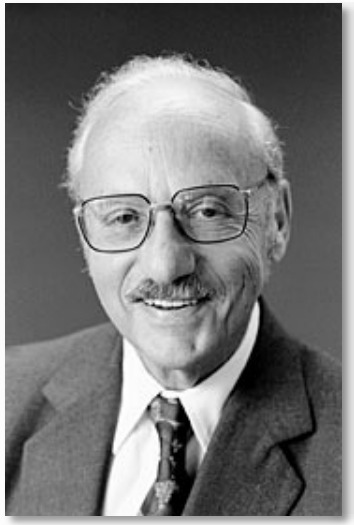
EPFL Summer School on Data Science, Optimization and Operations Research
August 15-20, 2021

Link for all lecture slides: https://odi.inf.ethz.ch/zinal

# Lecture 1: Introduction to Reinforcement Learning

Niao He, D-INFK, ETH Zurich

# The Era of Data, Uncertainties, Decisions



> *Those of us who were doing the planning right from the very beginning understood that the real problem was to be able to do planning under uncertainty. This, I feel, is the real field we should all be working on.*

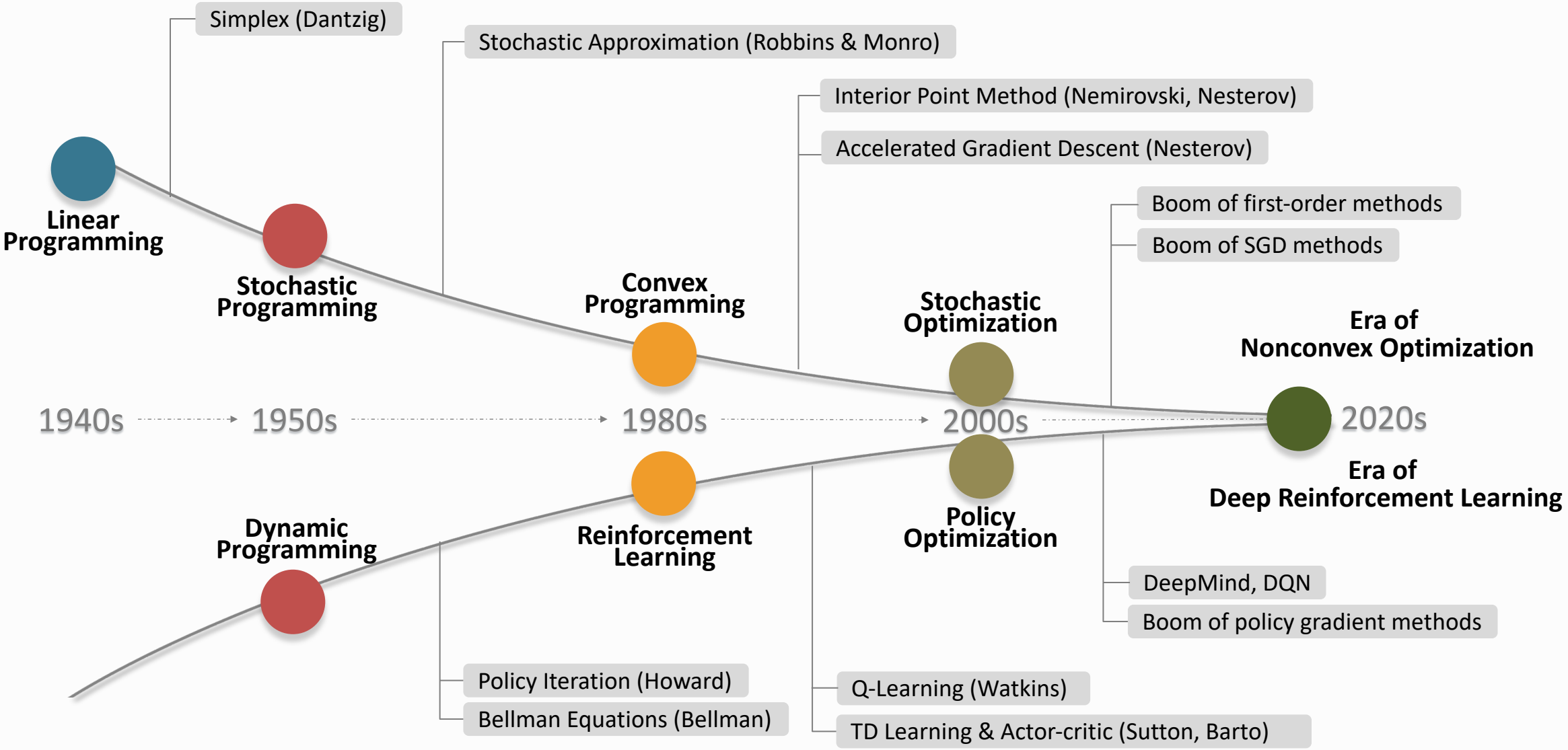
*G. B. Dantzig, E-Optimization (2001)*



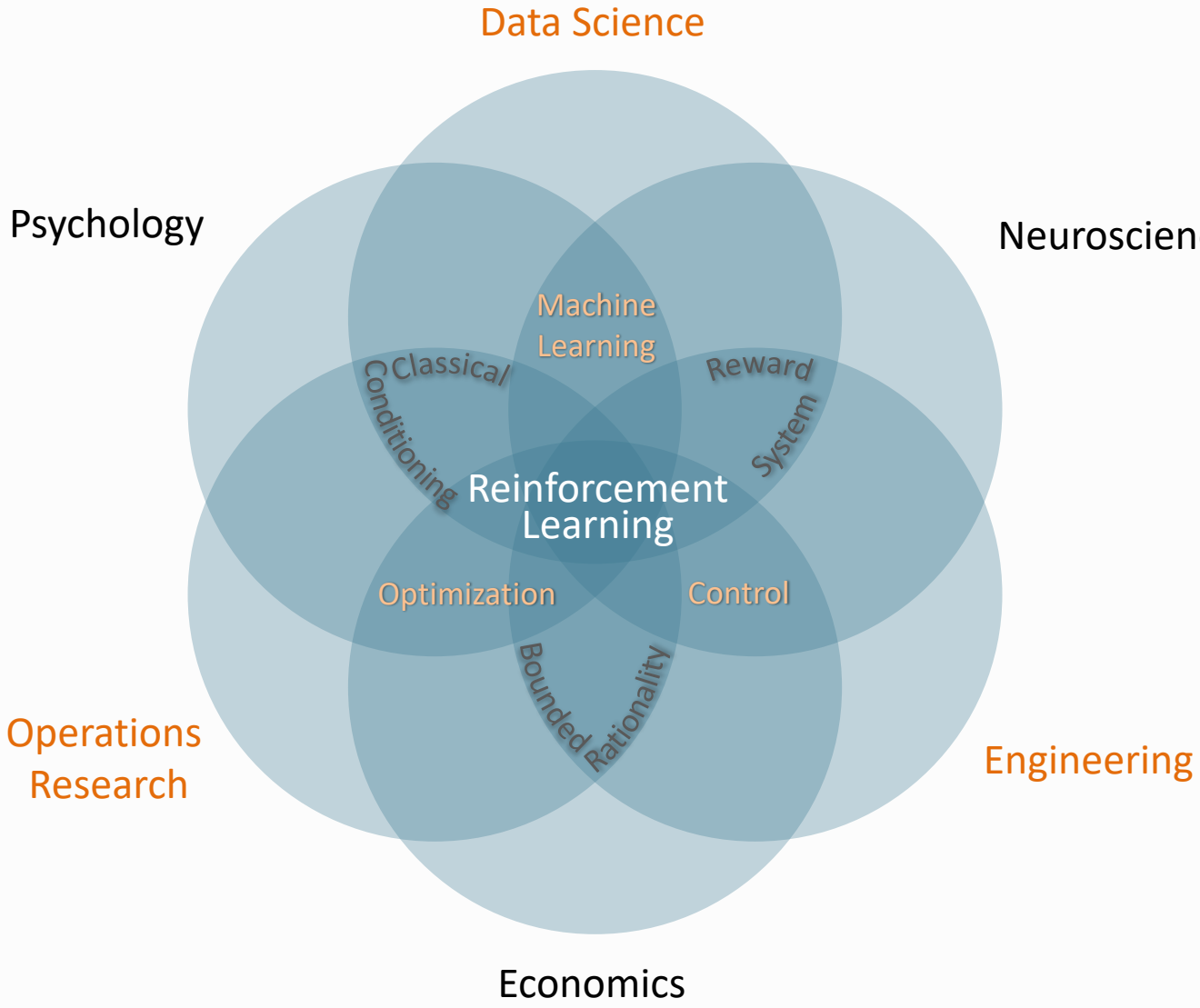
> *Part of the appeal of reinforcement learning is that it is in a sense the whole AI problem in a microcosm. The task is that of an autonomous learning agent interacting with its world to achieve a goal.*

*R. S. Sutton, Machine Learning (1992)*

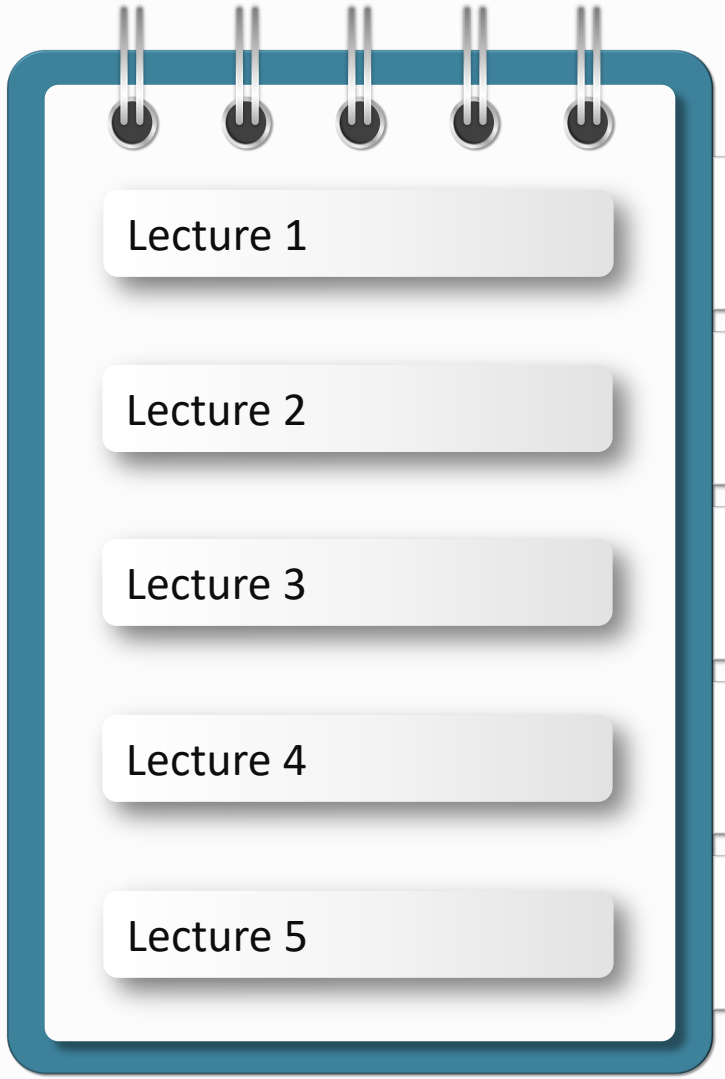
# The two journeys and crossroad



# The multi-facet of Reinforcement Learning (RL)

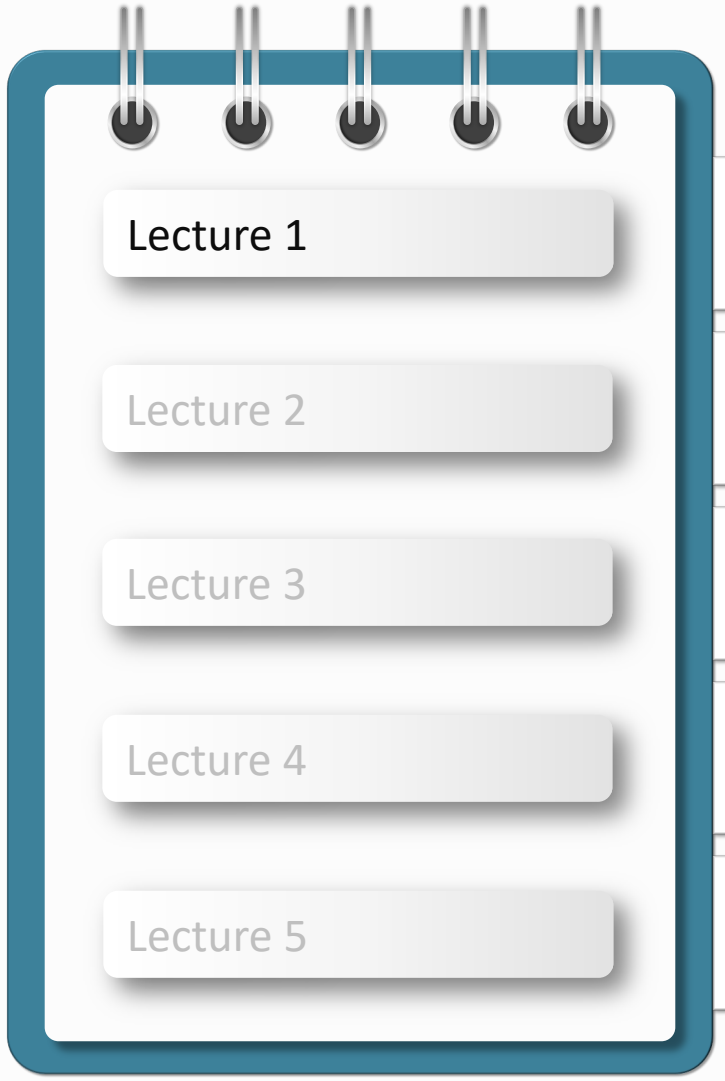


# Outline of Lecture Series



| | |
|---|---|
| Lecture 1 | Introduction to RL |
| Lecture 2 | RL from Control Perspectives<br>*- Value-based RL* |
| Lecture 3 | RL from Optimization Perspectives<br>*- Policy-based RL* |
| Lecture 4 | RL from Learning Perspectives<br>*- Deep RL* |
| Lecture 5 | RL from Game Perspectives |

# Outline of Lecture Series



Lecture 1

Lecture 2

Lecture 3

Lecture 4

Lecture 5

Introduction to RL

RL from Control Perspectives
- *Value-based RL*

RL from Optimization Perspectives
- *Policy-based RL*

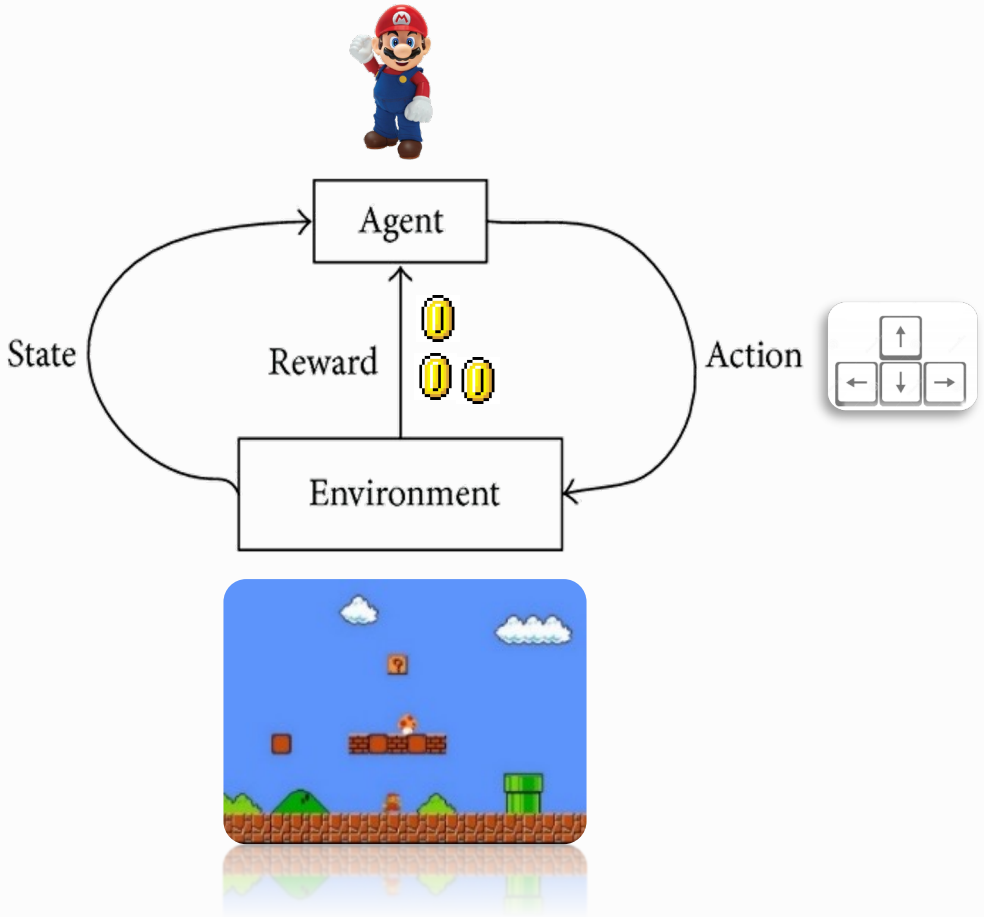
RL from Learning Perspectives
- *Deep RL*

RL from Game Perspectives

- Basics of RL
- Exact solution methods for planning
- Value-based vs policy-based methods
- Modern challenges and open questions

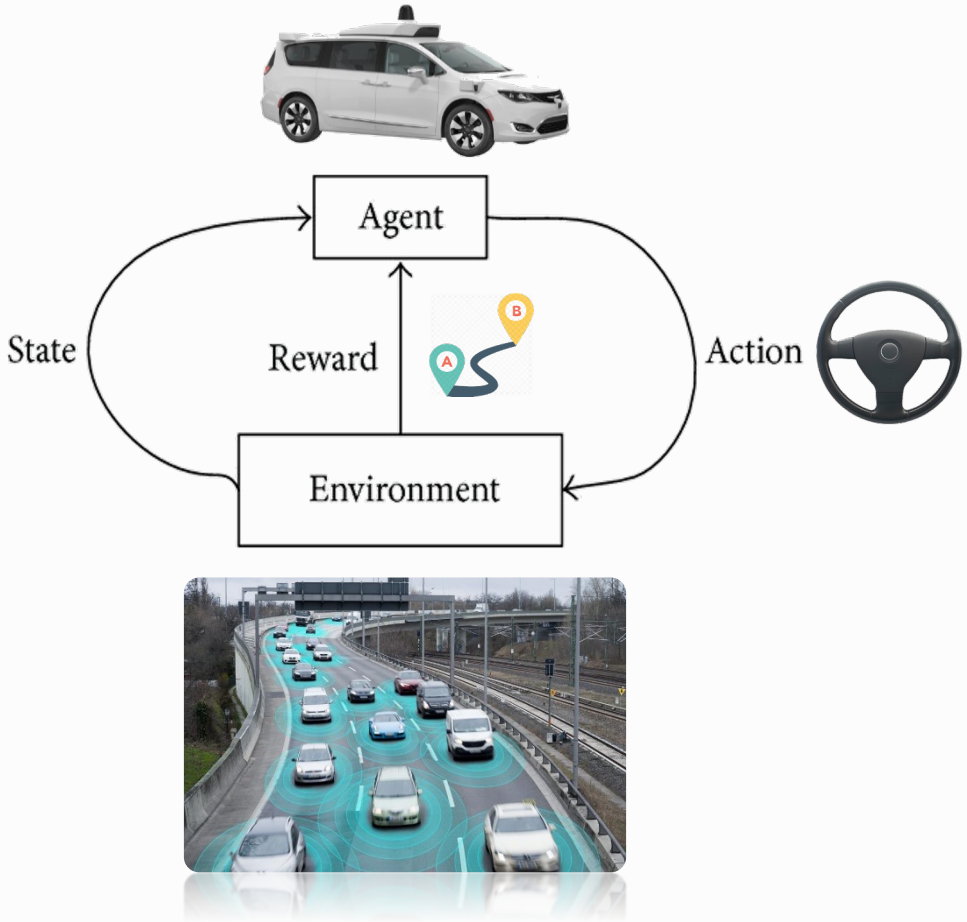
# Reinforcement Learning

An agent learns to act optimally by interacting with the uncertain environment



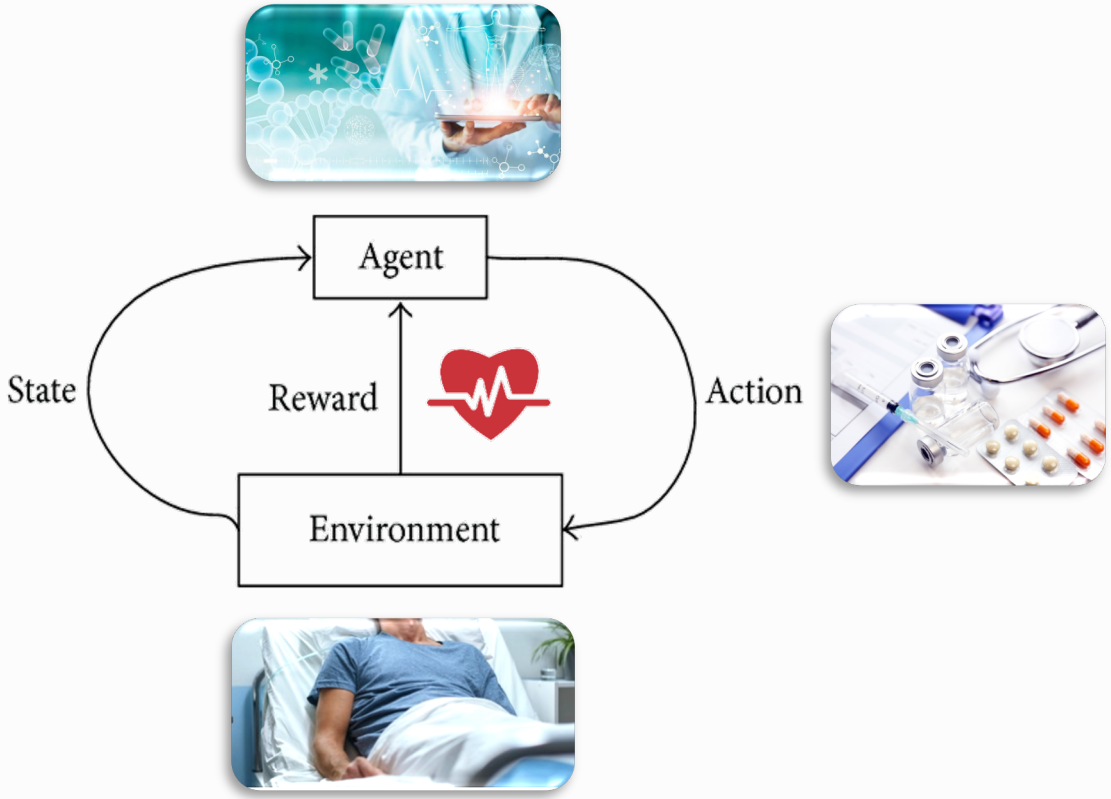
# Reinforcement Learning

An agent learns to act optimally by interacting with the uncertain environment



# Reinforcement Learning

An agent learns to act optimally by interacting with the uncertain environment

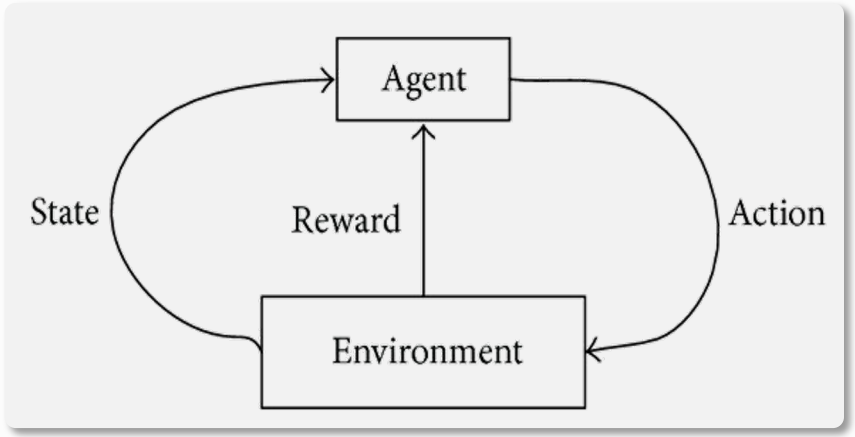


# Reinforcement Learning

An agent learns to act by interacting with the environment

Unknown Environment

Nonstationary Data




Delayed feedback

Trial & Error

# RL in the limelight

2015

Atari Game from DeepMind:
- high-dimensional input space



2017



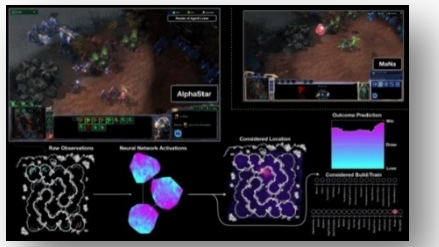

AlphaGo/ AlphaGo Zero/ Alpha Zero:
- self learning & large-scale training

2018

Dota 2 w/ OpenAI Five



2019



StarCraft II w/Alpha Star

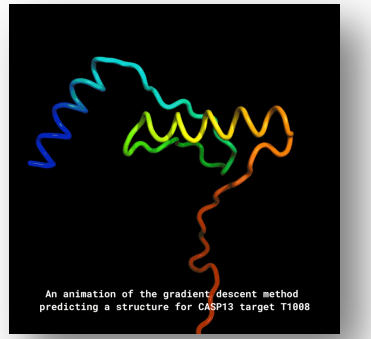
- real-time strategy

2019

Rubik's Cube from OpenAI:
- simulation to real robotic arms



2020




AlphaFold:
- 3D Protein Structures

# RL progress in practice




source: unite.ai



source: therobotreport.com



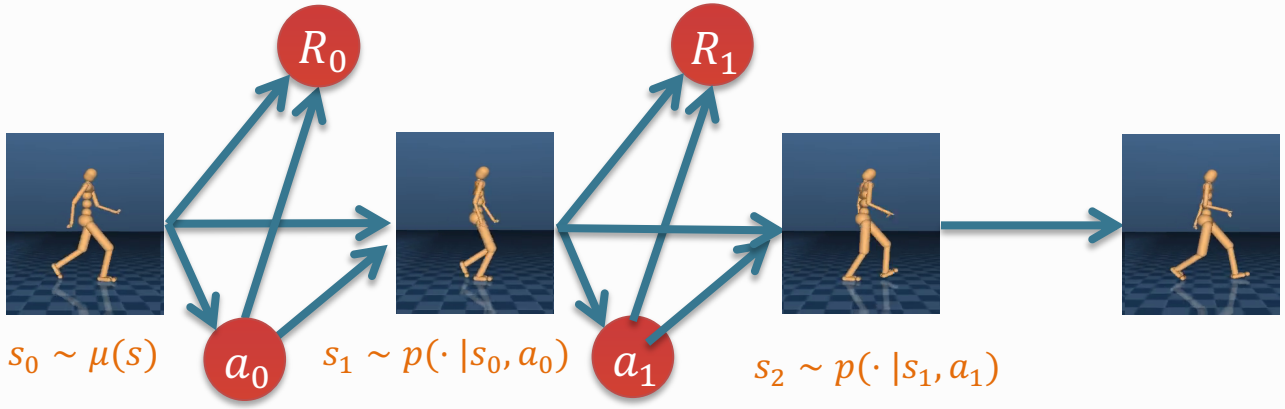
source: ieor8100.github.io




source: tradingmarket

# Markov Decision Processes

- **MDP** $(S, \mathcal{A}, P, R, \mu, \gamma)$
  - State $(S)$: a (finite) set of states
  - Action $(\mathcal{A})$: a (finite) set of actions
  - Probability transition matrix $(P_{ss'}^{a})$: $P(s_{t+1} = s' | s_t = s, a_t = a)$
  - Reward function $(R)$: the immediate reward $R(s, a)$
  - Initial distribution $(\mu)$: initial state $s_0 \sim \mu(\cdot)$
  - Discount factor $(\gamma)$: $\gamma \in [0,1]$



# Key Components

- **Policy ($\pi$):** the agent's behavior strategy
  - Deterministic policy: $\pi$: $S \to \mathcal{A},\ a = \pi(\cdot|s)$
  - Stochastic policy: $\pi$: $S \to \Delta(\mathcal{A}),\ a \sim \pi(\cdot|s)$

- **Value function**: how good is each state and/or action
  - Infinite horizon

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_t \sim \pi(s_t), s_{t+1} \sim P(\cdot|s_t, a_t)\right]$$

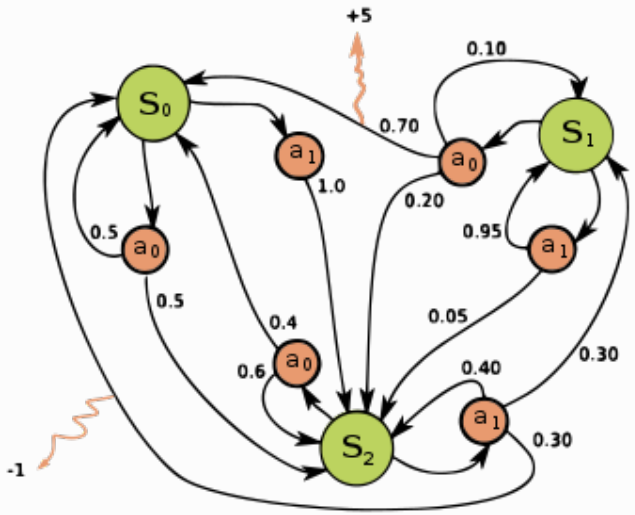
$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | (s_0, a_0) = (s, a), a_t \sim \pi(s_t), s_{t+1} \sim P(\cdot|s_t, a_t)\right]$$

  - Finite horizon / average reward

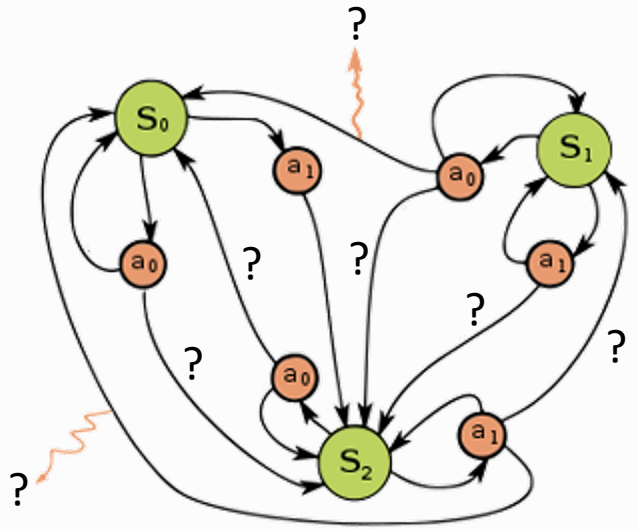
- **Model:** representation of the environment
  - Transition dynamics $P_{ss'}^{a} = P(s_{t+1} = s' | s_t = s, a_t = a)$ and reward $R(s, a)$

# Planning vs. Reinforcement Learning

- **Planning**
  - Fully observable state and reward
  - Known reward distribution and transition probabilities




- **Reinforcement Learning**
  - Observable trajectories
  - Unknown reward distribution
  - Unknown transition probabilities



# Bellman Equation

Policy Evaluation



Bellman Expectation Equation

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s\right]$$
$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a\right]$$

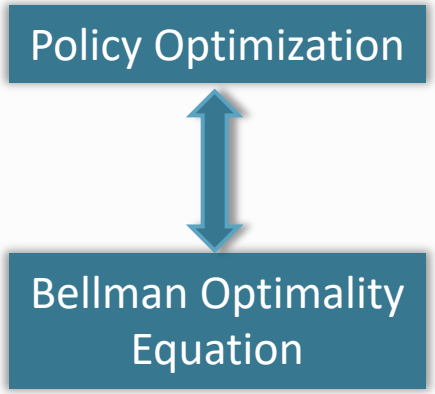
$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi(s)}\left[R(s,a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)} V^{\pi}(s')\right]$$
$$Q^{\pi}(s,a) = R(s,a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\mathbb{E}_{a'\sim\pi(s')} Q^{\pi}(s',a')\right]$$

*Proof :*

$$Q^{\pi}(s,a) = R(s,a) + \mathbb{E}_{\pi}\left[\sum_{t=1}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a\right]$$
$$= R(s,a) + \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') \cdot \gamma \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1}, a_{t+1}) | s_1 = s', a_1 = a'\right]$$
$$= R(s,a) + \gamma \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') Q^{\pi}(s',a')$$

Matrix form: $Q^{\pi} = R + \gamma P^{\pi} Q^{\pi} \iff Q^{\pi} = (I - \gamma P^{\pi})^{-1} R$

# Bellman Optimality Equation




$$V^*(s) := \max_\pi V^\pi(s), \quad Q^*(s,a) := \max_\pi Q^\pi(s,a)$$

$$V^*(s) = \max_{a \in \mathcal{A}} \left[ R(s,a) + \gamma \mathbb{E}_{s'|s,a} V^*(s') \right]$$

$$Q^*(s,a) = R(s,a) + \mathbb{E}_{s'|s,a} \left[ \gamma \max_{a' \in \mathcal{A}} Q^*(s',a') \right]$$

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s,a)$$

*Proof* ($\Leftarrow$):

$$\begin{aligned} Q^* - Q^\pi &= Q^* - (I - \gamma P^\pi)^{-1} R && \Leftarrow \quad Q^\pi = R + \gamma P^\pi Q^\pi \\ &= (I - \gamma P^\pi)^{-1} [(I - \gamma P^\pi) - (I - \gamma P^{\pi^*})] Q^* && \Leftarrow \quad Q^* = R + \gamma P^{\pi^*} Q^* \\ &= \gamma (I - \gamma P^\pi)^{-1} (P^{\pi^*} - P^\pi) Q^* \end{aligned}$$

$(I - \gamma P^\pi)^{-1} \geq 0$ (element-wise), $(P^{\pi^*} - P^\pi) Q^* \geq 0$ (element-wise)

# Fixed Point Viewpoint

- **Bellman operator:** $\mathcal{T}Q(s,a) := R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[\max_{a' \in \mathcal{A}} Q(s',a')\right]$

- **Contraction:** $\|\mathcal{T}Q_1 - \mathcal{T}Q_2\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty$

*Proof :*

$$\begin{aligned}
\|\mathcal{T}Q_1 - \mathcal{T}Q_2\|_\infty &\leq \max_{s,a} \gamma \sum_{s'} P(s'|s,a) \left|\max_{a'} Q_1(s',a') - \max_{a'} Q_2(s',a')\right| \\
&\leq \gamma \max_{s'} |\max_{a'} Q_1(s',a') - \max_{a'} Q_2(s',a')| \\
&\leq \gamma \max_{s',a'} |Q_1(s',a') - Q_2(s',a')| \\
&\leq \gamma ||Q_1 - Q_2||_\infty
\end{aligned}$$

- **Fixed Point Interpretation:**

$$\mathcal{T}Q^* = Q^*$$

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^{\pi^*}(s,a)$$

# Planning – Dynamic Programming

## Value Iteration

$$Q_{t+1} = \mathcal{T} Q_t$$

- Linear convergence:

$$\|Q_{t+1} - Q^*\|_\infty \leq \gamma^t \|Q_0 - Q^*\|_\infty$$

- Per-iteration complexity: $S^2 A$

## Policy Iteration

$$\pi_{t+1}(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q^{\pi_t}(s, a), \forall s$$

- Linear convergence:

$$\|Q^{\pi_{t+1}} - Q^*\|_\infty \leq \gamma^t \|Q^{\pi_0} - Q^*\|_\infty$$

- Per-iteration complexity: $S^3 + S^2 A$

# Linear Programming Formulation

[Puterman,1994]

- Primal LP:

$$\min_{V \in \mathbb{R}^{|\mathcal{S}|}} (1-\gamma)\langle \mu_0, V \rangle$$
$$\text{s.t. } V(s) \geq r(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s'), \forall s, a$$

- Dual LP:

$$\max_{\rho \in \mathbb{R}_+^{|\mathcal{S}|\times|\mathcal{A}|}} \langle r, \rho \rangle$$
$$\text{s.t. } \sum_a \rho(s',a) = (1-\gamma)\mu_0(s') + \gamma \sum_{s,a} \rho(s,a)P(s'|s,a), \forall s'$$

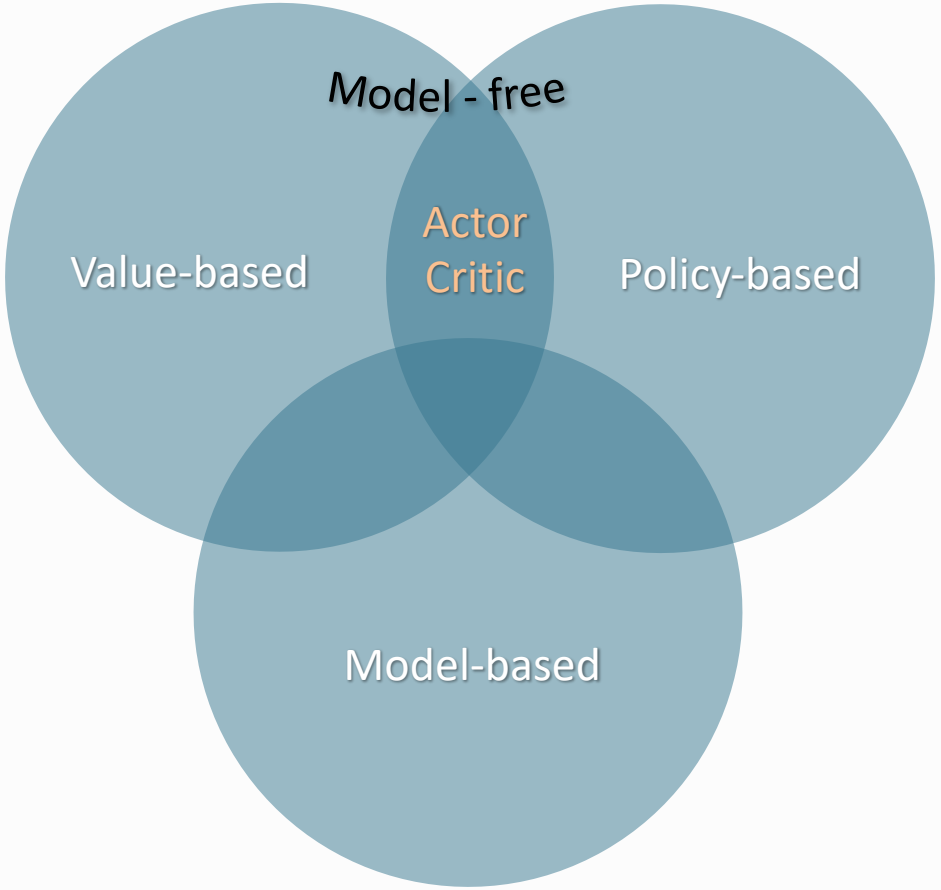
$$\pi^*(a|s) = \frac{\rho^*(s,a)}{\sum_a \rho^*(s,a)}$$

- Bilinear saddle point problem:

$$\max_{\rho \in \mathbb{R}_+^{|\mathcal{S}|\times|\mathcal{A}|}} \min_{V \in \mathbb{R}^{|\mathcal{S}|}} (1-\gamma)\langle \mu_0, V \rangle + \sum_{s,a} \rho(s,a)\delta[V](s,a)$$

$$\delta[V](s,a) := r(s,a) + \gamma \mathbb{E}_{s'|s,a}[V(s')] - V(s)$$

# Overview of Reinforcement Learning Approaches

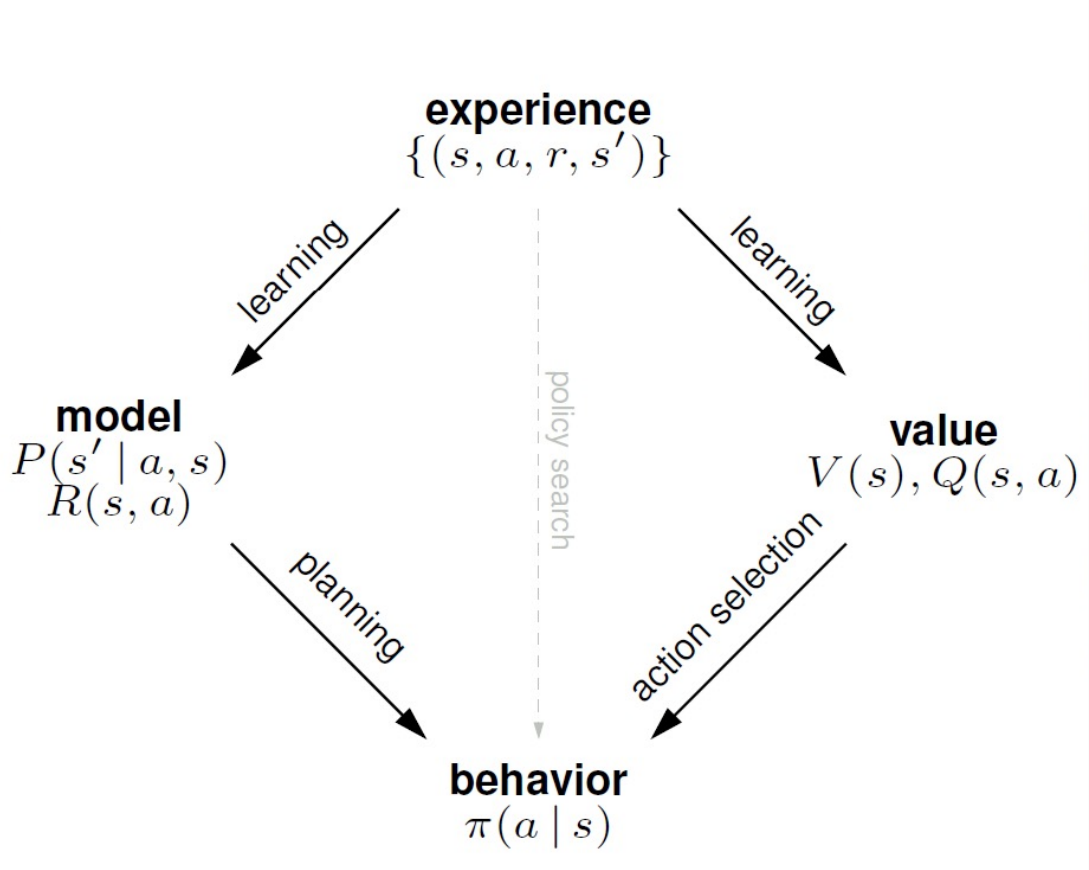



- **Value-based RL**
  - Estimate the optimal value function $Q^*(s, a)$
  - Example: Q-learning
- **Policy-based RL**
  - Search directly the optimal policy $\pi^*(\cdot\,|s)$
  - Example: Policy Gradient Method
- **Model-based RL**
  - First estimate the model $P, R$ and then do planning
  - Example: Dyna-Q algorithm

# Model-based vs Model-free Methods

- Make full use of experiences
- Can reason about model uncertainty
- More sample efficient for easy dynamics

Model-based




Model-free

- Direct and simple
- Not affected by bad model estimation
- Computationally efficient

# Value-based RL

**Goal:** Given samples $\{(s_k, a_k, s_{k+1}, r_k)\}$, want to estimate $Q^*$.

- Standard Q-learning [Watkins, 1989]:

$$Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha_k \left[r_k + \gamma \cdot \max_{a'} Q(s_{k+1}, a') - Q(s_k, a_k)\right]$$

- Double Q-learning [van Hasselt, 2010]:

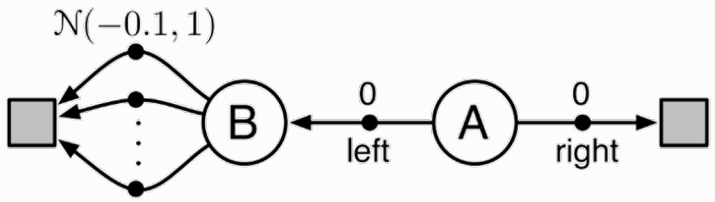
Randomly update

$$Q^A(s_k, a_k) \leftarrow Q^A(s_k, a_k) + \alpha_k \left[r_k + \gamma \cdot Q^B\left(s_{k+1}, \arg\max_a Q^A(s_{k+1}, a)\right) - Q^A(s_k, a_k)\right]$$

$$Q^B(s_k, a_k) \leftarrow Q^B(s_k, a_k) + \alpha_k \left[r_k + \gamma \cdot Q^A\left(s_{k+1}, \arg\max_a Q^B(s_{k+1}, a)\right) - Q^B(s_k, a_k)\right]$$

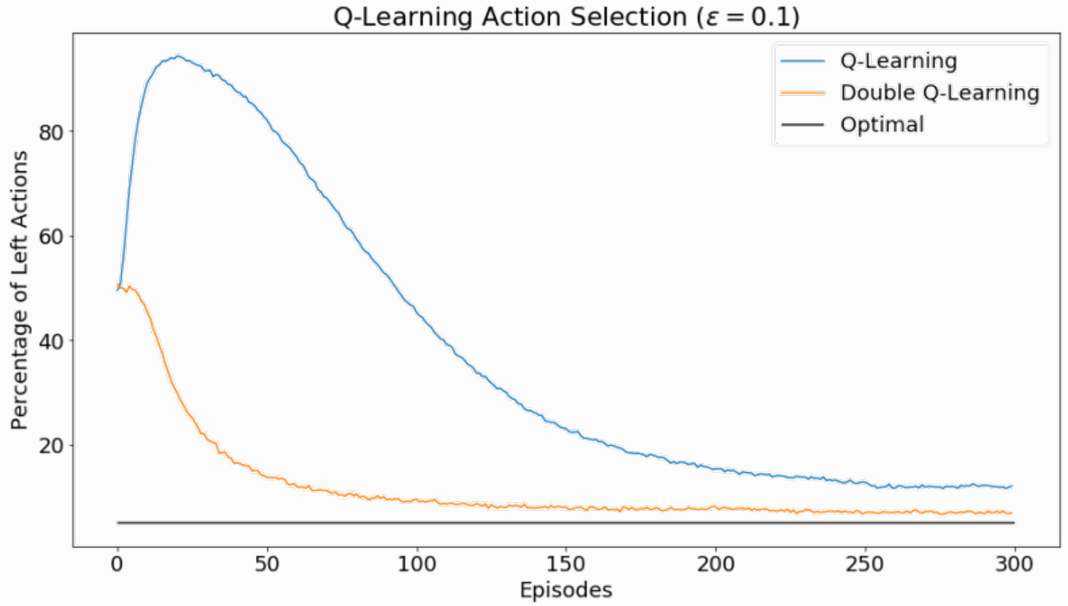
# Value-based RL

- Double Q-learning helps reduce the maximization bias.




Example

[Sutton and Barto, 2018]



# Policy-based RL

**Goal:** directly search optimal policy $\pi^*$ over policy space

$$\max_{\pi \in \Delta(\mathcal{A})^{|S|}} J(\pi) := \mathbb{E}_{s \sim \mu}[V^{\pi}(s)]$$

- Projected Policy Gradient:

$$\pi_{k+1} = \mathrm{Proj}_{\Delta(\mathcal{A})^{|S|}} \left(\pi_k + \eta_k \nabla J(\pi_k)\right)$$

- Softmax Policy Graident:

$$\pi_\theta(a|s) = \frac{\exp(\theta_{s,a})}{\sum_a \exp(\theta_{s,a})}$$

$$\theta_{k+1} = \theta_k + \eta_k \nabla_\theta J(\pi_{\theta_k})$$

# Policy Gradient

- Monte Carlo Policy gradient [William 1992]

$$\frac{\partial J(\pi_\theta)}{\partial \theta} = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}(\tau)} \left[ R(\tau) \sum_{t=0}^{\infty} \nabla \log \pi_\theta(a_t|s_t) \right]$$

$$R(\tau) := \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$$

  - Use sample trajectories $\tau = (s_0, a_0, r_0, s_1, a_1, \dots)$ under the policy
  - Score function estimator : $\nabla_\theta E_{\tau \sim p_\theta(\tau)}[f(\tau)] = E_{\tau \sim p_\theta(\tau)} f(\tau) \nabla_\theta \log p_\theta(\tau)]$
  - Suffers from high variance

- Policy gradient as weighted Q functions [Sutton 1999]

$$\frac{\partial J(\pi_\theta)}{\partial \theta} = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_\mu^{\pi_\theta}, a \sim \pi_\theta(\cdot|s)} \left[ Q^{\pi_\theta}(s,a) \nabla \log \pi_\theta(a|s) \right]$$

State-visitation distribution: $d_\mu^\pi(s) = \mathbb{E}_{s_0 \sim \mu} \left[ (1-\gamma) \sum_{k=0}^{\infty} \gamma^k P(s_k = s | s_0, \pi) \right]$

# Derivation of Policy Gradient

*Proof:*

$$\nabla_\theta V^{\pi_\theta}(s_0) = \nabla_\theta \sum_{a_0} \pi_\theta(a_0|s_0) Q^{\pi_\theta}(s_0, a_0)$$

$$= \underbrace{\sum_{a_0} \nabla_\theta \pi_\theta(a_0|s_0) Q^{\pi_\theta}(s_0, a_0)}_{:=\phi(s_0)} + \sum_{a_0} \pi_\theta(a_0|s_0) \nabla_\theta Q^{\pi_\theta}(s_0, a_0)$$

$$= \phi(s_0) + \sum_{a_0} \pi_\theta(a_0|s_0) \nabla_\theta \left[ R(s_0, a_0) + \sum_{s_1} P(s_1|s_0, a_0) V^{\pi_\theta}(s_1) \right]$$

$$= \phi(s_0) + \gamma \sum_{s_1} P(s_1|s_0, \pi_\theta) \nabla_\theta V^{\pi_\theta}(s_1)$$

$$= \phi(s_0) + \gamma \sum_{s_1} P(s_1|s_0, \pi_\theta) \left[ \phi(s_1) + \gamma \sum_{s_2} P(s_2|s_1, \pi_\theta) V^{\pi_\theta}(s_2) \right]$$

$$= \cdots$$

$$= \sum_{t=0}^{\infty} \sum_{s_t} \gamma^t P(s_t|s_0, \pi_\theta) \phi(s_t)$$

$$= \sum_{s} d_{s_0}^{\pi_\theta}(s) \sum_{a} Q^{\pi_\theta}(s, a) \nabla_\theta \pi_\theta(a|s)$$



# Actor-Critic RL

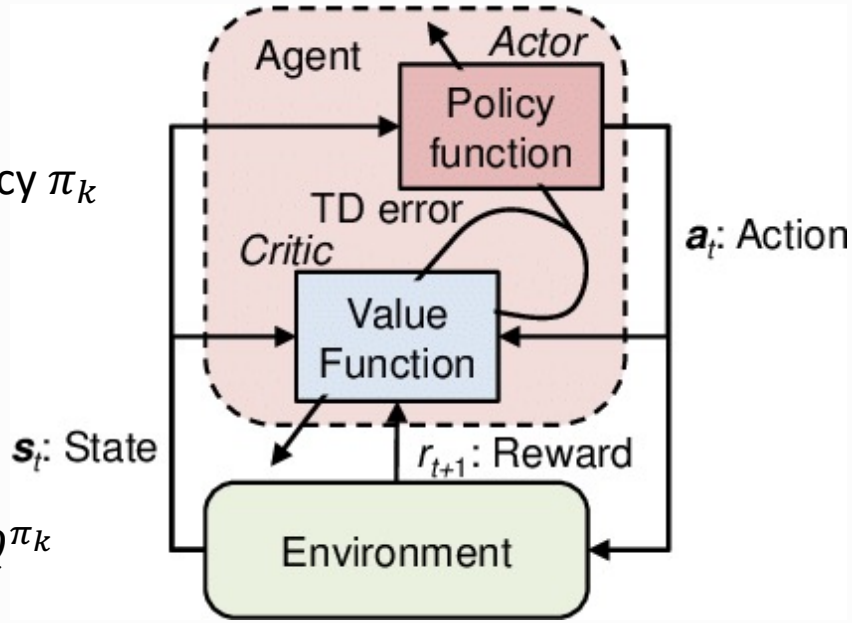
- Actor-critic Methods: combine value-based and policy-based methods
  - Critic update: estimate state (action)-value function under current policy $\pi_k$

$$V(s_k) \leftarrow V(\mathrm{s_k}) + \alpha_k \left[r_k + \gamma V(s_{k+1}) - V(s_k)\right]$$

$$Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha_k \left[r_k + \gamma \cdot Q(s_{k+1}, a_{k+1}) - Q(s_k, a_k)\right]$$

  - Actor update: policy gradient based on value function estimator $V^{\pi_k}$, $Q^{\pi_k}$

$$\theta_{k+1} = \theta_k + \eta_k \nabla_\theta \hat{J}(\pi_{\theta_k})$$



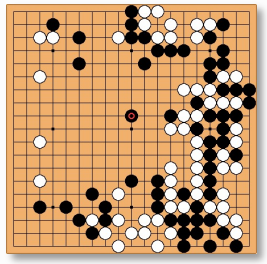

[Sutton and Barto, 2018]

# The first grand challenge – Representation

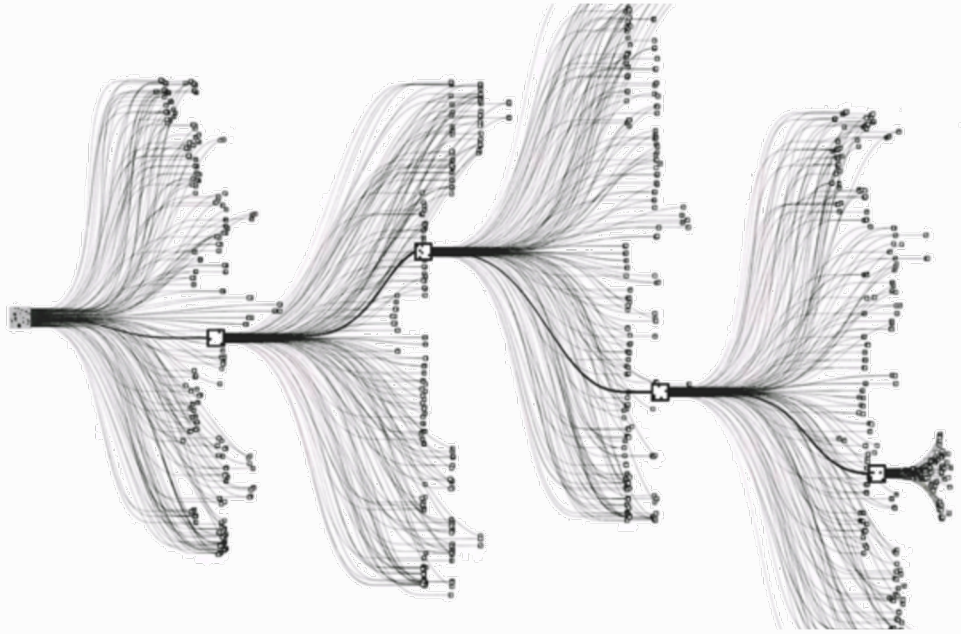
- Huge state space
- Huge policy space

Chess: $10^{120}$





Go: $3^{361}$



In practice : using (nonlinear) function approximation!

# RL with Function Approximation

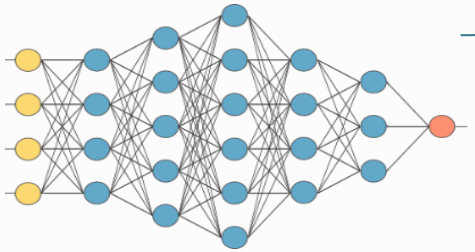
- Linear function approximation
- Neural networks
- Reproducing kernel Hilbert space (RKHS)
- Decision tree, nearest neighbor, etc.
- Generative models

State $s$



Action $a$

$$[\phi_1(s,a),\dots,\phi_d(s,a)]\begin{bmatrix}\theta_1\\ \vdots\\ \theta_d\end{bmatrix}$$



$V_\theta(s)$

$Q_\theta(s,a)$

$\pi_\theta(a|s)$

$P_\theta(s,s',a)$

Learn parameter $\theta \in R^d$ $(d \ll |S||A|)$

# Example: temporal-difference Learning

Given policy $\pi$, want to estimate $V^{\pi}$ or $Q^{\pi}$.

- TD(0)-leaning:

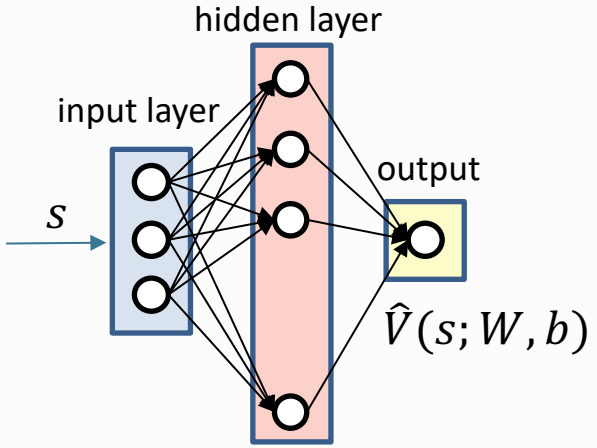
$$V(s_k) \leftarrow V(\mathrm{s_k}) + \alpha_k \left[r_k + \gamma V(s_{k+1}) - V(s_k)\right]$$

- TD learning with linear function approximation: $V(s) \approx \phi(s)^T\theta$

$$\theta_{k+1} = \theta_k + \alpha_k \left[r_k + \gamma \phi(s_{k+1})^T\theta_k - \phi(s_k)^T\theta_k\right]\phi(s_k)$$

- Neural TD learning: $\hat{V}(s; W, b) = \sum_{i=1}^{m} b_i \cdot \sigma(W_i^T\phi(s))$

$$W_{k+1} = W_k + \alpha_k \left[r_k + \gamma \hat{V}_k(s_{k+1}) - \hat{V}_k(s_k)\right]\nabla_W \hat{V}_k(s_k)$$



# Example: Q-learning algorithms

Given a trajectory, want to estimate $Q^*$.

- Q-leaning:

$$Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha_k \left[r_k + \gamma \cdot \max_{a'} Q(s_{k+1}, a') - Q(s_k, a_k)\right]$$

- Q-learning with linear function approximation: $Q(s, a) \approx \phi(s, a)^T \theta$

$$\theta_{k+1} = \theta_k + \alpha_k \left[r_k + \gamma \max_{a'} \phi(s_{k+1}, a')^T \theta_k - \phi(s_k, a_k)^T \theta_k\right] \phi(s_k, a_k)$$

- Neural Q-learning: $Q(s, a) \approx f\big(\theta; \phi(s, a)\big),\ f(\theta; x) = W_L \sigma_L(W_{L-1} \cdots \sigma_1(W_1 x))$

$$\theta_{k+1} = \theta_k + \alpha_k \left[r_k + \gamma \max_{a'} f(\theta_k; \phi(s_{k+1}, a')) - f(\theta_k; \phi(s_k, a_k))\right] \nabla_\theta f(\theta_k; \phi(s_k, a_k))$$

# Example: policy gradient method

$$\max_{\theta \in \Theta} \; J(\theta) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(s_k, a_k) \middle| s_0 \sim \mu, s_{k+1} \sim P(\cdot|s_k, a_k), a_k \sim \pi_\theta(\cdot|s_k)\right]$$

## Tabular MDPs

### Direct Parametrization

$$\pi_\theta(a|s) = \theta_{s,a}$$
$$\theta \in \Delta(\mathcal{A})^{|\mathcal{S}|}$$

### Softmax Parametrization

$$\pi_\theta(a|s) = \frac{\exp(\theta_{s,a})}{\sum_a \exp(\theta_{s,a})}$$

## Function Approximation

### Log-linear Policy

$$\pi_\theta(a|s) = \frac{\exp(\phi(s,a)^T\theta)}{\sum_a \exp(\phi(s,a)^T\theta)}$$

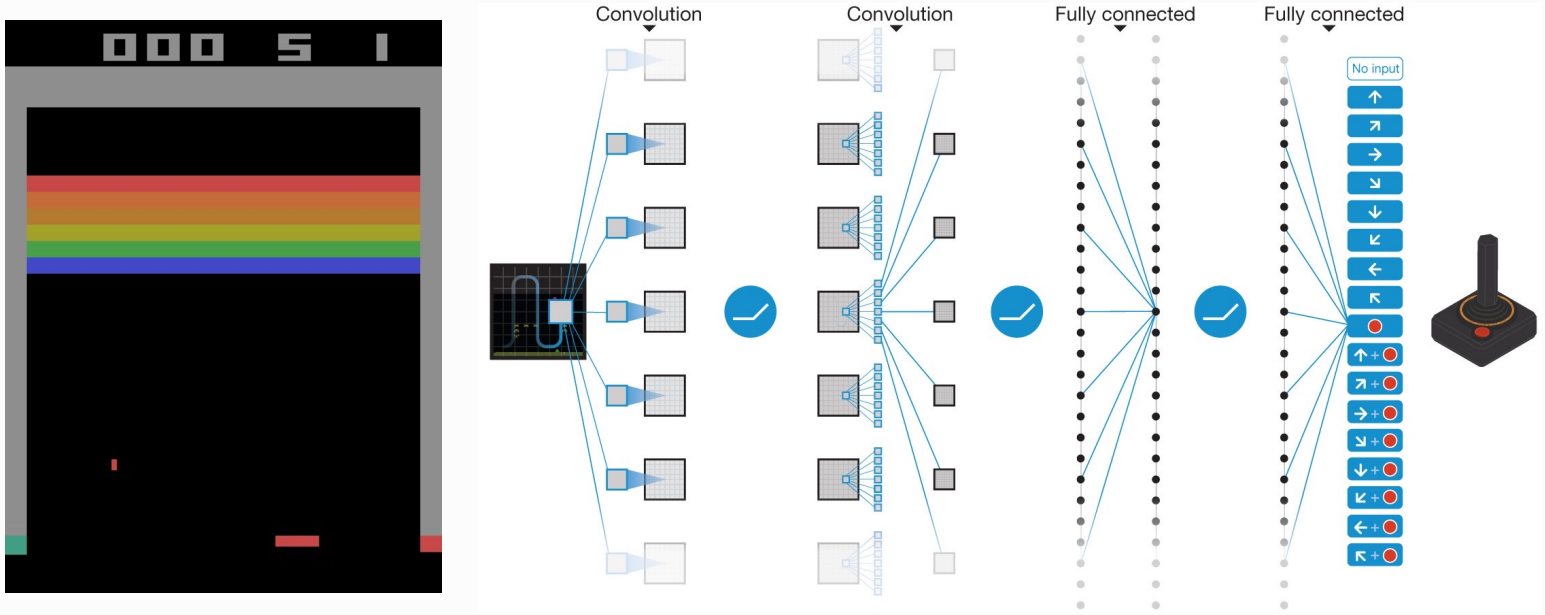
### Neural Policy

$$\pi_\theta(a|s) = \frac{\exp(f_\theta(s,a))}{\sum_a \exp(f_\theta(s,a))}$$

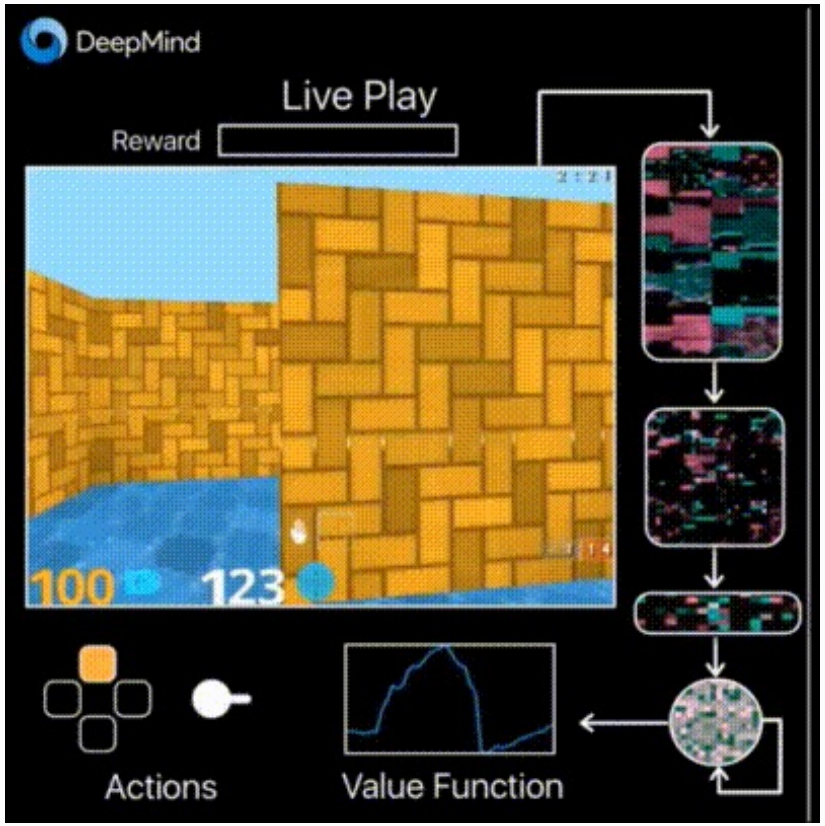
$f_\theta(s,a)$ is a neural network

# Empirical successes in RL

Deep Q-Networks




Asynchronous Advantage Actor-Critic (A3C)

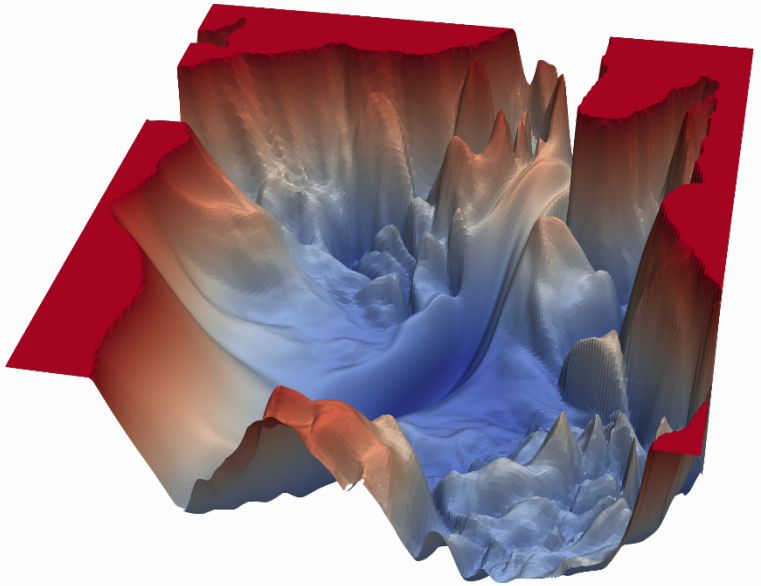



Mnih et al. "Human-level control through deep reinforcement learning." *Nature,* 2015.

Mnih et al. Asynchronous methods for deep reinforcement learning." ICML 2016.

# The second grand challenge – Optimization

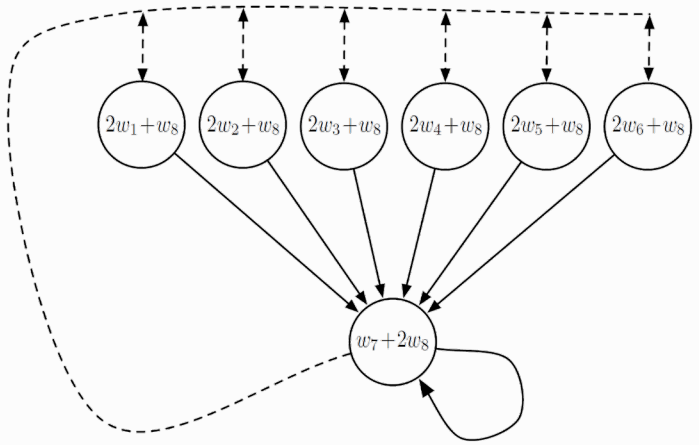
- Nonconvex objectives
- Biased gradients
- Intricate landscape



In practice : leveraging curvature, geometry, regularization, etc.

# The third grand challenge – Deadly Triad

- Divergence when combing off-policy data, bootstrapping, and function approximation



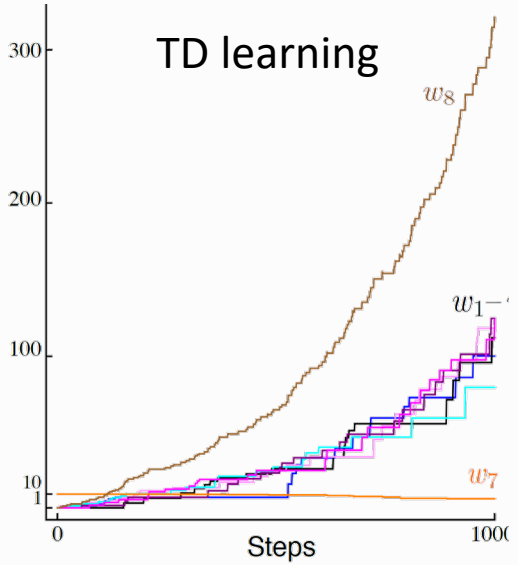

$\pi(\text{solid}|\cdot) = 1$

$b(\text{dashed}|\cdot) = 6/7$

$b(\text{solid}|\cdot) = 1/7$

$\gamma = 0.99$

Baird's Example



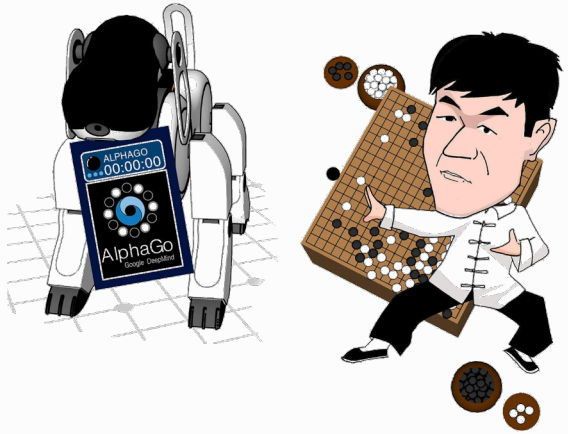

[Baird, 1995; Tsitsiklis and Van Roy, 1997; Sutton and Barto, 2018]

In practice: using experience replay, target network , multi-step returns, etc.

# The fourth grand challenge – Multi-agent Learning

- Equilibrium can be difficult to obtain
- Nonstationary of the environment
- Partial observability

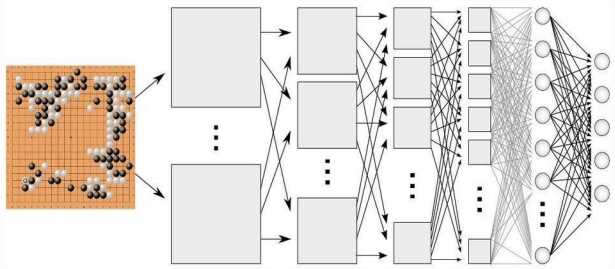


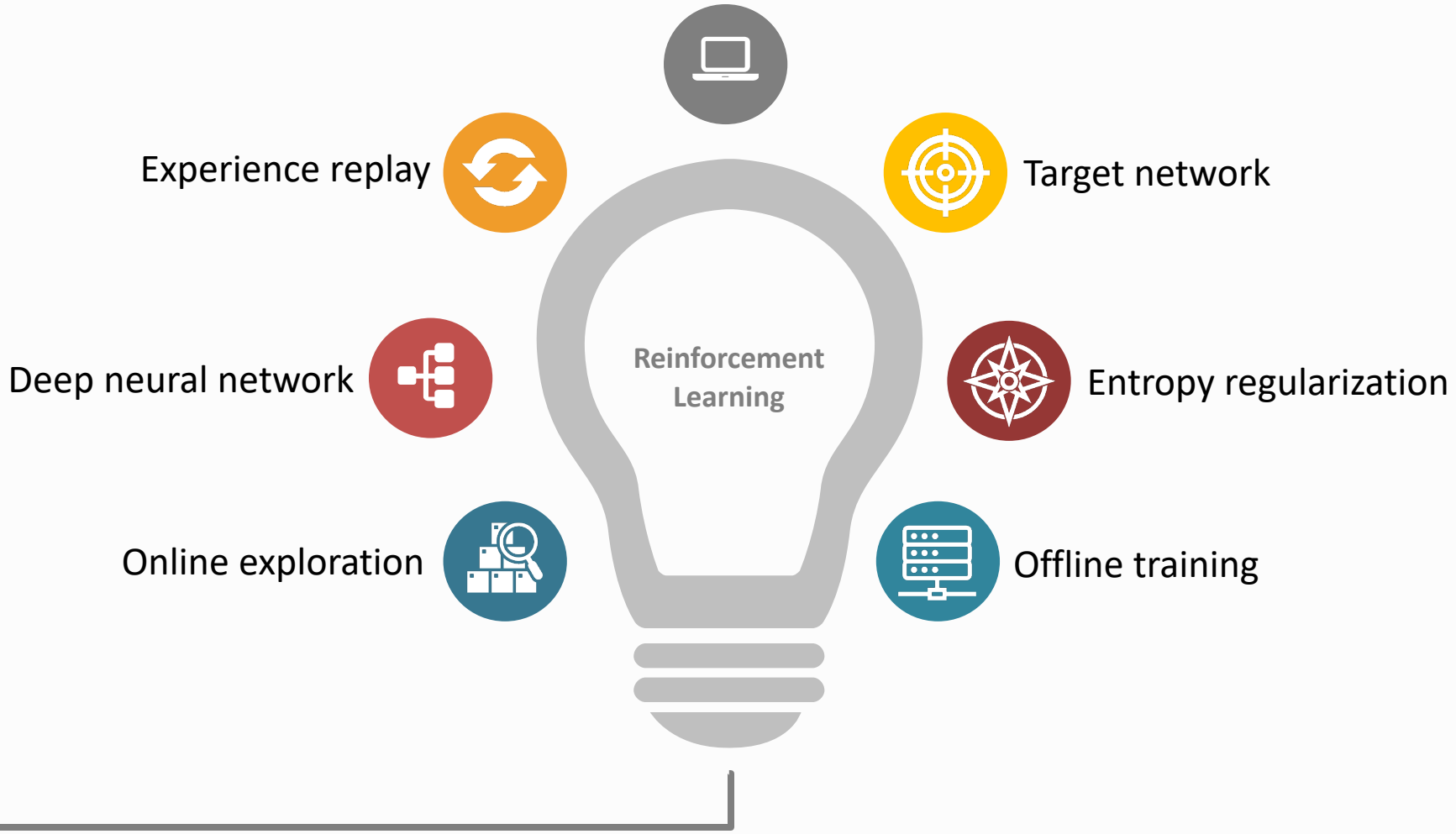



In practice: using independent learner, fictitious play, etc.

# Behind the scenes

Empirical successes in RL

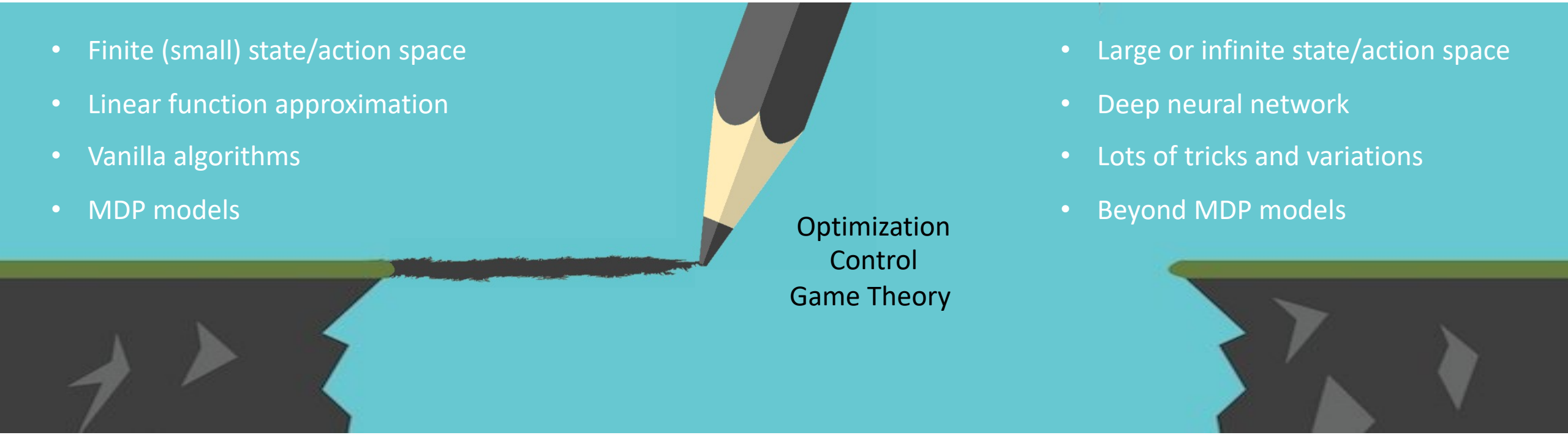




# The wide theory-practice gap

## RL Theory

- Finite (small) state/action space
- Linear function approximation
- Vanilla algorithms
- MDP models



Optimization
Control
Game Theory

## RL Practice

- Large or infinite state/action space
- Deep neural network
- Lots of tricks and variations
- Beyond MDP models

# The rest of this lecture series

To build a theoretical understanding of modern RL algorithms from various aspects:

- **Stability** – from dynamical system/control perspective
- **Global convergence** – from optimization perspective
- **Generalization** – from learning perspective
- **Equilibrium** – from game perspective

