Reinforcement Learning in Economics and Finance

Mirko Polato, PhD - 20/04/2021

Agenda

Part I - Introduction to Reinforcement Learning

Part II - Theoretical foundation of RL

Part III - Reinforcement Learning at scale

Part IV - RL applications in finance

Final remarks

Part I

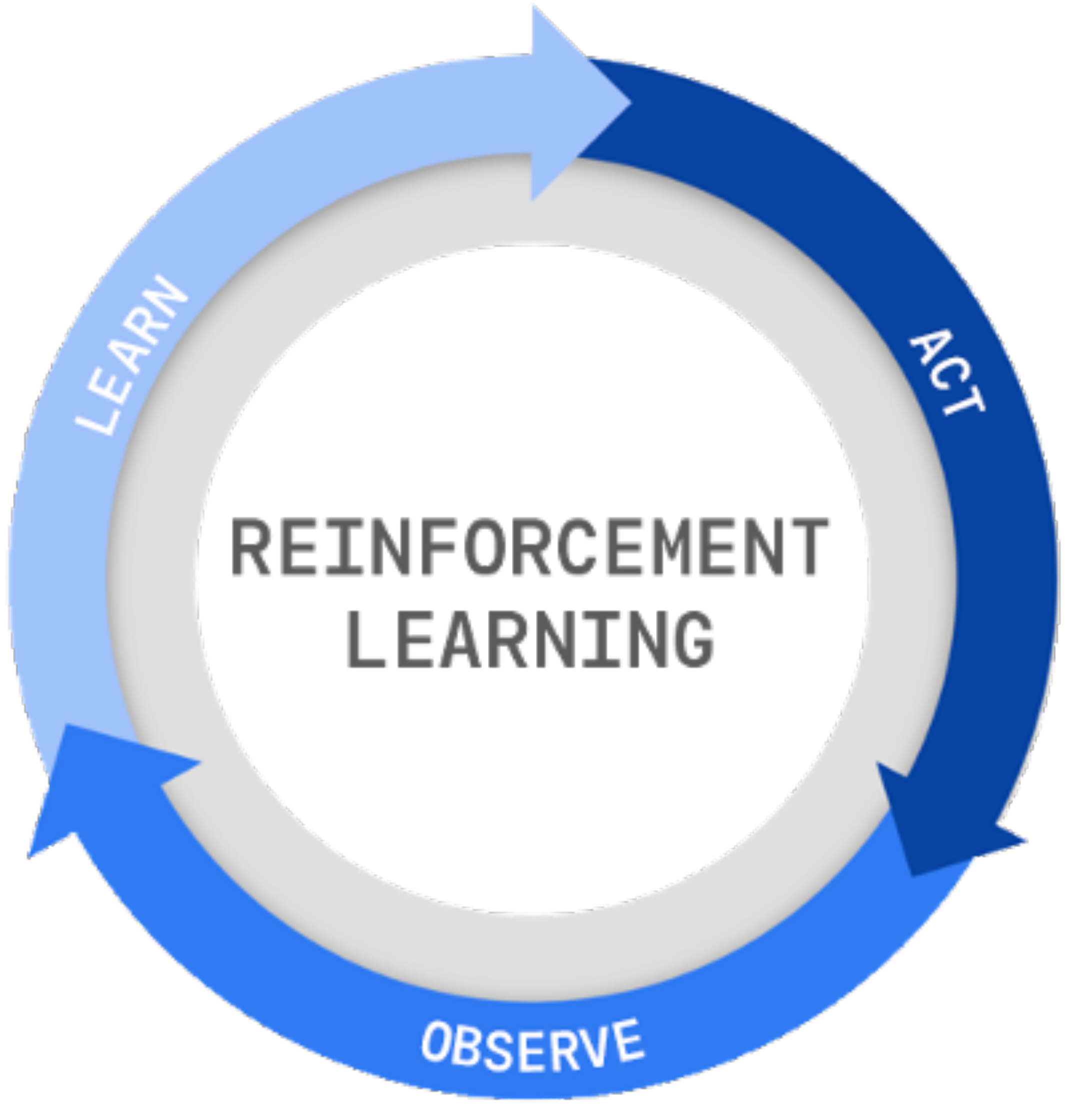
Reinforcement Learning - A gentle introduction

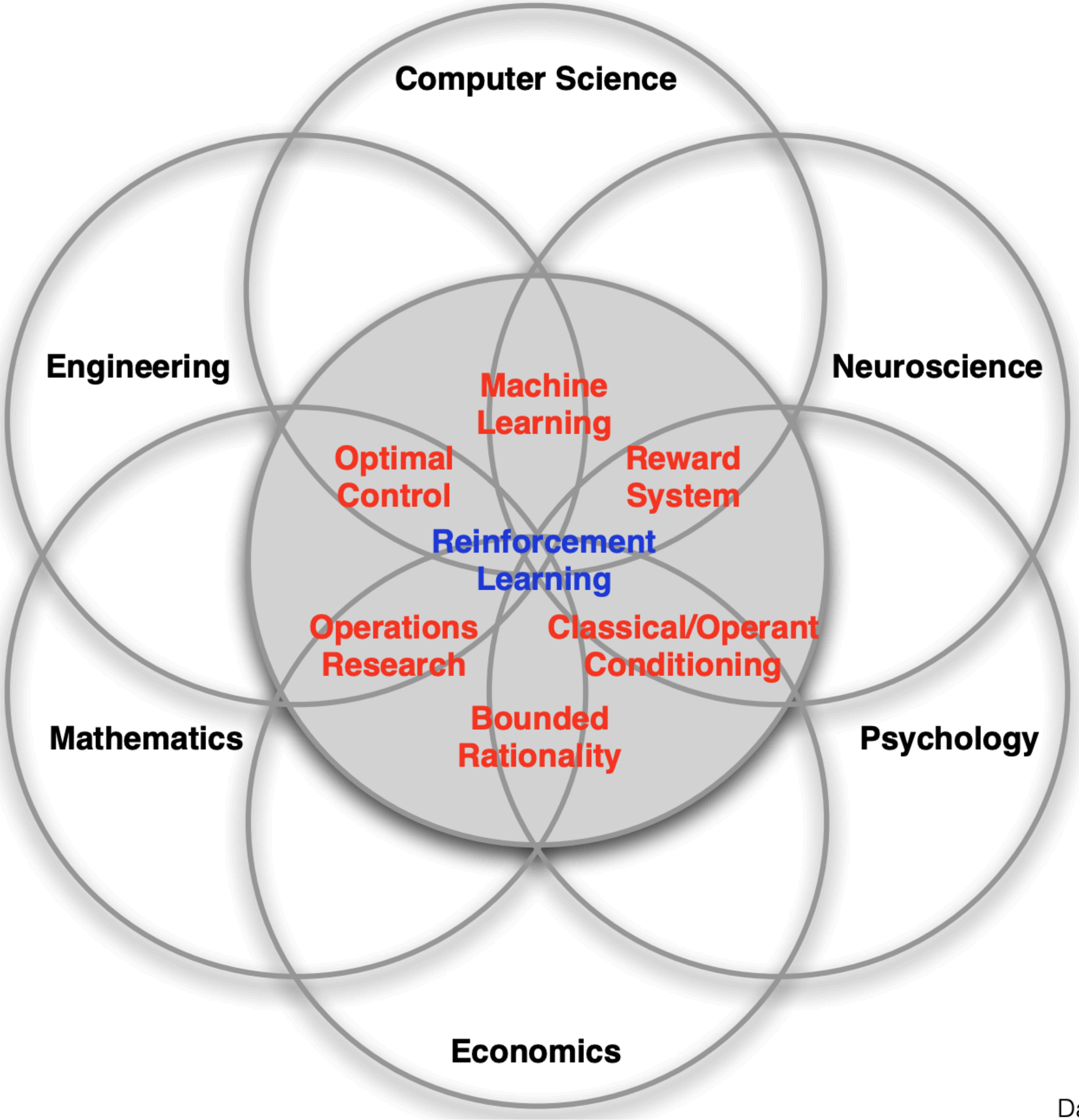
Reinforcement Learning

Reinforcement learning is learning what to do—how to map situations to actions—so as to maximise a numerical reward signal.

Sutton & Barto. Reinforcement learning: An introduction

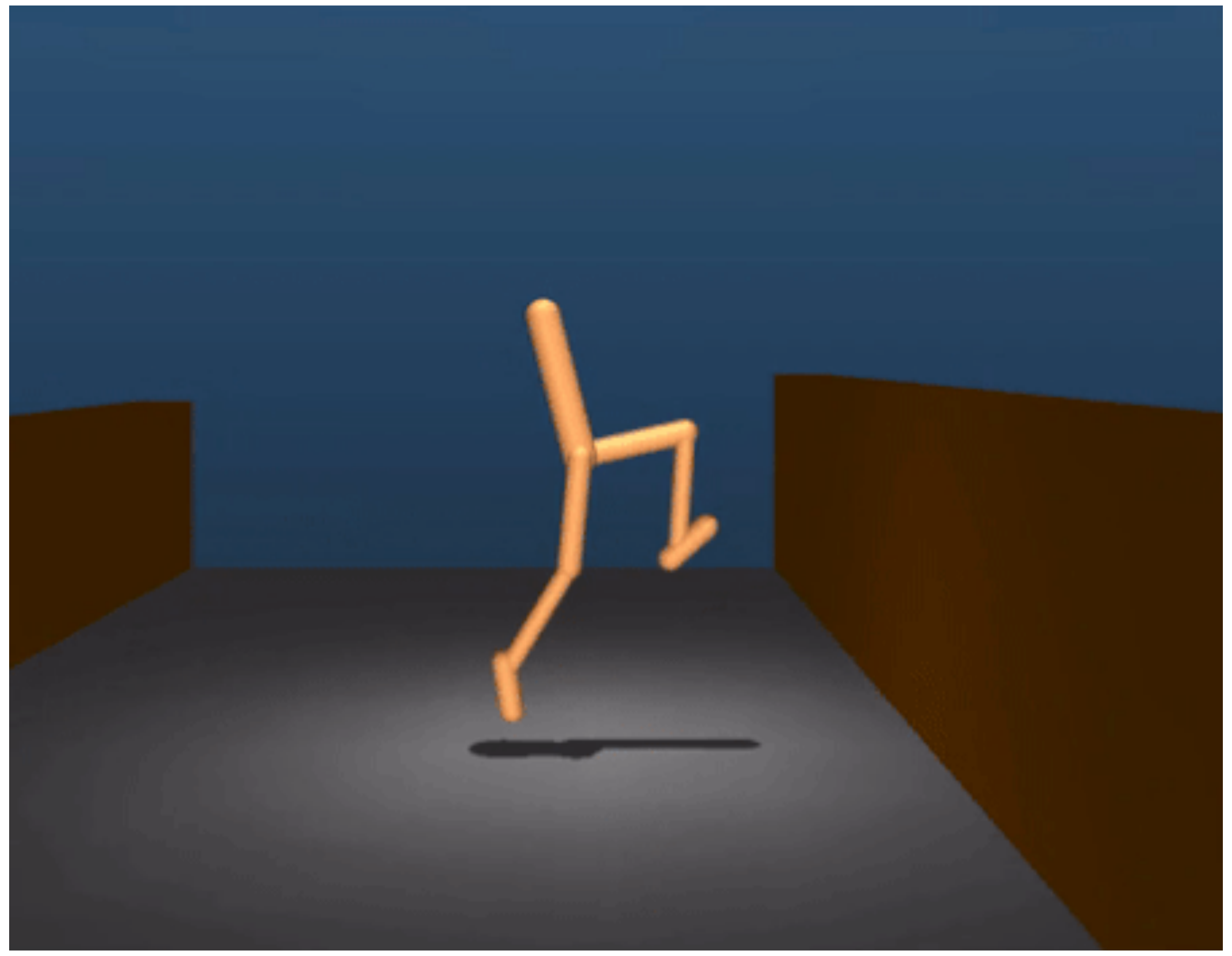
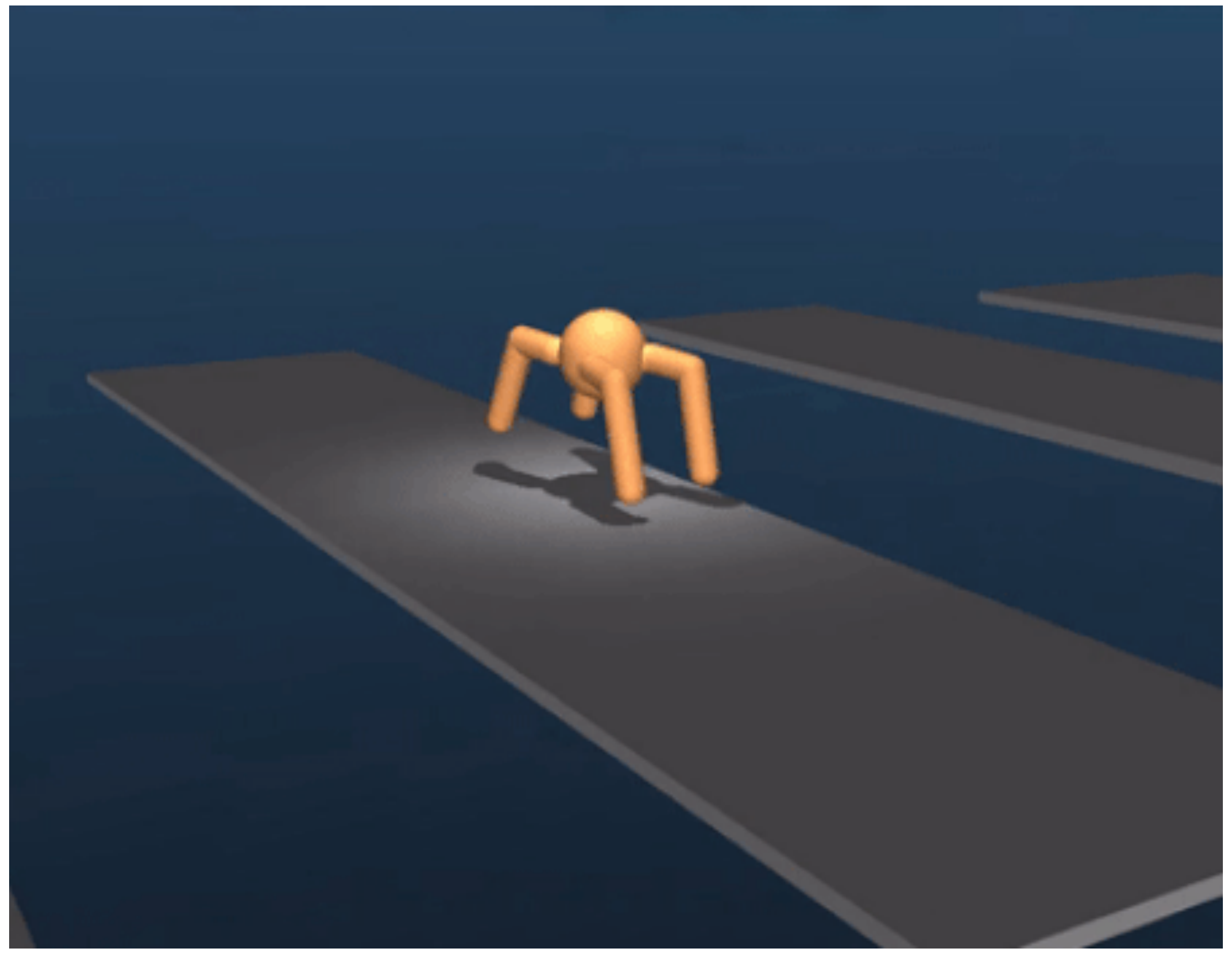
- **Agent-oriented learning**—learning by interacting with an environment to achieve a goal
- Learning by **trial and error**, with only delayed evaluative feedback (**reward**)
- **Sequential decision making**: non i.i.d data
- Agent's actions affect the subsequent data it receives (i.e., by acting it may change the environment)





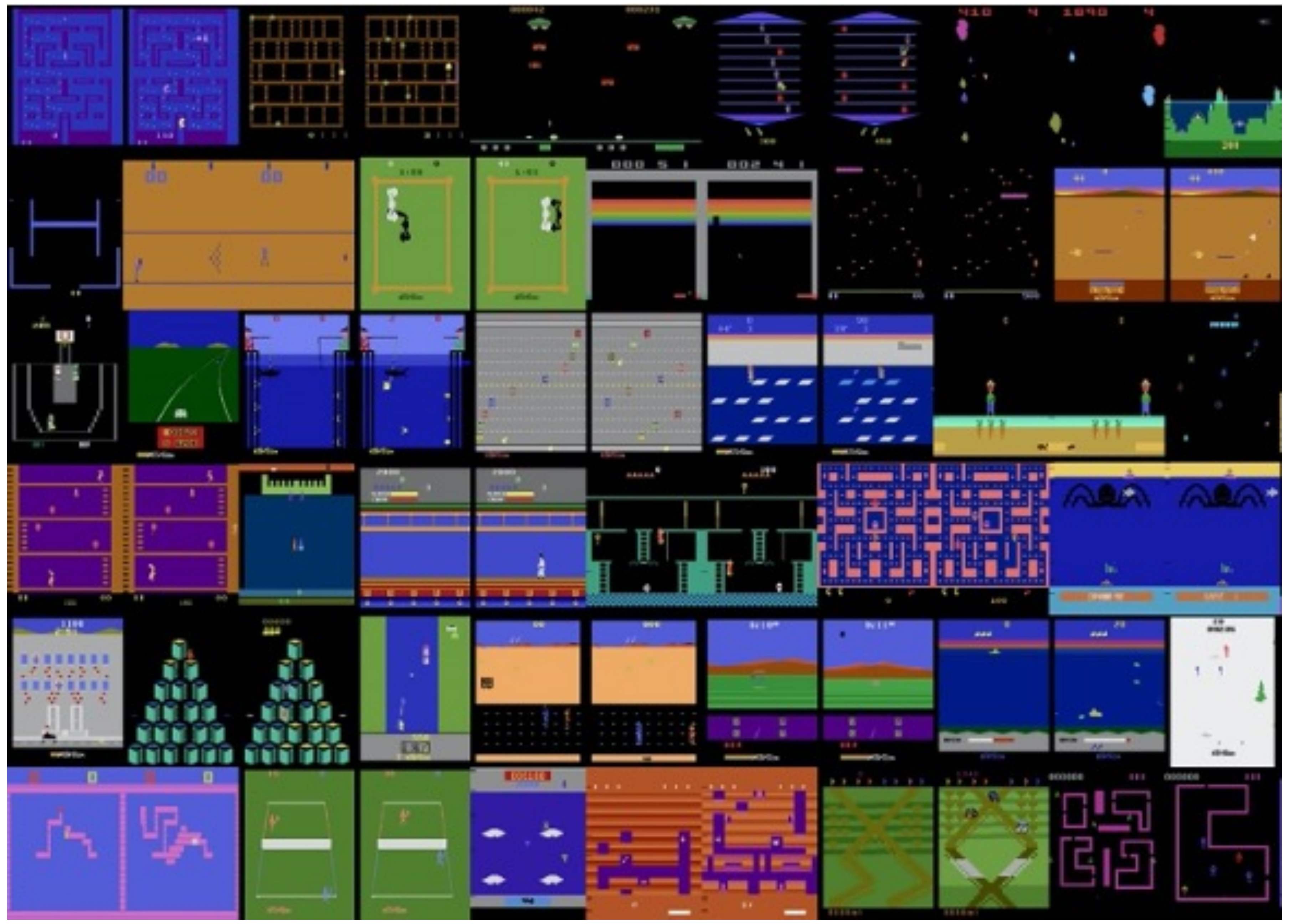
David Silver 2015

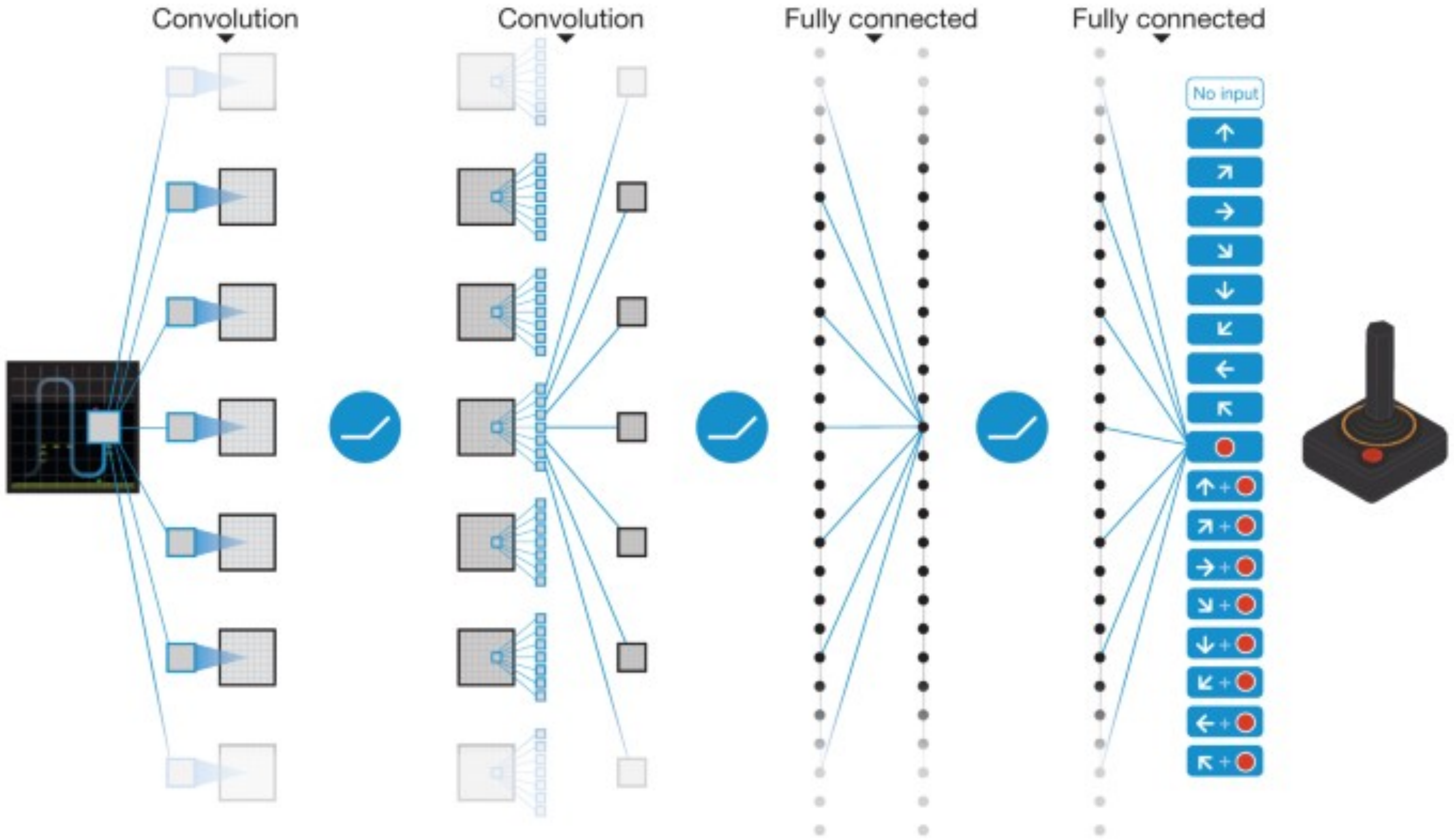
RL success: Learning to walk


Haarnoja, T., Zhou, A., Ha, S., Tan, J., Tucker, G., & Levine, S. (2019). Learning to Walk via Deep Reinforcement Learning. ArXiv, abs/1812.11103.

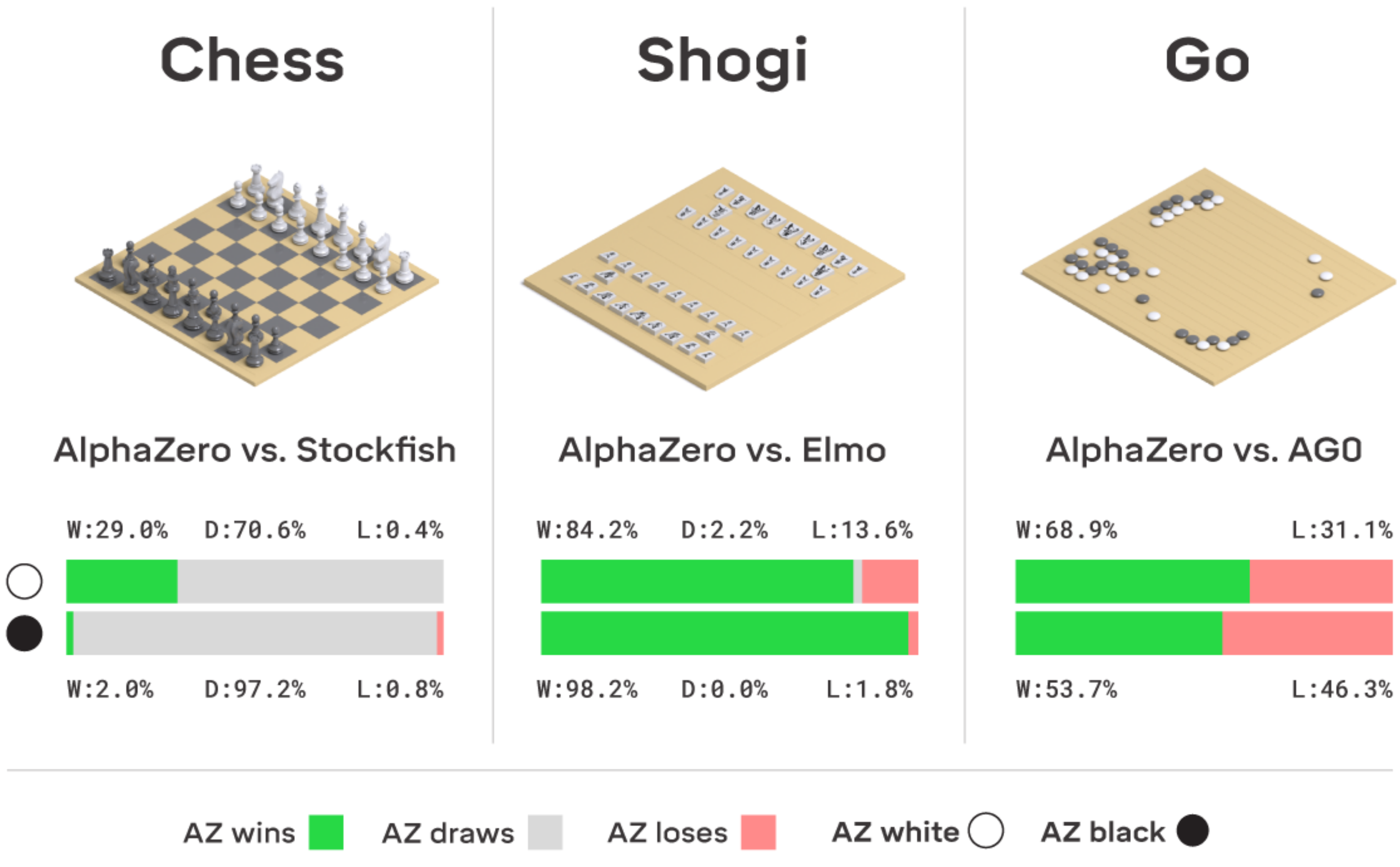
RL success: Playing ATARI games





Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S. & Hassabis, D. (2015), 'Human-level control through deep reinforcement learning', Nature 518 (7540), 529--533.

RL success: Mastering Go, Chess, …

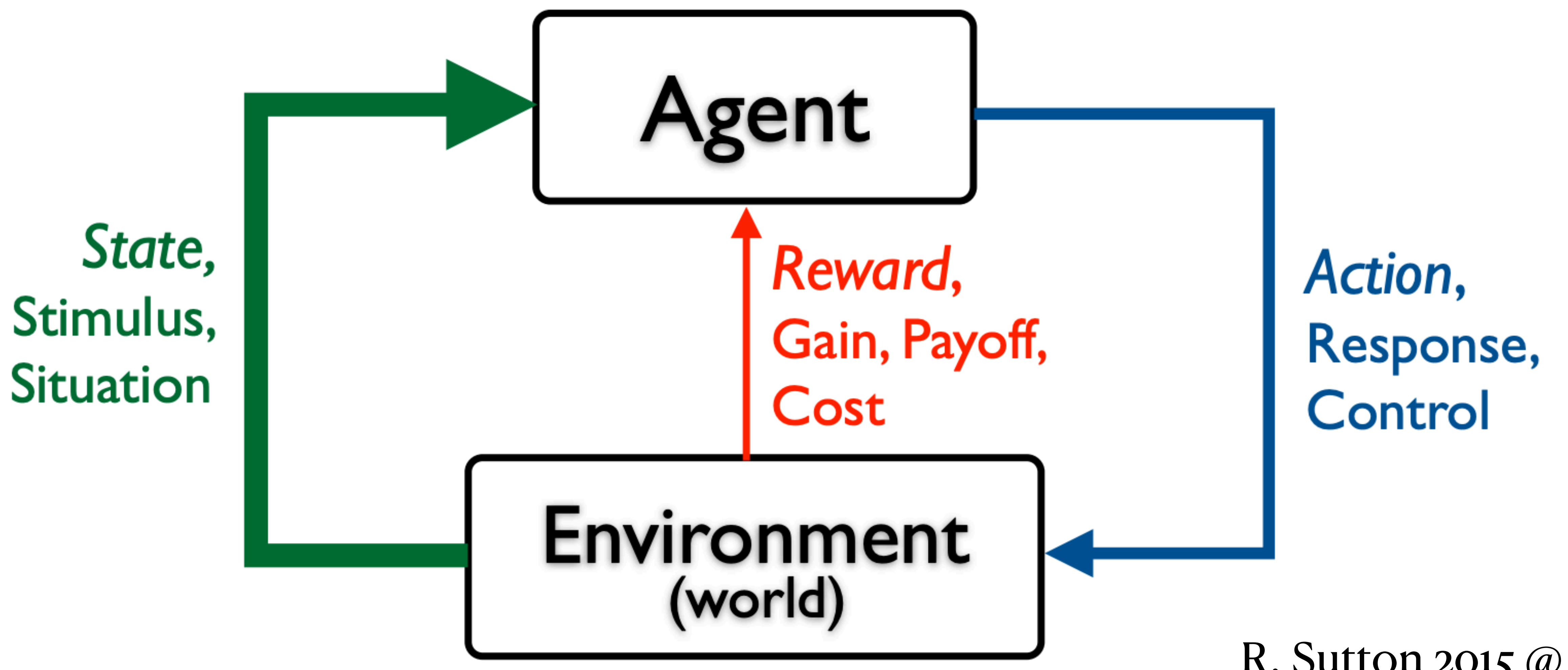


Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. ArXiv, abs/1712.01815.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T. & Hassabis, D. (2016), 'Mastering the Game of Go with Deep Neural Networks and Tree Search', Nature 529 (7587), 484--489.

The RL interface

a.k.a. the agent-environment interface



R. Sutton 2015 @ NIPS

- Environment may be unknown, nonlinear, stochastic and complex
- **Trajectory**/History: sequence of **Observation, Reward, Action**...
- States are **Markovian**, i.e., the state is a sufficient statistic of the future

[Finite] Markov Decision Process

The foundational RL framework

A **Markov Decision Process** is a tuple $\langle S, A, R, P, \gamma \rangle$

- S - finite set of **states**
- A - finite set of **actions**
- R - finite set of **rewards** (or a reward function)
- P - **transition probability matrix** that describes a Markov dynamics

The future is independent of the past given the present

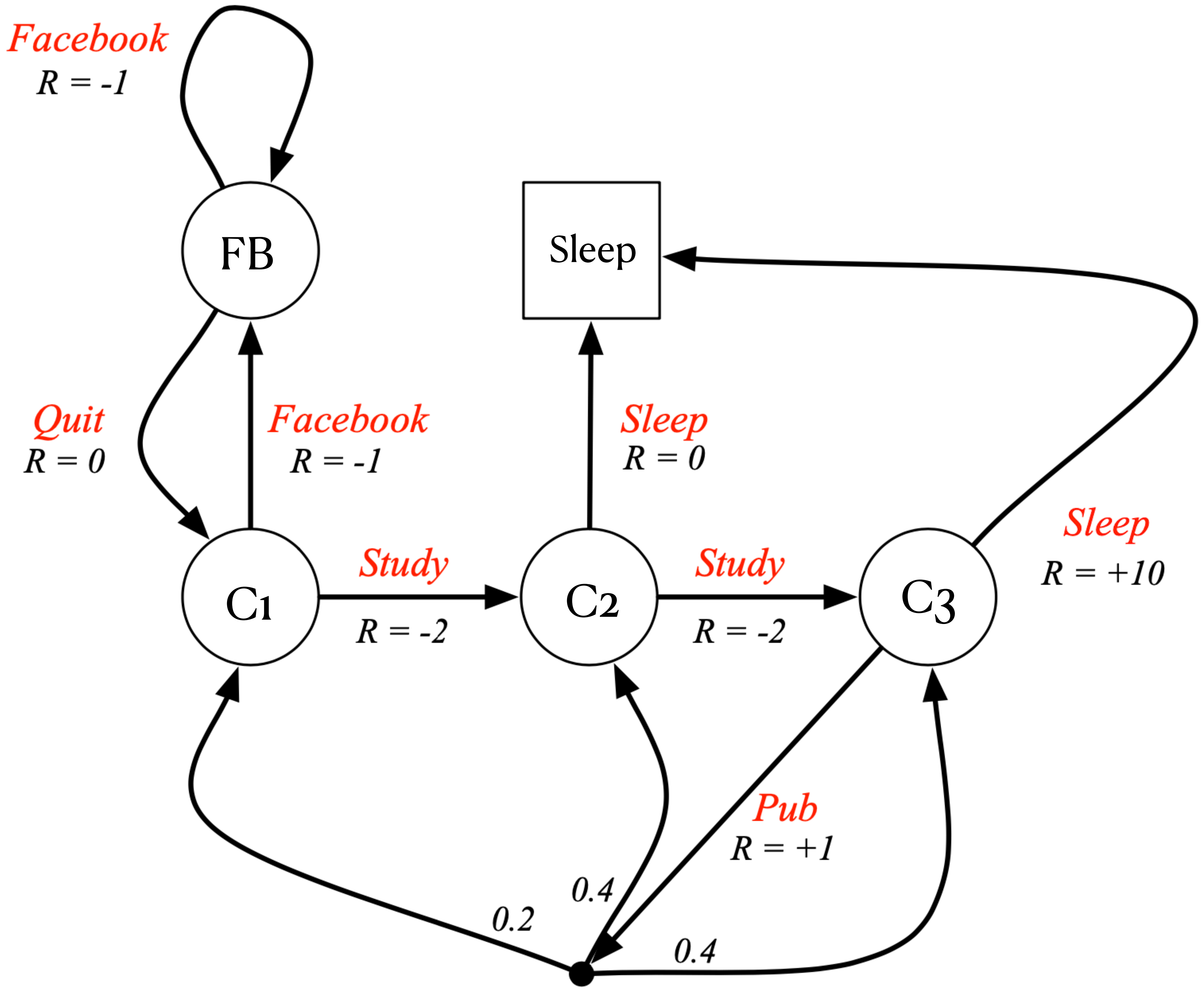
$$p(s \mid s', a) = \mathbb{P}(S_{t+1} = s' \mid S_t = s, A_t = a)$$

- $\gamma \in [0,1]$ - **discount rate**

Example of MDP: Student MDP

- **Goal**: Prepare for the exam and go to sleep
- **Episodic MDP**: each trajectory ends on the *Sleep* state (loop forever with reward zero)
- Example of episode :

  C1→FB→C1→C2→C3→C2→Sleep



D. Silver 2015

Return (G)

The agent aims to maximize it

- **Reward hypothesis**: All goals can be described by the maximisation of the **expected cumulative reward**
- Formally: the agent seeks to maximise the so-called **expected return**
- **Return** at time step t is defined as

$$
\begin{aligned}
G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \\
&= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \\
&= R_{t+1} + \gamma G_{t+1}
\end{aligned}
$$

Why the discount rate?

There are a bunch of reasons...

- **Mathematically convenient** to discount rewards (i.e., bounded return)
- **Avoids infinite returns** in cyclic Markov processes
- It encodes the **uncertainty** about the future
- We can adjust γ to our needs. E.g, in financial applications, immediate rewards may earn more interest than delayed rewards
- … however it is possible to use undiscounted reward (i.e. $\gamma = 1$)

Policy (π)

The agent's behaviour

- Describes the **agent's behaviour**
- It is a map from state to action
- **Deterministic policy**: given the current state s, the agent performs a *specific* action

$$\pi(s) = a$$

- **Stochastic policy**: given the current state s, the agent acts according to a probability distribution over all possible actions

$$\forall a \in A, \pi(a \mid s) = \mathbb{P}(A_t = a \mid S_t = s) \quad \text{s.t.} \sum_{a \in A} \pi(a \mid s) = 1$$

Value Function

How good/bad is the situation?

- Used to **evaluate the goodness/badness** of states according to a policy
- Formally, the value function of a state s under a policy π is **the expected return when starting in** s **and following** π **thereafter**
- **State-value function**

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

- **Action-value function**

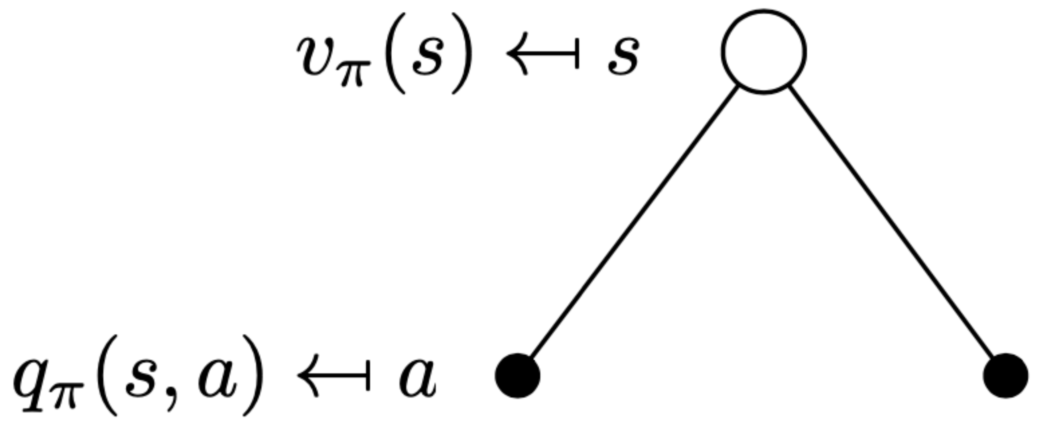
$$q_\pi(s, a) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

Bellman Expectation Equation

Breaks down the value function into two parts: the immediate reward plus the discounted future values

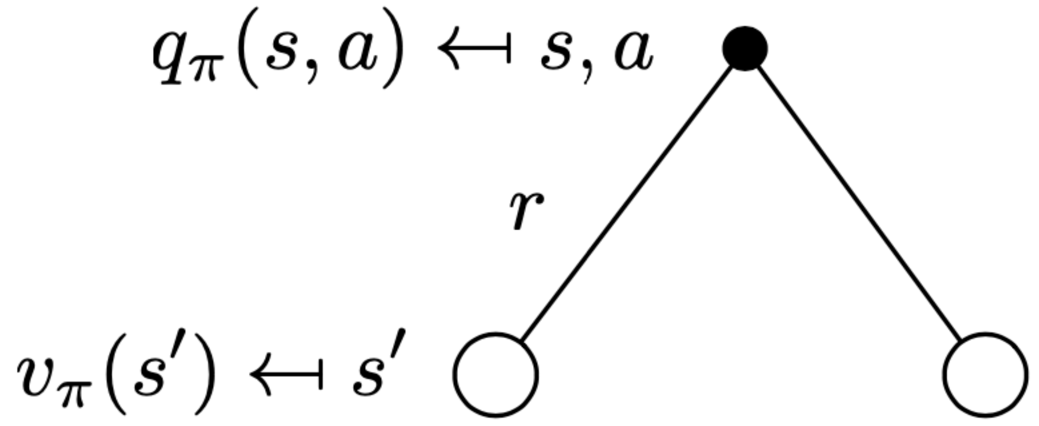
- BEE for v_π

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s \right] \\
&= \sum_{a \in A} \pi(a \mid s) q_\pi(s, a)
\end{aligned}
$$



- BEE for q_π

$$
\begin{aligned}
q_\pi(s,a) &= \mathbb{E}_\pi \left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right] \\
&= R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s, a) v_\pi(s)
\end{aligned}
$$



Optimal Value Function

What is the best we can do?

- The **optimal state-value** function $v_*(s)$ is the maximum value function over all policies

$$v_*(s) = \max_\pi v_\pi(s)$$

- The **optimal action-value** function $q_*(s, a)$ is the maximum action-value function over all policies

$$q_*(s, a) = \max_\pi q_\pi(s, a)$$

Optimal Policy

The final goal of the agent

- "**Greedyfication**": given an optimal action-value function q_* , the optimal policy can be computed by greedily selecting the action according to q_*

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

- The optimal policy is **deterministic**
- There is always at least one deterministic optimal policy

Bellman Optimality Equation

This is what we want to solve

- BOE for v_*

$$\begin{aligned}
v_*(s) &= \max_{a \in A(s)} q_{\pi_*}(s, a) \\
&= \max_a \mathbb{E}_{\pi_*} \left[G_t \mid S_t = s, A_t = a \right] \\
&= \max_a \mathbb{E}_{\pi_*} \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a \right] \\
&= \max_a \mathbb{E} \left[R_{t+1} + \gamma v_* \left(S_{t+1} \right) \mid S_t = s, A_t = a \right] \\
&= \max_a \sum_{s'} p\left(s' \mid s, a\right) \left[R(s, a) + \gamma v_*\left(s'\right) \right]
\end{aligned}$$

- BOE for q_*

$$\begin{aligned}
q_*(s, a) &= \mathbb{E} \left[R_{t+1} + \gamma \max_{a'} q_* \left(S_{t+1}, a' \right) \mid S_t = s, A_t = a \right] \\
&= \sum_{s'} p\left(s' \mid s, a\right) \left[R(s, a) + \gamma \max_{a'} q_* \left(s', a'\right) \right]
\end{aligned}$$

Part II

Tabular methods - The theoretical foundation of RL

Tabular methods

They use tables!

- When the underlying MDP is small, v_π and q_π can be stored in a table, called **V-Table** and **Q-Table**, respectively
- Popular tabular methods:
  - **Monte-Carlo**
  - **Temporal-Difference Learning**
  - **Q-Learning**
  - SARSA

How does the agent learn?

a.k.a. Fantastic policies and how to find them

- **GOAL**: solving the Bellman Optimality Equation!
- **PROBLEM**: BOE is non-linear and it has no closed form solution (in general)
- **SOLUTIONS**:
  - **Known MDP**: we know how the world works (unrealistic in practice!), thus we can solve BOE using *Dynamic Programming*
  - **Unknown MDP**: we can only experience the environment. **Trial-and-error based learning**: the agent explore the environment and incrementally improves its own policy

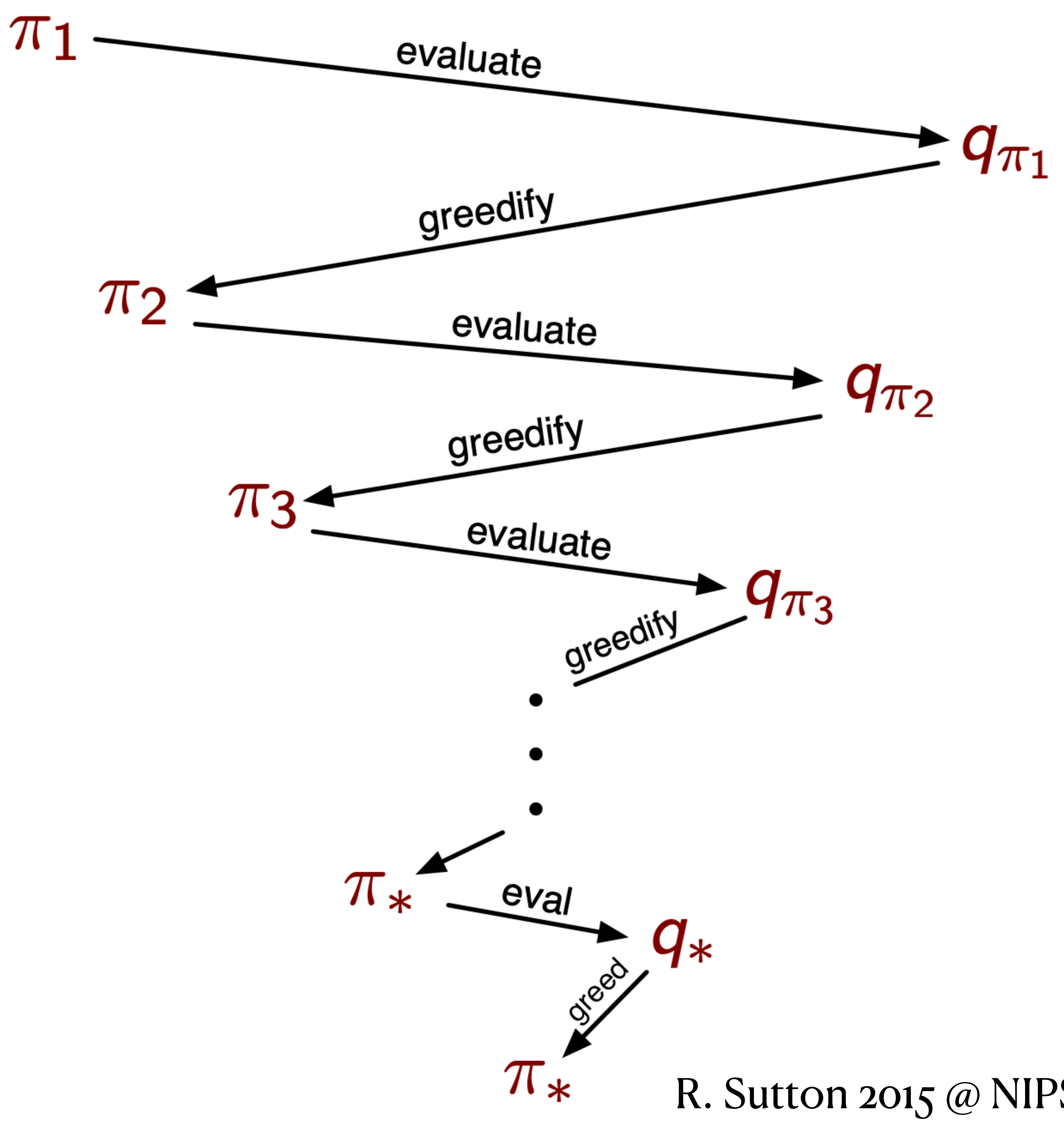
(Generalised) Policy Iteration

The dance of policy and value *(Cit. R. Sutton)*

- Two-step procedure until convergence (i.e., no improvements) starting from an arbitrary policy:

1. **Policy evaluation**: compute the value function(s) for all states according to the current policy

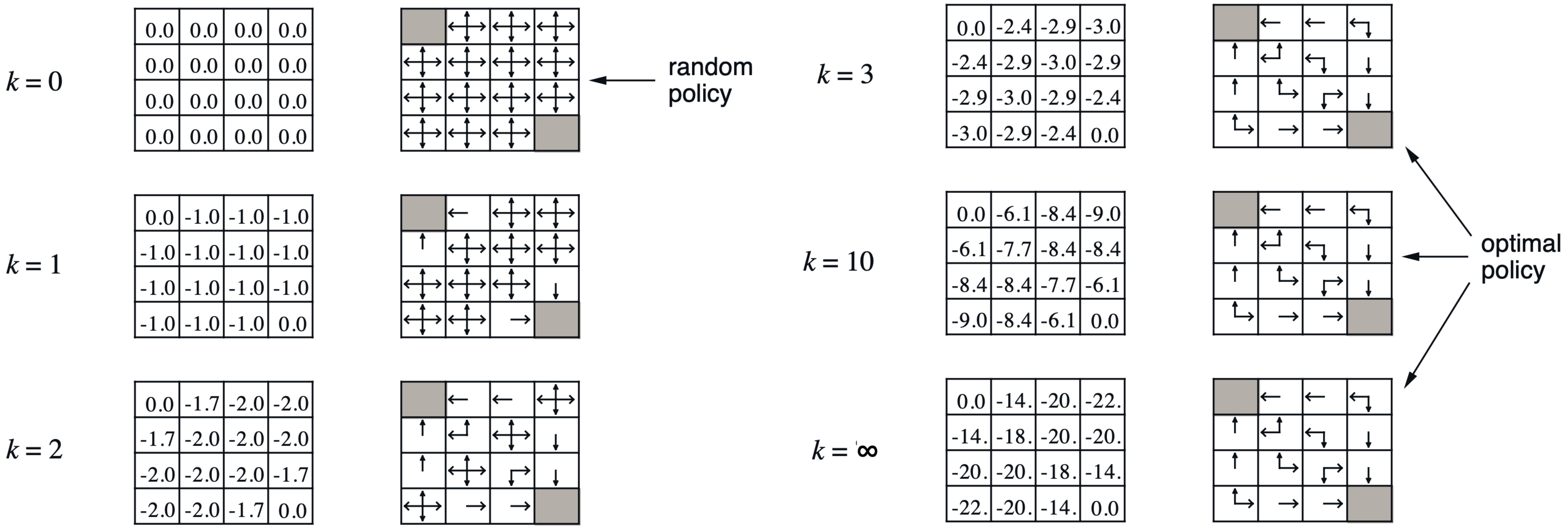
2. **Policy improvement**: greedyfication according to the current estimation of the value function



R. Sutton 2015 @ NIPS

Policy Iteration on Grid World

Reward: -1 for each step, 0 if the target state is reached



What if we do not know the MDP?

Explore & Exploit

- The agent has to **learn from experience**!
- The agent acts in the environment and adjusts its policy on the basis of the obtained rewards
- **Model-based**: the agent use its experience to create a model of the environment and then it simulates/plans over the learned model
- **Model-free**: the agent does not care about building a model of the world, it simply wants to know how to act in every situation!

Monte-Carlo control

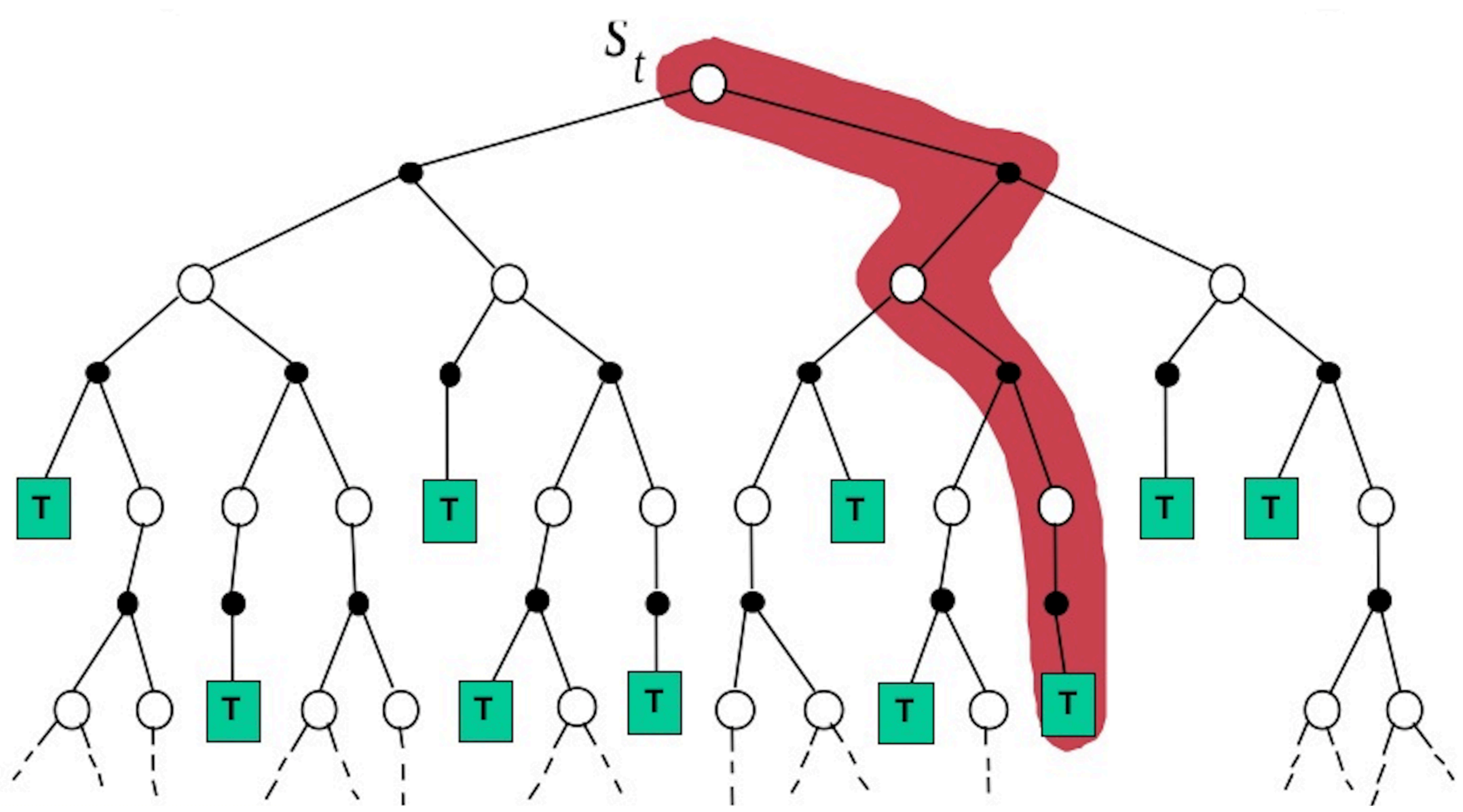
One of the easiest control methods (for episodic MDPs*)

- Given an arbitrary policy π, and a Q-Table randomly initialised. Repeat:

  ➡ Sample an episode by following the policy π

  ➡ For each transition $\langle s_t, a_t, r_t, s_{t+1} \rangle$ update the average return from s to a, i.e.,
  $q(s, a) \leftarrow$ average G from s with action a

  ➡ Update π according to q



- In the limit (# episodes $\rightarrow \infty$), and ensuring that **all actions in all states are selected infinitely often**

$$q(s, a) \rightarrow q_*(s, a)$$

Exploration vs Exploitation

We do not want to leave anything behind

- Pure **greedyfication** may lead to **poor exploration** of the environment with a consequent risk to learn a suboptimal policy
- We need to ensure that all actions are taken in all states (infinitely often in the limit)
- **ϵ-greedy exploration**

$$a_t = \begin{cases} \arg\max_{a \in A} q_\pi(a, s_t) & \text{with probability } 1 - \epsilon \\ \text{any action} & \text{with probability } \epsilon \end{cases}$$

Temporal-Difference learning

If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference (TD) learning.

Andrew Barto and Richard S. Sutton

- **Model-free** method that is based on value function updates similar to SGD
- Starting from an arbitrary value function, at every time step t (i.e., after each action with transition $\langle s, a, r, s' \rangle$) update the value-function as

$$v(s) \leftarrow v(s) + \underbrace{\alpha}_{\text{Learning rate}} \overbrace{\left[\underbrace{r + \gamma v(s')}_{\text{TD Target}} - v(s)\right]}^{\text{TD Error}}$$

- **IDEA**: improve our estimate of v (or q) using the new gathered experience

Q-Learning: Off-policy TD control

One of the early breakthrough in RL

- **Off-policy**: The agent learns the optimal policy while acting according to an "arbitrary" policy, i.e., the update rule does not depend on the used policy!
- Temporal-Difference based update $\left(s \xrightarrow{a} s'\right)$

$$q(s,a) \leftarrow q(s,a) + \alpha \left[r + \gamma \max_{a'} q\left(s', a'\right) - q(s,a) \right]$$

- Needs exploration: ϵ-**greedy policy**

Q-Learning algorithm

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

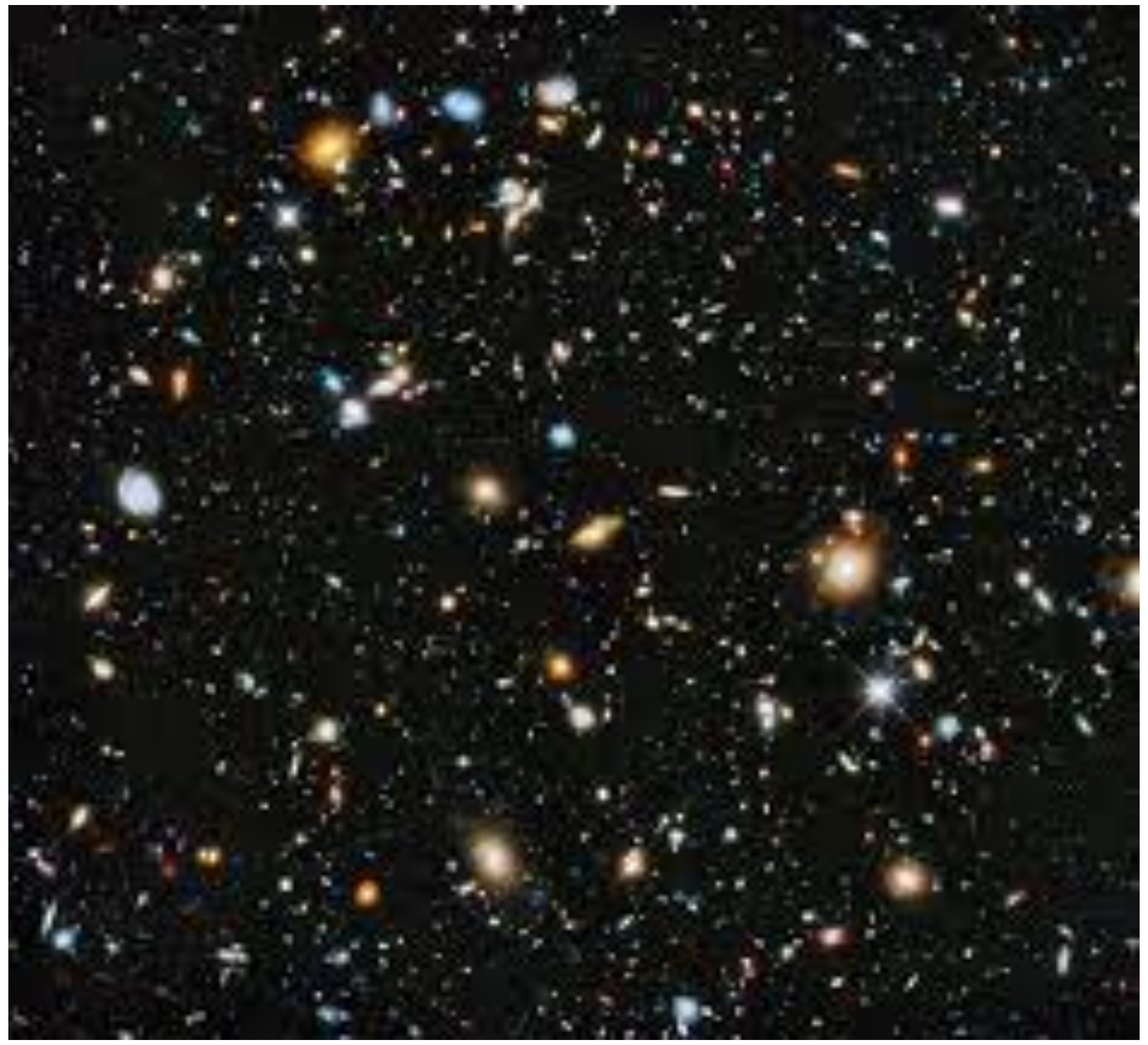
Part III

Approximated methods - RL @ scale

What's wrong with tabular solutions?

They do not scale!

- **Real-world problems are too big for being stored in tables!**
  - Backgammon: 10^{20} states
  - Chess: 10^{71} states
  - Go: 10^{170} states



- Moreover, we need more flexible ways to represent the states!

Value Function Approximation

This is how we scale up

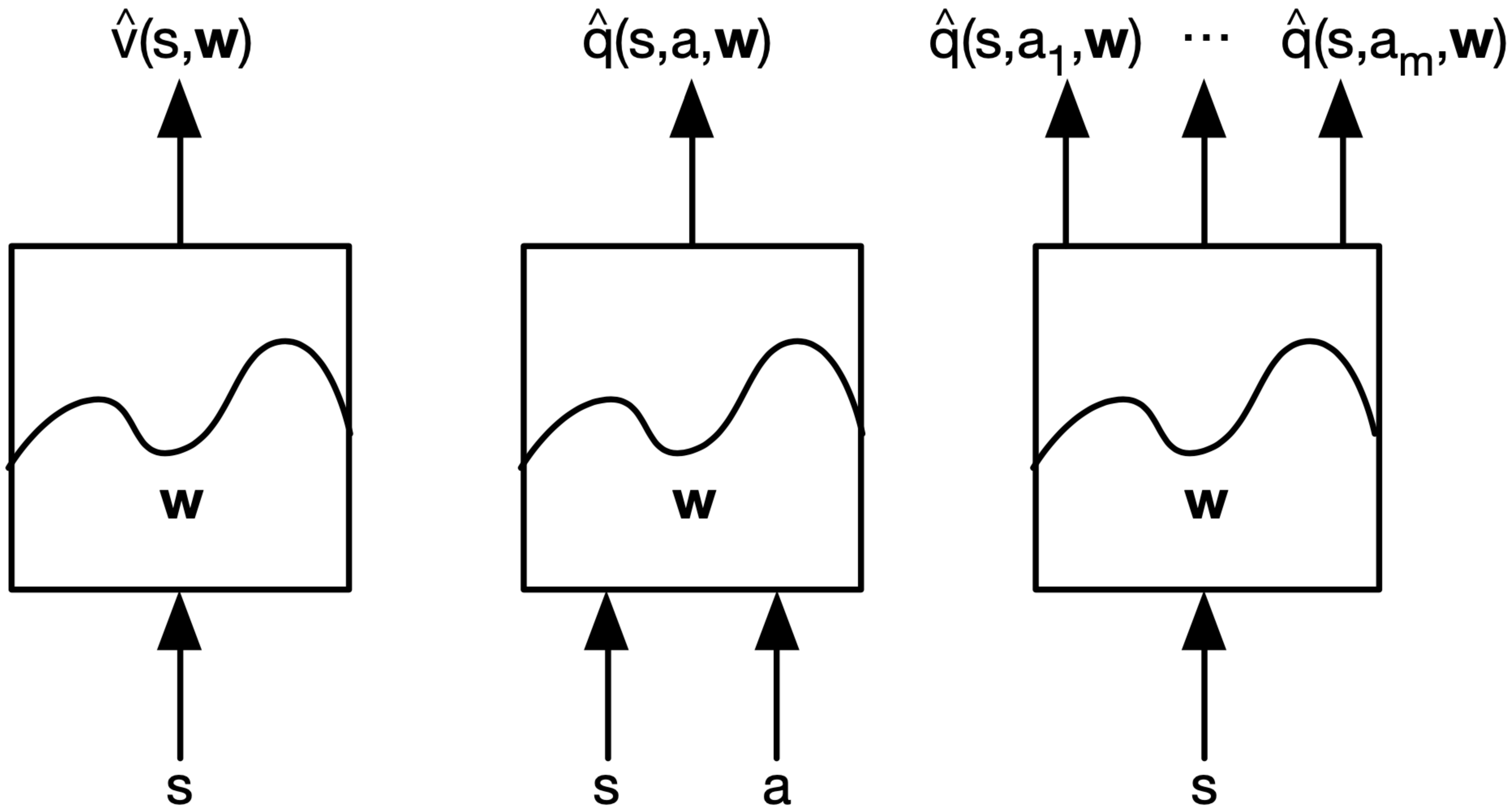
- Approximate the value function using a **function approximator**

$$\hat{v}(s, \theta) \approx v_\pi(s)$$

$$\hat{q}(s, a, \theta) \approx q_\pi(s, a)$$

- The approximator can be whatever you want
  - Linear combination of features
  - **Neural Networks**
  - Decision trees
  - …

Type of approximations



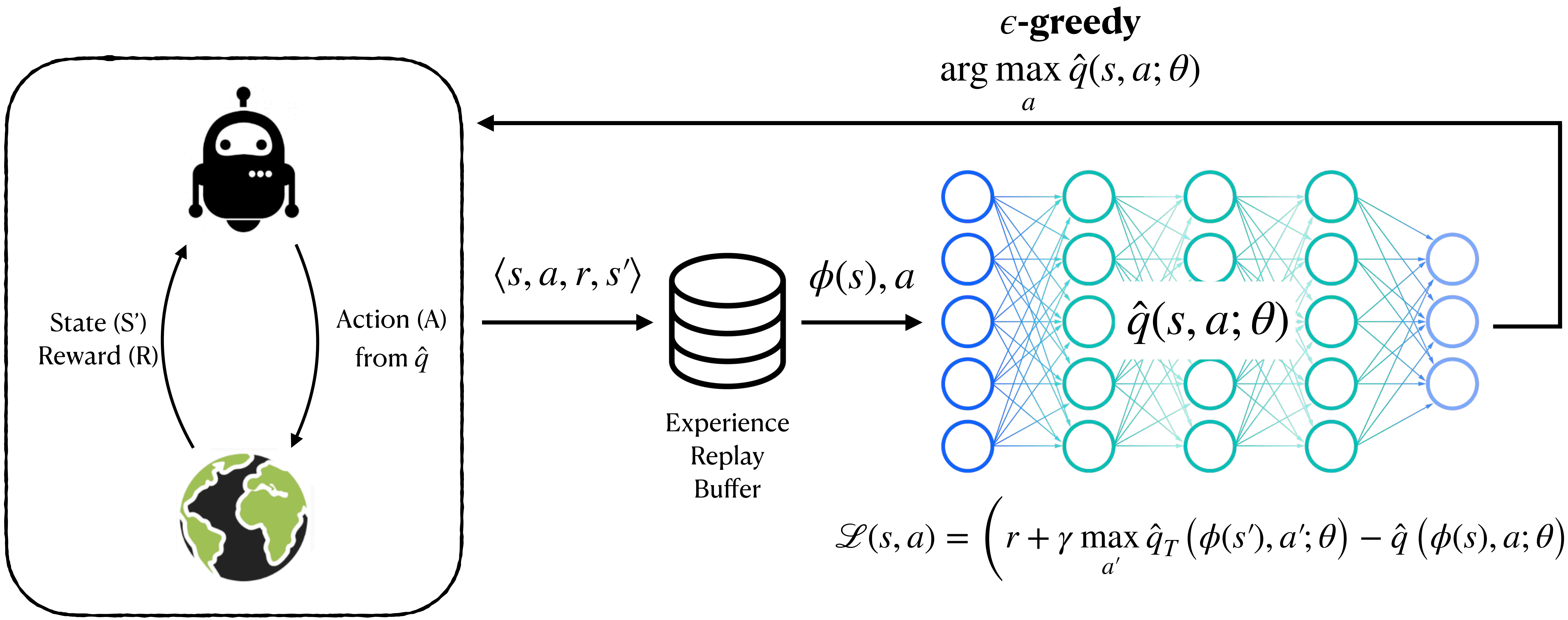
D. Silver 2015

Deep Q-Networks (DQN)

Q-Learning at scale

- DQN is based on the Q-Learning algorithm where the **Q-Table is approximated by a (deep) neural network**
- DQN has two key enhancements w.r.t. the Q-learning algorithm to actually make it work:
  - **Experience replay buffer**: to reduce the instability caused by training on highly correlated sequential data, store transition tuples $\langle s, a, r, s' \rangle$ buffer. Cut down correlations by randomly sampling the buffer for mini-batches of training data.
  - **Freeze the target network**: to address the instability caused by chasing a moving target, freeze the target network and only update it periodically with the latest parameters from the trained estimator.

DQN in a figure



$$\mathscr{L}(s, a) = \left(r + \gamma \max_{a'} \hat{q}_T \left(\phi(s'), a'; \theta\right) - \hat{q}\left(\phi(s), a; \theta\right) \right)^2$$

$\hat{q}_T$: the target network (old version of $\hat{q}$) to alleviate the moving target problem

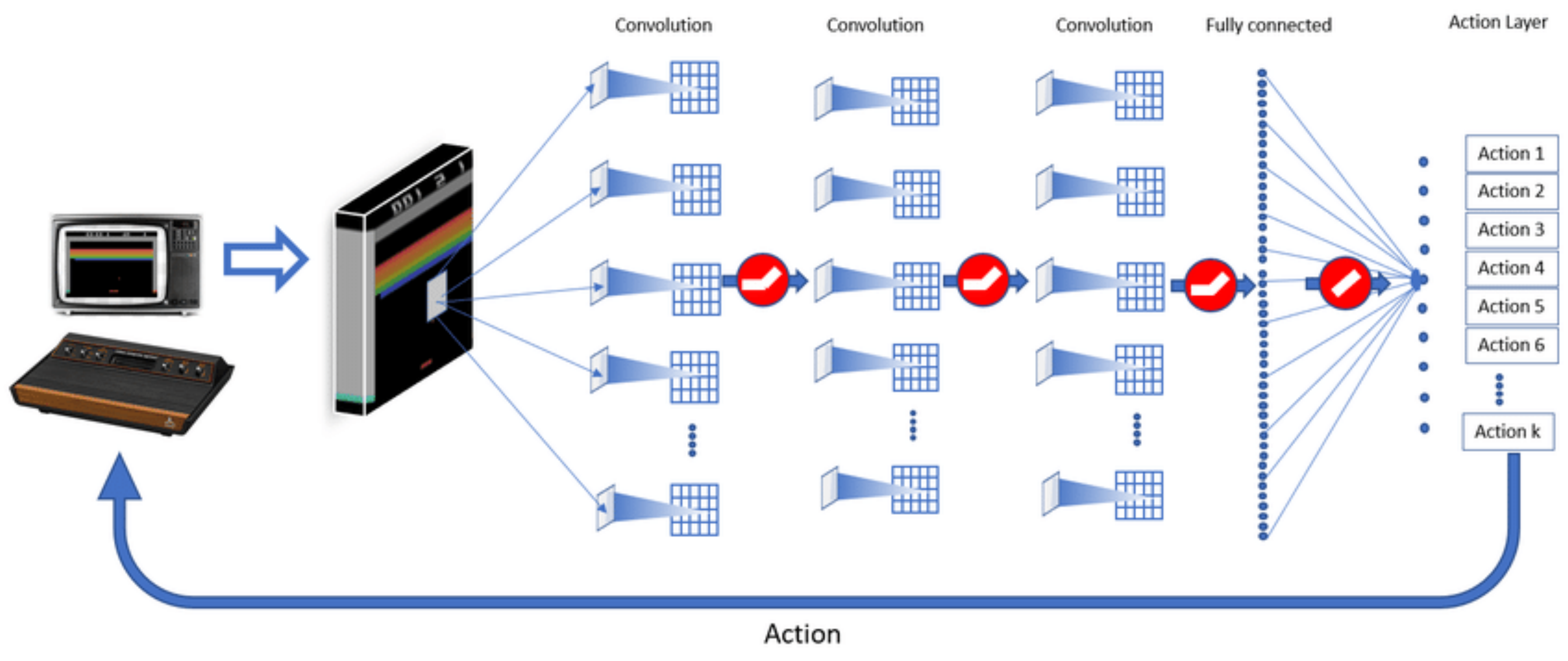
DQN Algorithm

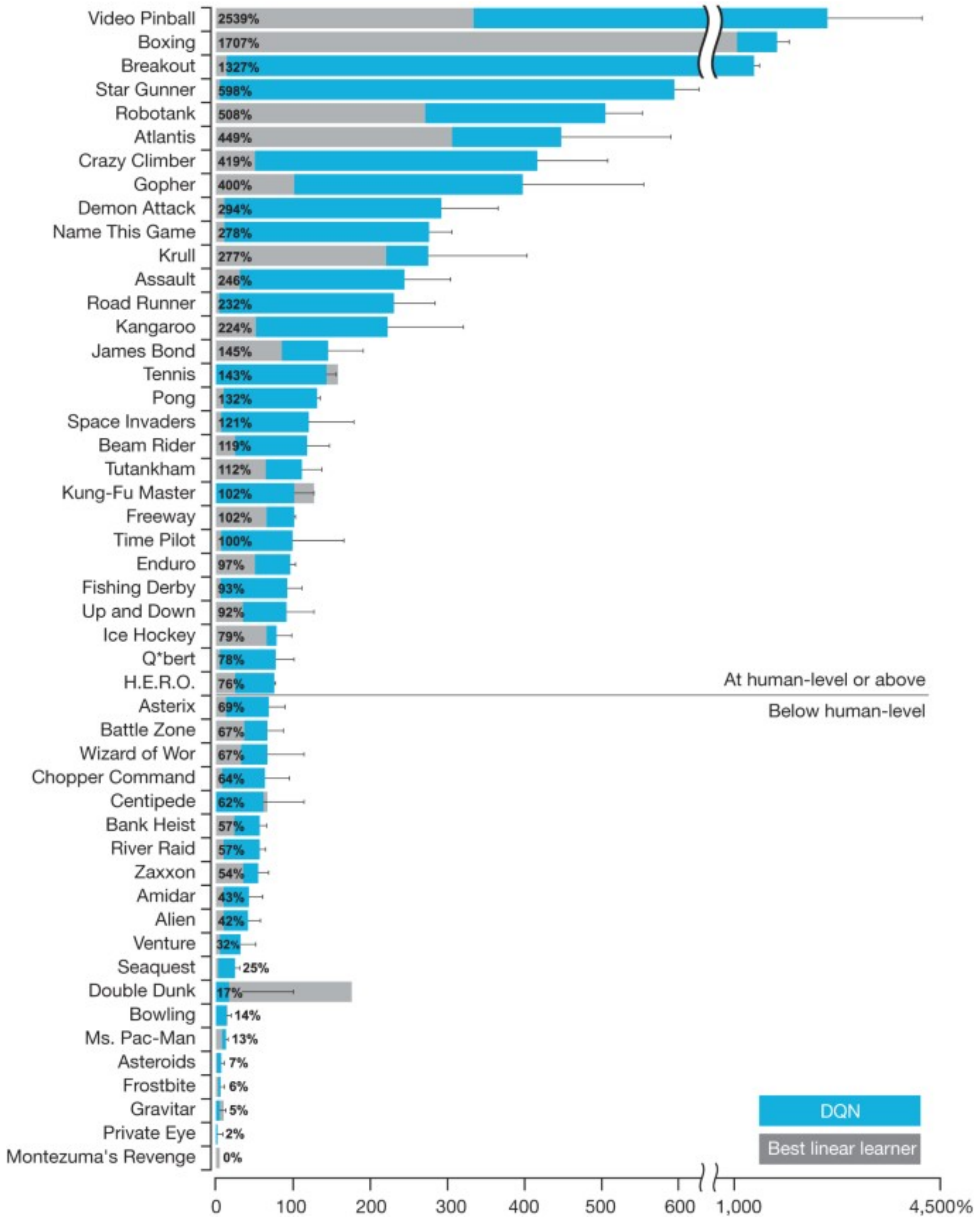
Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

DQN for ATARI

A classic example

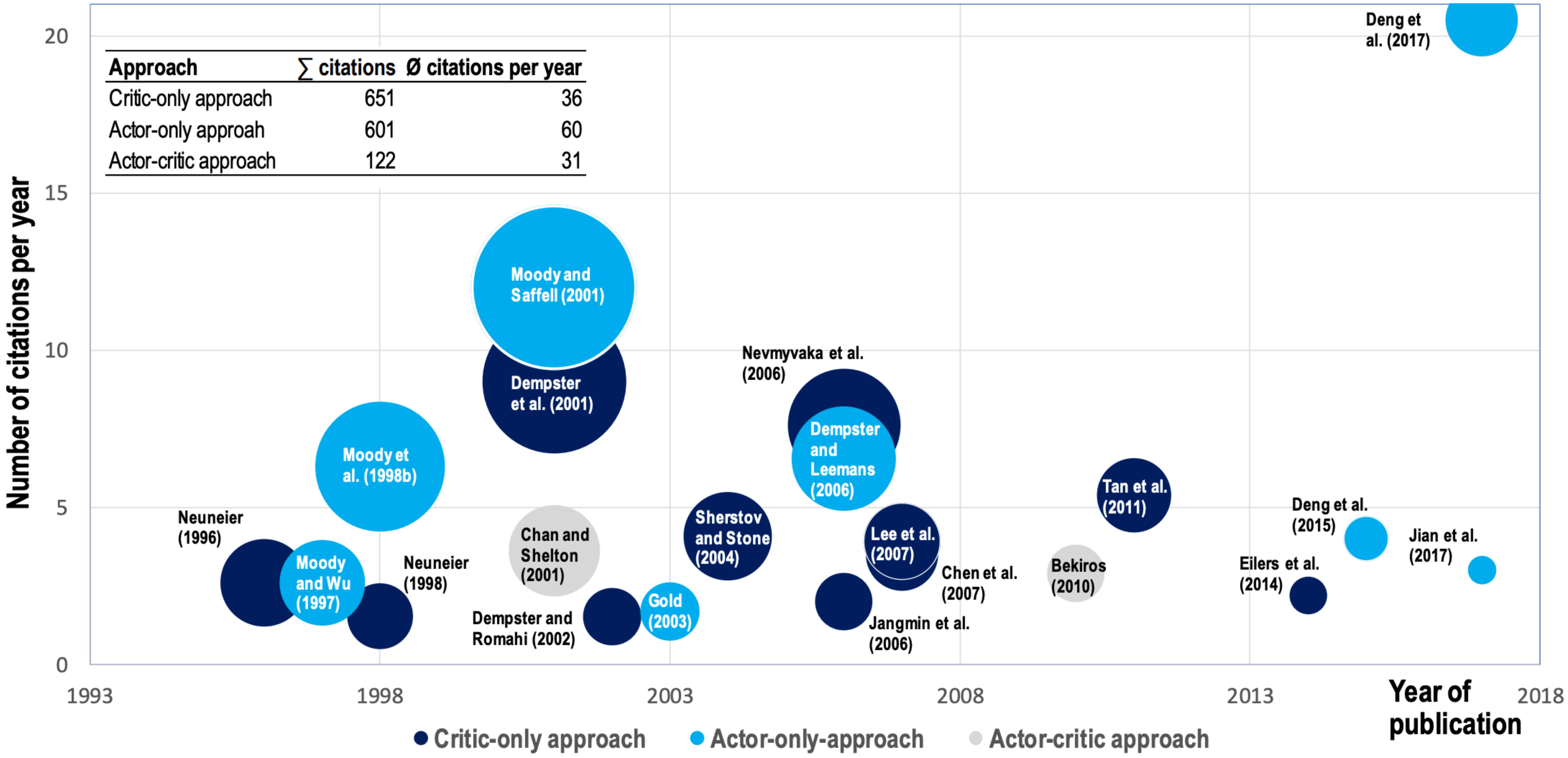




Part IV

RL application in finance - Some examples

RL in economics and finance



Fischer, Thomas G., 2018. "Reinforcement learning in financial markets - a survey," FAU Discussion Papers in Economics 12/2018, Friedrich-Alexander University Erlangen-Nuremberg, Institute for Economics.

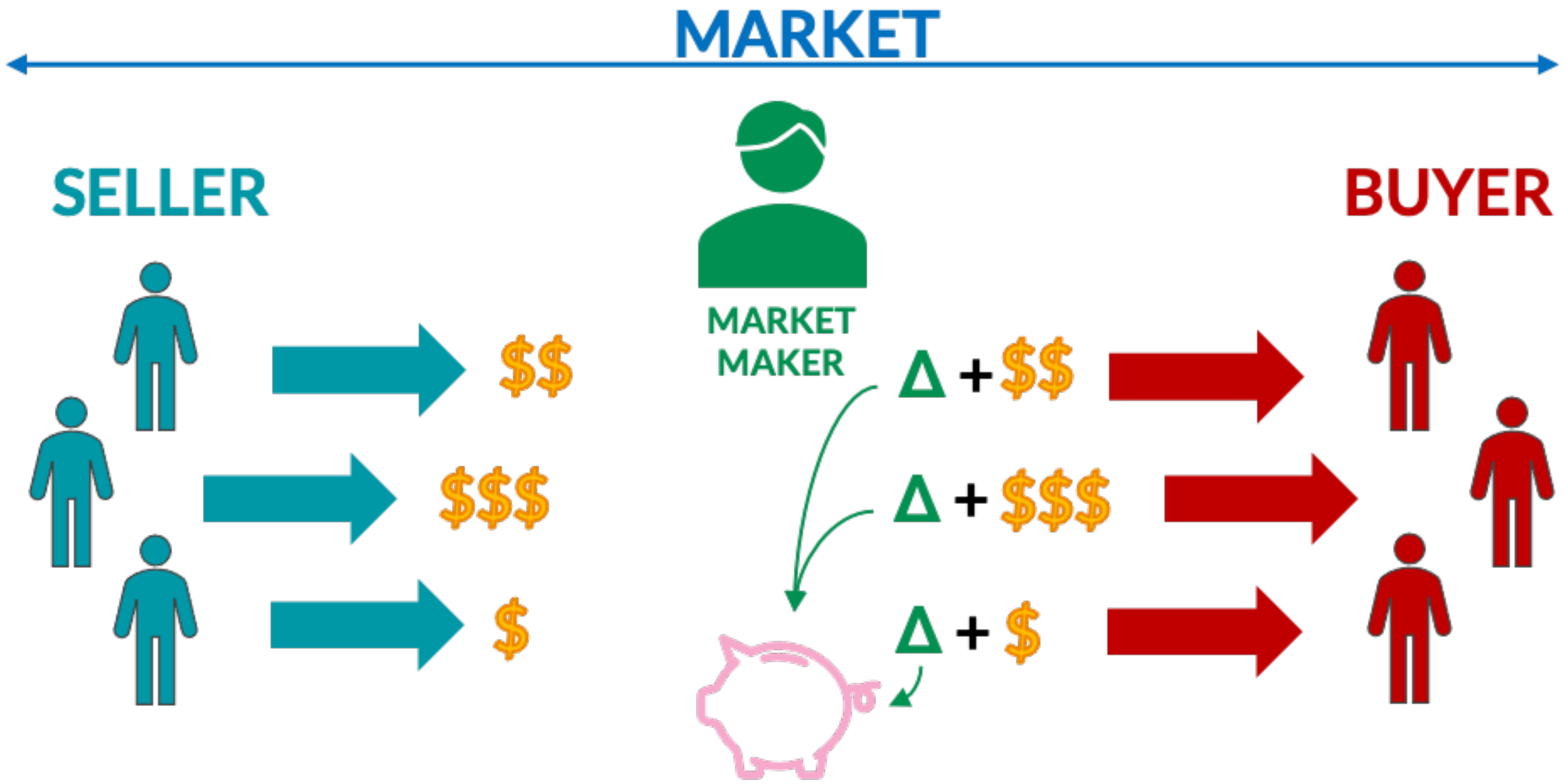
Market Making via Reinforcement Learning

Thomas Spooner, John Fearnley, Rahul Savani, and Andreas Koukorinis. 2018. Market Making via Reinforcement Learning. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 434–442.

Market Maker

Traders who profit from facilitating exchange in a particular asset and exploit their skills in executing trades

Cartea, Jaimungal, & Penalva. Algorithmic and High-Frequency Trading



Profit & risks of a Market Maker

Profits

- **Spread** (Δp)
- **Favorable market**: increasing of the value of the owned financial instruments

Risks

- **Non-zero inventory**: bought financial instruments are never sold, or viceversa
- **Unfavorable market**: decreasing of the value of the owned financial instruments

State space

The observable variables for the market maker

Agent/Market maker state

- **Inv(t)**: the amount of stock currently owned or owed by the agent
- Effective values of the **control parameters,** $\theta_{a,b}$, after going forward in the simulation

Market/Environment state

- Market (bid/ask) **spread** (Δs)
- **Mid-price move** (Δm)
- **Volatility**
- **RSI**

PROBLEM: huge (continuous) state space → **Tile coding**

Reward function

The quantity the market maker wants to maximise

PnL REWARD: the money lost/gained through executions of the orders relative to the mid-price

agent's quoted spread + inventory increment – dampening factor

price

$$R_t = X_t^a \cdot \left[p_t^a - m_t\right] + X_t^b \cdot \left[m_t - p_t^b\right] + \boxed{I_t \Delta m_t} - \boxed{\eta D\left(I_t\right)}$$

volume matched (executed) against the agent's orders since t–1 in the order books

mid-price

Action space

What the market maker can do

Action ID	0	1	2	3	4	5	6	7	8
Ask (θ_{a})	1	2	3	4	5	1	3	2	5
Bid (θ_{b})	1	2	3	4	5	3	1	5	2
Action 9	MO with $\mathrm{Size}_{\mathrm{m}} = -\mathrm{Inv}(t_i)$								

clear its inventory using a Market Order

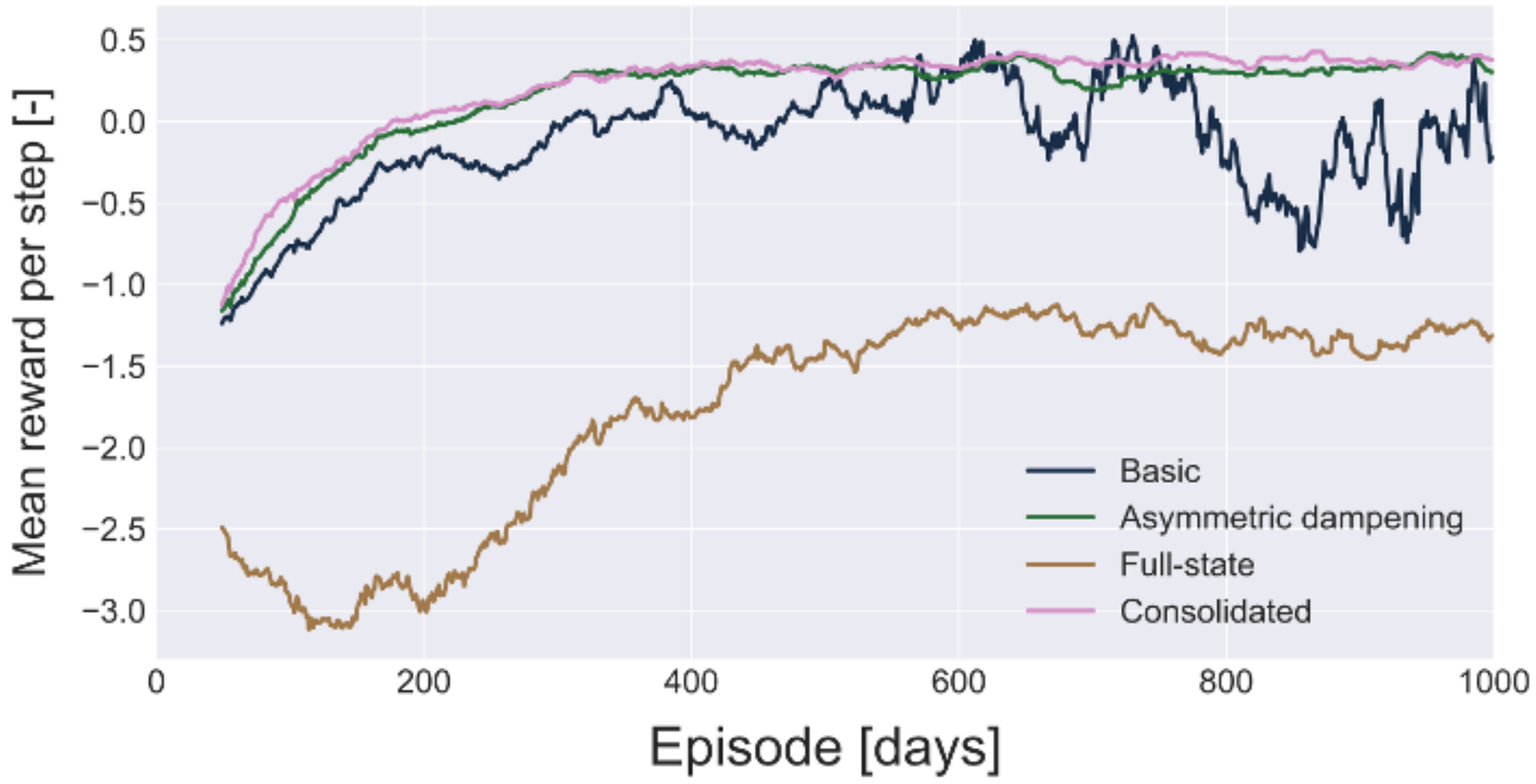
Agent's pricing strategy $$p_t^{a,b} = m_t + \frac{1}{2}\theta_t^{a,b}\Delta s_t$$

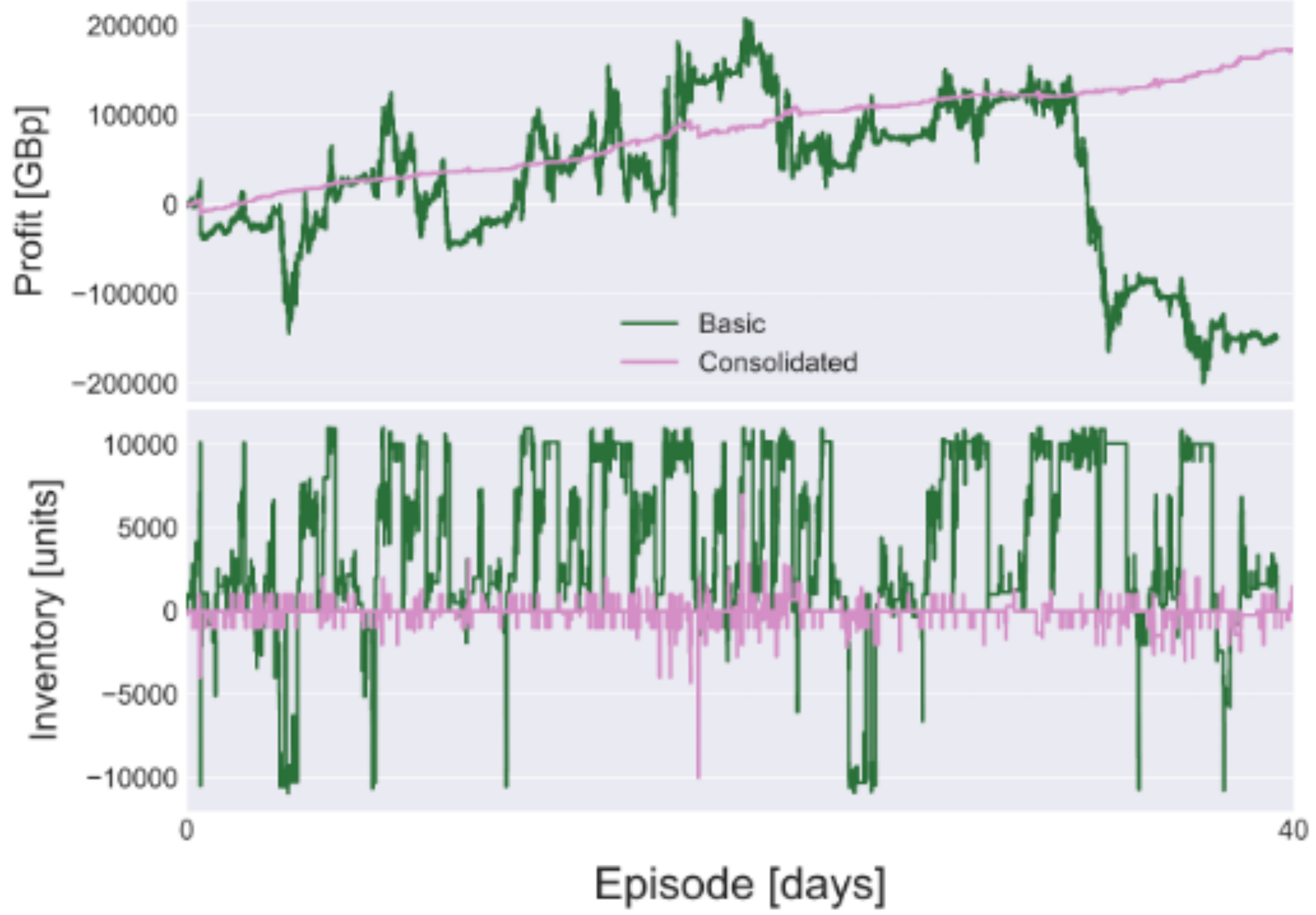
Experimental setting

- **Simulated data** of a financial market via direct reconstruction of the limit order book from historical data (January — August 2010) of 10 securities from 4 different sectors
- Tested RL models:
  - **Q-learning**
  - SARSA
  - R-learning
  - Variants of the previous approaches
  - Consolidated agent: SARSA + ad-hoc state representation

Results - PnL

	CRDI.MI	GASI.MI	GSK.L	HSBA.L	ING.AS	LGEN.L	LSE.L	NOK1V.HE	SAN.MC	VOD.L
Double Q-learning	−5.04 ± 83.90	5.46 ± 59.03	6.22 ± 59.17	5.59 ± 159.38	58.75 ± 394.15	2.26 ± 66.53	16.49 ± 43.10	−2.68 ± 19.35	5.65 ± 259.06	7.50 ± 42.50
Expected SARSA	0.09 ± 0.58	3.79 ± 35.64	−9.96 ± 102.85	25.20 ± 209.33	6.07 ± 432.89	2.92 ± 37.01	6.79 ± 27.46	−3.26 ± 25.60	32.28 ± 272.88	15.18 ± 84.86
R-learning	5.48 ± 25.73	−3.57 ± 54.79	12.45 ± 33.95	−22.97 ± 211.88	−244.20 ± 306.05	−3.59 ± 137.44	8.31 ± 23.50	−0.51 ± 3.22	8.31 ± 273.47	32.94 ± 109.84
Double R-learning	19.79 ± 85.46	−1.17 ± 29.49	21.07 ± 112.17	−14.80 ± 108.74	5.33 ± 209.34	−1.40 ± 55.59	6.06 ± 25.19	2.70 ± 15.40	32.21 ± 238.29	25.28 ± 92.46
On-policy R-learning	0.00 ± 0.00	4.59 ± 17.27	14.18 ± 32.30	9.56 ± 30.40	18.91 ± 84.43	−1.14 ± 40.68	5.46 ± 12.54	0.18 ± 5.52	25.14 ± 143.25	16.30 ± 32.69





Deep Reinforcement Learning for trading

Zihao Zhang, Stefan Zohren, Roberts Stephen, 2020. Deep Reinforcement Learning for Trading.
The Journal of Financial Data Science. DOI: https://doi.org/10.3905/jfds.2020.1.030

Trading

"This falls under the framework of optimal control theory and forms a classical sequential decision-making process."

- **Profit from buying & selling** different financial instruments
- Deals with probability never certainty
- Trading vs Investing: holding period

Goal of a trader (*)

Maximize some expected utility (U) of final wealth

$$\mathbb{E}\left[U\left(W_T\right)\right] = \mathbb{E}\left[U\left(W_0 + \sum_{t=1}^{T} \delta W_t\right)\right]$$

Goal of RL

Maximize the expected return G, i.e., the expected discounted cumulative rewards

$$\mathbb{E}[G] = \mathbb{E}\left[\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k\right]$$

(*) Modern portfolio theory:
- Arrow, K. J. "The Theory of Risk Aversion." In Essays in the Theory of Risk-Bearing, pp. 90–120. Chicago: Markham, 1971.
- Pratt, J. W. "Risk Aversion in the Small and in the Large." In Uncertainty in Economics, pp. 59–79. Elsevier, 1978.
- Ingersoll, J. E. Theory of Financial Decision Making, vol. 3. Lanham, MD; Rowman & Littlefield, 1987.

Action space

- -1: maximally short position — **SELL**
- 0: no holdings — **DO NOTHING**
- +1: maximally long position — **BUY**
- If $a_t = a_{t+1}$: no transaction costs
- If $a_t = -a_{t+1}$: double transaction costs
- In the **continuous case** the action can be anything in the **range [-1, 1]**

Reward function

Profits representing a risk-insensitive trader

cost rate: β=10-4

price

volatility target

$$R_t = A_t \frac{\sigma_{tgt}}{\sigma_{t-1}} \left(p_t - p_{t-1}\right) - \beta p_{t-1} \left| \frac{\sigma_{tgt}}{\sigma_{t-1}} A_{t-1} - \frac{\sigma_{tgt}}{\sigma_{t-2}} A_{t-2} \right|$$

additive profit

Transaction cost

ex ante volatility
calculated using a weighted moving std with a 60-day window on the additive profit

State space

- Normalized **close price series**
- Normalized **returns** over the past 1, 2, 3 and 12 months
- **MACD(*) indicator** which "measures" the momentum, direction and duration of the trend of the price.
- **RSI indicator** in [0, 100] with a look-back window of 30 days
    - ≤ 20: oversold
    - ≥ 80: overbought

(*) Baz, J., N. Granger, C. R. Harvey, N. Le Roux, and S. Rattray. "Dissecting Investment Strategies in the Cross Section and Time Series." SSRN 2695101, 2015.

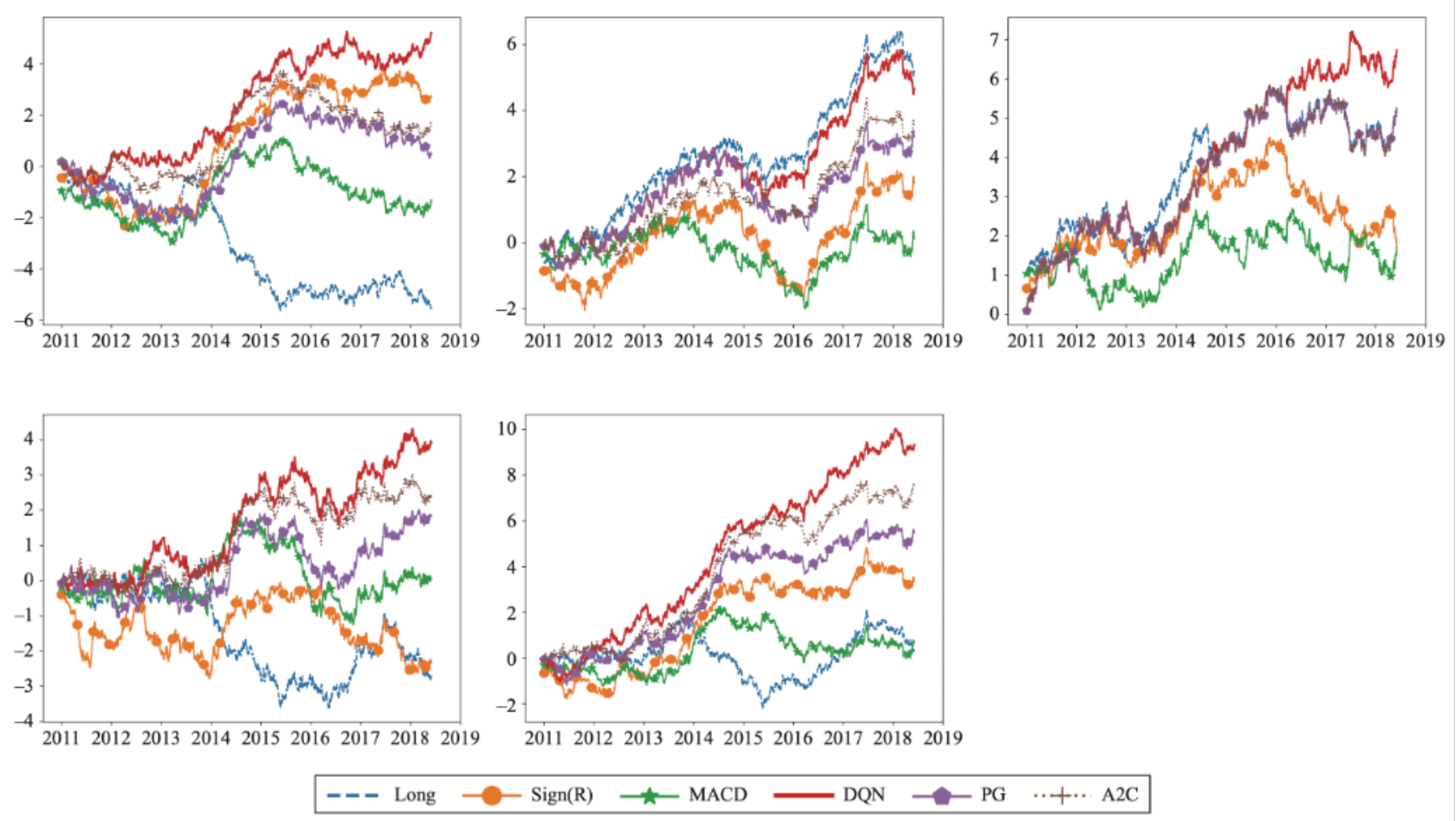
Experimental setting

- Dataset: CLC Database (*2019) that ranges from 2005 to 2019 and consists of a variety (4) of asset classes

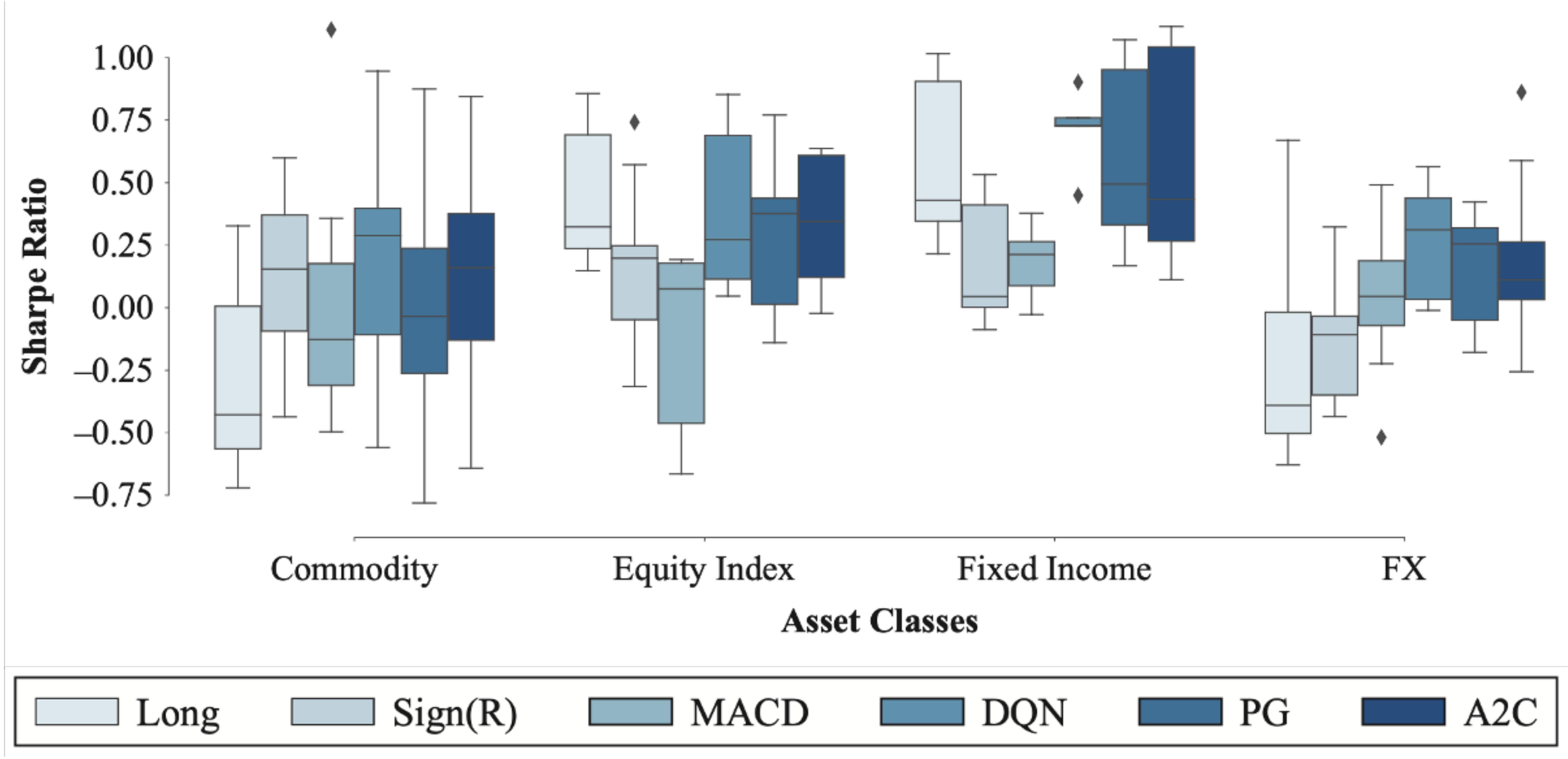
- Function approximator: 2-layer LSTM with 64/32 units, and Leaky-RELU
- RL techniques: **DQN**, A2C and PG
- **A separate model for each asset class is trained**
- The **portfolio is equally distributed** over all the asset classes

(*2019) CLC Database. Pinnacle Data Corp, 2019, https://pinnacle- data2.com/clc.html.

Results - Cumulative trade return



Results - Sharp ratio



Final Remarks

a.k.a. Take home message

Conclusions

- RL is the ML **paradigm for sequential decision making**
- RL highly **differs from standard learning** like supervised and unsupervised ML
- RL represents a **“natural” way of learning**
- RL shares many characteristics with **Game Theory** (used in mathematical economics)
- RL shows **high potential in financial applications** and it is currently an hot topic

Useful References

In addition to the ones reported as footnotes

1. *[BOOK]* Sutton, R.S. & Barto, A.G., 2018. Reinforcement learning: An introduction, MIT press
2. *[PREPRINT]* Arthur Charpentier, Romuald Elie, Carl Remlinger. Reinforcement Learning in Economics and Finance, 2020. https://arxiv.org/pdf/2003.10014.pdf
3. *[PREPRINT]* Mosavi, A.; Faghan, Y.; Ghamisi, P.; Duan, P.; Ardabili, S.F.; Salwana, E.; Band, S.S. Comprehensive Review of Deep Reinforcement Learning Methods and Applications in Economics. Mathematics 2020, 8, 1640.
4. *[VIDEO]* Arthur Charpentier presentation on *RL for economics*. https://www.youtube.com/watch?v=vd7Pj9Ejyws

Thank you!

Questions are welcome!

The only stupid question is the one you were afraid to ask but never did.

Richard Sutton

Mirko Polato, PhD
Deparment of Mathematics
University of Padova
mpolato@math.unipd.it