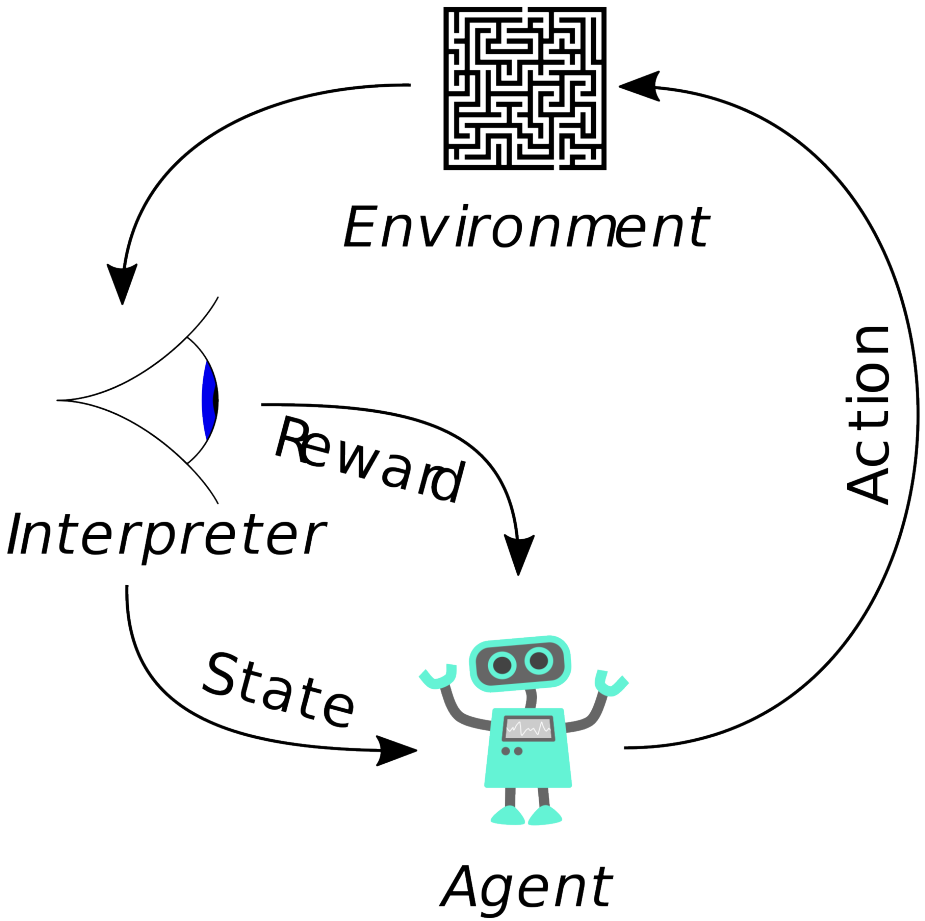


Reinforcement Learning

张江

北京师范大学系统科学学院教授

集智俱乐部、集智学园创始人

集智研究中心理事长

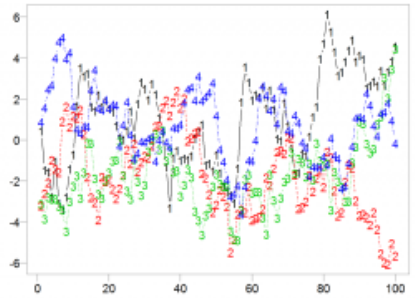
Outline

- Basic Notions
- AlphaGo
- Deep Q-Learning
- Reinforcement Learning and Causality
- World Models

Why RL?

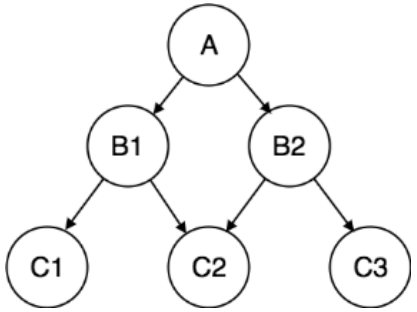
Complex System

Behavior (observations)



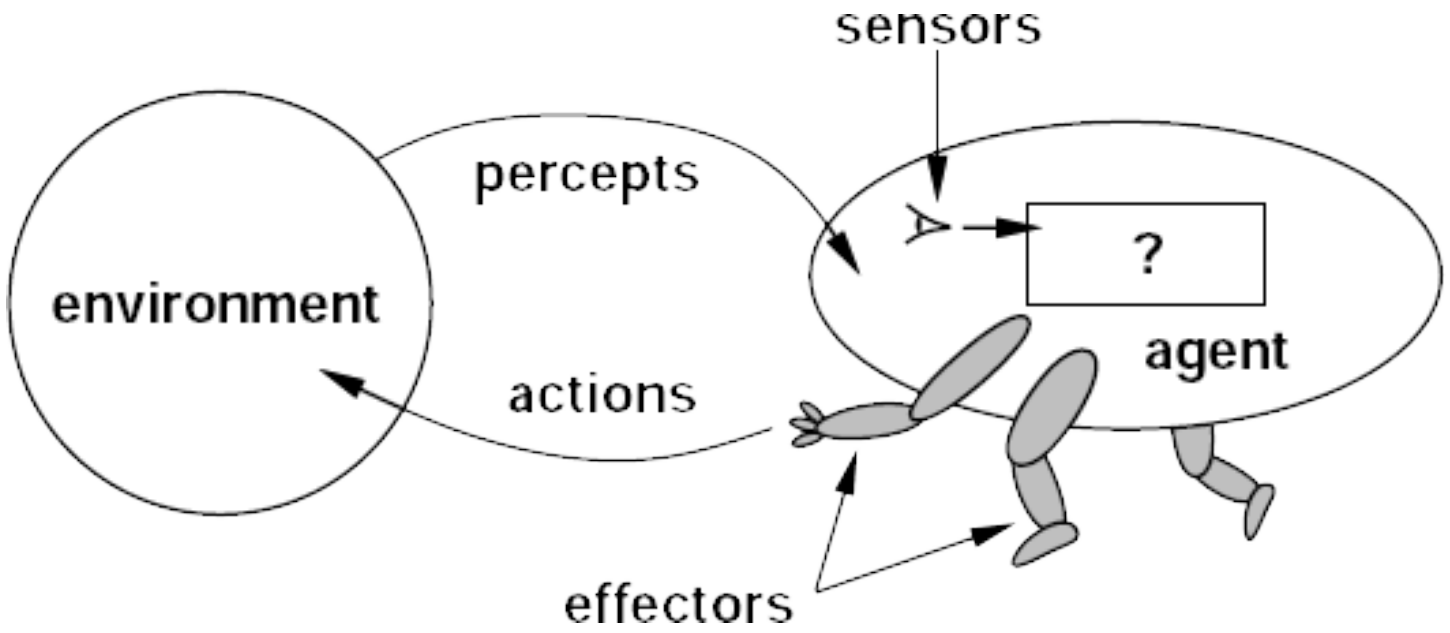
AI systems

Models



Prediction, Simulation,
Optimization, Controls

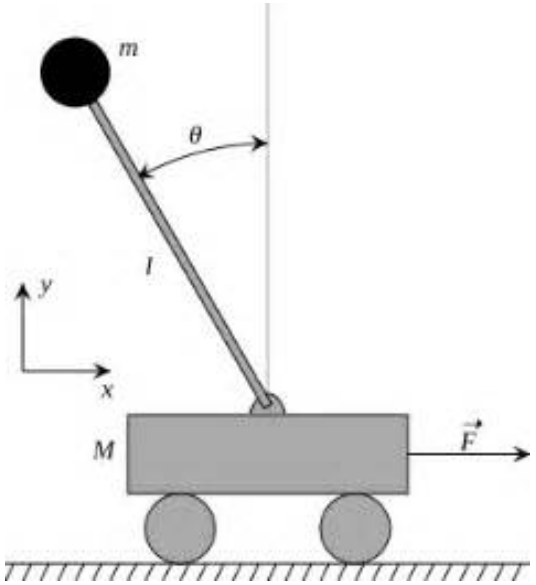
A Complete Agent

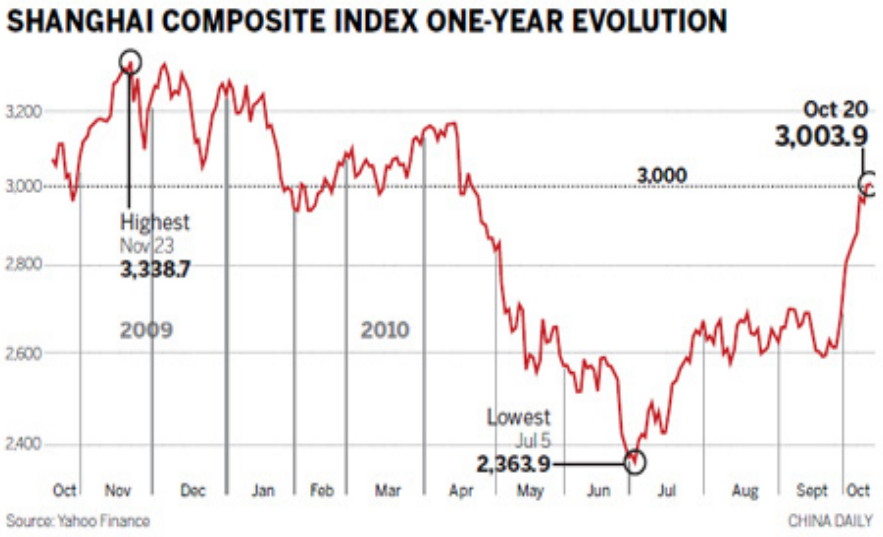


智能体（Intelligent agent）应该是一个完整的可以感知、决策、行动、学习的整体，而不是分开的一系列任务

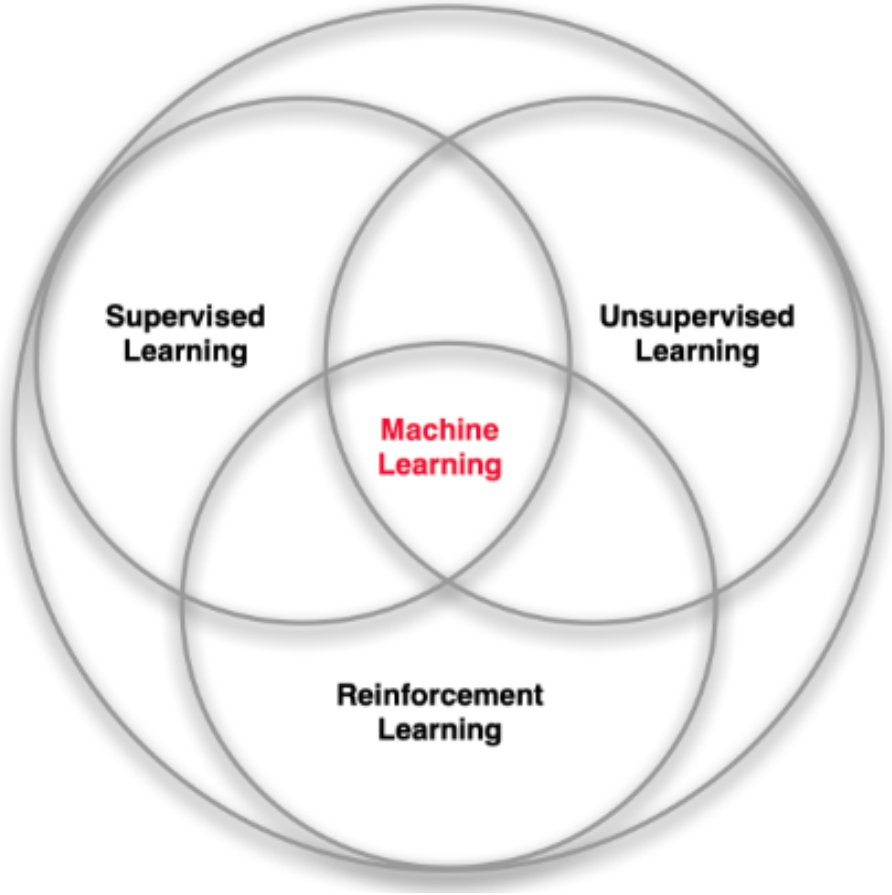
Some Cases


柯洁 KE JIE
00:20:36
ALPHAGO
01:57:08


m
θ
l
y
x
M
F

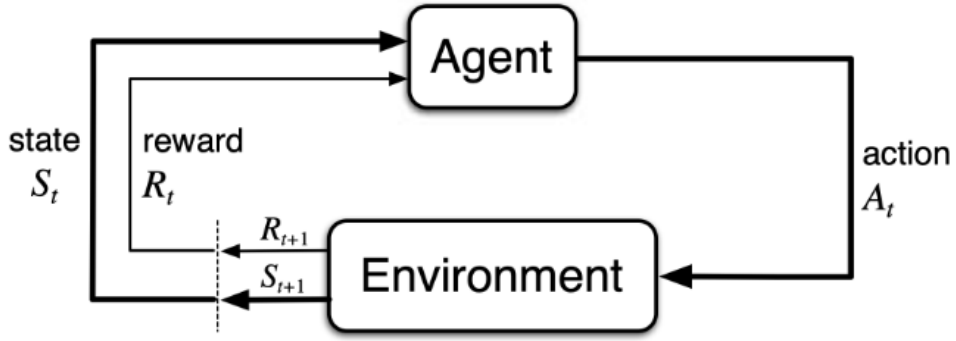

SHANGHAI COMPOSITE INDEX ONE-YEAR EVOLUTION
3,200
3,000
2,800
2,600
2,400
Oct 20
3,003.9
3,000
Highest
Nov 23
3,338.7
2009
2010
Lowest
Jul 5
2,363.9
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sept
Oct
Source: Yahoo Finance
CHINA DAILY

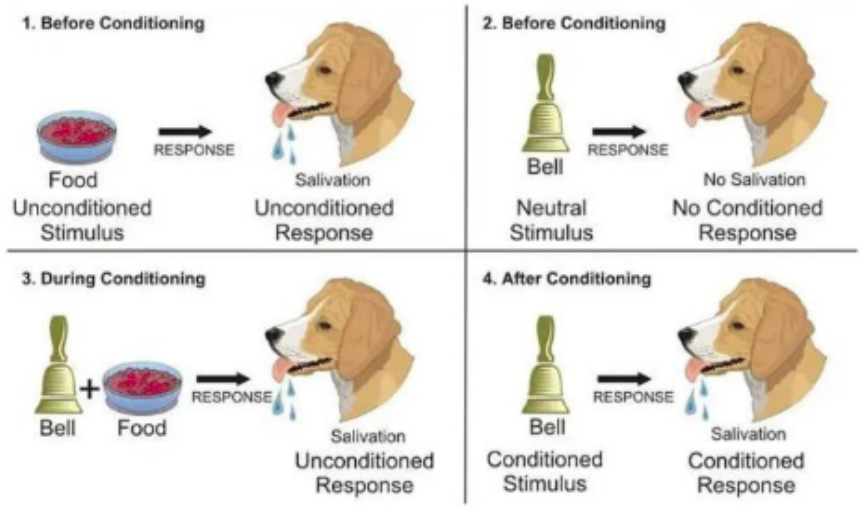
Territory


Supervised Learning
Unsupervised Learning
Machine Learning
Reinforcement Learning

Early histories of RL

- Trial-and-Error learning in Psychology
- Optimal control and its solution with dynamic programming
- Temporal difference learning

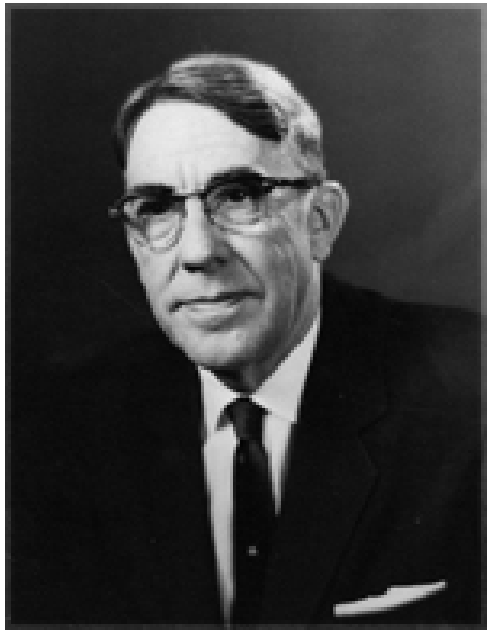




Stimulus-response

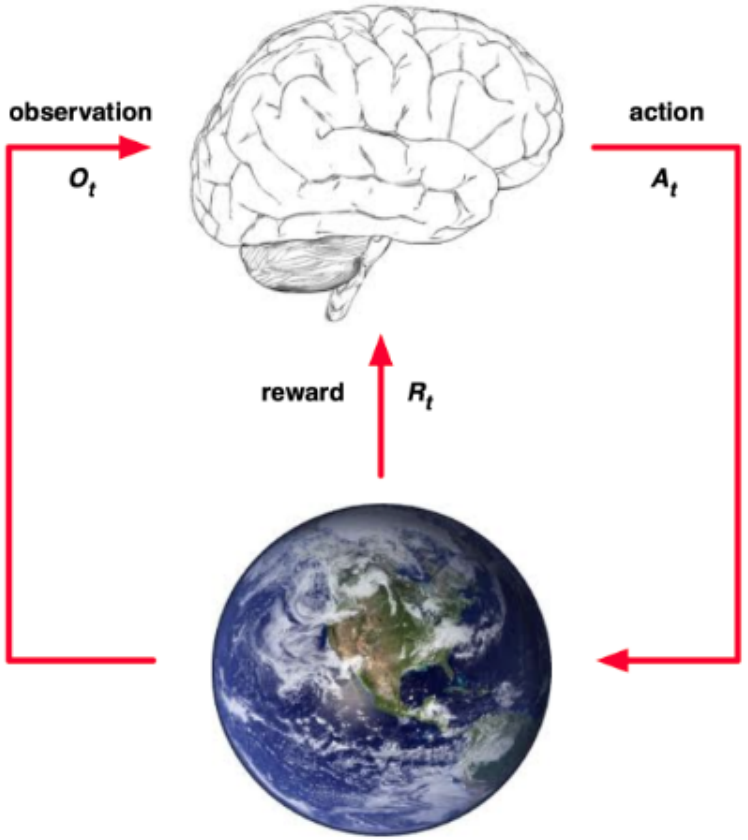
Early history of RL

Reinforcement learning (**RL**) is an area of machine learning inspired by behaviourist psychology, concerned with how software agents ought to take *actions* in an *environment* so as to maximize some notion of cumulative *reward*.



Arthur Samuel

What is RL? – Problem Setting



- At each step t the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at env. step

What is RL? – Problem Setting

- A reward R_t is a scalar feedback signal
- The agent's job is to maximize cumulative reward

Reward Hypothesis

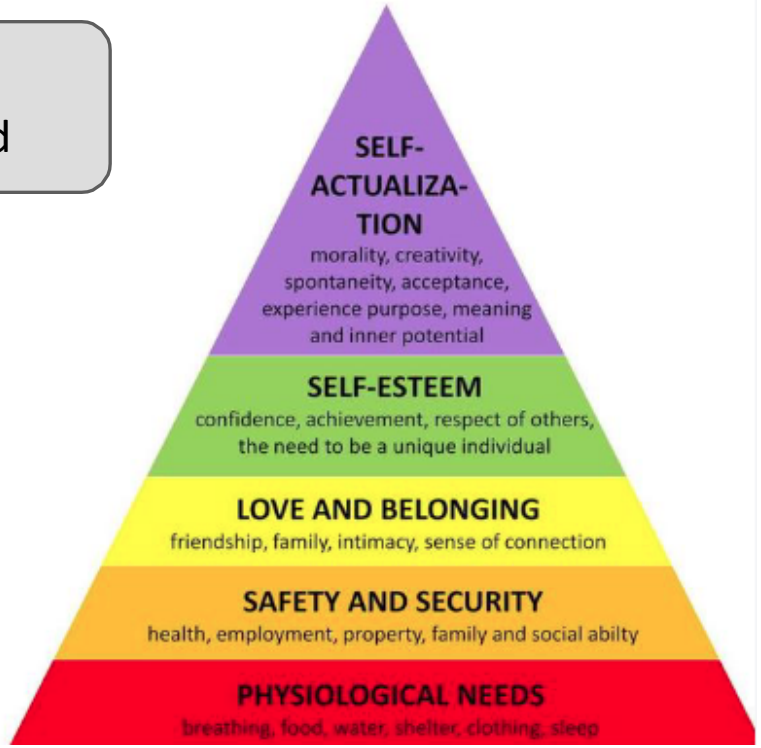
All goals can be described by the maximization of expected cumulative reward

Definition: Return (aka cumulative future reward).

- $G_t = R_t + R_{t+1} + R_{t+2} + R_{t+3} + \cdots$

Definition: Discounted return (aka cumulative discounted future reward).

- γ: discount factor (tuning hyper-parameter).
- $G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots$



Simplify – Markov Decision Process

- **Markov Property**
- **Markov Process**
- **Markov Reward Process**
- **Markov Decision Process**

Definition

A state S_t is *Markov* if and only if

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

Definition

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

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$

Definition

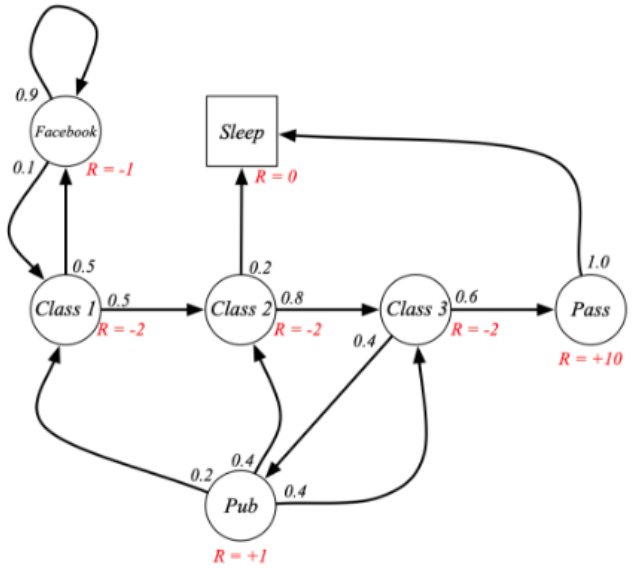
A *Markov Reward Process* is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

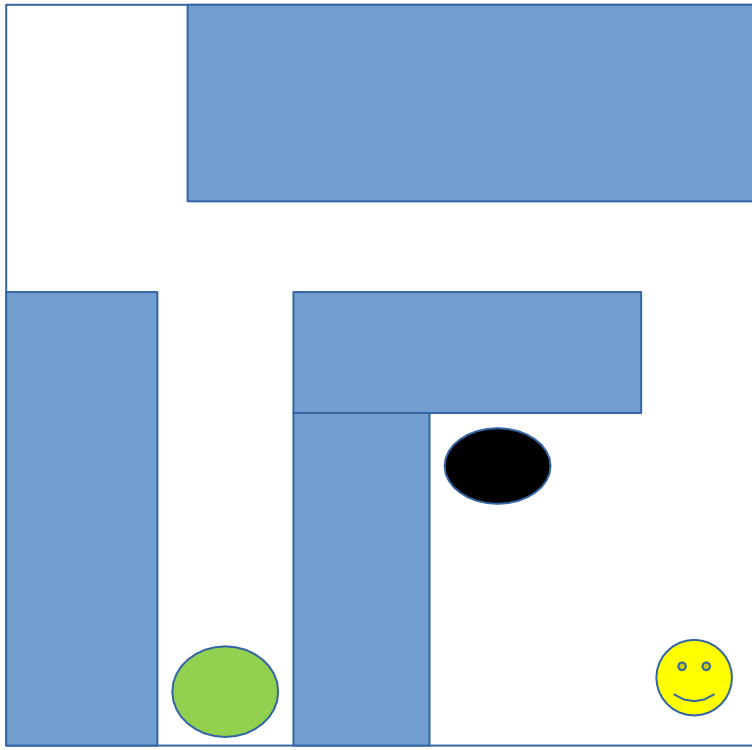
- $\mathcal{S}$ is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}[R_{t+1} \mid S_t = s]$
- γ is a discount factor, $\gamma \in [0, 1]$

Definition

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

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- γ is a discount factor $\gamma \in [0, 1]$.



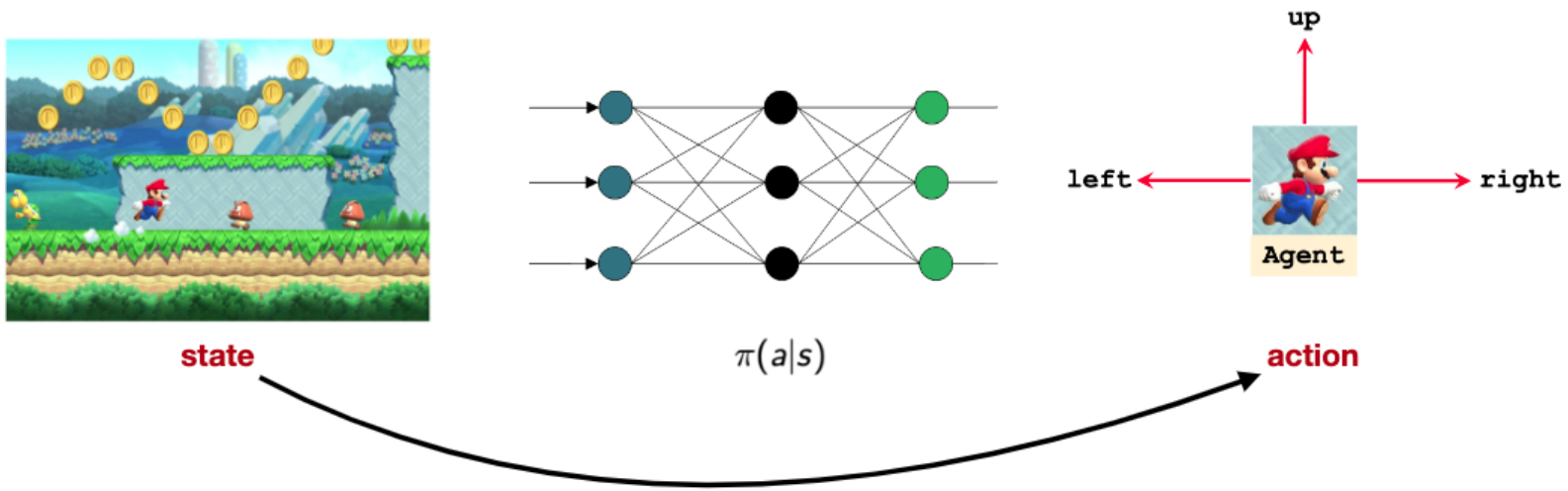


Major Components of an RL Agent

- An RL agent may contain one or more these components:
  - **Policy** : agent's behaviour function
  - **Value function** : how good is each state and/or action
  - **Model** : agent's representation of the environment

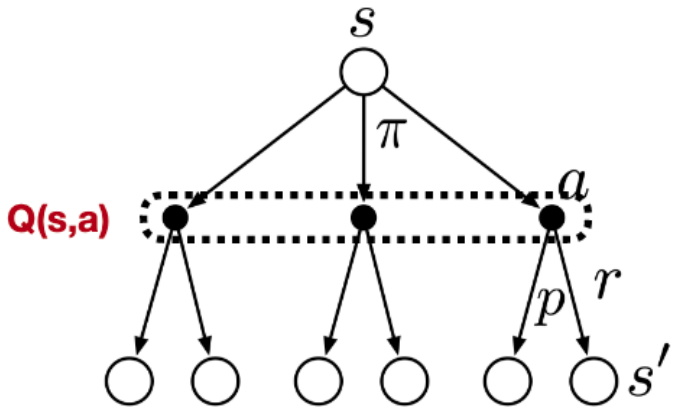
Policy

- A policy is the agent's behavior
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = P(A_t = a|S_t = s)$



Major Components of an RL Agent

- An RL agent may contain one or more these components:
  - **Policy** : agent's behaviour function
  - **Value function** : how good is each state and/or action
  - **Model** : agent's representation of the environment



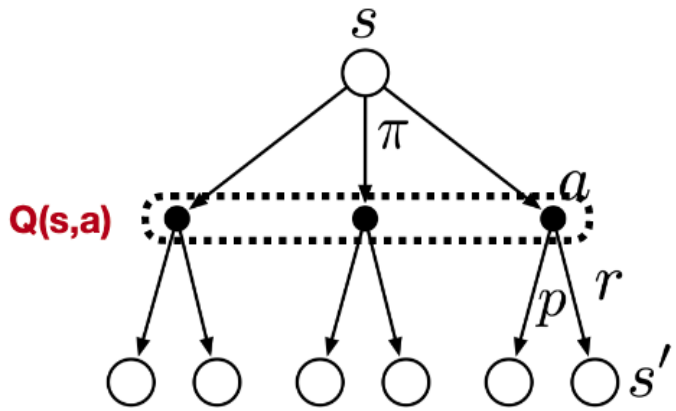
Value functions

State-value function: the state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and the following policy π:

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

Action-value function: the action-value function $q_\pi(s,a)$ is the expected return starting from state s, taking action a, and then following policy π:

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

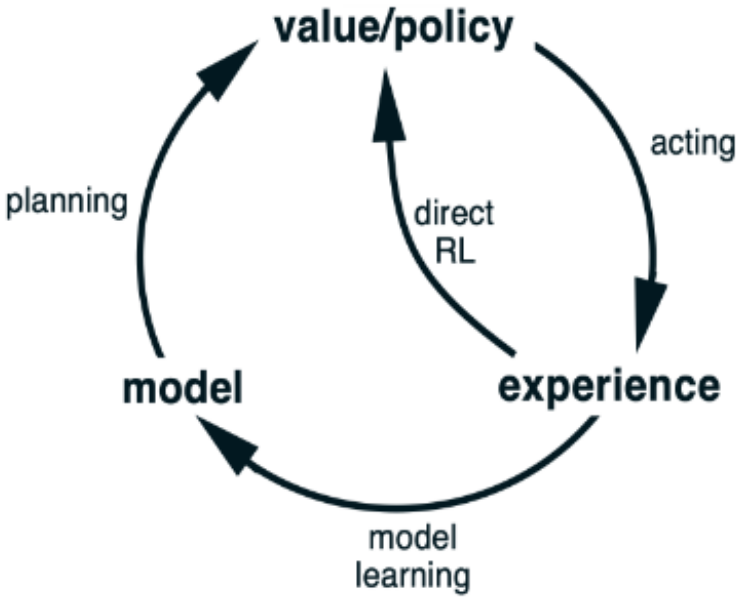


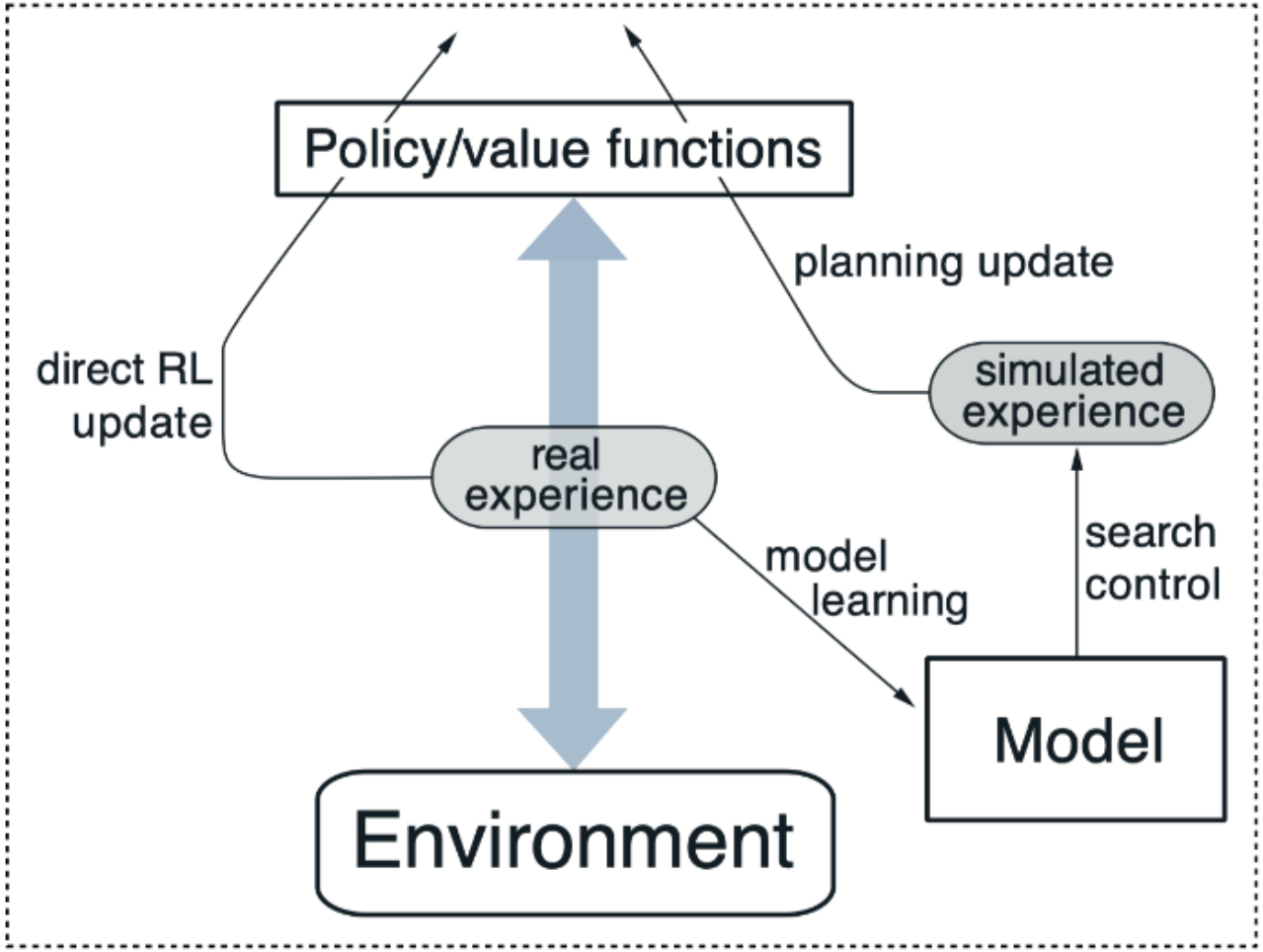
Major Components of an RL Agent

- An RL agent may contain one or more these components:
  - **Policy** : agent's behaviour function
  - **Value function** : how good is each state and/or action
  - **Model** : agent's representation of the environment

Model

- Learn model from experience
- Learn behavior in imagination
- Act in the environment





Sutton, R.S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Machine Learning Proceedings, 216-224, Elsevier, 1990

Summary: basic notions

- **MDP** $P^a_{ss'} = \mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a)$

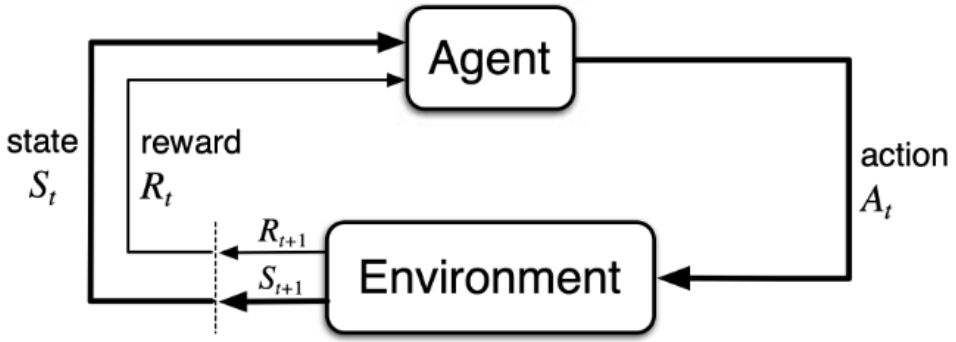
- **Return** $G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

- **Policy** $\pi(a|s) = P(A_t = a | S_t = s)$

- **Value function** $v_\pi(s) = \mathbb{E}_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s)$

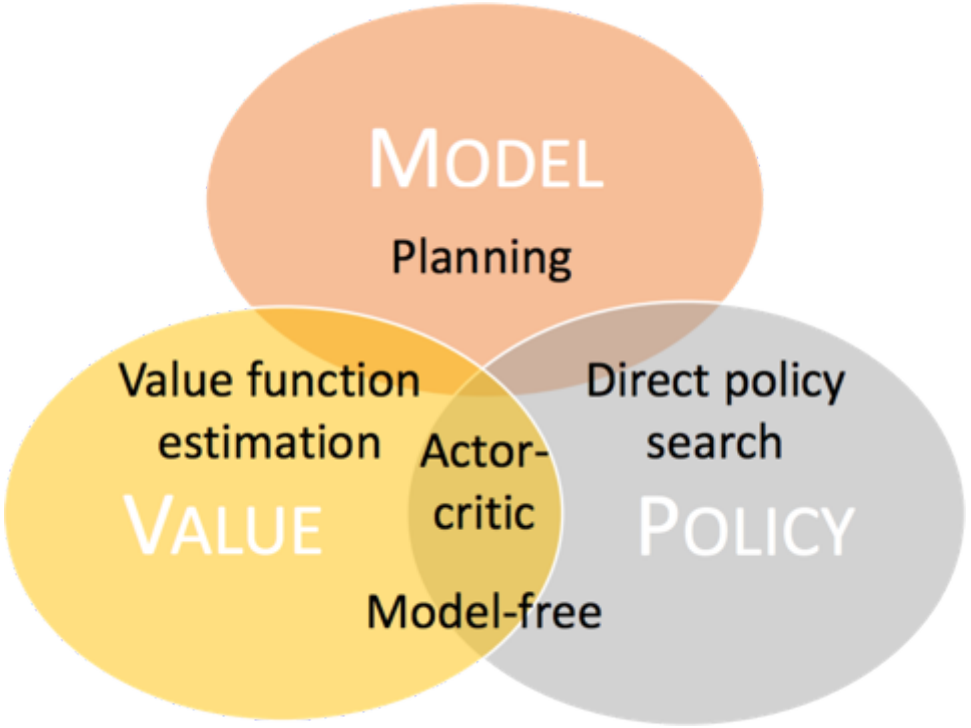
  $q_\pi(s, a) = \mathbb{E}_\pi(G_t | S_t = s, A_t = a)$

- **Model**

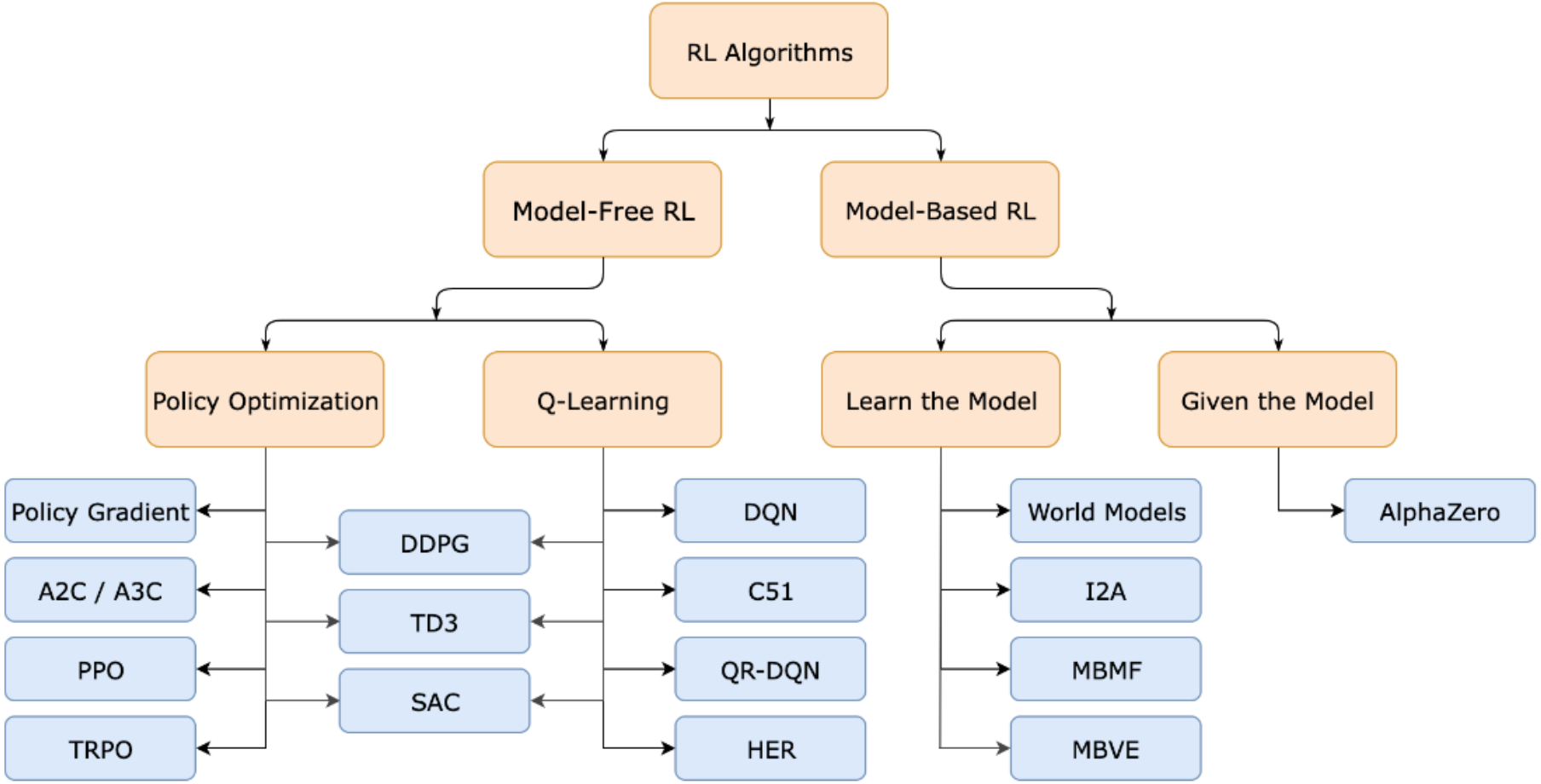


Classification

- 按照Agent对环境的理解程度
  - Model-free: Q-learning, DQN,…
  - Model-based: AlphaGo
- 按照学习对象的划分
  - Policy based: REINFORCE
  - Value based: DQN
  - Policy-value: AlphaGo
  - Actor-critic: A2C→PPO



Classification



https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

Outline

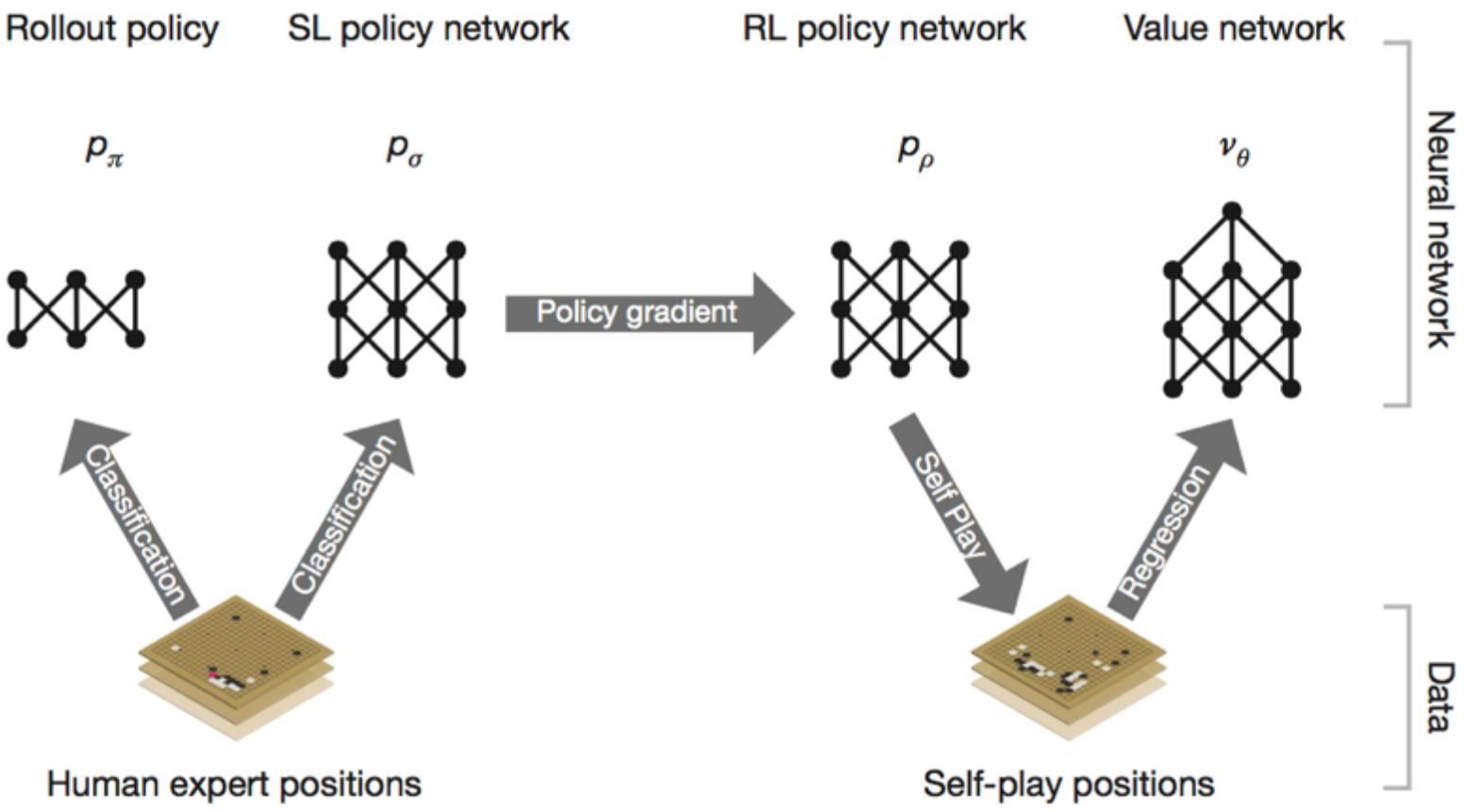
- Basic Notions
- AlphaGo
- Deep Q-Learning
- Reinforcement Learning and Causality
- World Models

Alpha Go

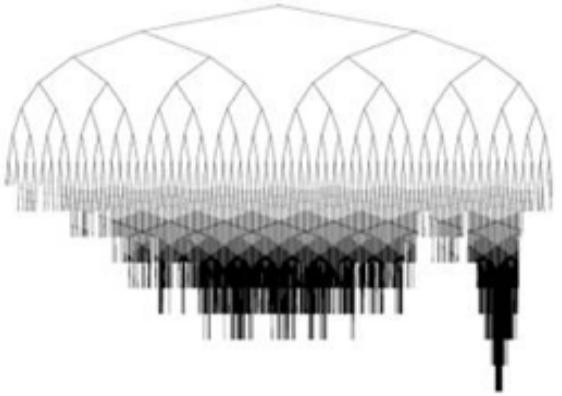
- Environment S: configuration
- Action of Agent: next move
- Transition of Environments: Depend on the opponent player
- Feedback: win or lose

Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

Architecture of Alpha Go



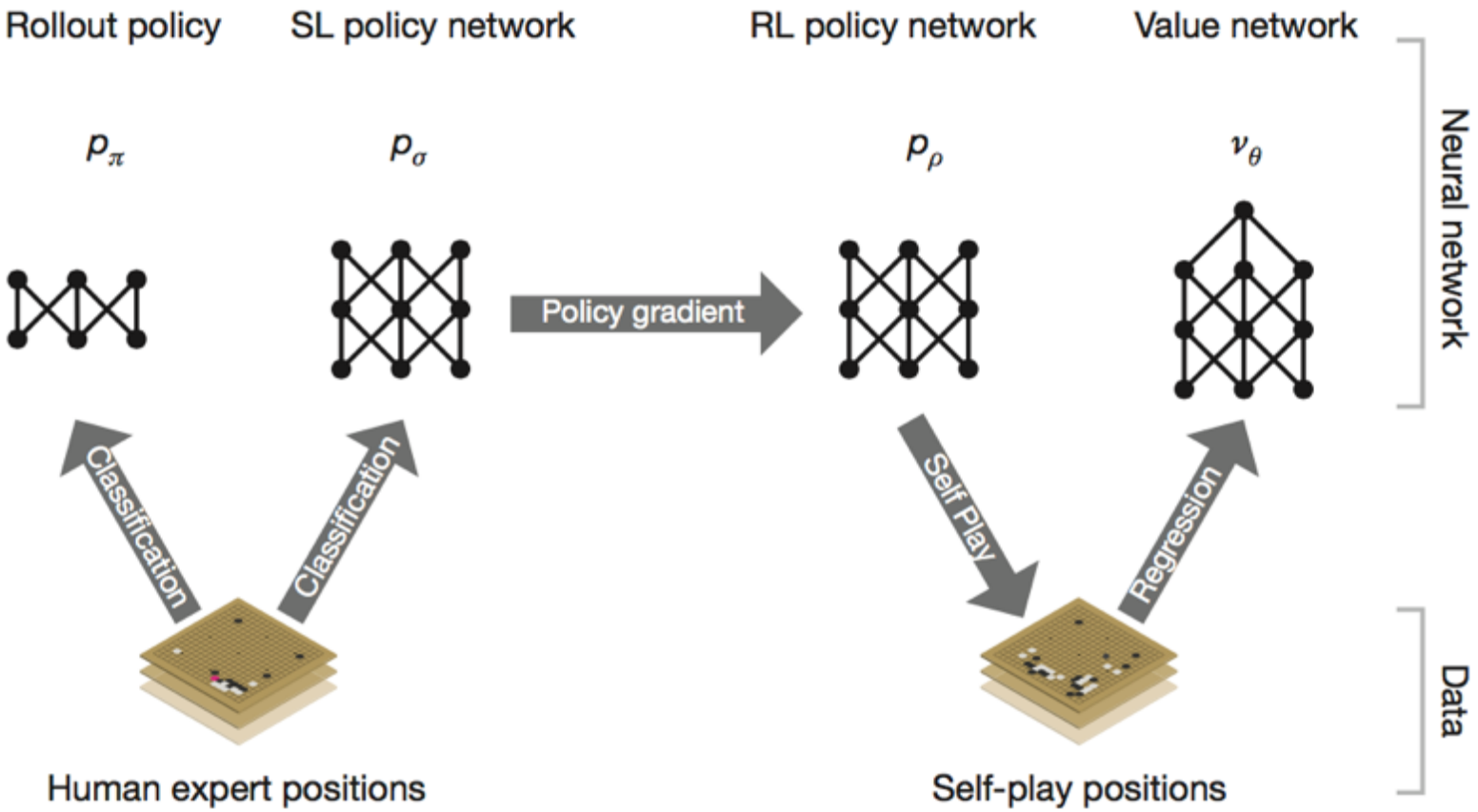
Learning Phase

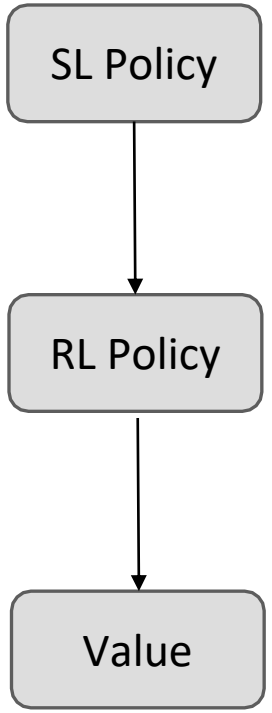


Searching Phase

Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

Architecture of Alpha Go

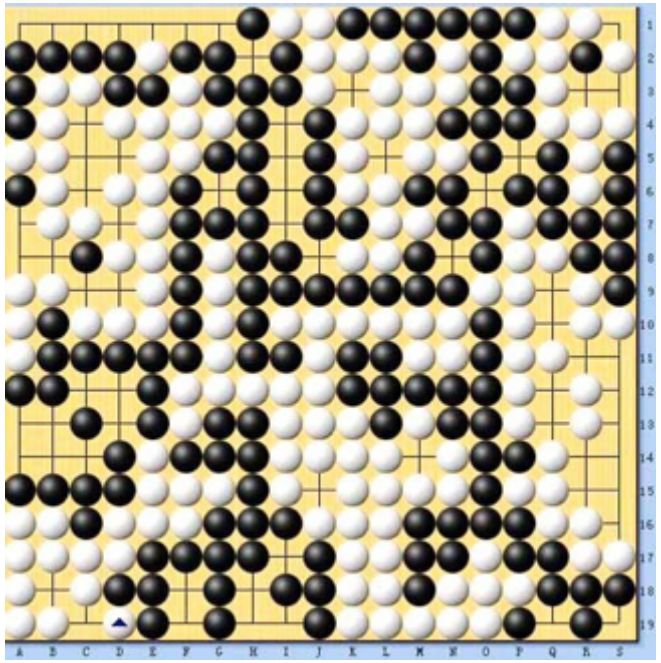




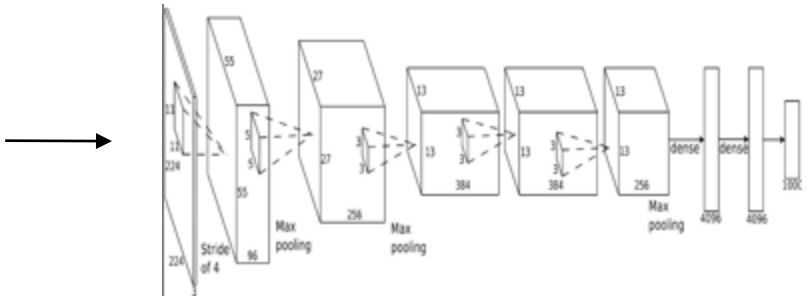
Learning Phase

Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

Supervised policy network



KBS platform of players



13 layers CNN

Probability of Next Move

Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

Reinforcement Learning Policy Network

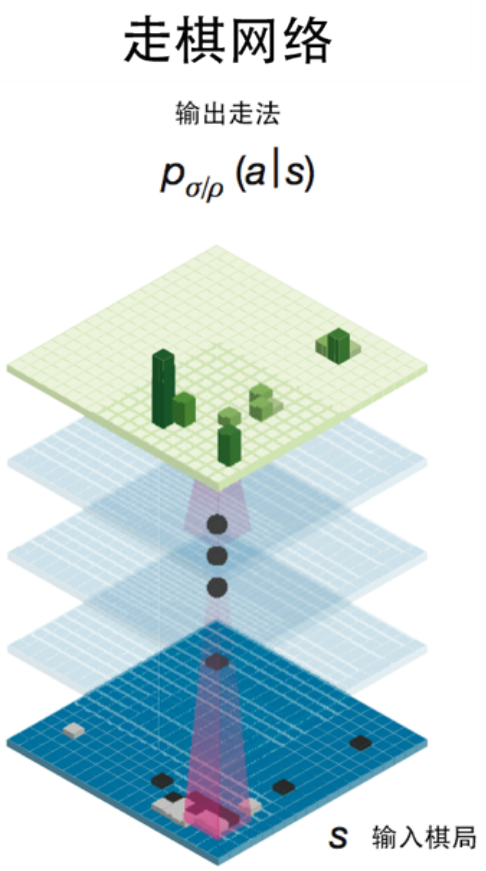
Self-play until the end: z=+-1:

- Current
- Random selected policy network

$$\Delta\rho \propto \frac{\partial \log p_\rho(a_t \mid s_t)}{\partial \rho} z_t$$

Weights

Policy gradient



Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

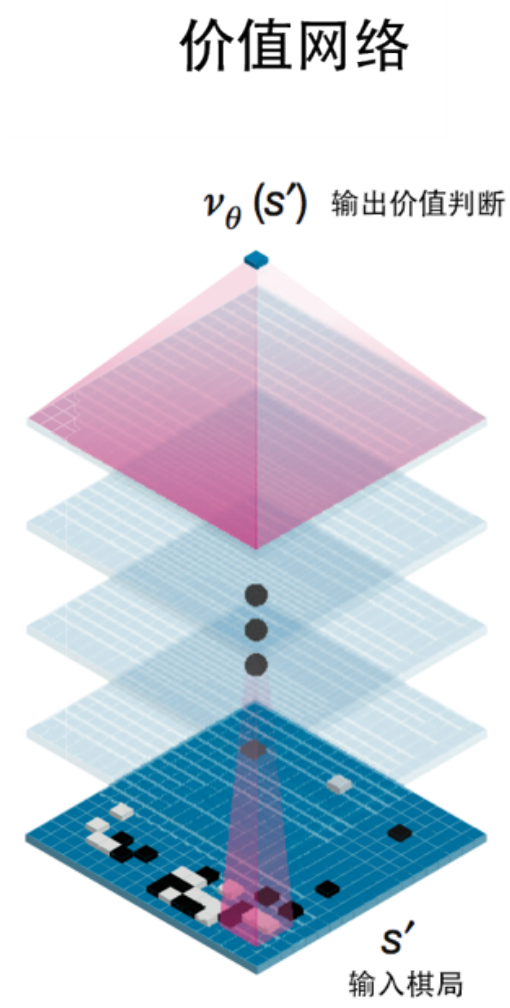
Reinforcement Learning Policy Network

- Self-play via policy network, until end: z=+-1,
- Get average value of z from 30 million games

$$v^p(s) = \mathbb{E}[z_t | s_t = s,\ a_{t...T} \sim p]$$

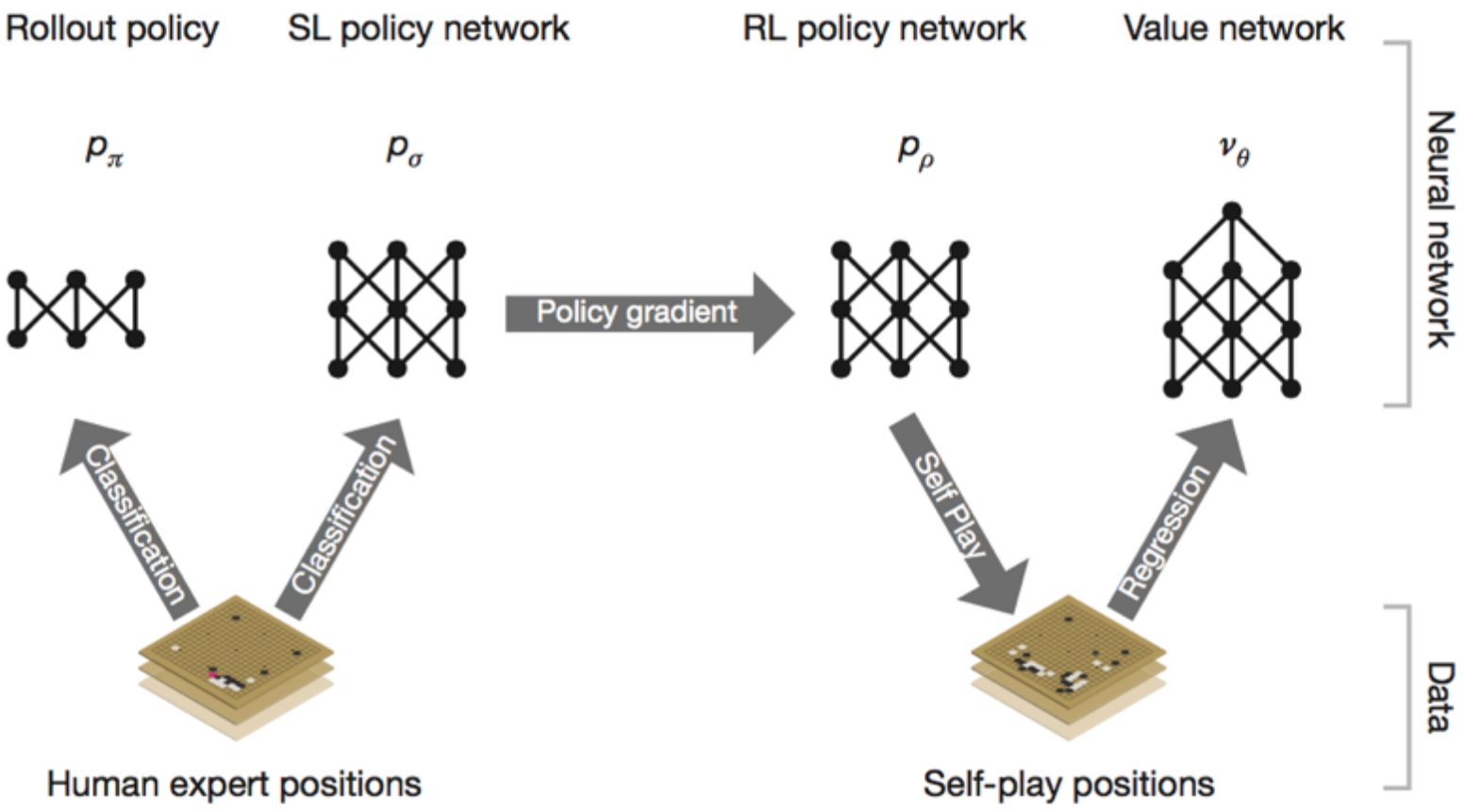
$$\Delta\theta \propto \frac{\partial v_\theta(s)}{\partial\theta}(z - v_\theta(s))$$

Weight update

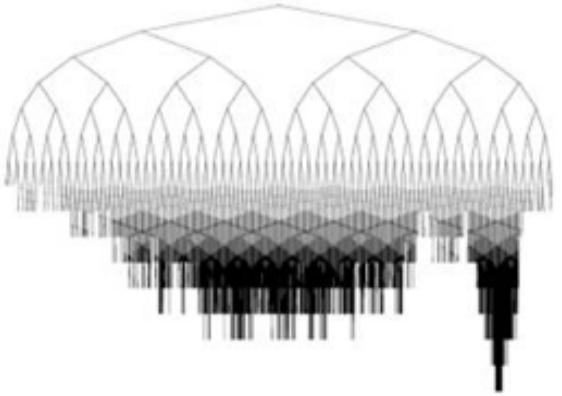


Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

Architecture of Alpha Go



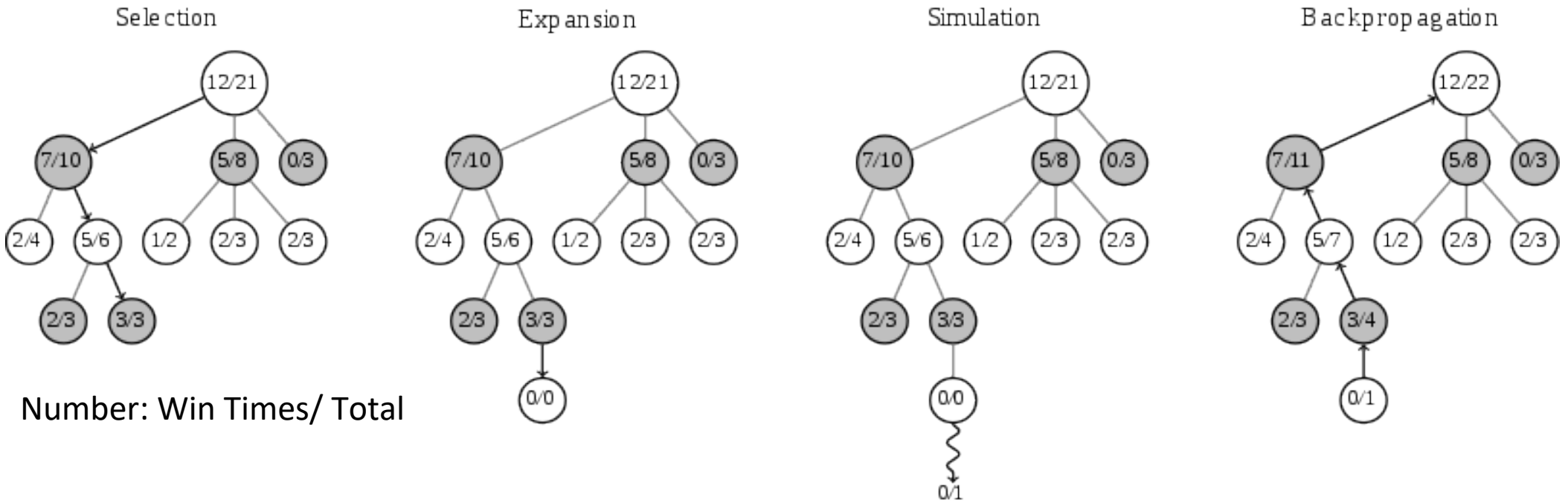
Learning Phase



Searching Phase

Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

Monte Carlo Tree Search

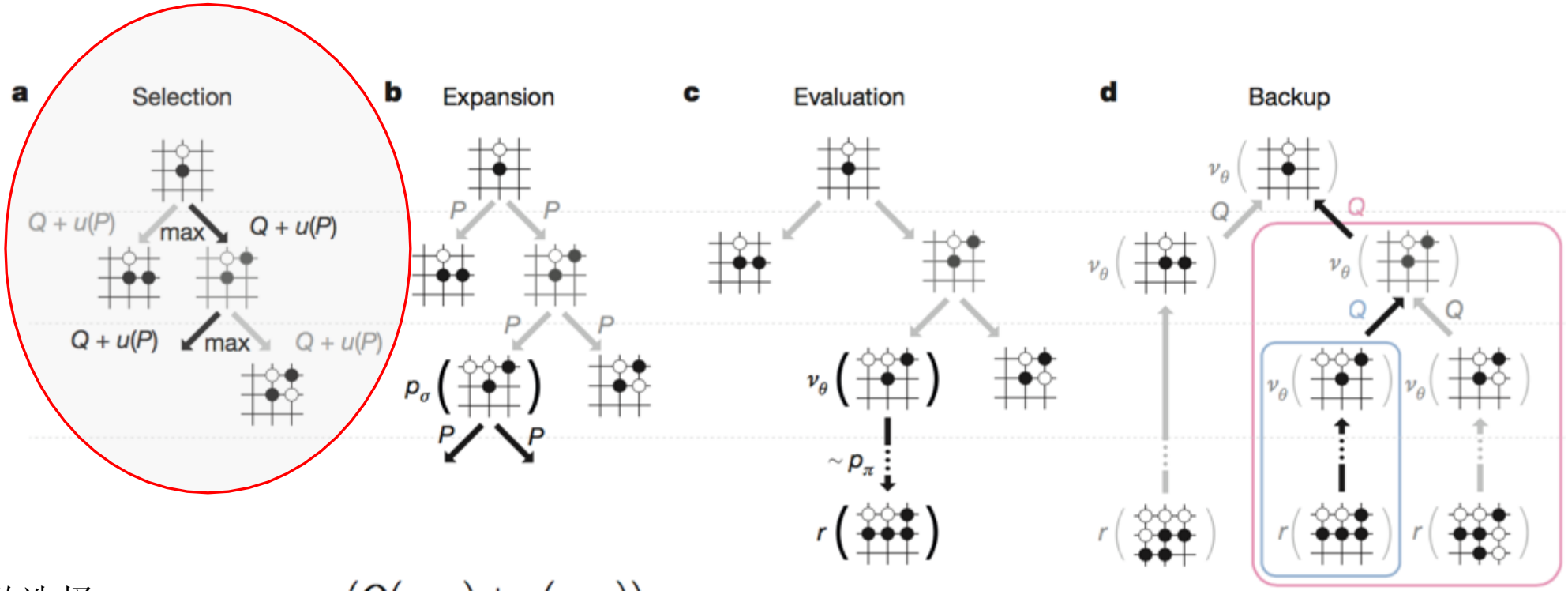


Number: Win Times/ Total

- Multi-times Simulation from one state
- Select a move with maximized value
- Fast simulation with at un-expanded node (Rollout Policy Network)
- Update all weights from bottom up

Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

Selection



做选择: $a_t = \underset{a}{\operatorname{argmax}}(Q(s_t, a) + u(s_t, a))$

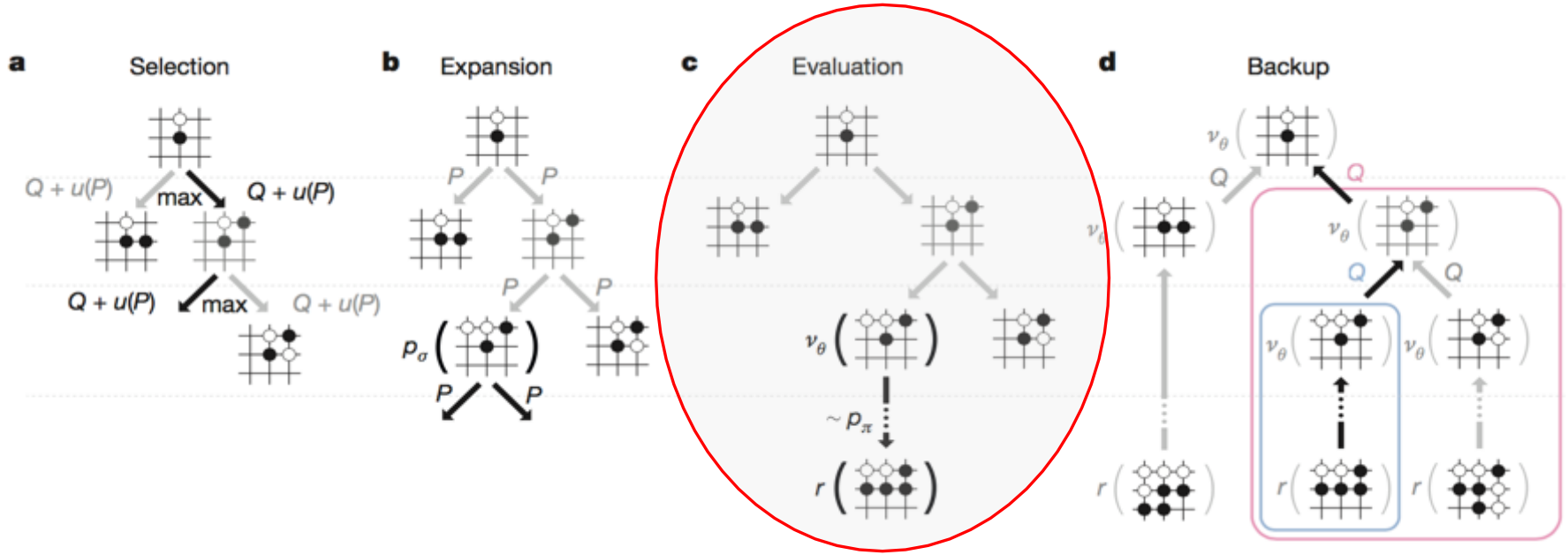
so as to maximize action value plus a bonus

$$u(s,a) \propto \frac{P(s,a)}{1+N(s,a)}$$

次数

$P(s,a) = p_\sigma(a|s)$ 用监督学习训练的走棋网络

Evaluation



For each leaf node:

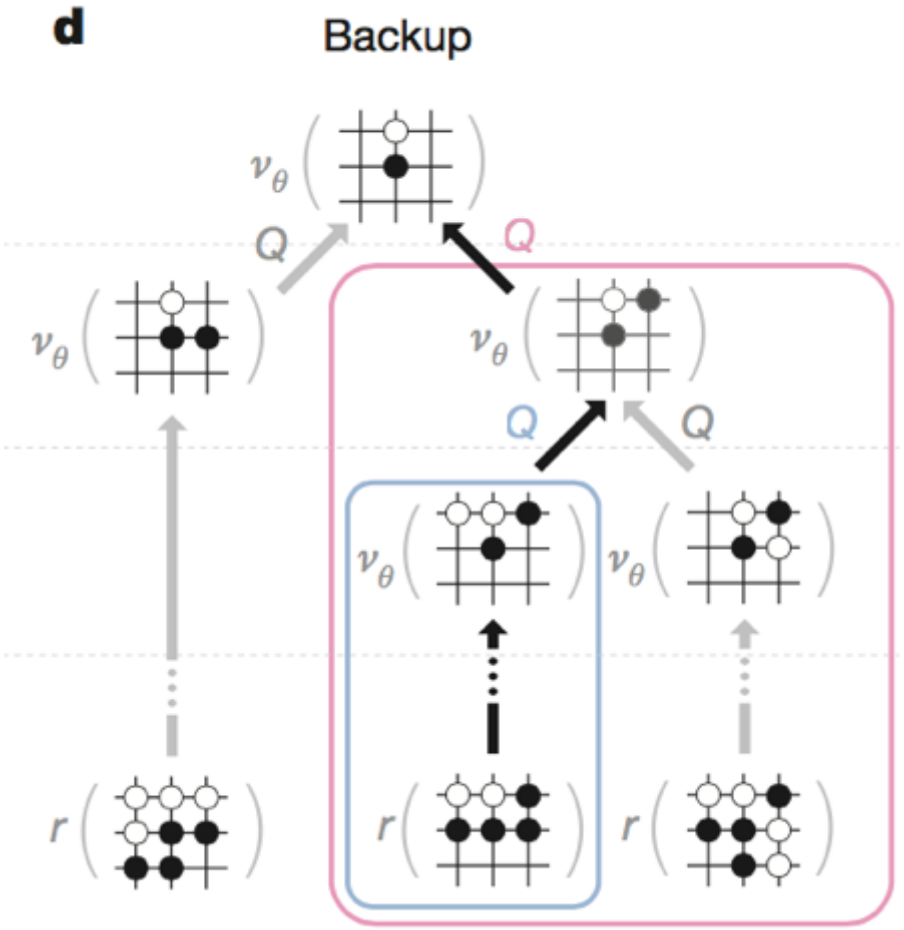
$$V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda z_L$$

Value Network

Rollout Policy Network

Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

Backup



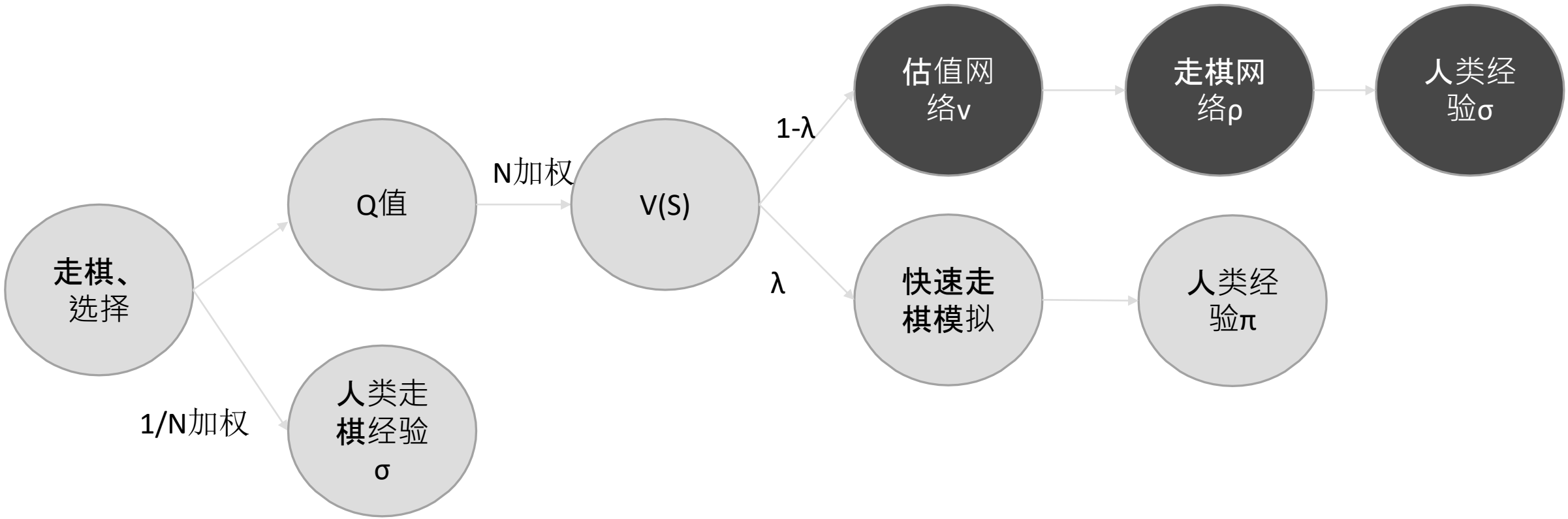
$$N(s,a) = \sum_{i=1}^{n} 1(s,a,i)$$

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i)$$

- 某一条连边被重复越多，权重越大，越相信它的估值

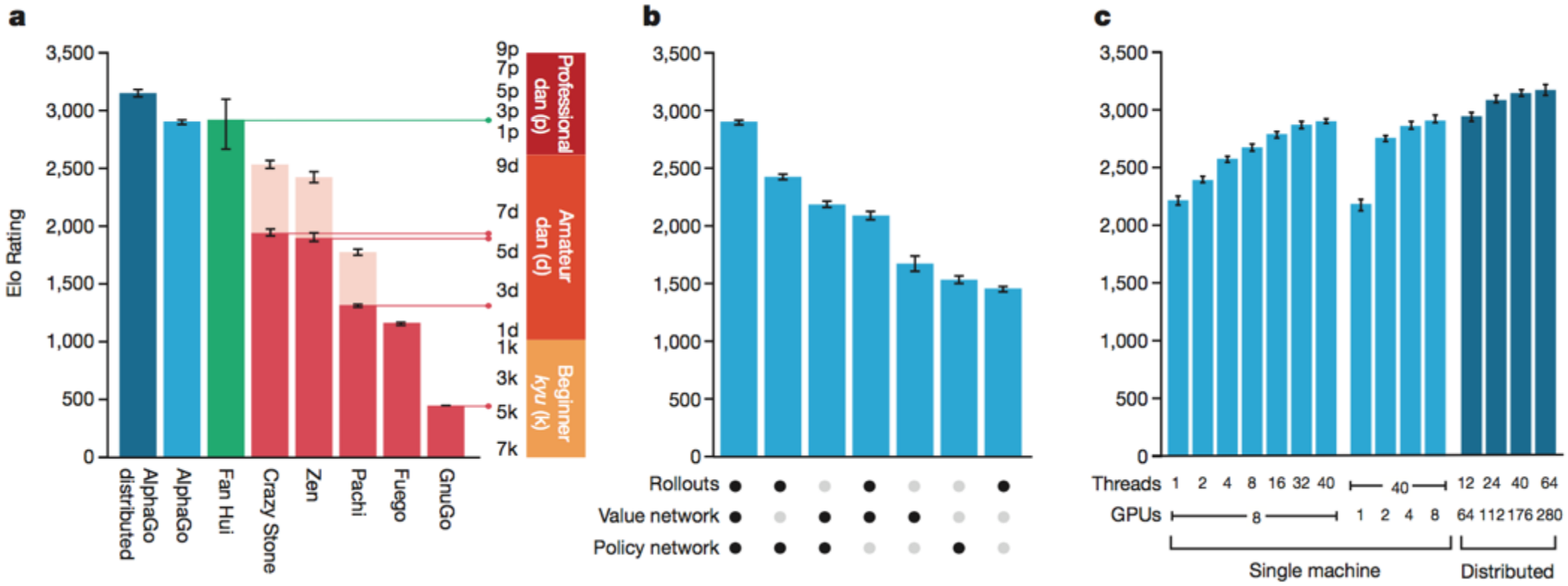
Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

Summary for Mixture



Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

Results



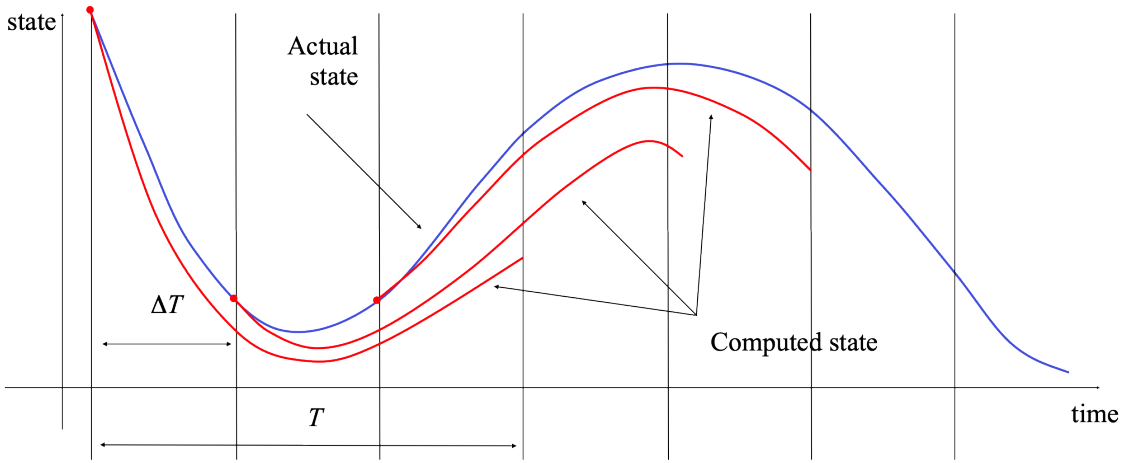
Silver, D., Huang, A., Maddison, C. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).

Outline

- Basic Notions
- AlphaGo
- Deep Q-Learning
- Reinforcement Learning and Causality
- World Models

Optimal Control and Dynamic Programming

Optimal Control



Min $V(\boldsymbol{x}, \boldsymbol{u})$

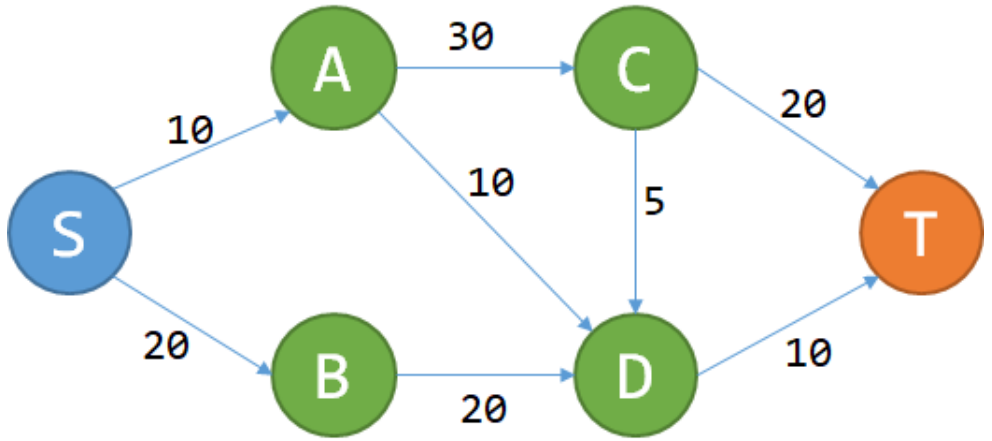
S.t. $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$

$\boldsymbol{x}(t_0)$ is given

$$V(\boldsymbol{x}, u) = \Phi\left(\boldsymbol{x}(t_f)\right) + \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)dt$$

General nonlinear control problem

Dynamic Programming



$$v_*(s) \leftarrow \max_{a \in \mathcal{A}} \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

https://www.zhihu.com/question/23995189

Temporal-difference Learning

- Goal: Learn v_π online from experience under policy π
- Incremental every-visit Monte-Carlo
  - Update value $V(S_t)$ toward actual return G_t
  - $V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$

- Simplest temporal-difference learning algorithm:
  - Update value value $V(S_t)$ toward estimated return $R_{t+1} + \gamma V(S_{t+1})$

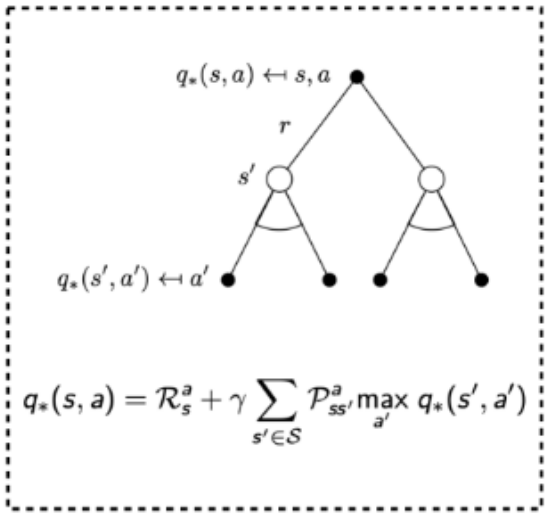
$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

  - $R_{t+1} + \gamma V(S_{t+1})$ is called the TD target
  - $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$ is called the TD error

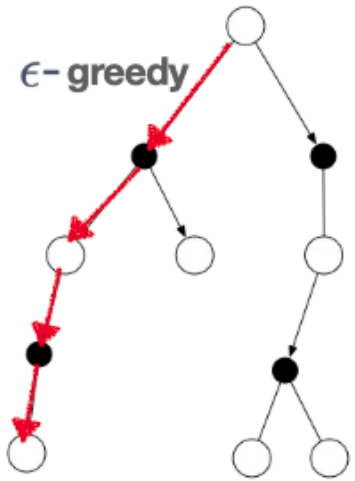
Arthur Samuel

Q-Learning Algorithm

Bellman Optimality Equation

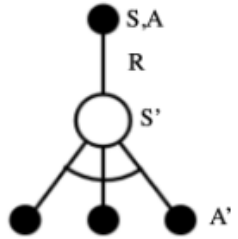


Act

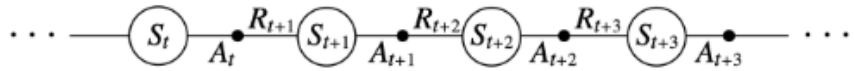


$$\pi(a|s) = \begin{cases} \epsilon/|\mathcal{A}| + 1 - \epsilon & \text{如果} a = \arg\max_{a'} Q(s,a') \\ \epsilon/|\mathcal{A}| & \text{其他动作} \end{cases}$$

Learn



$$Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma \max_{a'} Q(S',a') - Q(S,A)\right)$$



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 $Q(terminal,\cdot) = 0$

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'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
    $S \leftarrow S'$
  until S is terminal

Deep Q Network

LETTER

doi:10.1038/nature14236

Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]

The theory of reinforcement learning provides a normative account[1], deeply rooted in psychological[2] and neuroscientific[3] perspectives on animal behaviour, of how agents may optimize their control of an environment. To use reinforcement learning successfully in situations approaching real-world complexity, however, agents are confronted with a difficult task: they must derive efficient representations of the environment from high-dimensional sensory inputs, and use these to generalize past experience to new situations. Remarkably, humans and other animals seem to solve this problem through a harmonious combination of reinforcement learning and hierarchical sensory processing systems[4,5], the former evidenced by a wealth of neural data revealing notable parallels between the phasic signals emitted by dopaminergic neurons and temporal difference reinforcement learning

agent is to select actions in a fashion that maximizes cumulative future reward. More formally, we use a deep convolutional neural network to approximate the optimal action-value function

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s,\ a_t = a,\ \pi\right],$$

which is the maximum sum of rewards r_t discounted by γ at each time-step t, achievable by a behaviour policy $\pi = P(a|s)$, after making an observation (s) and taking an action (a) (see Methods)[19].

Reinforcement learning is known to be unstable or even to diverge when a nonlinear function approximator such as a neural network is used to represent the action-value (also known as Q) function[20]. This instability has several causes: the correlations present in the sequence of observations, the fact that small updates to Q may significantly change

Human-level control through deep reinforcement learning
https://www.nature.com/articles/nature14236

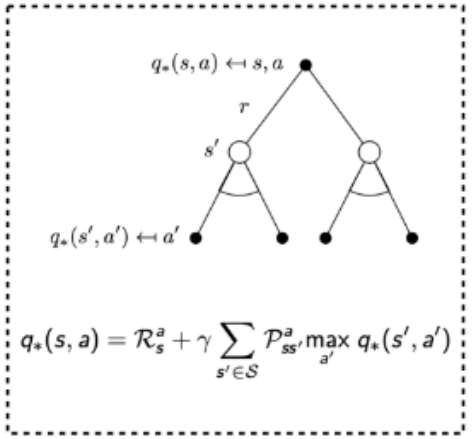
Deep Q Network

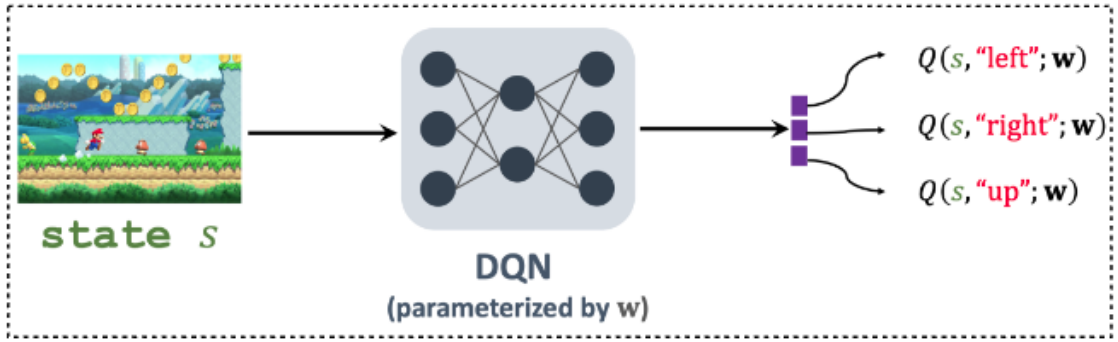
In Q-learning, we use temporal difference to update the target incrementally

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

Extend Q-learning to deep Q network, DQN

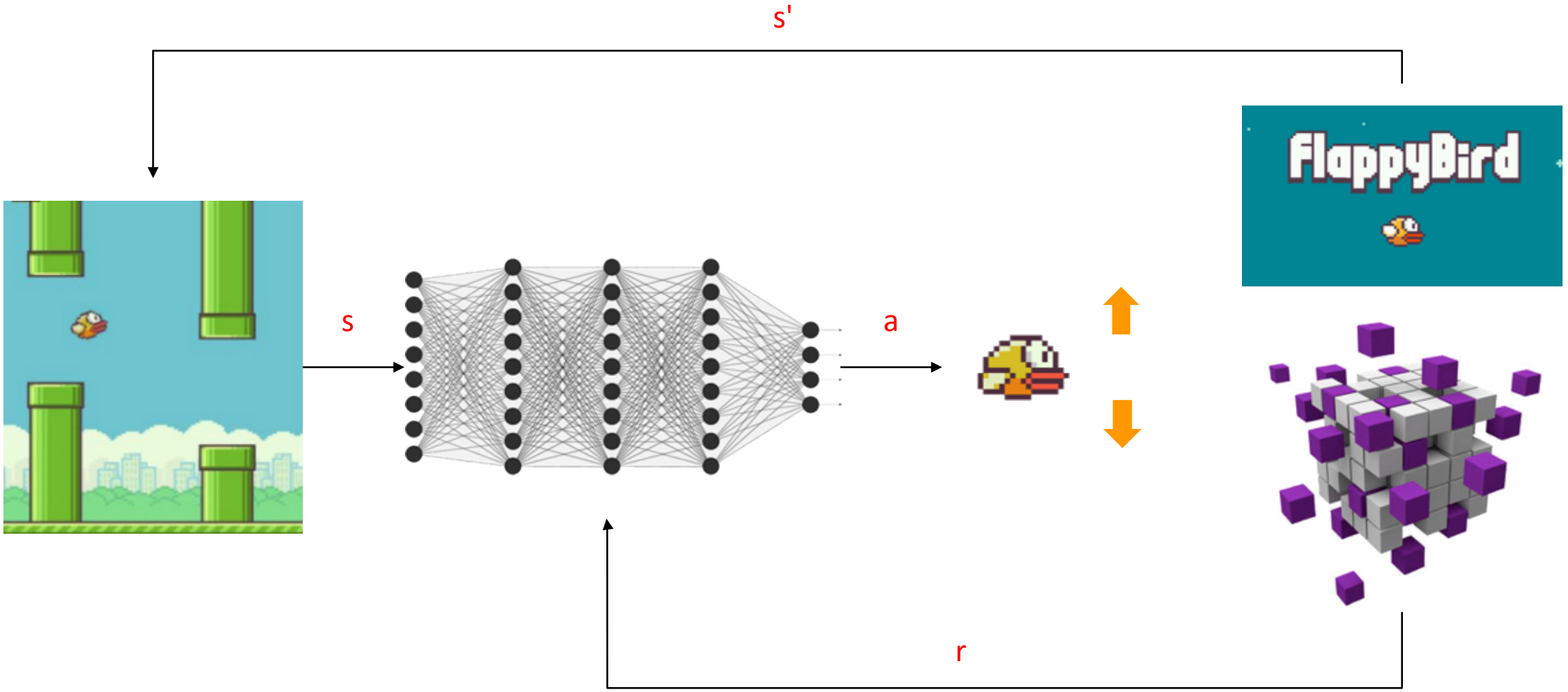
$$\omega^* = \arg\min_{\omega} \frac{1}{2N} \sum_{i=1}^{N} \left[Q_\omega(s_i,a_i) - \left(r_i + \gamma \max_{a'} Q_\omega(s_i',a') \right) \right]^2$$





Human-level control through deep reinforcement learning
https://www.nature.com/articles/nature14236

Deep Q Network



Human-level control through deep reinforcement learning
https://www.nature.com/articles/nature14236