(more on) Reinforcement Learning

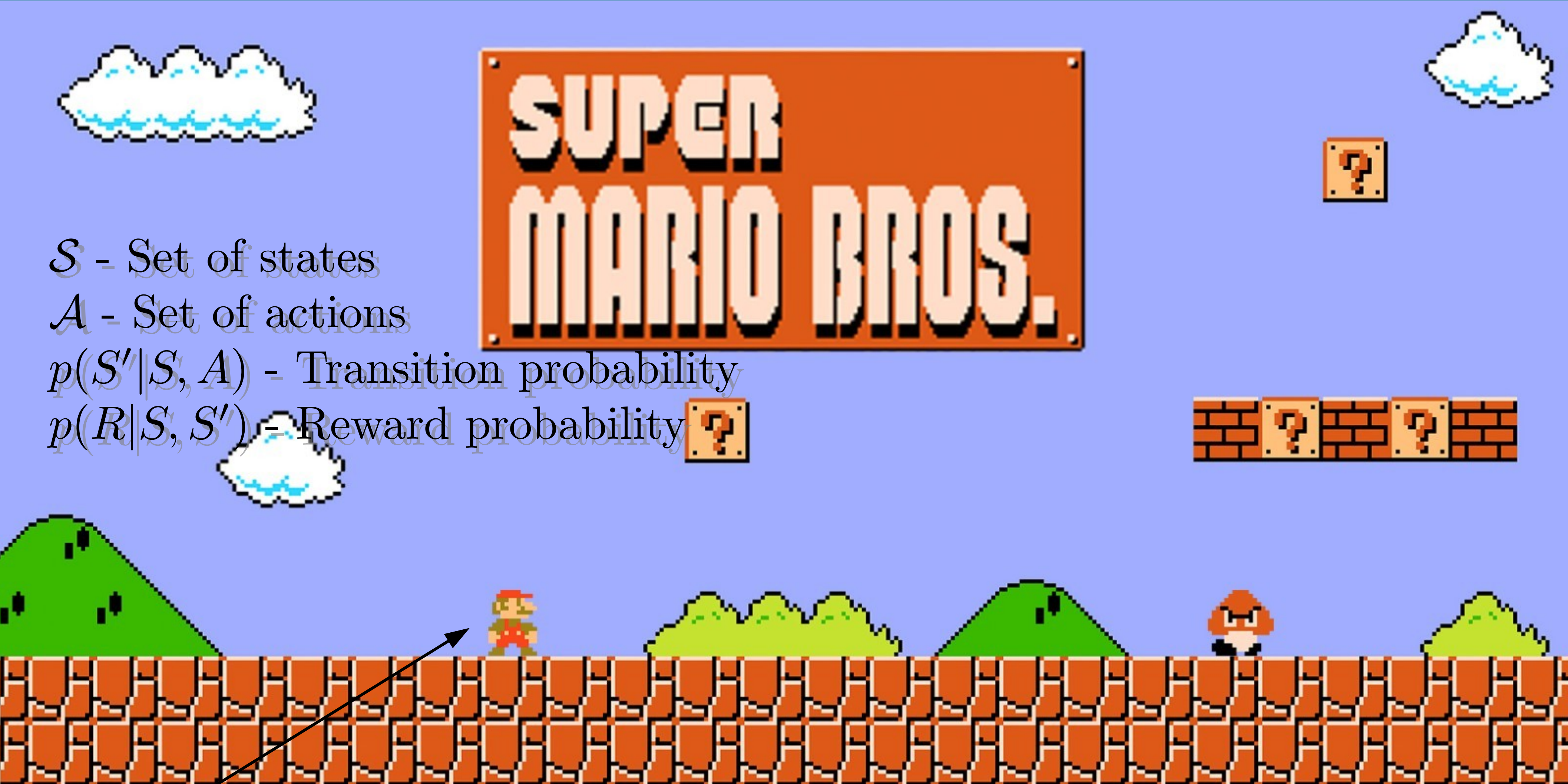
Winter School on Quantitative Systems Biology

19th November 2018

David Hofmann
david.hofmann@emory.edu

EMORY UNIVERSITY

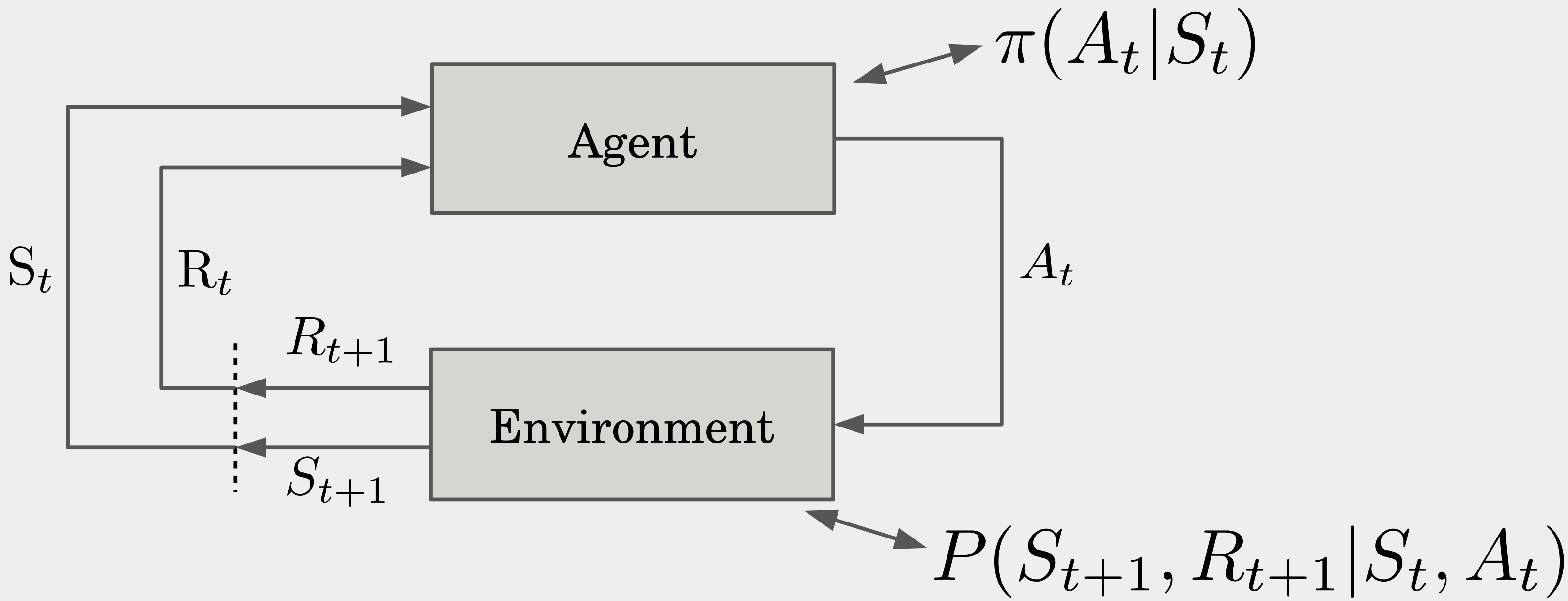
Complex Environment



$\mathcal{S}$ - Set of states
$\mathcal{A}$ - Set of actions
$p(S'|S, A)$ - Transition probability
$p(R|S, S')$ - Reward probability

$\pi(A|S)$ - Policy

Markov Decision Processes (MDP)



- An MDP is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$
- Actions lead to state transitions
- Rewards are released on state transitions

The simple RL problems

No actions:

- Classical conditioning
  (Markov Reward Process)

- Multiple actions $a \in \mathcal{A}$
- Each action a leads to a reward r with probability $P(R_t|A_t)$

$\rightarrow$ Arguably simplest RL problem
(it is state-less)

No states:

- Bandits!

One-armed bandit:
1899 "Liberty Bell" machine
[Wikimedia Commons]

Action Selection

Which of the k arms should I play?

Compute value of arms:

- Simplest algorithm: "Averaging"

$$Q_t(a) = \frac{R_1^a + \ldots + R_{N_t(a)-1}^a}{N_t(a) - 1}$$

- Iterative algorithm:

$$Q_{t+1}(a) = Q_t(a) + \frac{1}{N_t(a)}[R_{N_t(a)}^a - Q_t(a)]$$

Select an action:

- Greedy: $a_t = \text{argmax}_a Q_t(a)$
- Purely exploitative, no exploration

Exploration vs. Exploitation

- Epsilon greedy

$$a_t = \begin{cases} \text{argmax}_a Q_t(a), & \text{with probability } 1-\epsilon \\ \text{random } a, & \text{with probability } \epsilon \end{cases}$$

- Usually the greedy action is chosen
- But with probability ϵ choose a random action
- $\rightarrow$ Stochastic exploration

Exploration vs. Exploitation

- Upper Confidence Bound

$$a_t = \operatorname{argmax}_a \left(Q_t(a) + c\sqrt{\frac{\ln(t)}{N_t(a)}} \right)$$

- On top of quality of action uncertainty is also
- $\rightarrow$ Deterministic exploration

Exploration vs. Exploitation

- Bayesian approach

$$
\begin{aligned}
p_{\text{posterior}} &= \frac{p_{\text{likelihood}}}{p_{\text{evidence}}} p_{\text{prior}} \\
p_t(Q(a)|\mathcal{D}_t) &= \frac{p(r_t|Q(a))}{p(r_t)} p_t(Q(a)|\mathcal{D}_{t-})
\end{aligned}
$$

- Thompson Sampling:
  Sample from the posterior distribution.

$$Q_t(a) \sim p(Q(a)|\mathcal{D}_t)$$

$$a_t = \text{argmax}_a Q_t(a)$$

- $\rightarrow$ Stochastic exploration

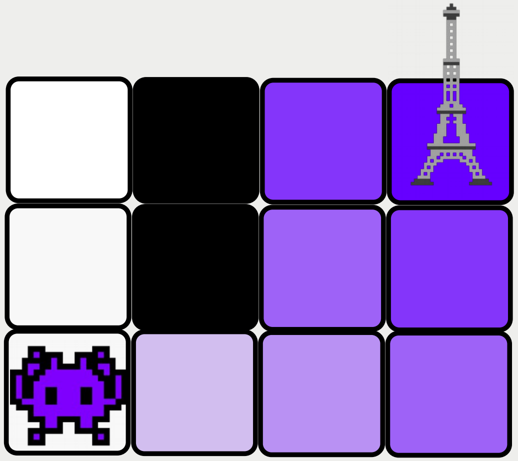
Bellman Equation

- Extend to discrete, finite state space
- Table values of state action pairs

$$Q_\pi(s,a) = \mathrm{E}_\pi\left(\sum_{t=0}^{\infty}\gamma^t R_{t+1}|S_0 = s, A_0 = a\right)$$

$$= \sum_{\{a_{t+1},s_{t+1},r_{t+1}\}} p(\{R_{t+1}, S_{t+1}, A_{t+1}\}|S_0 = s, A_0 = a)\sum_t^{\infty}\gamma^t R_{t+1}$$

$$Q_\pi(s,a) = \mathrm{E}_\pi\left(R_1|S_0 = s, A_0 = a\right) + \\ + \mathrm{E}_\pi\left(\sum_{t=1}^{\infty}\gamma^t R_{t+1}|S_1 = s', A_1 = a'\right)$$

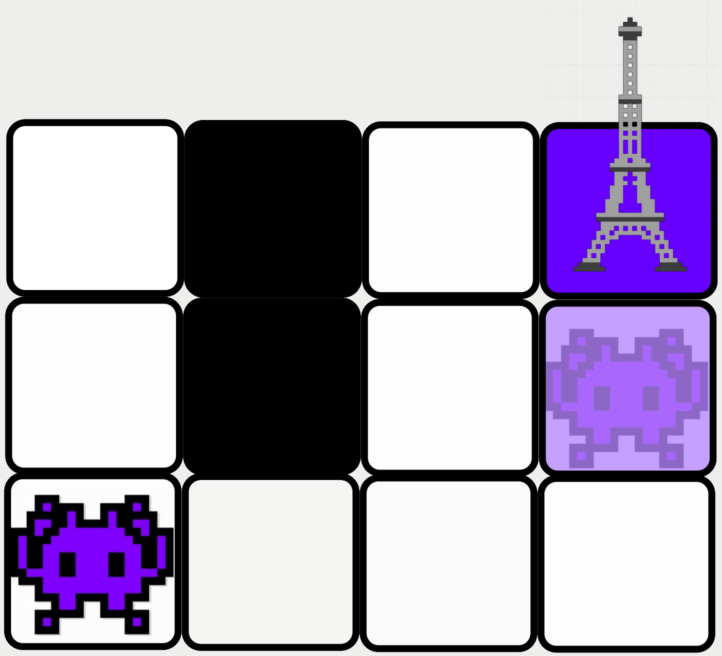


$$Q_\pi(s,a) = \mathrm{E}_\pi\left(R_1|S_0 = s, A_0 = a\right) + \gamma \mathrm{E}_\pi Q_\pi(s', a')$$

On-policy and Off-policy learning

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha_t \delta_{t+1}(Q_t) \mathbb{I}_{\{S_t=s, A_t=a\}}$$

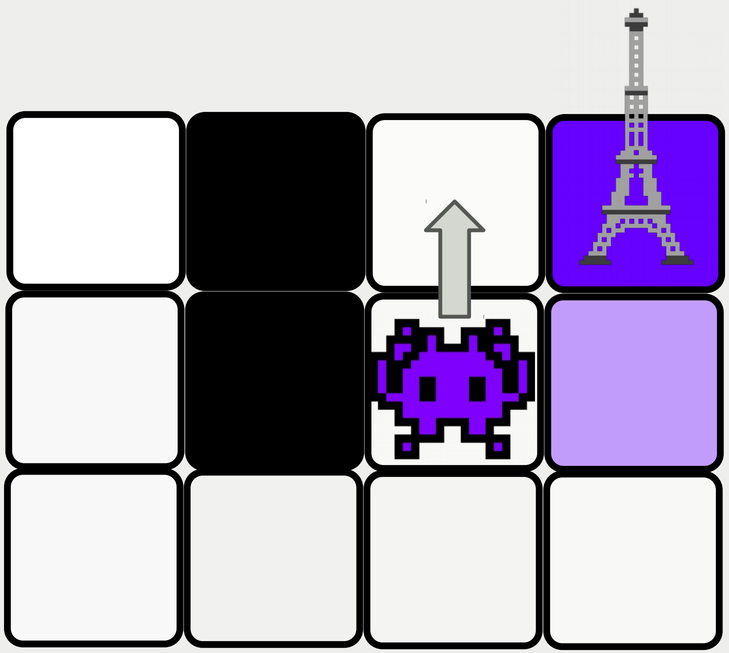
SARSA: $$\delta_{t+1} = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$



On-policy and Off-policy learning

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha_t \delta_{t+1}(Q_t) \mathbb{I}_{\{S_t = s, A_t = a\}}$$

Q-learning: $\delta_{t+1} = R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q(S'_{t+1}, a') - Q(S_t, A_t)$



On-policy and Off-policy learning

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha_t \delta_{t+1}(Q_t)\mathbb{I}_{\{S_t=s, A_t=a\}}$$

SARSA: $$\delta_{t+1} = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$

Q-learning: $$\delta_{t+1} = R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q(S'_{t+1}, a') - Q(S_t, A_t)$$

SARSA (on-policy)

- Regular TD learning for action-value functions
- Policy iteration through sampling the quintuplet $\{S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}\}$

Q-learning (off-policy)

- Q-learning is an instance of TD learning
- S' can be S but doesn't need to.
- Allows for sampling

Large or Continuous State Spaces

- So far we had discrete state space S.
  What if it is continuous?

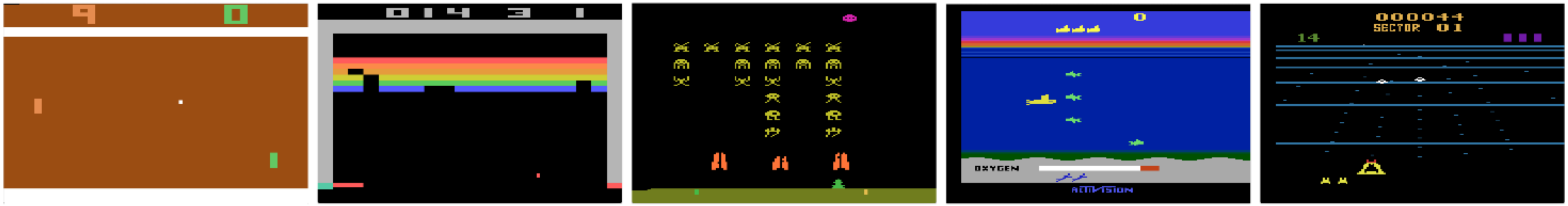
- Common approach: "discretize"!
  By function approximation.

$$Q(s,a) = \sum_i \theta_i f_i(s,a)$$

$f_i(s,a)$ … basis functions

θ_i … weights

Deep Q-learning



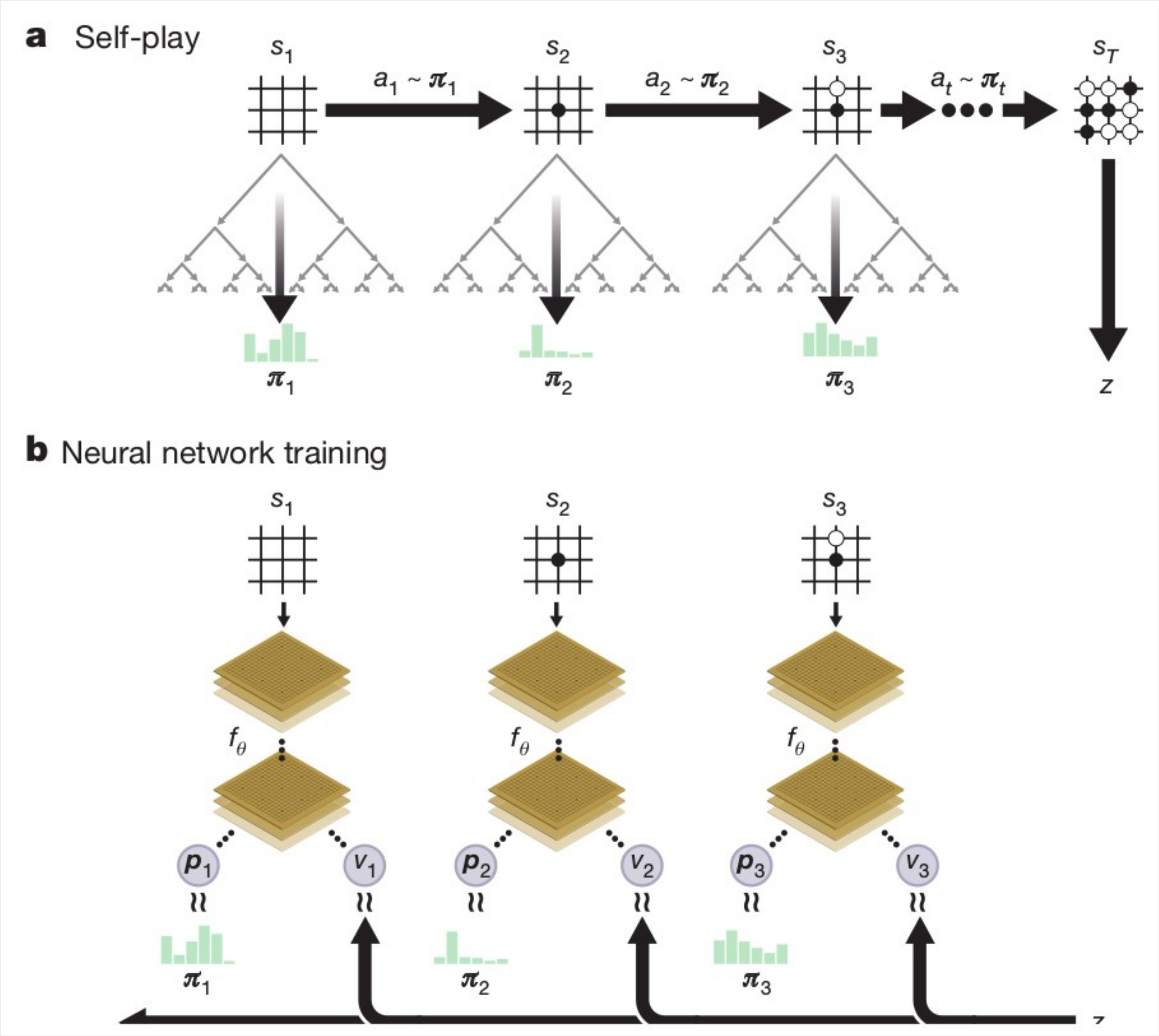
[Minh et al. 2013]

- Use DNNs to represent the value function
- “Experience replay” to make more efficient use of collected information
- (Explain SGD bit)

Advantages:

- Consecutive samples have strong correlations (little information)
- Not only one state update but propagating new information.
- Better convergence behavior when using function approximation (as DNN)
- When on-policy then training distribution depends on selected actions (can result in unwanted feedback loops)

Alpha Go Zero



[Silver et al. 2017]

References

- Reinforcement Learning: An Introduction. **Sutton R. and Barto A.**, 2nd Edition, 2018
- Algorithms for Reinforcement Learning. **Szepesvári C.** 2012
- A Tutorial on Thompson Sampling. **Russo et al.** *Foundations and Trends in Machine Learning* 2018.
- Mastering the game of Go without human knowledge. **Silver et al.** *Nature* 2017
- Playing Atari with Deep Reinforcement Learning **Mnih et al.** arXiv:1312.5602 [cs] 2013