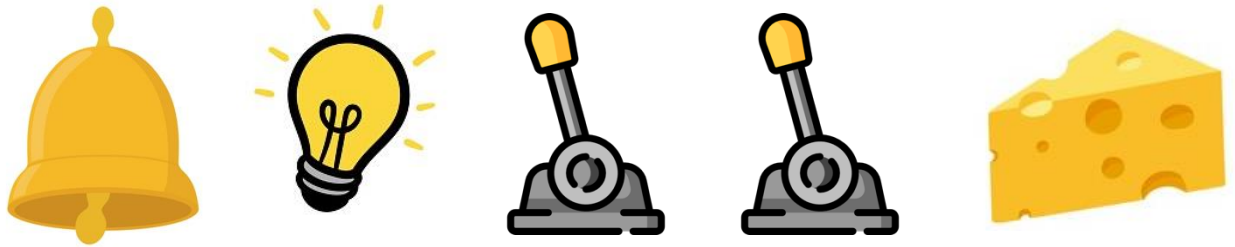
D. Silver 'Reinforcement Learning' @UCL

i. Markov Processes: State Definition, example

Episode A:



Episode B:



Episode C:

D. Silver 'Reinforcement Learning' @UCL

State formalization (i): last event/2 events/3 events ->

State formalization (ii): counter of events ->

State formalization (iii): last 4 events -> undefined next state in Episode C

i. Markov Processes: Markov State

In Markov Processes, we don't have rewards and actions: the agent has no agency in this case (we are not dealing with a RL problem yet))

However, transitions from one state s to its successor s' are determined by a state transition probability (stochastic behaviour!)

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

State transition matrix $\mathcal{P}$ defines all the transition probabilities.

In $\mathcal{P}$ each row characterize everything from a particular state and sums to 1!

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

i. Markov Processes: Markov Chain

In other words…

A Markov process is a memoryless random process: a sequence of random states $S_1, S_2, \ldots$ with the Markov Property

Definition

A Markov Process (or Markov Chain) is a tuple $\langle \mathcal{S}, \mathcal{P} \rangle$ such that:

- $\mathcal{S}$ is a finite* set of states
- $\mathcal{P}$ is a state transition probability matrix with entries

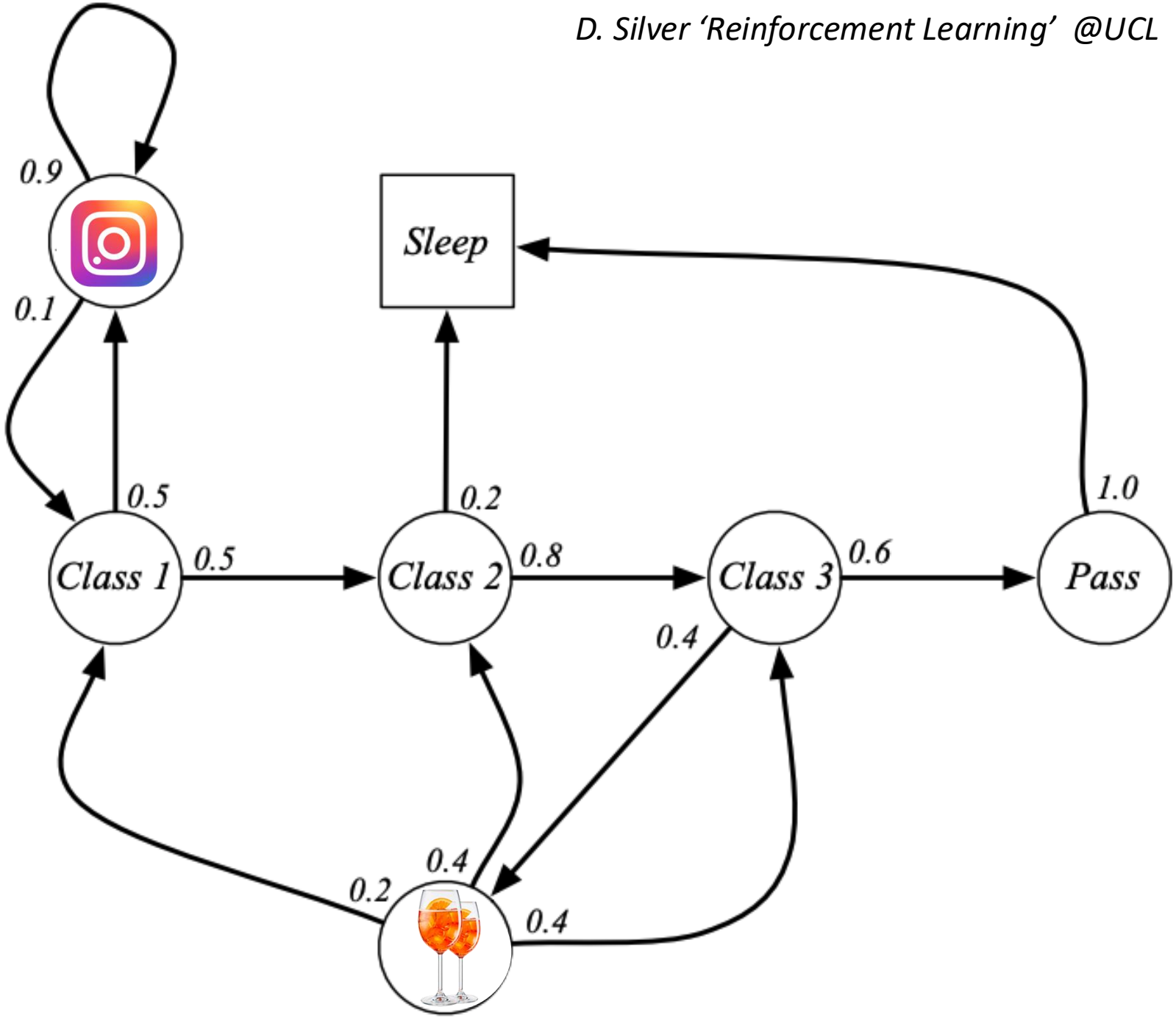
$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

* We will always consider finite cases when dealing with episodis tasks

i. Markov Processes:

Student Markov Chain

D. Silver 'Reinforcement Learning' @UCL

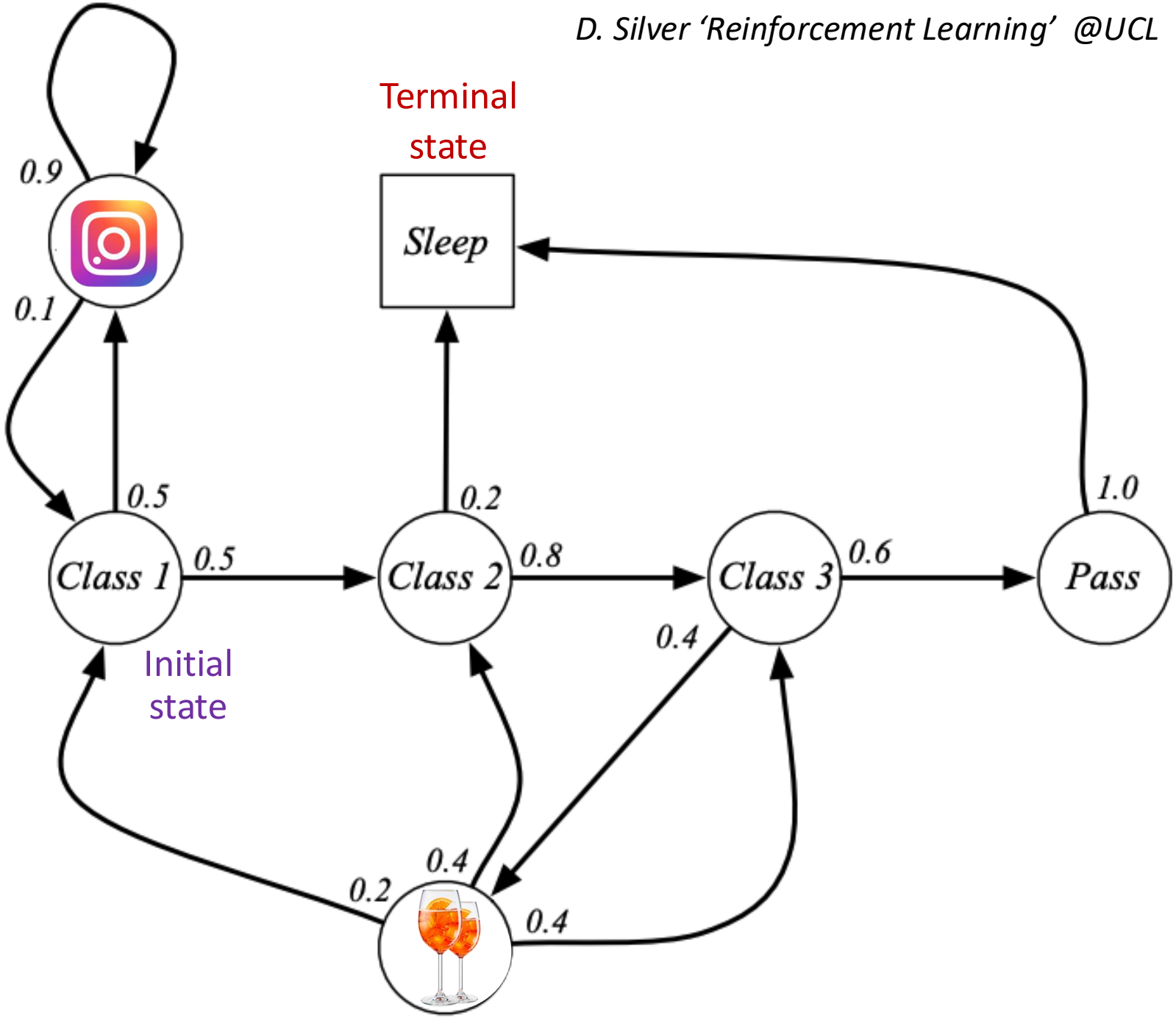


i. Markov Processes: Student Markov Chain

In this example (not always true)

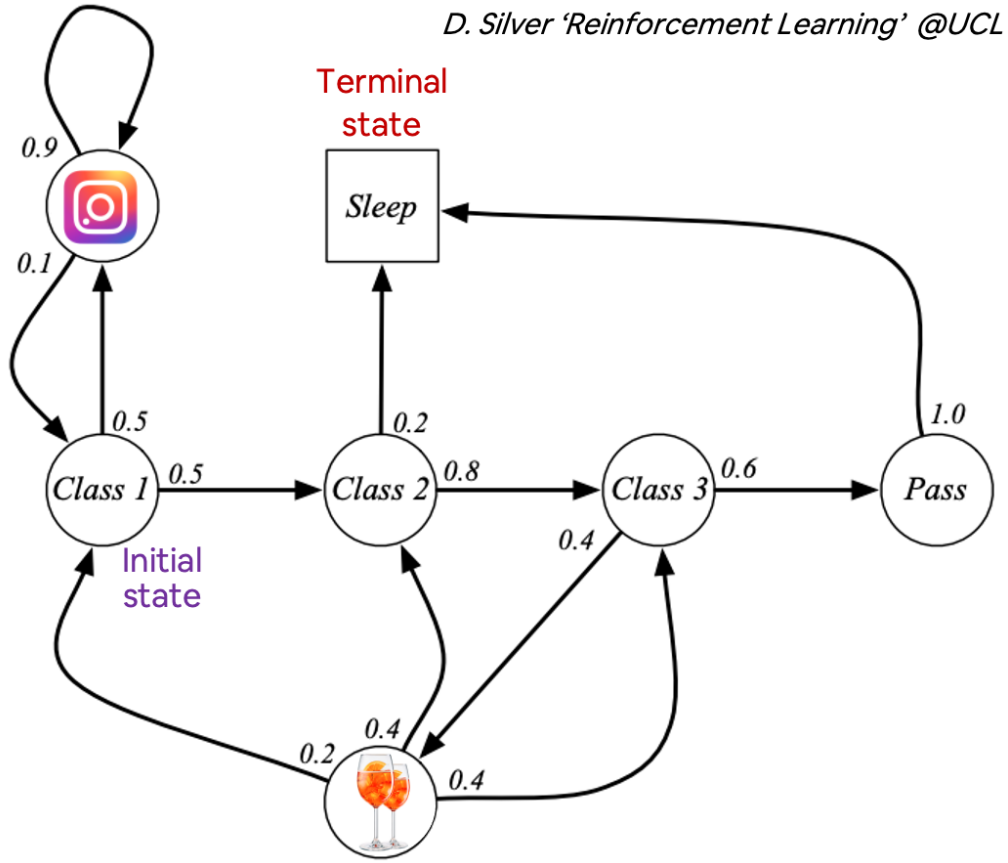
- One initial states (*Class 1*)
- One terminal state. (*Sleep*)

Remember, there is no agency: think of this as the lecturer p.o.v.



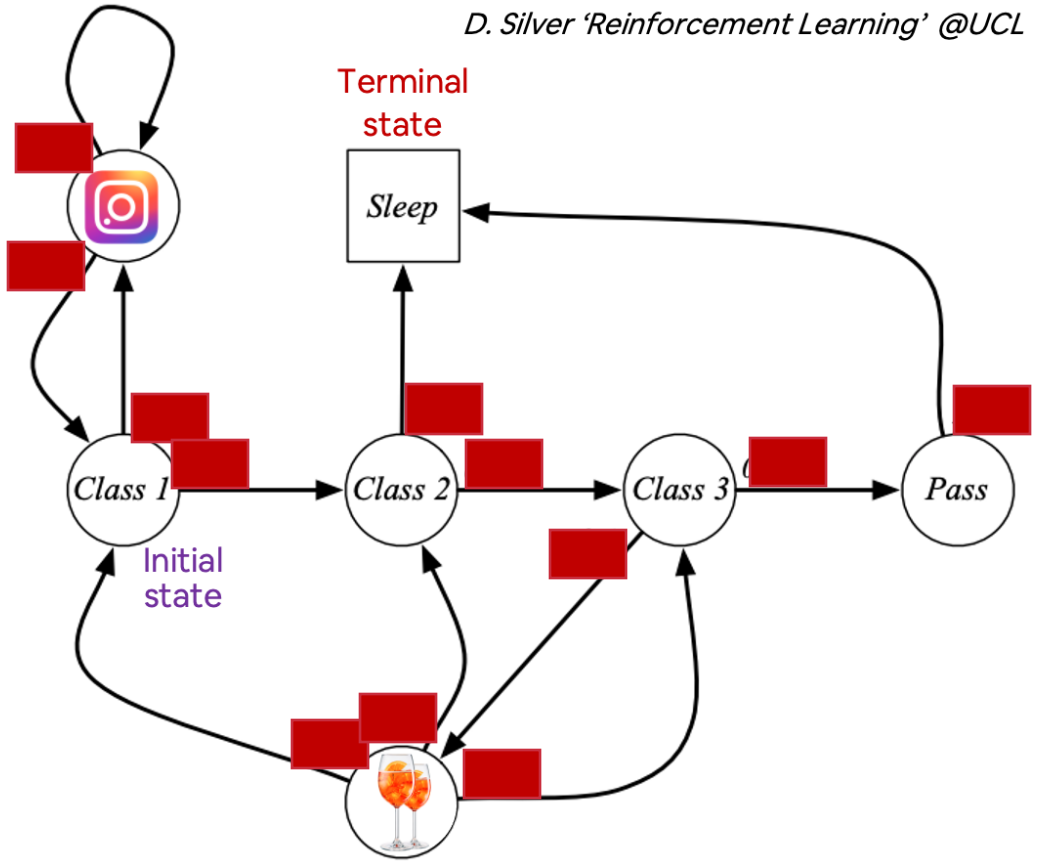
i. Markov Processes: Student Markov Chain

$$\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Spritz \\ IG \\ Sleep \end{array} \begin{array}{c} \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Spritz & IG & Sleep \end{array} \\ \begin{bmatrix} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{array}$$



i. Markov Processes: Student Markov Chain

$$\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Spritz \\ IG \\ Sleep \end{array} \begin{array}{c} \begin{array}{cccccccc} C1 & C2 & C3 & Pass & Spritz & IG & Sleep \end{array} \\ \left[\begin{array}{cccccccc} & & & & ? & & & \end{array}\right] \end{array}$$



A sneak peek: when dealing with the 'full RL' problem, the transition probabilities will not be known!

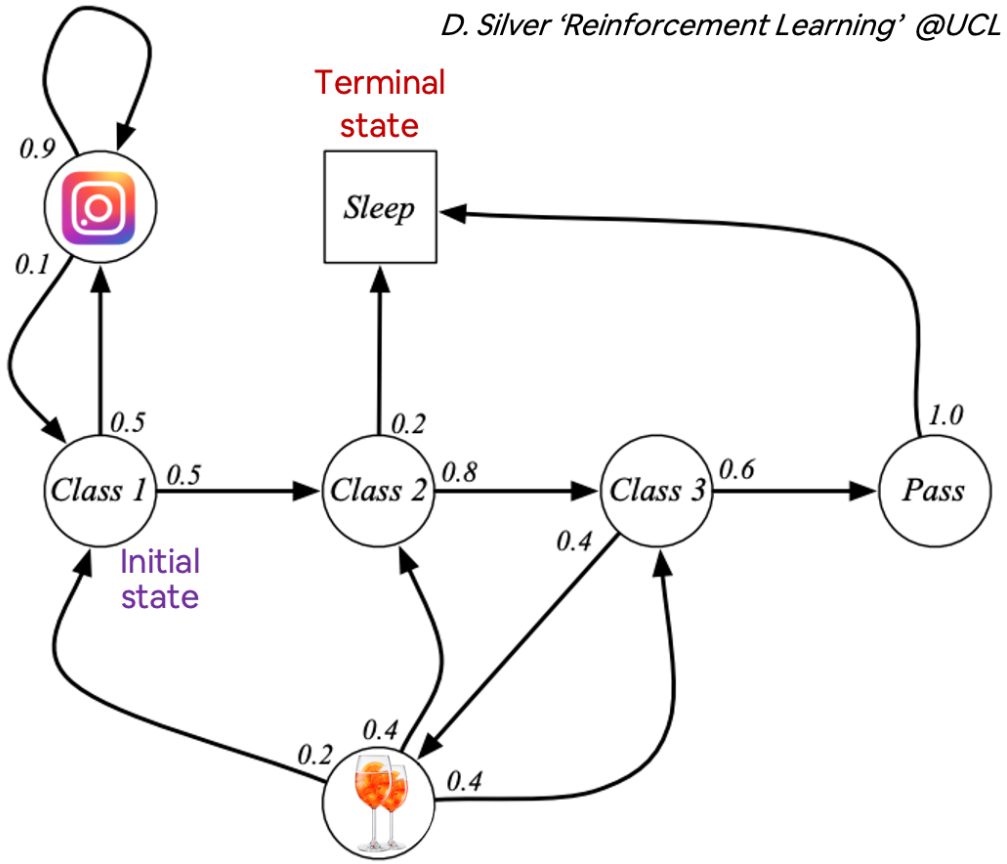
i. Markov Processes: Student Markov Chain

We can draw samples (sequences of states, episodes) from this Markov Chain:

- C1 C2 C3 Pass Sleep
- C1 IG IG C1 C2 Sleep
- C1 C2 C3 Spritz C2 C3 Pass Sleep
- C1 IG IG C1 C2 C3 Spritz C1 IG IG IG C1 C2 C3 Spritz C2 Sleep

(The samples are random, governed by the transition probability)

With data, we can estimate the transition probabilities!



i. Markov Processes: Student Markov Chain

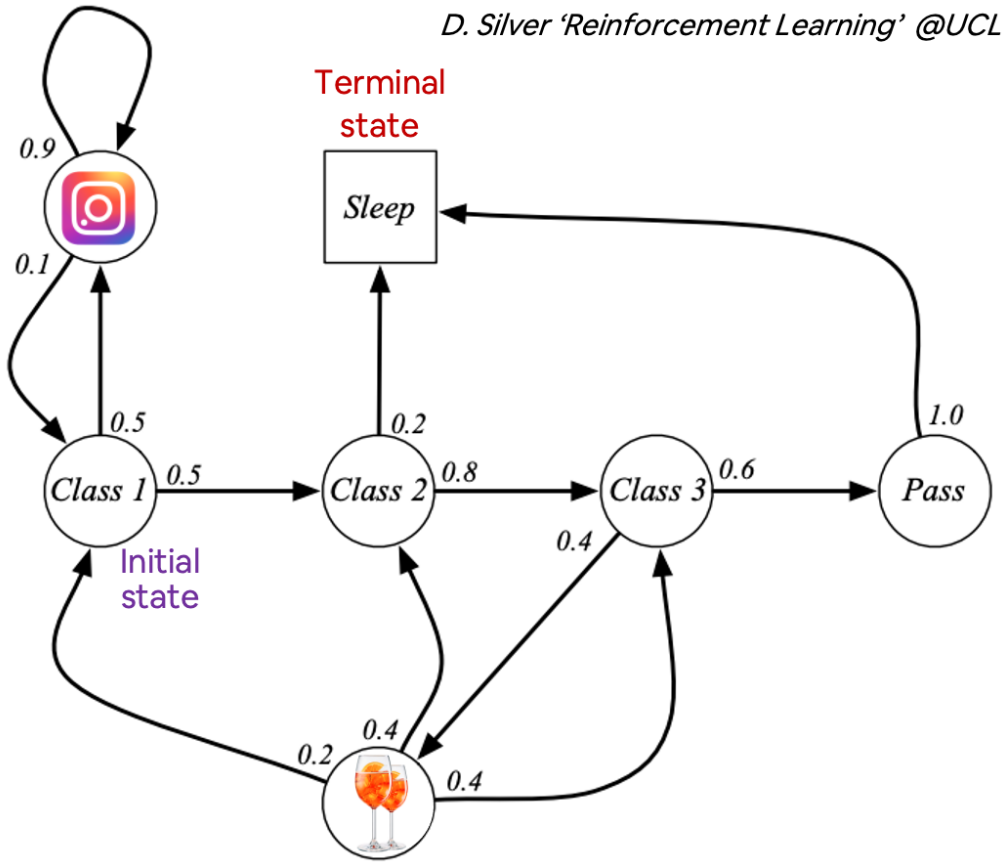
We can draw samples (sequences of states, episodes) from this Markov Chain:

- C1 C2 C3 Pass Sleep
- C1 IG IG C1 C2 Sleep
- C1 C2 C3 Spritz C2 C3 Pass Sleep
- C1 IG IG C1 C2 C3 Spritz C1 IG IG IG C1 C2 C3 Spritz C2 Sleep

(The samples are random, governed by the transition probability)

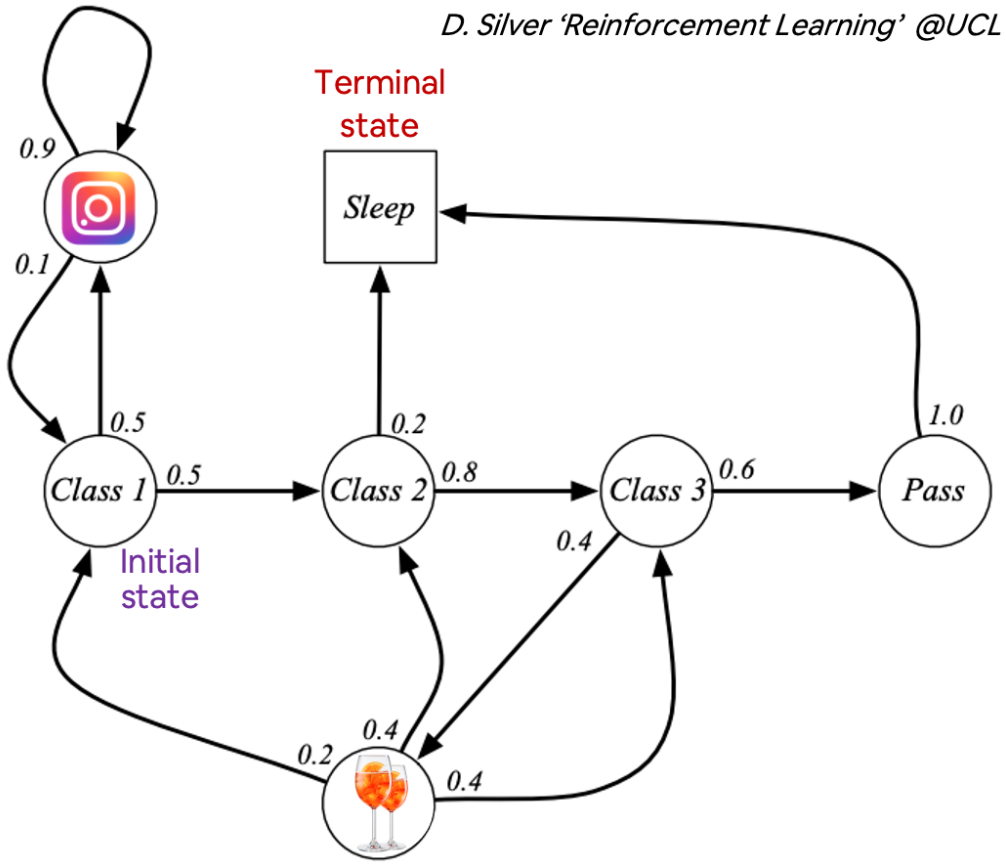
With data, we can estimate the transition probabilities!

However, we will not do it, our goal will not be to have a mathematical model of the environment (too costly) but just deriving an 'optimal' strategy!



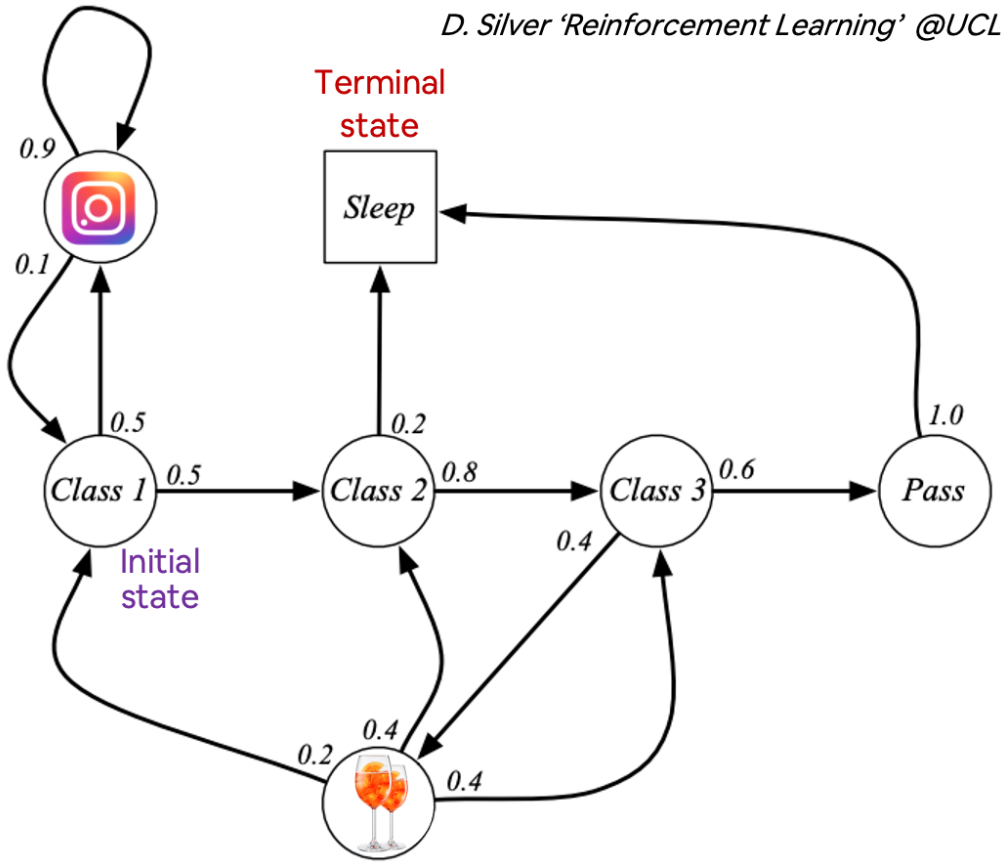
i. Markov Processes: Remarks

- Markov Processes can also model complex scenarios with many states
- There are scenarios where probabilities change over time. The are non-stationary Markov Process versions.



i. Markov Processes: Remarks

- Markov Processes can also model complex scenarios with many states
- There are scenarios where probabilities change over time. The are non-stationary Markov Process versions.



What is missing?

ii. Markov Reward Processes

Let's add rewards: a Markov Reward process is a Markov Chain with reward values

Definition

A Markov Reward Process is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ such that:

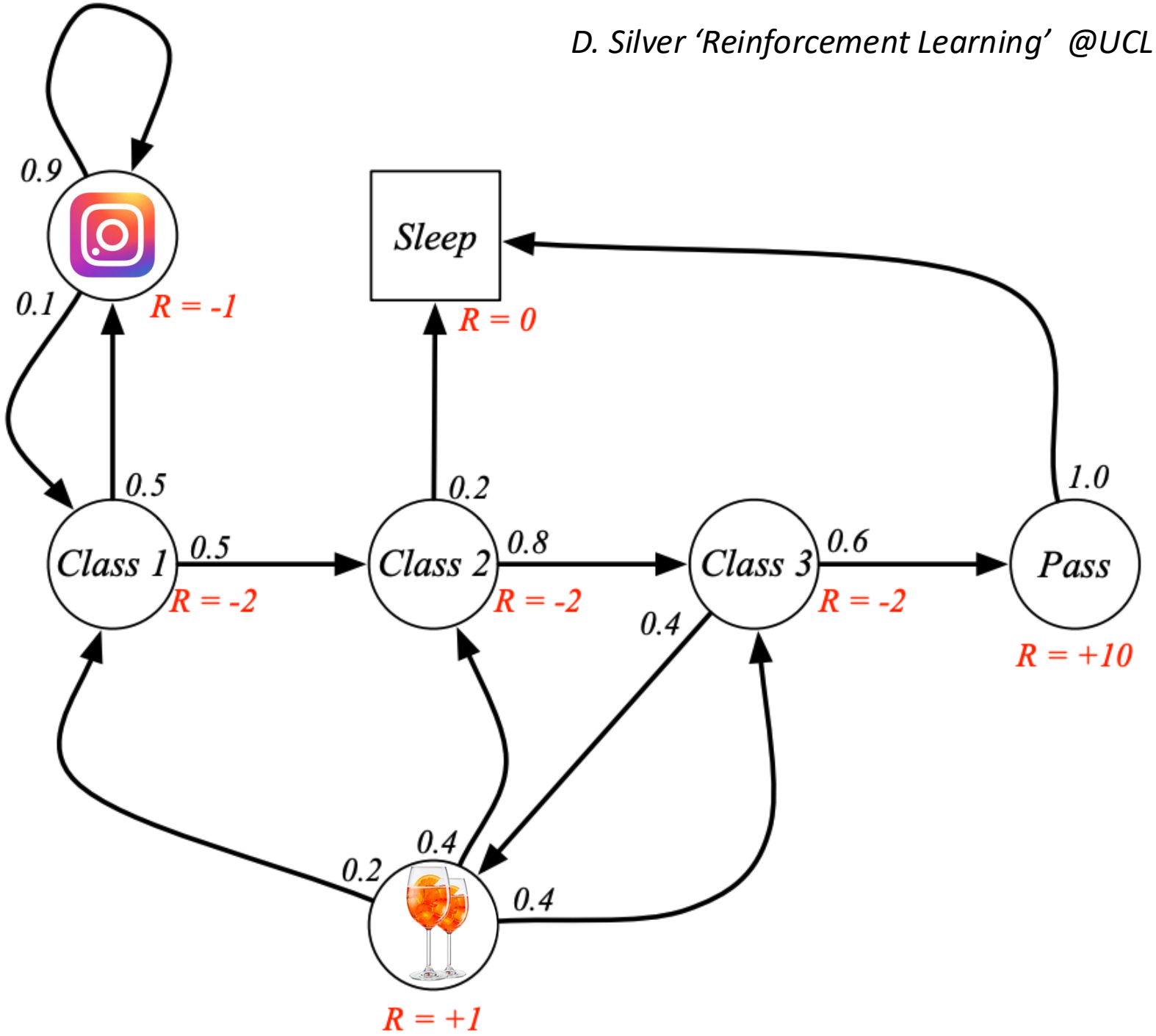
- $\mathcal{S}$ is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix with entries

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}\left[R_{t+1} | S_t = s\right]$ (it is just the immediate reward, in that specific state)
- γ is a discount factor, $\gamma \in [0,1]$

D. Silver 'Reinforcement Learning' @UCL

ii. Markov Reward Processes : Student Markov Chain



ii. Markov Reward Processes: Return

Definition

The Return G_t is the total discounted reward from time-step t

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where γ is a discount factor, $\gamma \in [0,1]$ (0 = myopic, 1 = far-sigthed)

- In RL we are not interested in maximizing the value of a single step, but we want to maximize the return (return = goal of RL).

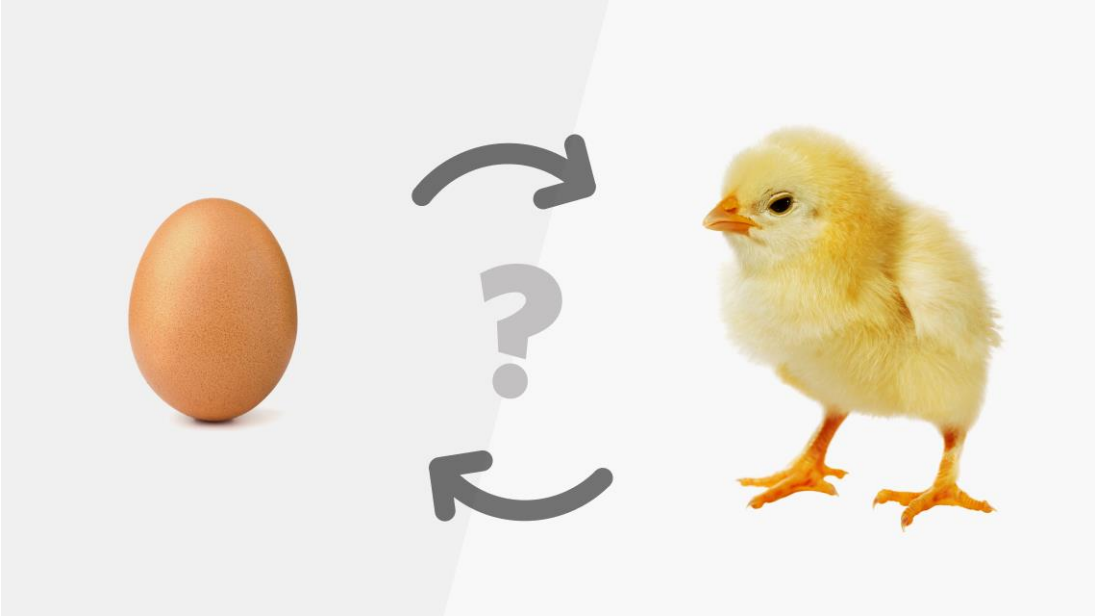
- The discount is the present value of future rewards:

$\gamma = 0$ is the 'myopic' case (we give value only to present reward)

$\gamma = 1$ is the 'far-sighted' case (all rewards are important, even if far away in the future)

ii. Markov Reward Processes: Return

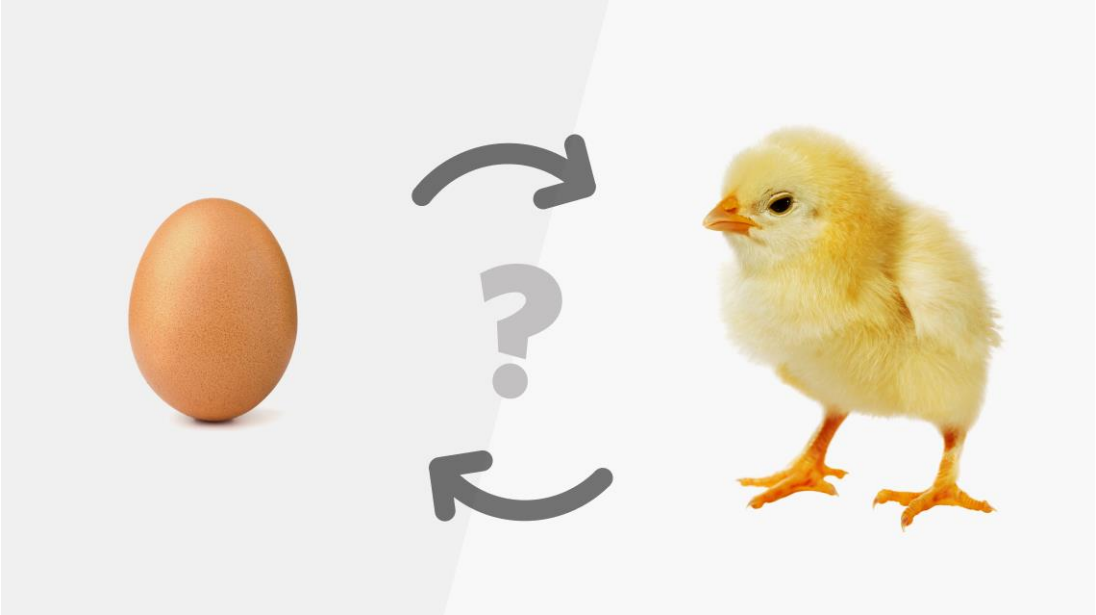
Why discounts?



ii. Markov Reward Processes: Return

Why discounts?

- Mathematically convenient: analysis can be simplified by the presence of rewards (for example we can avoid infinite returns in Markov Processes with cycles)
- Uncertainty about the future may not be fully represented
- Financial inspiration: immediate rewards may earn more interest than delayed rewards
- Animals and humans show preference for immediate rewards
- It is a general formulation: if $\gamma = 1$ we are considering the undiscounted Markov reward processes



ii. Markov Reward Processes: Value Function

The value function $v(s)$ gives the long-term value of state s

Definition

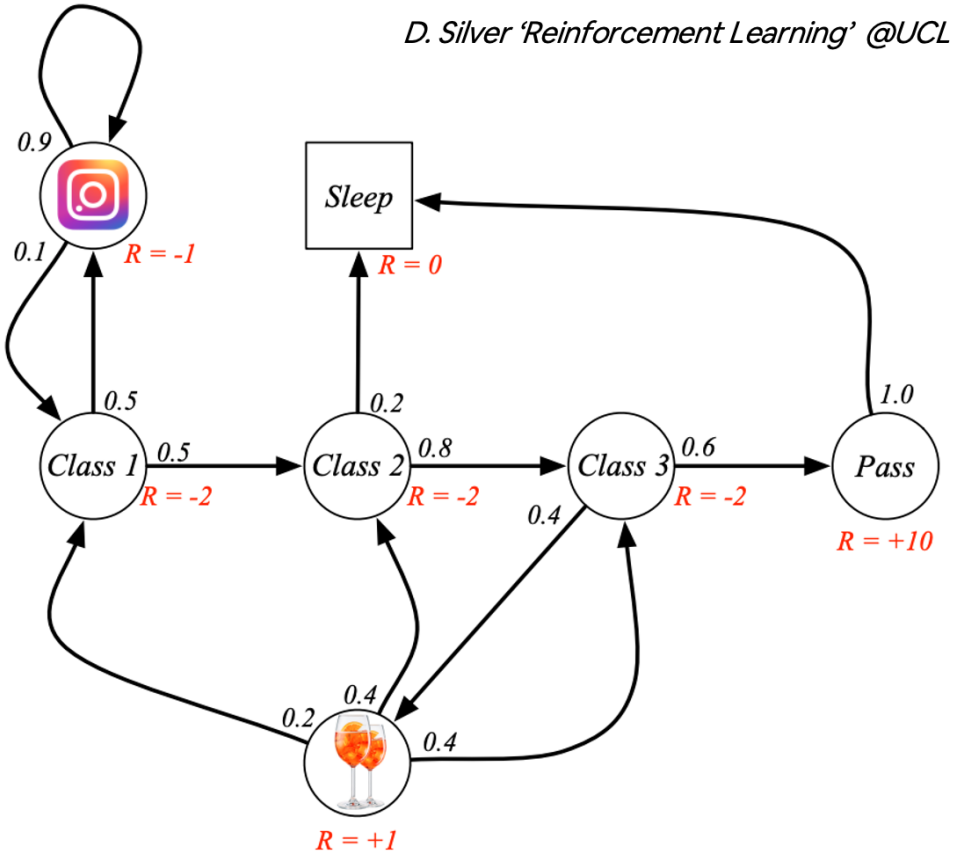
The state value function $v(s)$ of a Markov Reward Process is the expected return starting from state s

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

since there's no agency, the expected value is taken over transitions

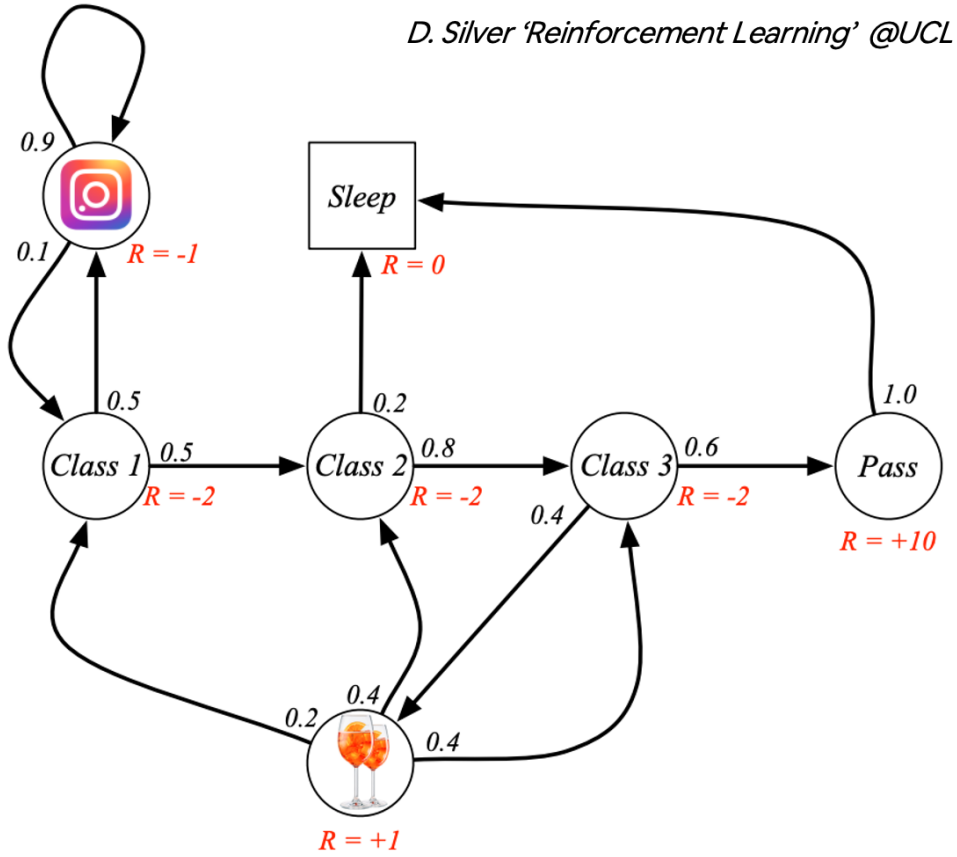
Please note the expectation: this is fundamental since we are in stochastic settings

ii. Markov Reward Processes : Student Markov Chain



How to compute state value functions from a Markov Reward Process?

ii. Markov Reward Processes : Student Markov Chain



Let's consider $\gamma = 1/2$ and the return obtain with the available samples

C1 C2 C3 Pass Sleep -> v(C1) = -2-2/2-2/4+10/8 = -2.25

C1 IG IG C1 C2 Sleep -> v(C1) = -2-1/2-1/4-2/8-2/16 = -3.125

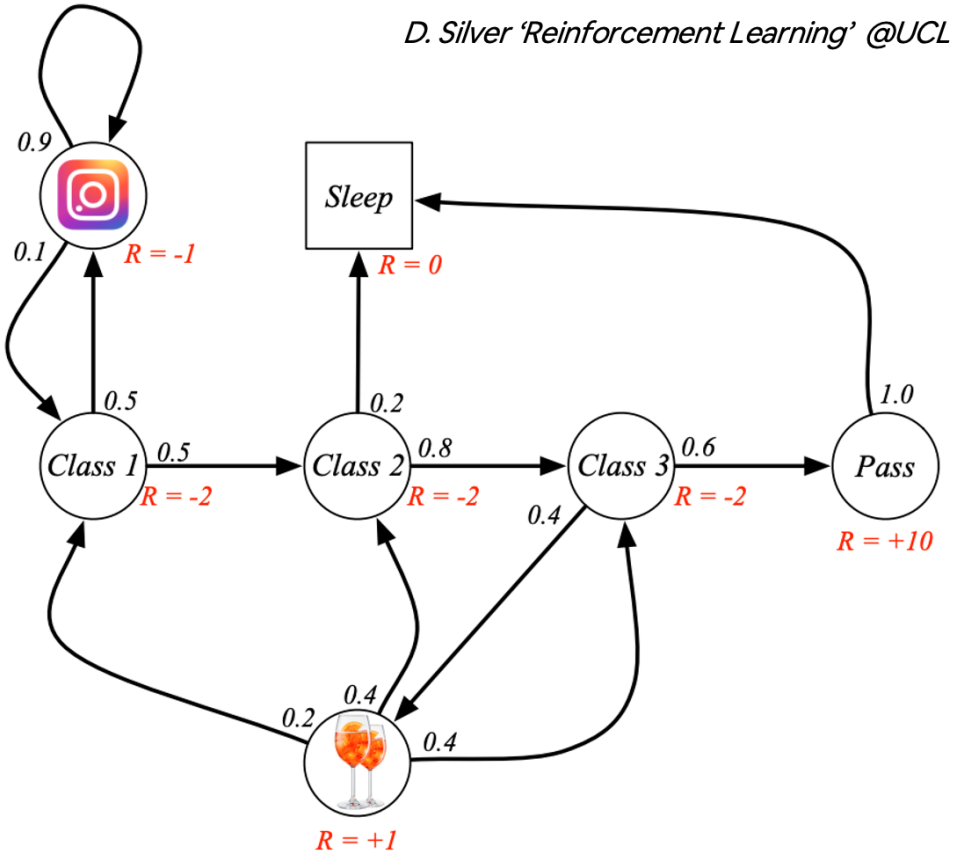
C1 C2 C3 Spritz C2 C3 Pass Sleep ->

v(C1) = -2-2/2-2/4+1/8-2/16-2/32+10/64 = -3.41

C1 IG IG C1 C2 C3 Spritz C1 IG IG IG C1 C2 C3 Spritz C2 Sleep

v(C1) = -2-1/2-1/4-2/8-2/16+ … = -3.20

ii. Markov Reward Processes : Student Markov Chain



Let's consider $\gamma = 1/2$ and the return obtain with the available samples

C1 C2 C3 Pass Sleep -> v(C1) = -2-2/2-2/4+10/8 = -2.25

C1 IG IG C1 C2 Sleep -> v(C1) = -2-1/2-1/4-2/8-2/16 = -3.125

C1 C2 C3 Spritz C2 C3 Pass Sleep ->

v(C1) = -2-2/2-2/4+1/8-2/16-2/32+10/64 = -3.41

C1 IG IG C1 C2 C3 Spritz C1 IG IG IG C1 C2 C3 Spritz C2 Sleep

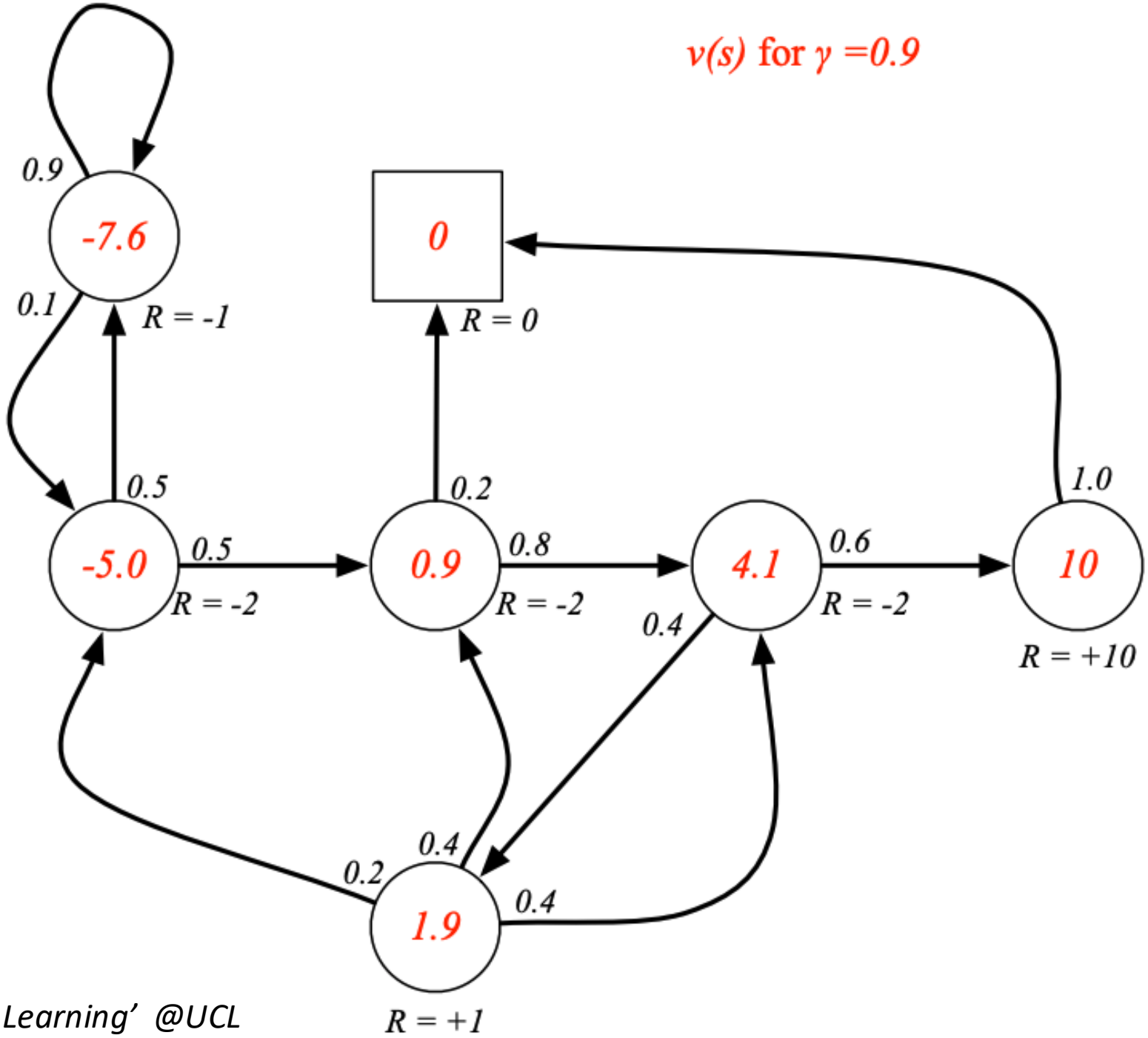
v(C1) = -2-1/2-1/4-2/8-2/16+ … = -3.20

If we have samples, an estimate of the value function for state s is provided by the sampled average of the returns seen from that state.

Ie. In this case v(C1) = (-2.25-3.125-3.41-3.20)/4 = -3

ii. Markov Reward Processes : Student Markov Chain

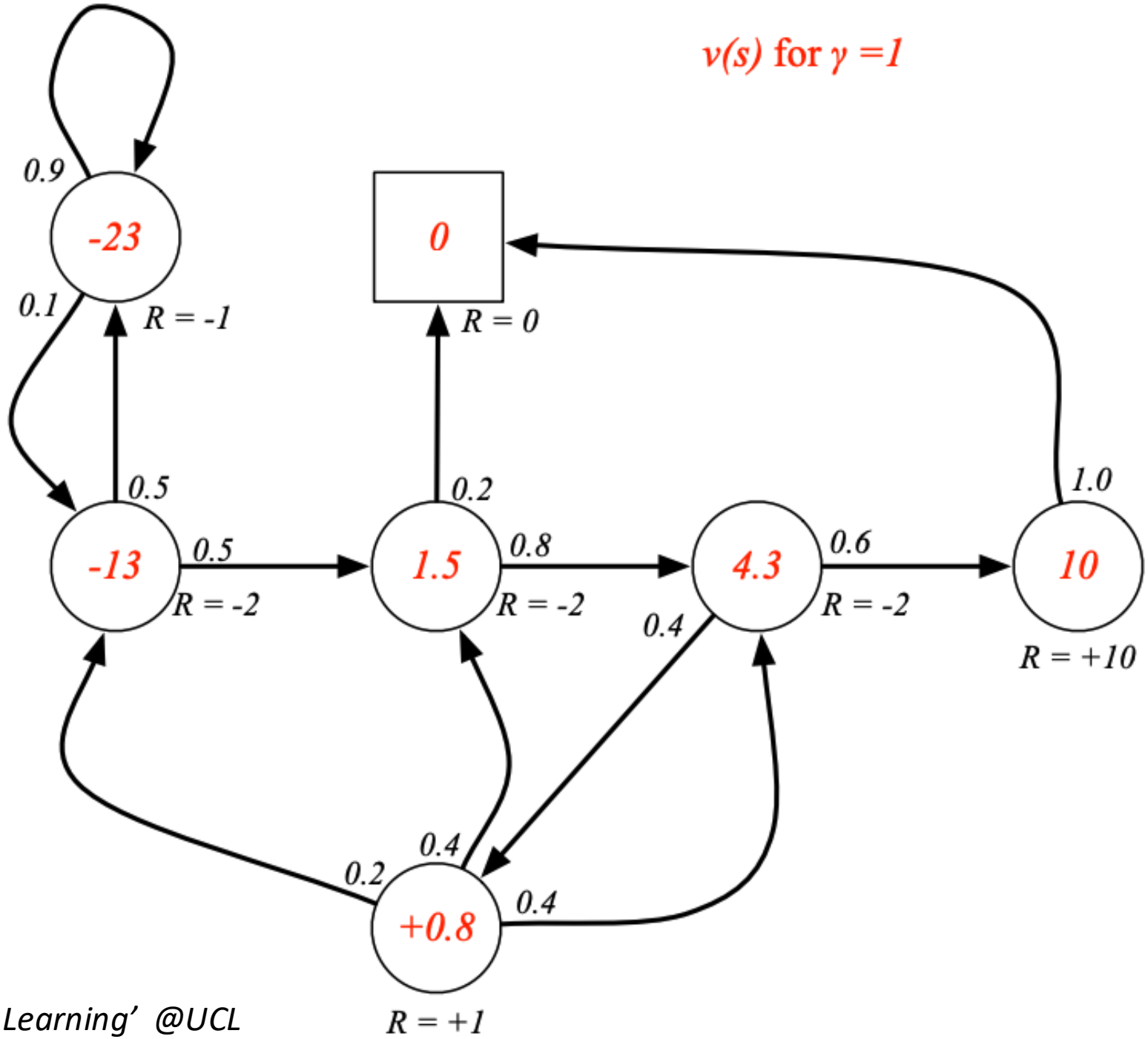
Different discounts values = different state values functions!



D. Silver 'Reinforcement Learning' @UCL

ii. Markov Reward Processes : Student Markov Chain

Different discounts values = different state values functions!



D. Silver 'Reinforcement Learning' @UCL

ii. Markov Reward Processes: Bellman Equation

The value function $v(s)$ can be decomposed into 2 parts:

- The immediate reward R_{t+1}
- The discounted value of successor state $\gamma v(S_{t+1})$

Exploiting

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots\right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

We have that

$$\begin{aligned} v(s) = \mathbb{E}[G_t | S_t = s] &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \end{aligned}$$

ii. Markov Reward Processes: Bellman Equation

The value function $v(s)$ can be decomposed into 2 parts:

- The immediate reward R_{t+1}
- The discounted value of successor state $\gamma v(S_{t+1})$

Exploiting

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots\right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

We will use 'iterative definitions' many times throughout the course!

We have that (law of iterated expectations)

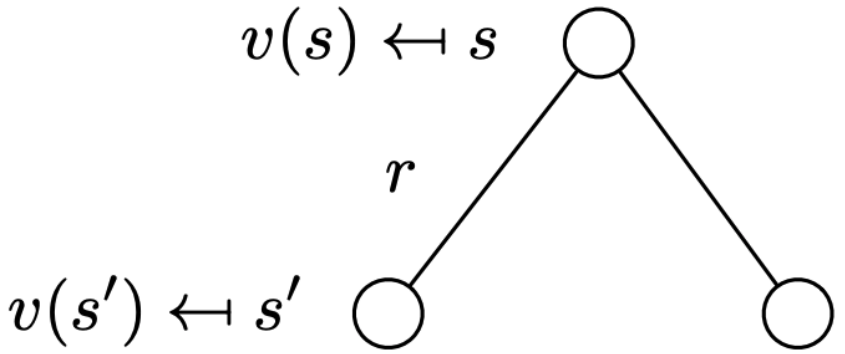
$$\begin{aligned} v(s) = \mathbb{E}[G_t|S_t = s] &= \mathbb{E}[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1})|S_t = s] \end{aligned}$$

ii. Markov Reward Processes: Bellman Equation

The definition of the value-function from the Bellman Equation can be seen a 1-step look ahead search

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

Backup diagrams: visual representations of different algorithms and models in RL

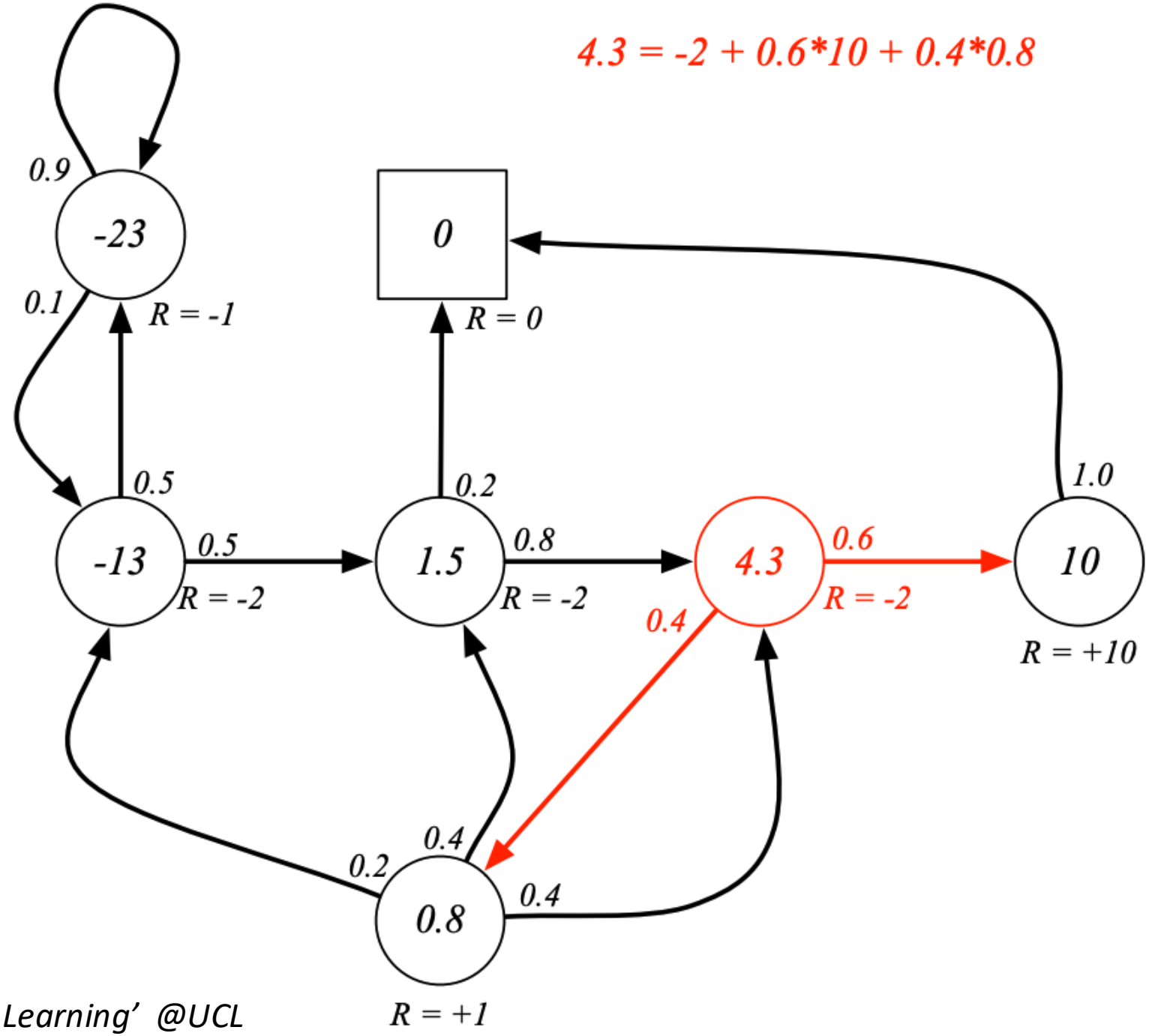


We need to consider all possible successor states with the related transition probabilities

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

ii. Markov Reward Processes : Student Markov Chain

- Undiscounted case
- If state value functions are provided we can use the Bellman Equation to verify if they are true



D. Silver 'Reinforcement Learning' @UCL

ii. Markov Reward Processes: Bellman Equation in Matrix Form

The Bellman Equation can be written concisely in matrix form

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

Where v is a column vector with one entry per state

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{11} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

ii. Markov Reward Processes: Solving the Bellman Equation

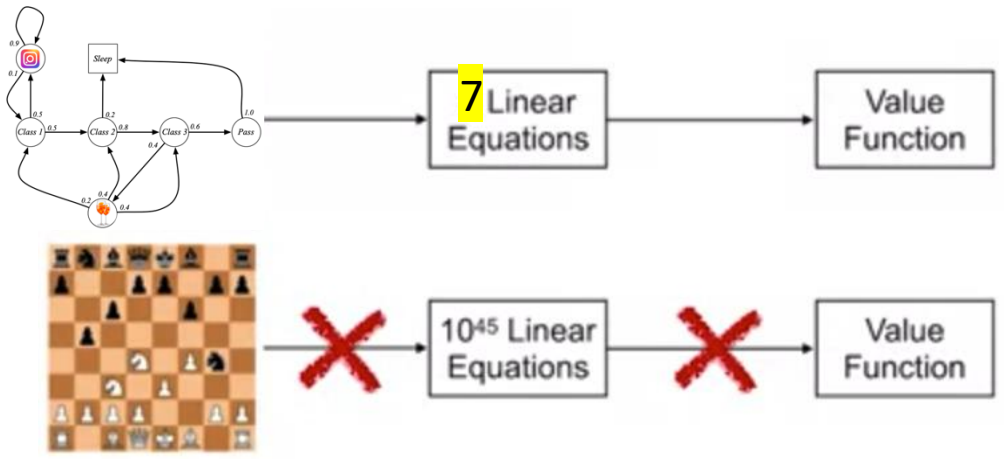
- The Bellman Equation (when we are dealing with Markov Reward Processes*) is linear
- We can solve the previous matrix directly
- If we have n states, the computational cost is $O[n^3]$: affordable only with small Markov Reward Processes (MRPs)
- For large MRPs we will look tor efficient methods (Dynamic Programming, Monte-Carlo evaluation, Temporal-Difference learning)

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

$$(I - \gamma \mathcal{P})\, v = \mathcal{R}$$

$$v = (I - \gamma \mathcal{P})^{-1} \mathcal{R}$$

**This will not be true when we will deal with optimization and maximization (when we will consider an agent making decisions!) in Markov Decision Processes*



A. White, M. White 'Fundamentals of Reinforcement Learning'

ii. Markov Reward Processes: Solving the Bellman Equation

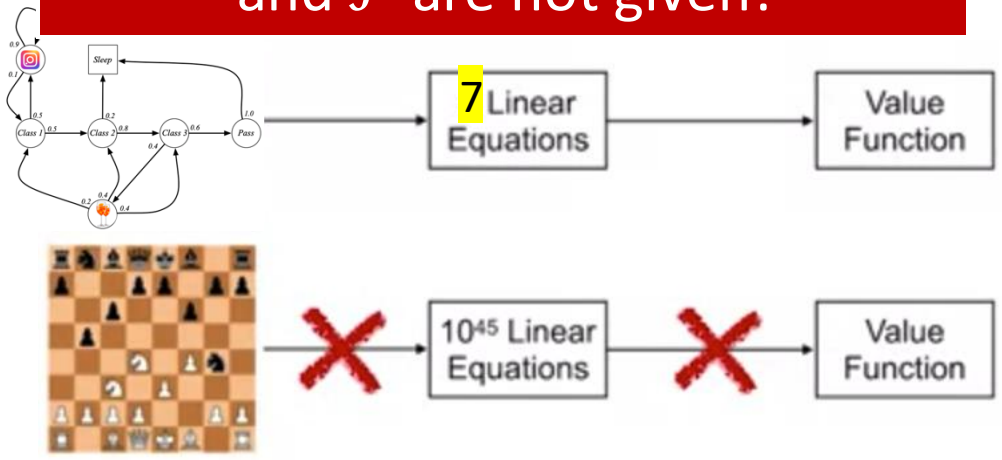
- The Bellman Equation (when we are dealing with Markov Reward Processes*) is linear
- We can solve the previous matrix directly
- If we have n states, the computational cost is $O[n^3]$: affordable only with small Markov Reward Processes (MRPs)
- For large MRPs we will look tor efficient methods (Dynamic Programming, Monte-Carlo evaluation, Temporal-Difference learning)

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

$$(I - \gamma \mathcal{P})\, v = \mathcal{R}$$

$$v = (I - \gamma \mathcal{P})^{-1} \mathcal{R}$$

Moreover, in true RL problems $\mathcal{R}$ and $\mathcal{P}$ are not given!



This will not be true when we will deal with optimization and maximization (when we will consider an agent making decisions!) in Markov Decision Processes

A. White, M. White 'Fundamentals of Reinforcement Learning'

iii. Markov Decision Processes (MDPs)

Let's add actions and decisions (a true RL problem!): a Markov Decision Process is a Reward Process with decisions. MPD is an environment in which all states are Markov.

Definition

A Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ such that:

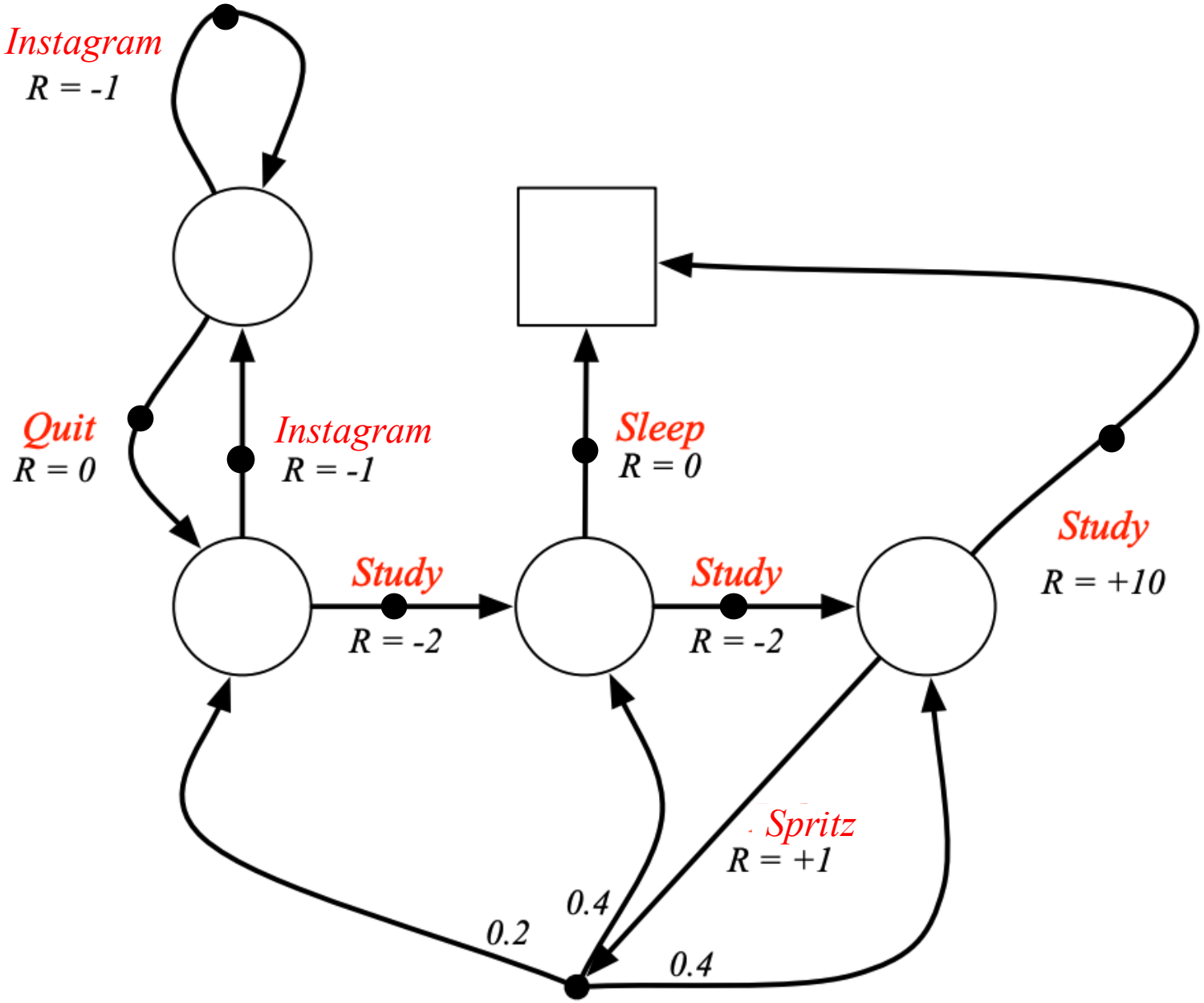
- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix with entries

$$\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

- $\mathcal{R}$ is a reward function, $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$ (it is just the immediate reward, in that specific state)
- γ is a discount factor, $\gamma \in [0,1]$

iil. Markov Decision Processes: Student Markov Chain

- Pay attention: we are reporting actions (typically indicated with a black dot)
- States are not reported here
- Now there is control and agency! Now we can try to maximize our reward!



D. Silver 'Reinforcement Learning' @UCL

iii. MDPs: (Stochastic) Policies

Definition

A Policy π is a distribution over actions given that we are in a state:

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$$

- A policy fully defines the behaviour of an agent
- MDP policies depend on the current state (we are considering Markov states in MDPs, history doesn't matter)
- In MDP policies are stationary (do not depend on t), however we can change our policy in future episodes
- We consider stochastic policies: this allow us for example to deal with exploration!
- Please not that there is no reward here: the policy can be given or we may have 'learned' the policy with a dedicated procedure

iii. MDPs: (Stochastic) Policies

Given an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ + a policy π:

- The state sequence $S_1, S_2, \ldots$ is a Markov Process $\langle \mathcal{S}, \mathcal{P}^\pi \rangle$ where

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

- The state and reward sequence $S_1, R_2, S_2, \ldots$ is a Markov Reward Process $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$ where

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

iii. MDPs: Value Function

We had value functions with Markov Reward Processes, but now that we have agency, value of a state depends on the policy!

Definitions

The state-value function $v_\pi(s)$ of an MDP is the expected return from state s if we follow policy policy π :

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

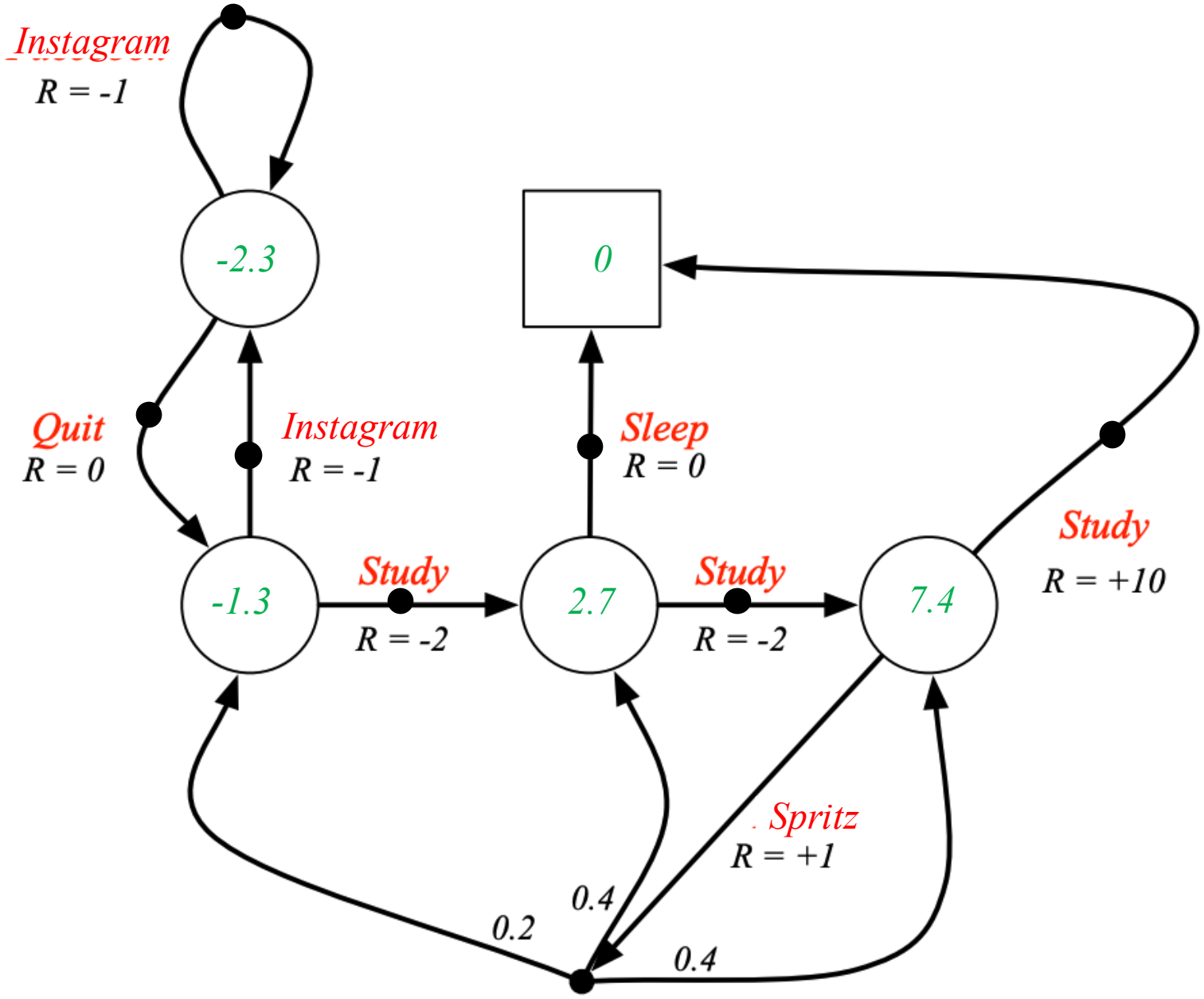
The action-value function $q_\pi(s,a)$ of an MDP is the expected return from state s if we take action a and then we follow policy policy π :

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

iii. Markov Decision Processes: Student Markov Chain

We consider the undiscounted MDP ($\gamma = 1$) and a uniform random policy: for each state (C1, IG, C2, C3) there are two possible actions, each one with probability 0.5

$\pi(a|s) = 0.5$ for all a, s



D. Silver 'Reinforcement Learning' @UCL

iii. MDPs: Bellman Expectation Equation

The value function $v_\pi(s)$ can again be decomposed into 2 parts:

- The immediate reward R_{t+1}
- The discounted value of successor state $\gamma v(S_{t+1})$

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

Similarly for the action-value function

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

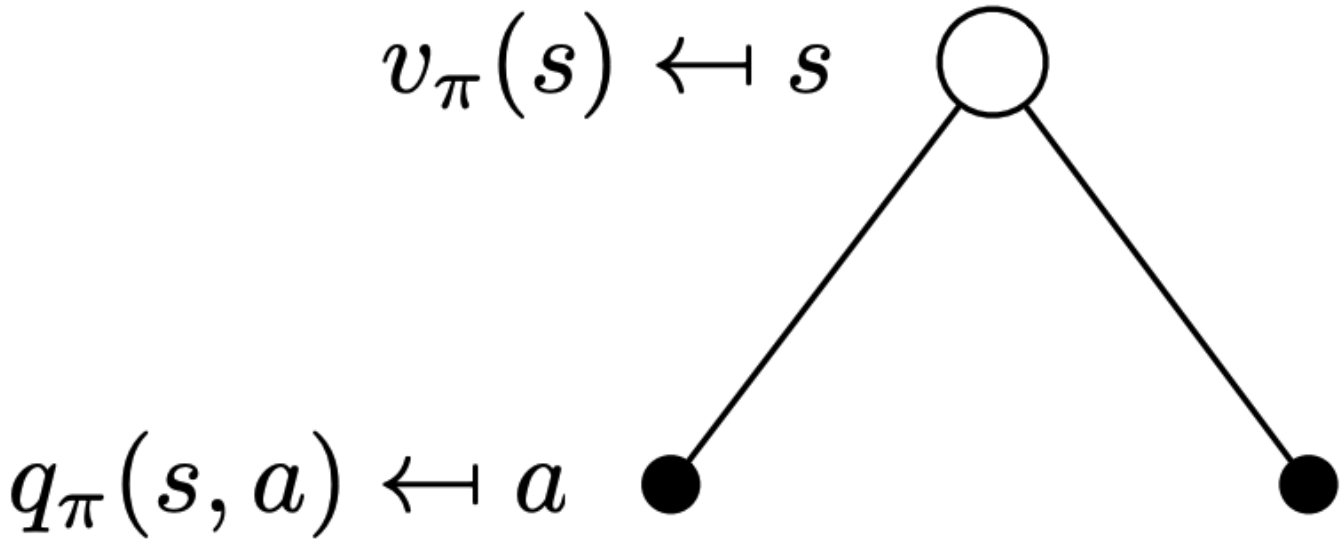
* Please note that previously we have considered $v(s)$

iii. MDPs: Bellman Expectation Equation

Let's see the relation between q_π and v_π (1 of 4)

states:

actions:



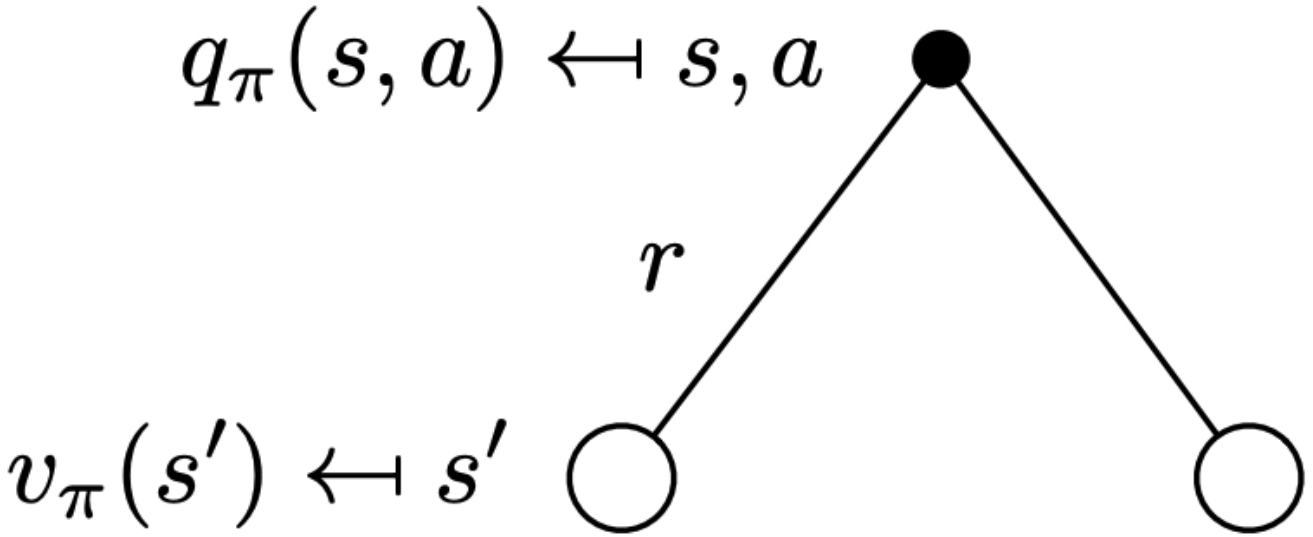
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

iii. MDPs: Bellman Expectation Equation

Let's see the relation between q_π and v_π (2 of 4)

states: ○

actions: ●



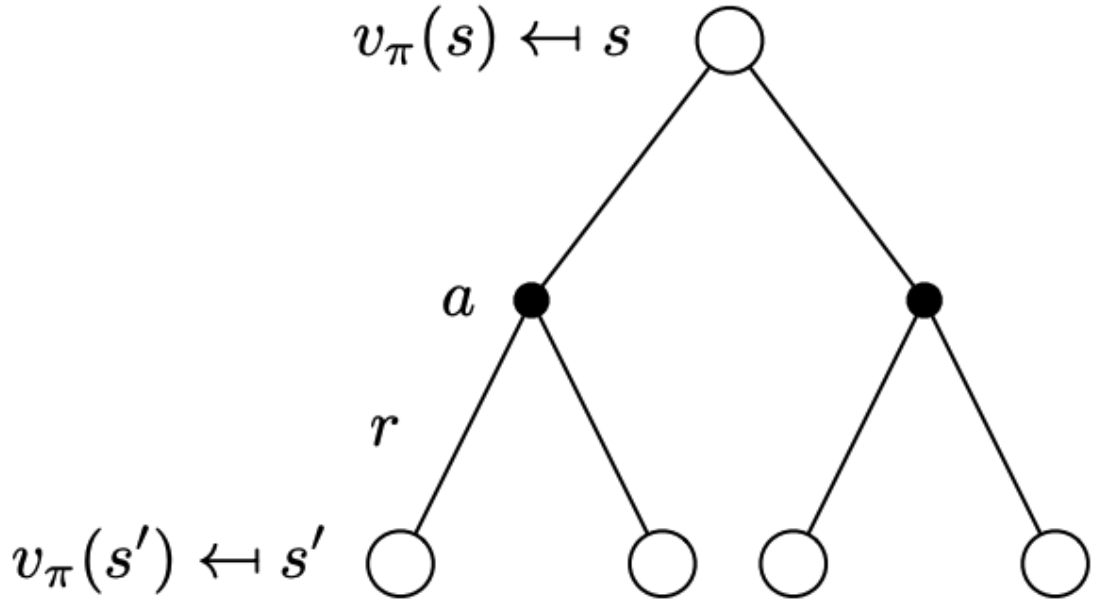
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

iii. MDPs: Bellman Expectation Equation

Let's see the relation betweer q_π and v_π (3 of 4)

states: ○

actions: ●



We obtain a recursive description of v_π

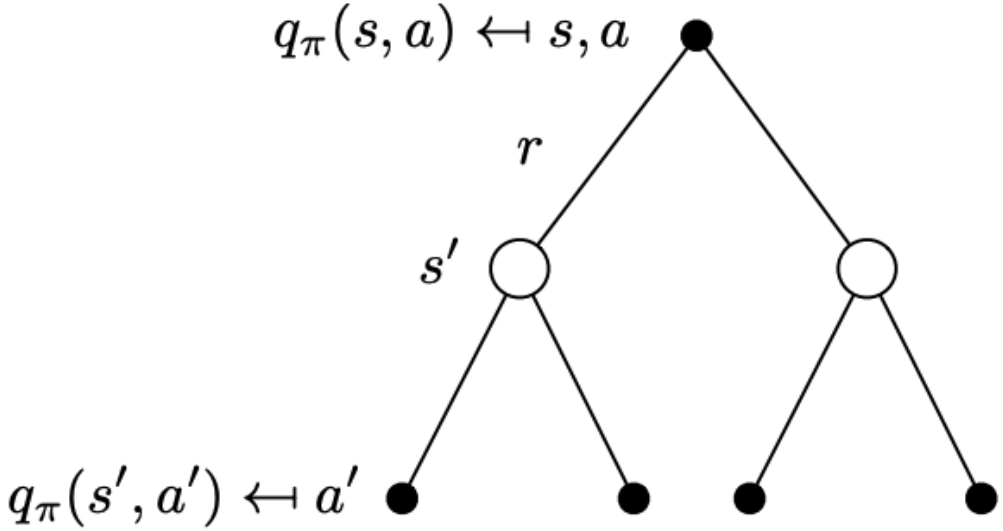
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

iii. MDPs: Bellman Expectation Equation

Let's see the relation between q_π and v_π (4 of 4)

states: ○

actions: ●



We obtain a recursive description of q_π

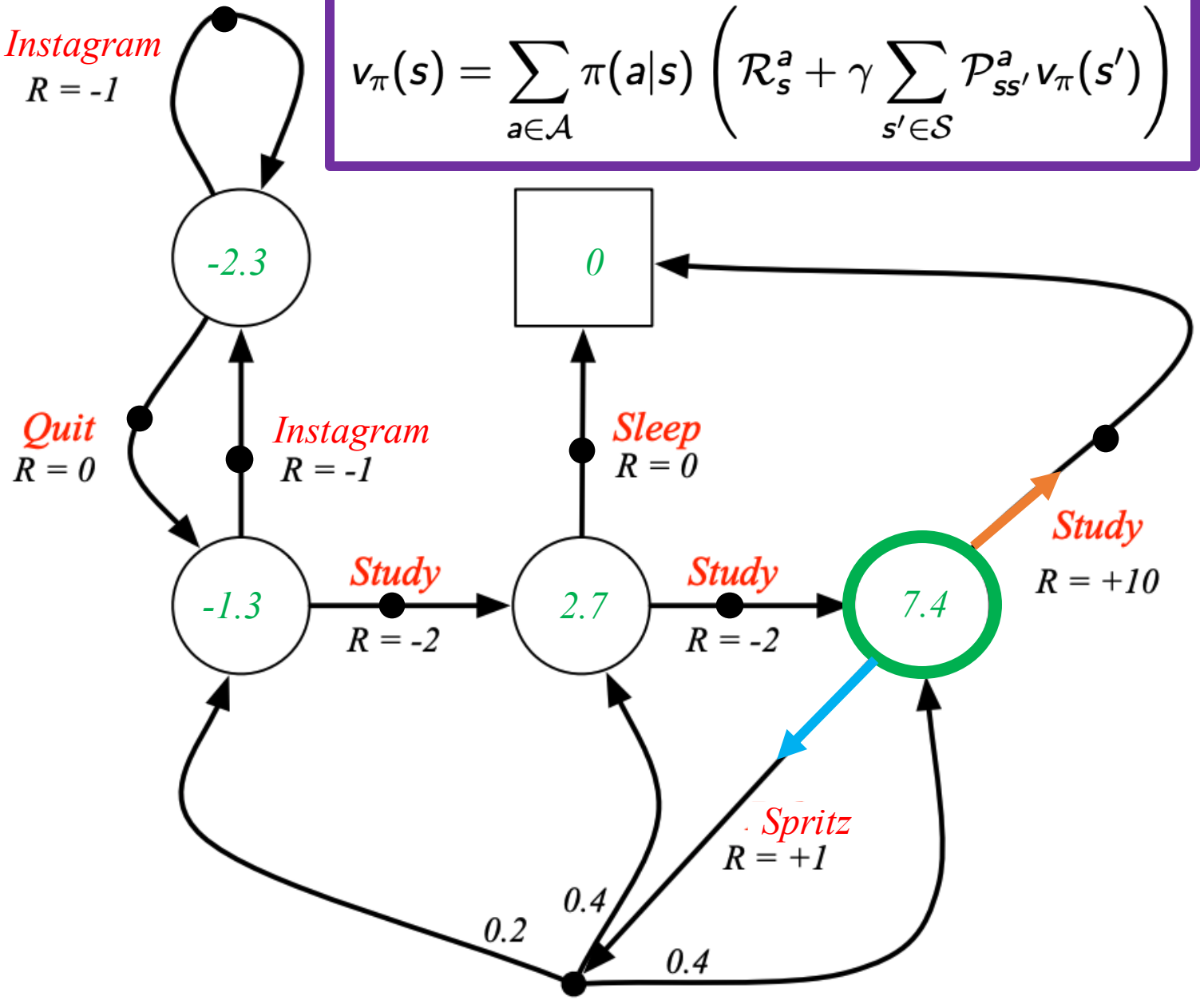
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

iiI. Markov Decision Processes: Student Markov Chain

Let's consider the previous case (undiscounted, uniform random policy) and let's verify with the recursive definition that $v_\pi(\mathrm{C3}) = 7.4$

7.4 = 0.5*10+0.5*(1+0.4*7.4-0.2*1.3+0.4*2.7)

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



D. Silver 'Reinforcement Learning' @UCL

iii. Markov Decision Processes: Bellman Expectation Equation in Matrix Form

Also the Bellman Expectation Equation can be written concisely in matrix form (we are resorting to the induced Markov Reward Process by using $\mathcal{R}_s^\pi = \sum_{a \in \mathcal{A}} \pi(a|s)\, \mathcal{R}_s^a$ seen before):

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

That can be solved:

$$v_\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

As said, a good part of the course will be dedicated to find efficient ways to avoid computing such set of linear equations.

What is missing?

We now have a formalization of the RL problem and (simple) ways to evaluate important quantities like v_π for a given policy.

But how is the policy chosen?

More importantly, how we choose an optimal policy? Ie. a policy that allows us to maximize cumulative rewards.

Credits

- Image of the course is taken from C. Mahoney ‘Reinforcement Learning’ https://towardsdatascience.com/reinforcement-learning-fda8ff535bb6

- Overall structure of the lecture and some content was inspired/adapted from D. Silver RL course a @ UCL

- Examples and applications were inspired by D. Mwiti https://neptune.ai/blog/reinforcement-learning-applications

AMCO
ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND CONTROL RESEARCH GROUP

Reinforcement Learning 2025/2026

RL2
Reinforcement Learning Research Lab @ UniPD

Thank you! Questions?

Lecture #03
Multi-armed Bandits + Markov Decision Processes

Gian Antonio Susto