Basic Q-Learning

Apr 16^{th}, 2019

Applied Deep Learning

YUN-NUNG (VIVIAN) CHEN
HTTP://ADL.MIULAB.TW

國立臺灣大學
National Taiwan University

Slides credited from Dr. David Silver & Hung-Yi Lee

Reinforcement Learning Approach

Value-based RL

- Estimate the optimal value function $Q^*(s,a)$

$Q^*(s,a)$ is maximum value achievable under any policy

Policy-based RL

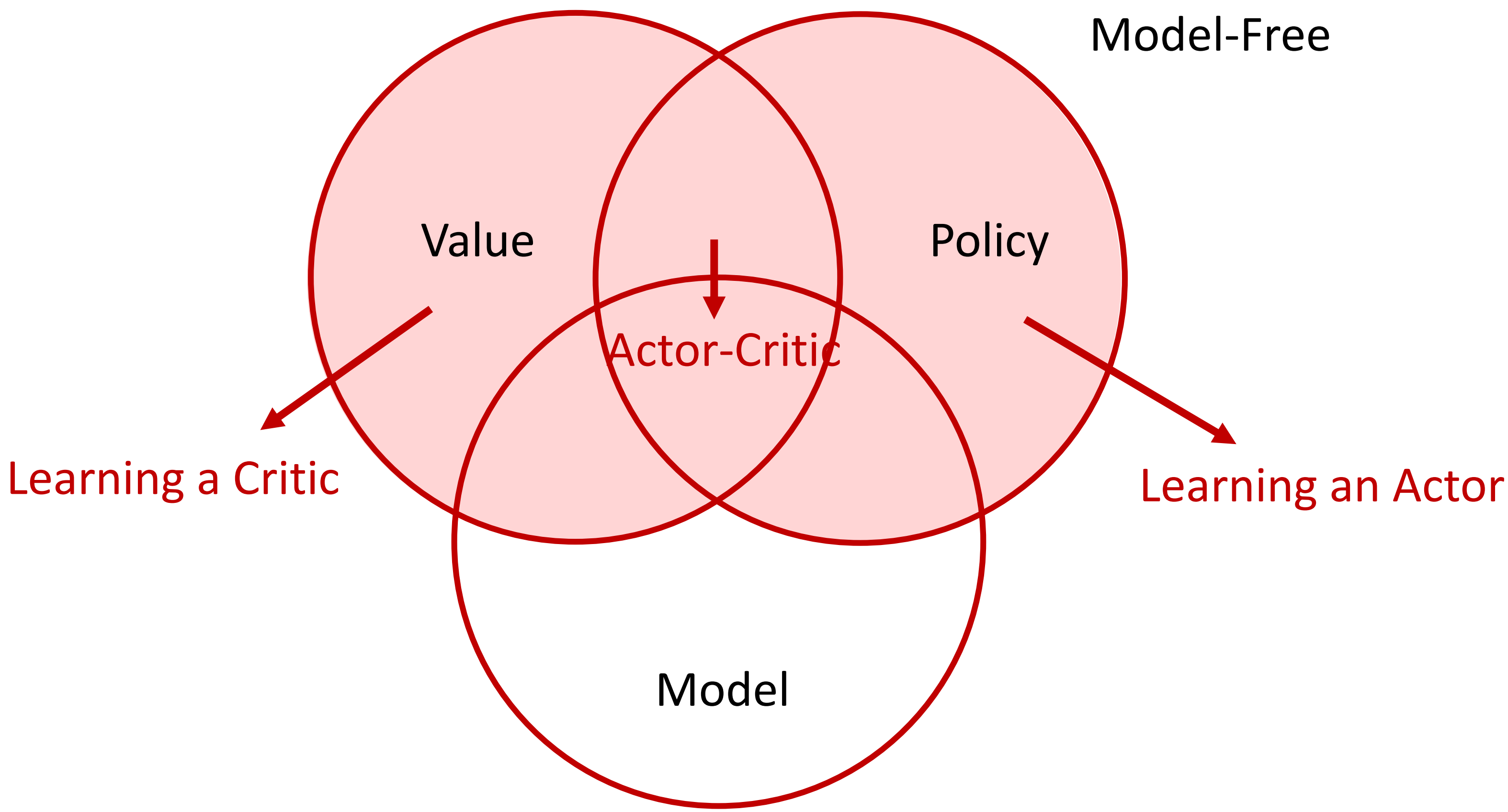
- Search directly for optimal policy π^*

π^* is the policy achieving maximum future reward

Model-based RL

- Build a model of the environment
- Plan (e.g. by lookahead) using model

RL Agent Taxonomy

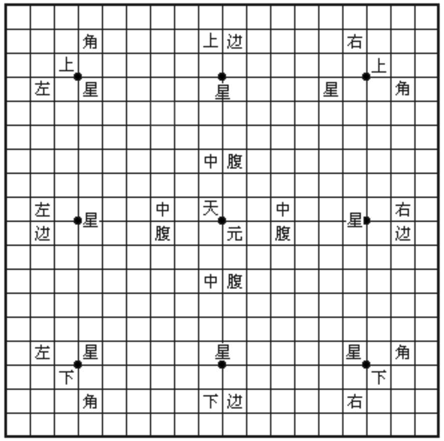


Value-Based Approach

LEARNING A CRITIC

Value Function

A value function is a prediction of future reward (with action *a* in state *s*)



Q-value function gives expected total reward

- from state s and action a
- under policy π
- with discount factor γ

$$Q^{\pi}(s,a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a\right]$$

Value functions decompose into a Bellman equation

$$Q^{\pi}(s,a) = \mathbb{E}_{s',a'}\left[r + \gamma Q^{\pi}(s',a') \mid s, a\right]$$

Optimal Value Function

An optimal value function is the maximum achievable value

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) = Q^{\pi^*}(s,a)$$

The optimal value function allows us act optimally

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

The optimal value informally maximizes over all decisions

$$\begin{aligned} Q^*(s,a) &= r_{t+1} + \gamma \max_{a_{t+1}} r_{t+2} + \gamma^2 \max_{a_{t+2}} r_{t+3} + \ldots \\ &= r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \end{aligned}$$

Optimal values decompose into a Bellman equation

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$$

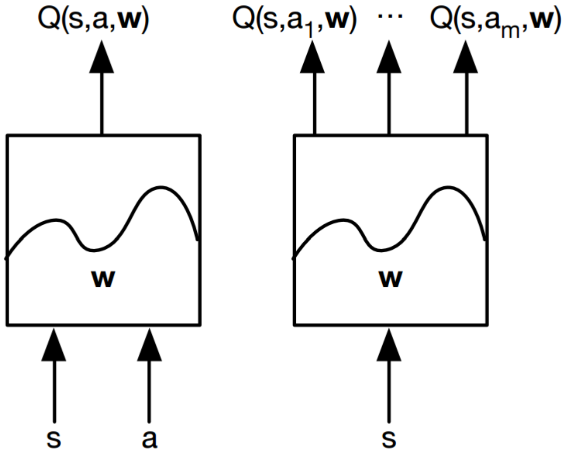
Value Function Approximation

Value functions are represented by a *lookup table*

$$Q(s, a) \;\; \forall s, a$$

- too many states and/or actions to store
- too slow to learn the value of each entry individually

Values can be estimated with *function approximation*

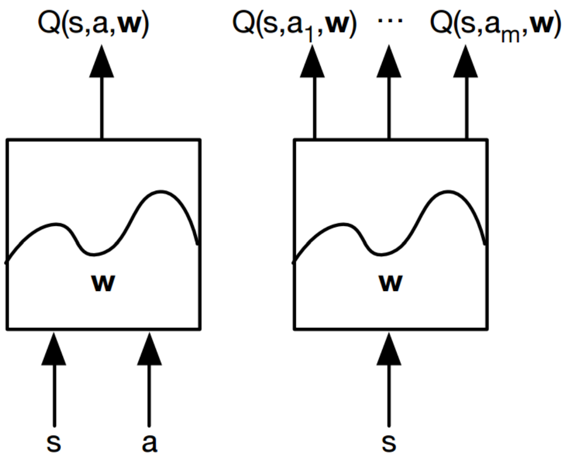


Q-Networks

Q-networks represent value functions with weights w

$$Q(s, a, w) \approx Q^*(s, a)$$

- generalize from seen states to unseen states
- update parameter w for function approximation



Q-Learning

Goal: estimate optimal Q-values

- Optimal Q-values obey a Bellman equation

$$Q^*(s, a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s', a') \mid s, a\right]$$

learning target

- *Value iteration* algorithms solve the Bellman equation

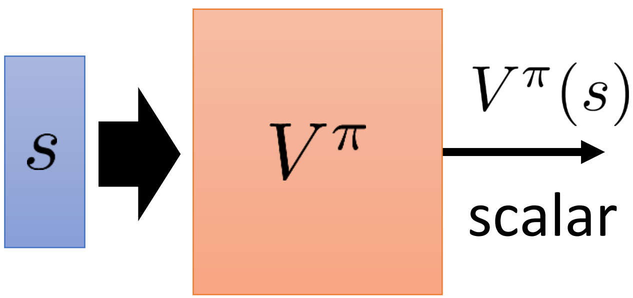
$$Q_{i+1}(s, a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q_i(s', a') \mid s, a\right]$$

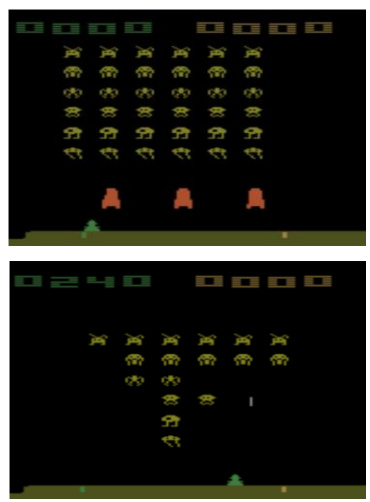
Critic = Value Function

Idea: how good the actor is

State value function: when using actor π, the *expected total reward* after seeing observation (state) s

$$V^{\pi}(s) \ \forall s \quad = \mathbb{E}[G_t \mid s_t = s]$$





larger $V^{\pi}(s)$

smaller $V^{\pi}(s)$

A critic does not determine the action
An actor can be found from a critic

Monte-Carlo for Estimating $V^{\pi}(s)$

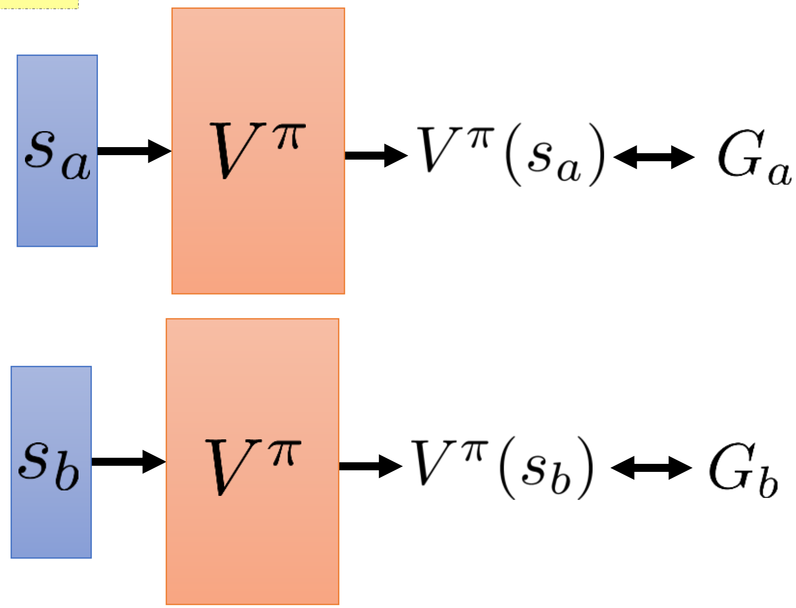
Monte-Carlo (MC)

- The critic watches π playing the game
- MC learns directly from *complete* episodes: no bootstrapping

Idea: value = *empirical mean* return

After seeing s_a,
until the end of the episode,
the cumulated reward is G_a

After seeing s_b,
until the end of the episode,
the cumulated reward is G_b



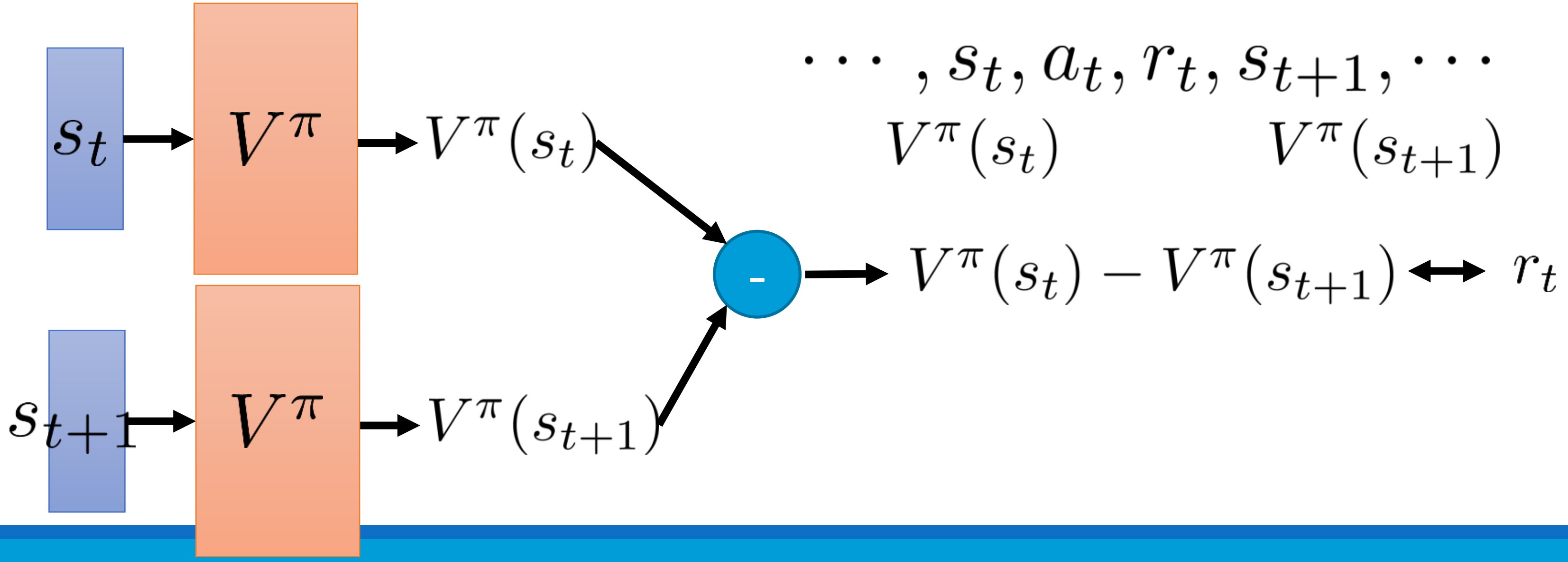
Issue: long episodes delay learning

Temporal-Difference for Estimating $V^\pi(s)$

Temporal-difference (TD)

- The critic watches π playing the game
- TD learns directly from *incomplete* episodes by *bootstrapping*
- TD updates a guess towards a guess

Idea: update value toward *estimated* return



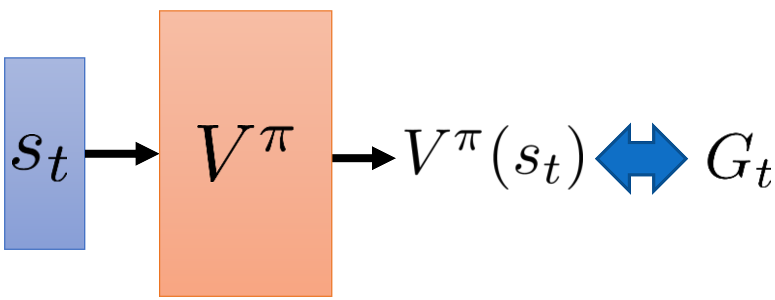
MC v.s. TD

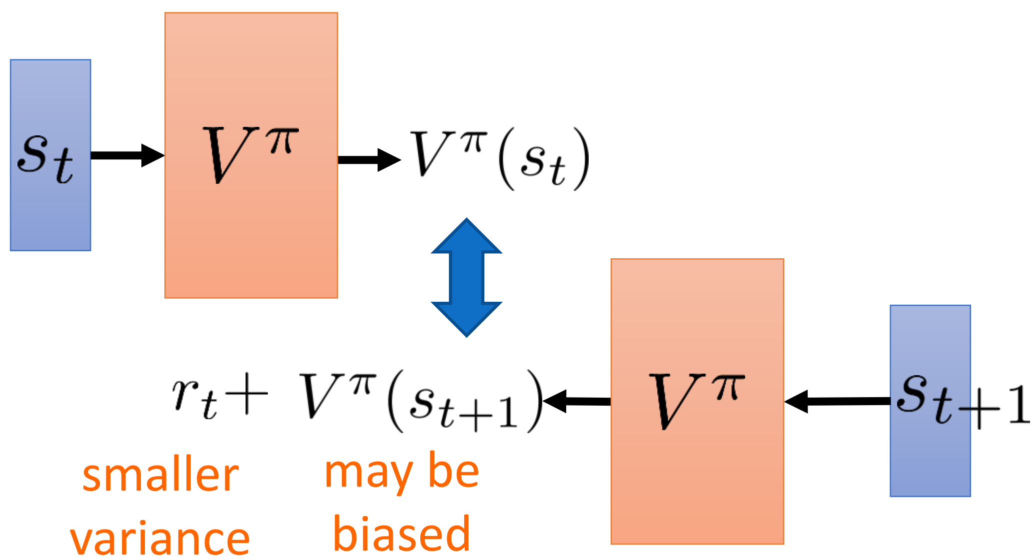
Monte-Carlo (MC)

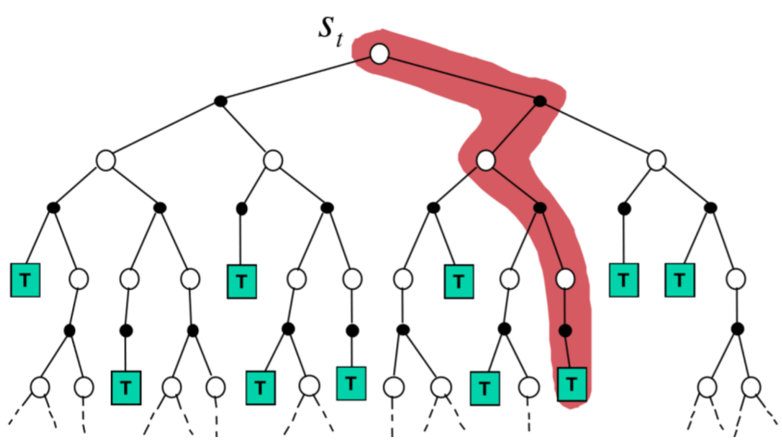
- Large variance
- Unbiased
- No Markov property

Temporal-Difference (TD)

- Small variance
- Biased
- Markov property

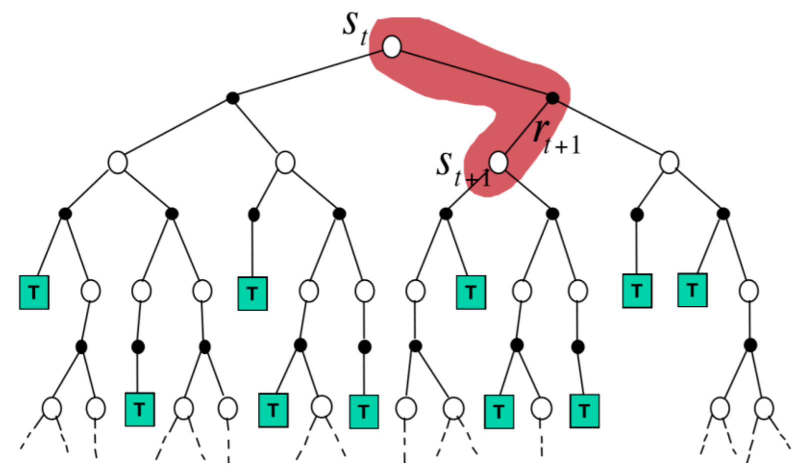






$V'^{\pi}(s_t)$

$= V^{\pi}(s_t) + \alpha(G_t - V^{\pi}(s_t))$



$V'^{\pi}(s_t)$

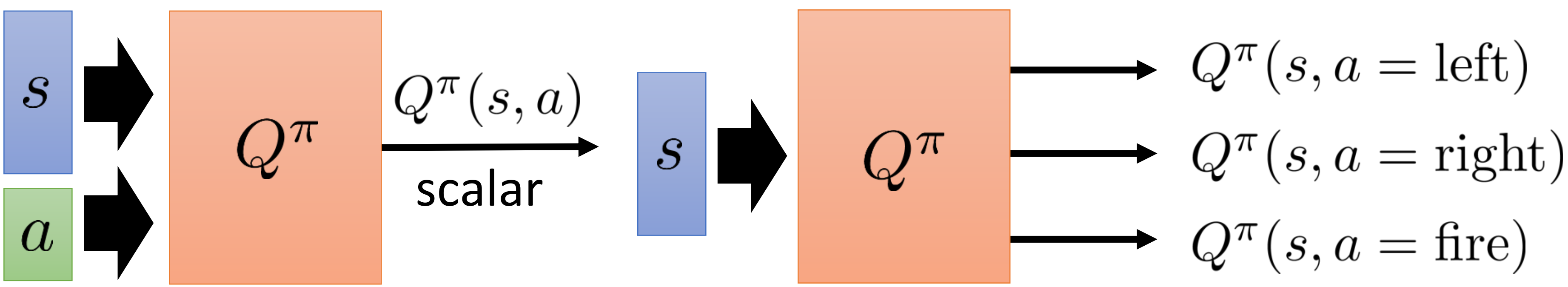
$= V^{\pi}(s_t) + \alpha(r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t))$

MC v.s. TD

Critic = Value Function

State-action value function: when using actor π, the *expected total reward* after seeing observation (state) s and taking action a

$$Q^\pi(s,a)\ \forall s,a = \mathbb{E}[G_t \mid s_t = s, a_t = a]$$



for discrete action only

Q-Learning

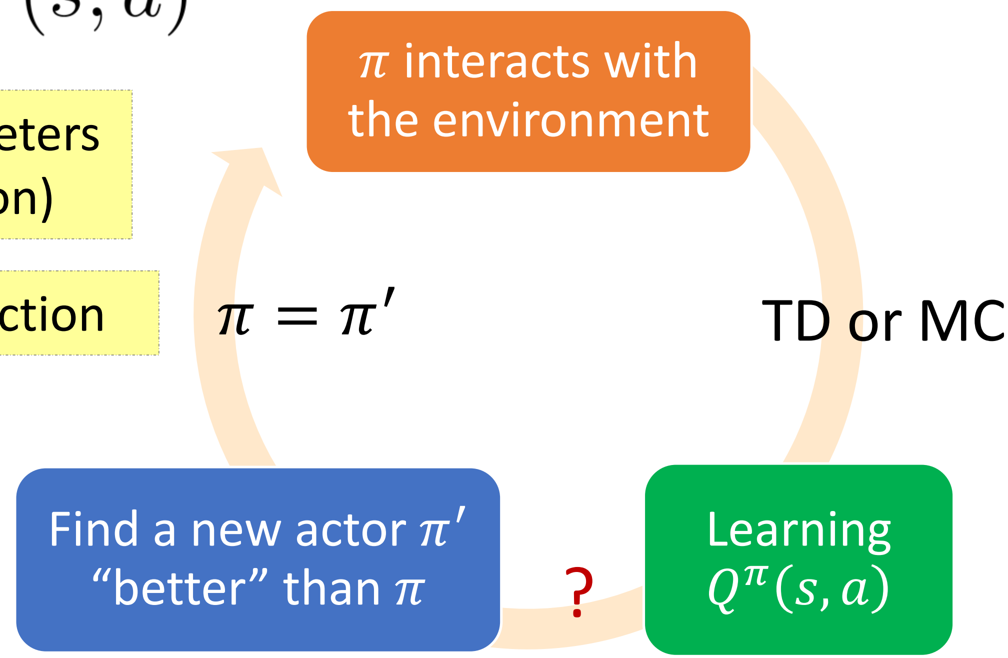
Given $Q^{\pi}(s,a)$, find a new actor π' "better" than π

$$V^{\pi'}(s) \geq V^{\pi}(s) \ \forall s$$

$$\pi'(s) = \arg\max_{a} Q^{\pi}(s,a)$$

π' does not have extra parameters (depending on value function)

not suitable for continuous action



Q-Learning

Goal: estimate optimal Q-values

- Optimal Q-values obey a Bellman equation

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a') \mid s,a\right]$$

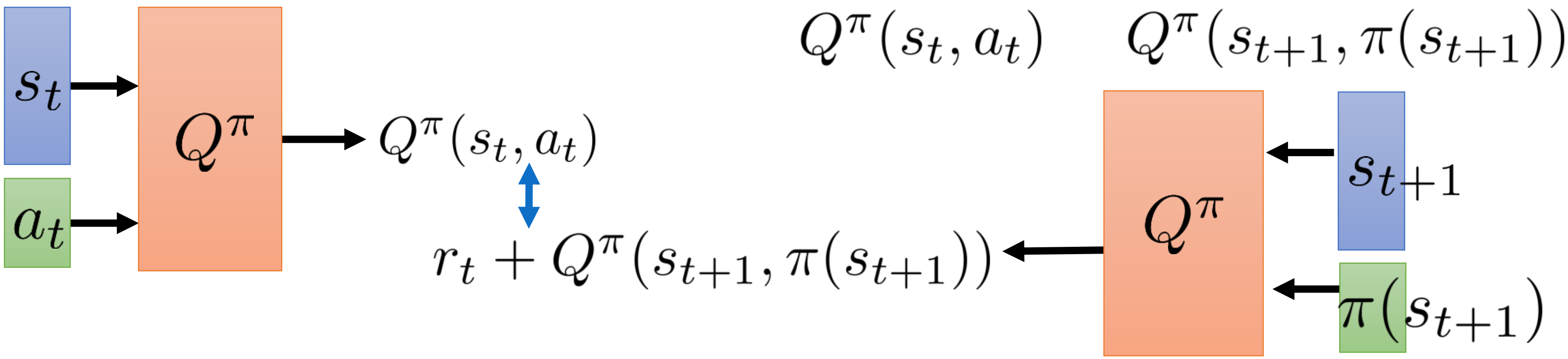
learning target

- *Value iteration* algorithms solve the Bellman equation

$$Q_{i+1}(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q_i(s',a') \mid s,a\right]$$

Deep Q-Networks (DQN)

Estimate value function by TD $\cdots, s_t, a_t, r_t, s_{t+1}, \cdots$



Represent value function by deep Q-network with weights w

$$Q(s, a, w) \approx Q^*(s, a)$$

Objective is to minimize MSE loss by SGD

$$\mathcal{L}(w) = \mathbb{E}\Big[\Big(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\Big)^2\Big]$$

Deep Q-Networks (DQN)

Objective is to minimize MSE loss by SGD

$$\mathcal{L}(w) = \mathbb{E}\Big[\Big(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\Big)^2\Big]$$

Leading to the following Q-learning gradient

$$\frac{\partial \mathcal{L}(w)}{\partial w} = \mathbb{E}\Big[\Big(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\Big) \frac{\partial Q(s, a, w)}{\partial w}\Big]$$

Issue: naïve Q-learning oscillates or diverges using NN due to:
1) correlations between samples 2) non-stationary targets

Stability Issues with Deep RL

Naive Q-learning oscillates or diverges with neural nets

1. Data is sequential
   - Successive samples are correlated, non-iid (independent and identically distributed)
2. Policy changes rapidly with slight changes to Q-values
   - Policy may oscillate
   - Distribution of data can swing from one extreme to another
3. Scale of rewards and Q-values is unknown
   - Naive Q-learning gradients can be unstable when backpropagated

Stable Solutions for DQN

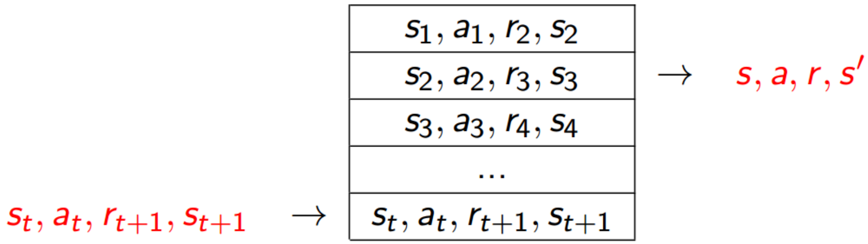
DQN provides a stable solutions to deep value-based RL

1. Use experience replay
   - Break correlations in data, bring us back to iid setting
   - Learn from all past policies
2. Freeze target Q-network
   - Avoid oscillation
   - Break correlations between Q-network and target
3. Clip rewards or normalize network adaptively to sensible range
   - Robust gradients

Stable Solution 1: Experience Replay

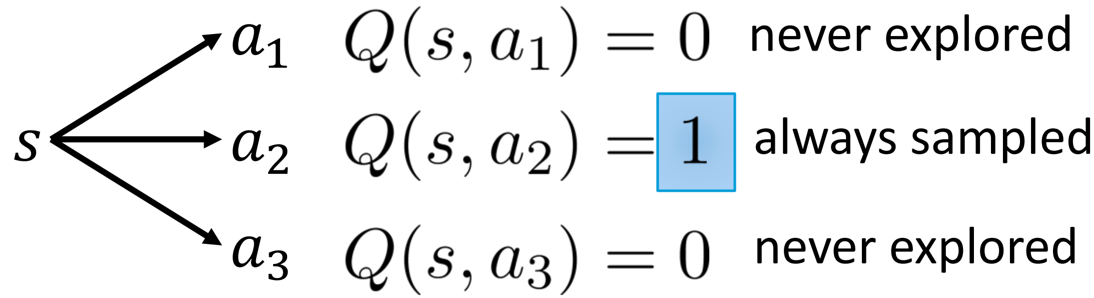
To remove correlations, build a dataset from agent's experience

- Take action at according to ϵ-greedy policy
  small prob for exploration
- Store transition (s_t, a_t, r_t, s_{t+1}) in replay memory D
- Sample random mini-batch of transitions (s, a, r, s') from D



- Optimize MSE between Q-network and Q-learning targets

$$\mathcal{L}(w) = \mathbb{E}_{s,a,r,s' \sim D}\left[\left(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w) \right)^2 \right]$$



Exploration

The policy is based on Q-function

$$a = \arg\max_a Q(s, a)$$

not good for data collection
→ inefficient learning

Exploration algorithms

- Epsilon greedy

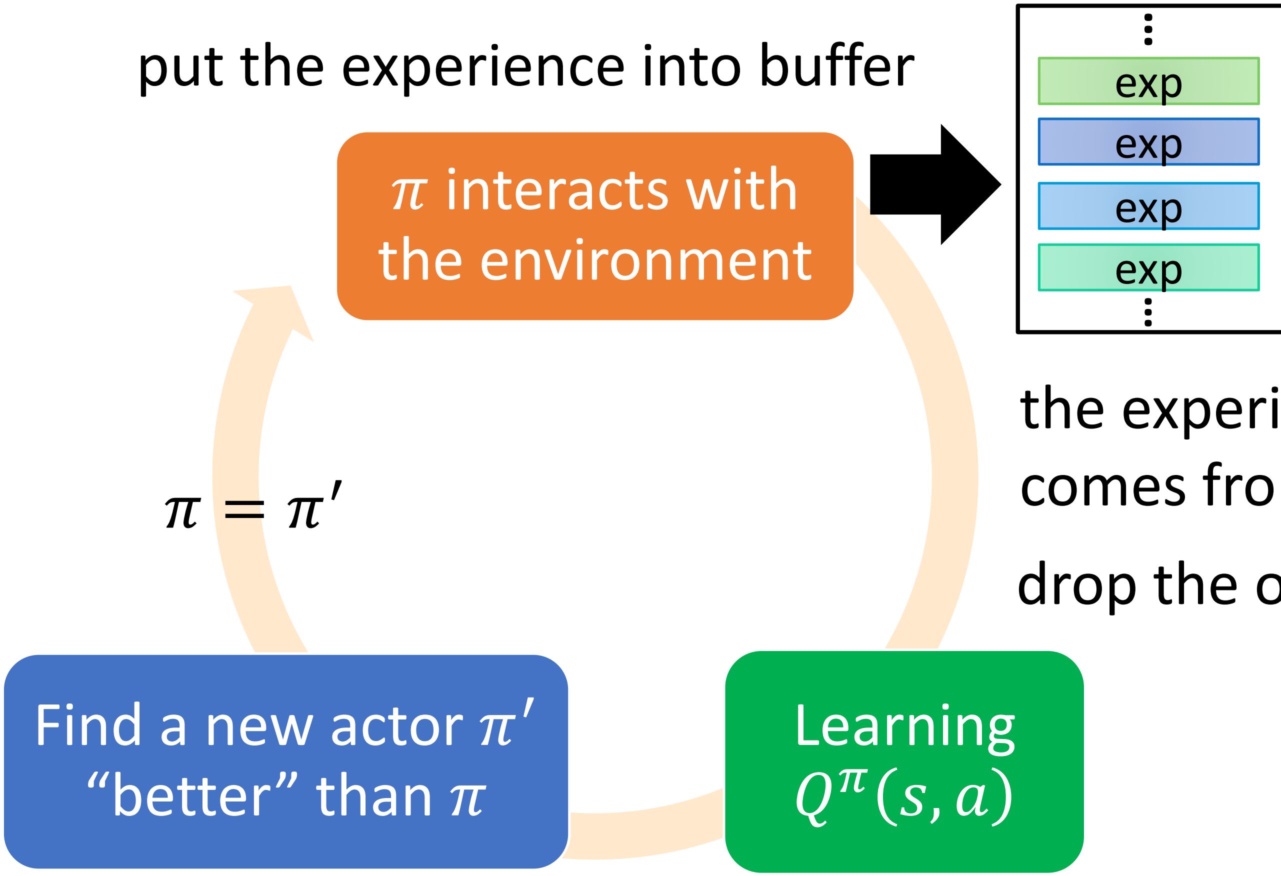
ε would decay during learning

$$a = \begin{cases} \arg\max_a Q(s, a), & \text{with } p = (1 - \epsilon) \\ \text{random}, & \text{otherwise} \end{cases}$$

- Boltzmann sampling

$$P(a \mid s) = \frac{\exp(Q(s, a))}{\sum_a \exp(Q(s, a))}$$

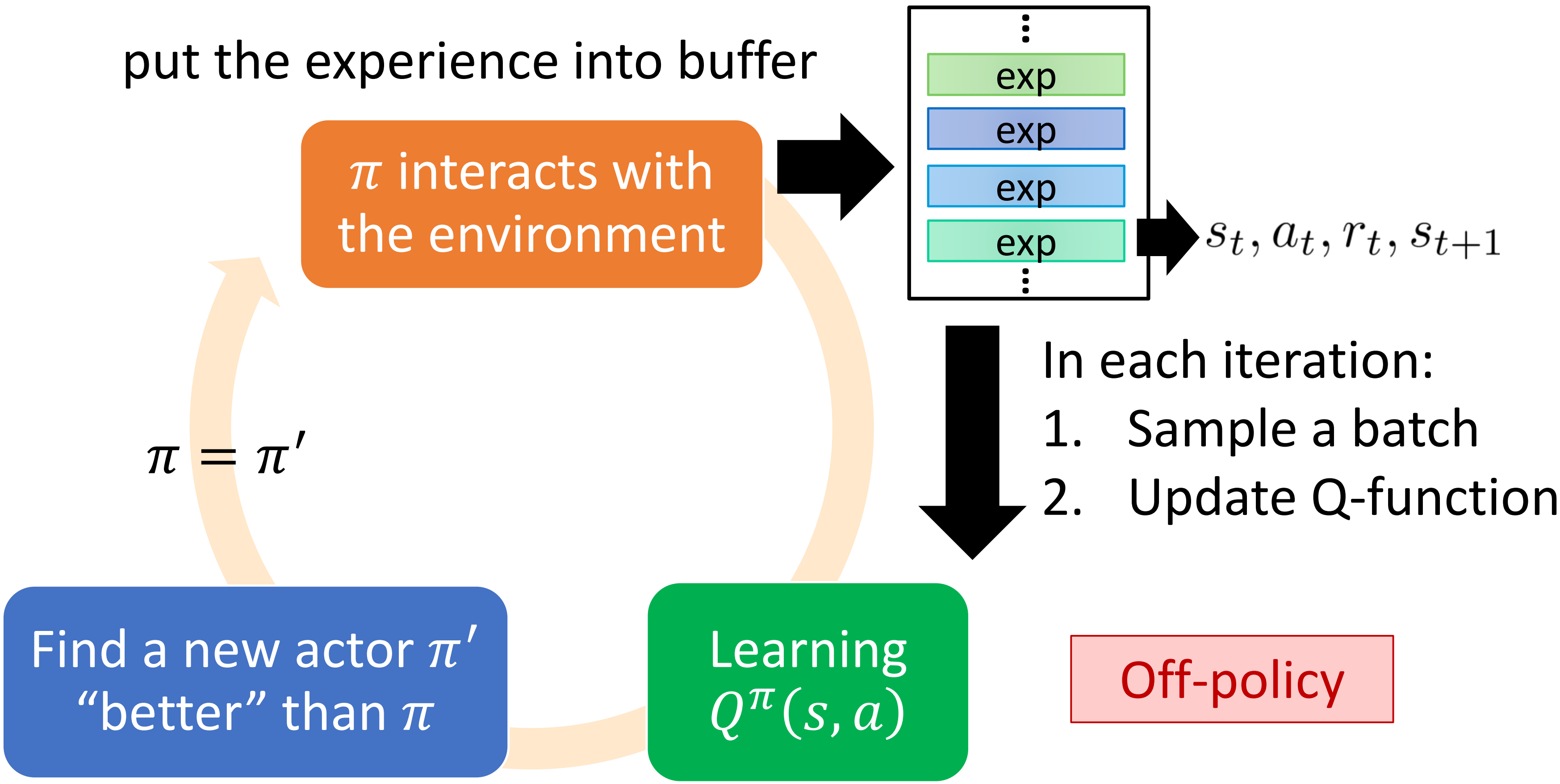
Replay Buffer



the experience in the buffer comes from different π

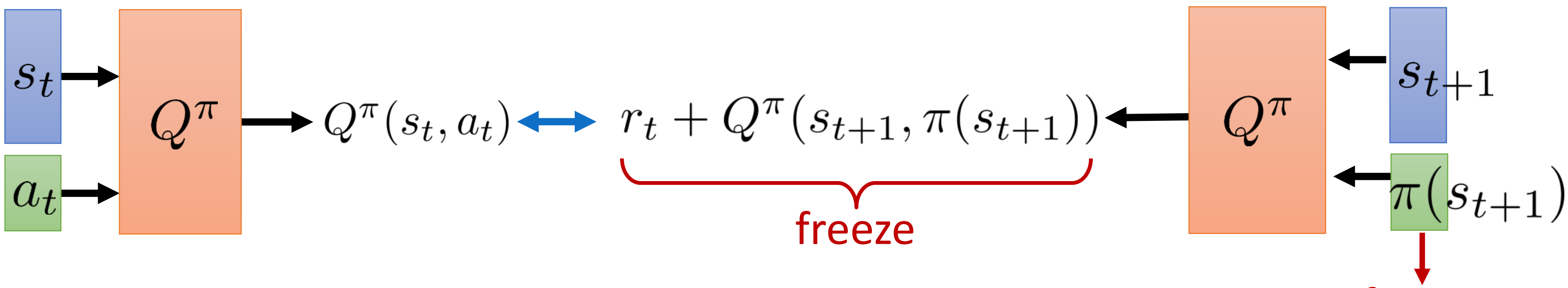
drop the old one if full

Replay Buffer



Stable Solution 2: Fixed Target Q-Network

To avoid oscillations, fix parameters used in Q-learning target



- Compute Q-learning targets w.r.t. old, fixed parameters w^-

$$r + \gamma \max_{a'} \hat{Q}(s', a', w^-)$$

- Optimize MSE between Q-network and Q-learning targets

$$\mathcal{L}(w) = \mathbb{E}_{s,a,r,s' \sim D}\left[\left(r + \gamma \max_{a'} \hat{Q}(s', a', w^-) - Q(s, a, w) \right)^2 \right]$$

- Periodically update fixed parameters $w^- \leftarrow w$

Stable Solution 3: Reward / Value Range

To avoid oscillations, control the reward / value range

- DQN clips the rewards to $[-1, +1]$
  - Prevents too large Q-values
  - Ensures gradients are well-conditioned

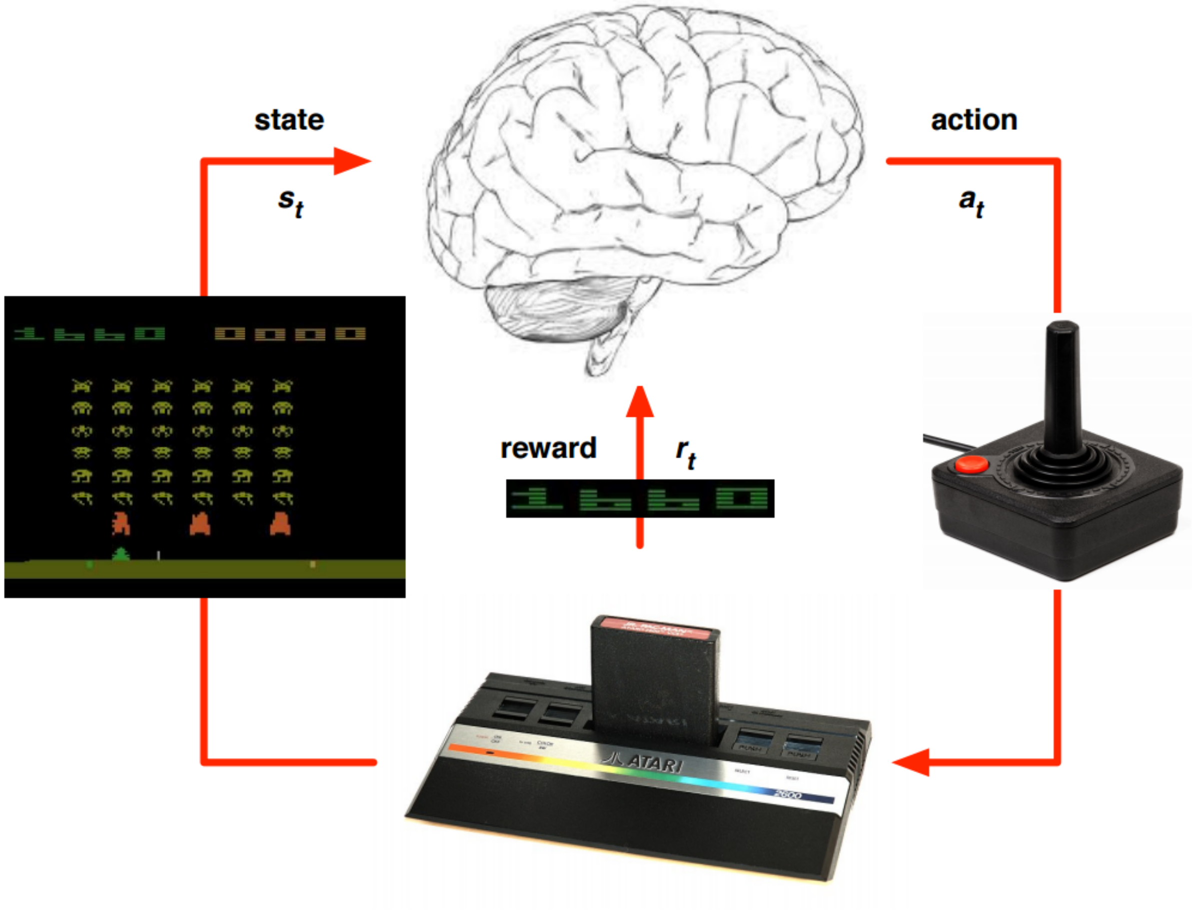
Typical Q-Learning Algorithm

Initialize Q-function Q, target Q-function $\hat{Q} = Q$

In each episode

- For each time step t
  - Given state s_t, take action a_t based on Q (epsilon greedy)
  - Obtain reward r_t, and reach new state s_{t+1}
  - Store (s_t, a_t, r_t, s_{t+1}) into buffer
  - Sample (s_i, a_i, r_i, s_{i+1}) from buffer (usually a batch)
  - Update the parameters of Q to make $Q(s_i, a_i) \approx r_i + \max_a \hat{Q}(s_{i+1}, a)$
  - Every C steps reset $\hat{Q} = Q$

Deep RL in Atari Games

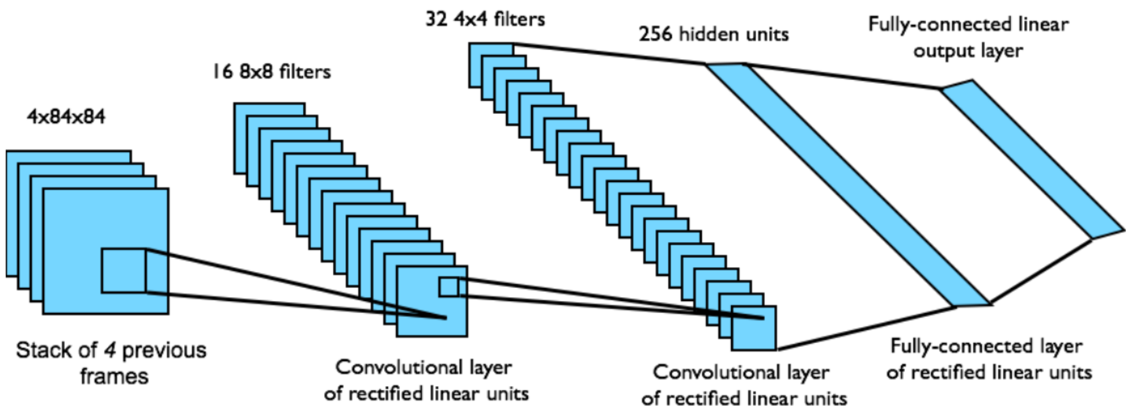


DQN in Atari

Goal: end-to-end learning of values $Q(s, a)$ from pixels

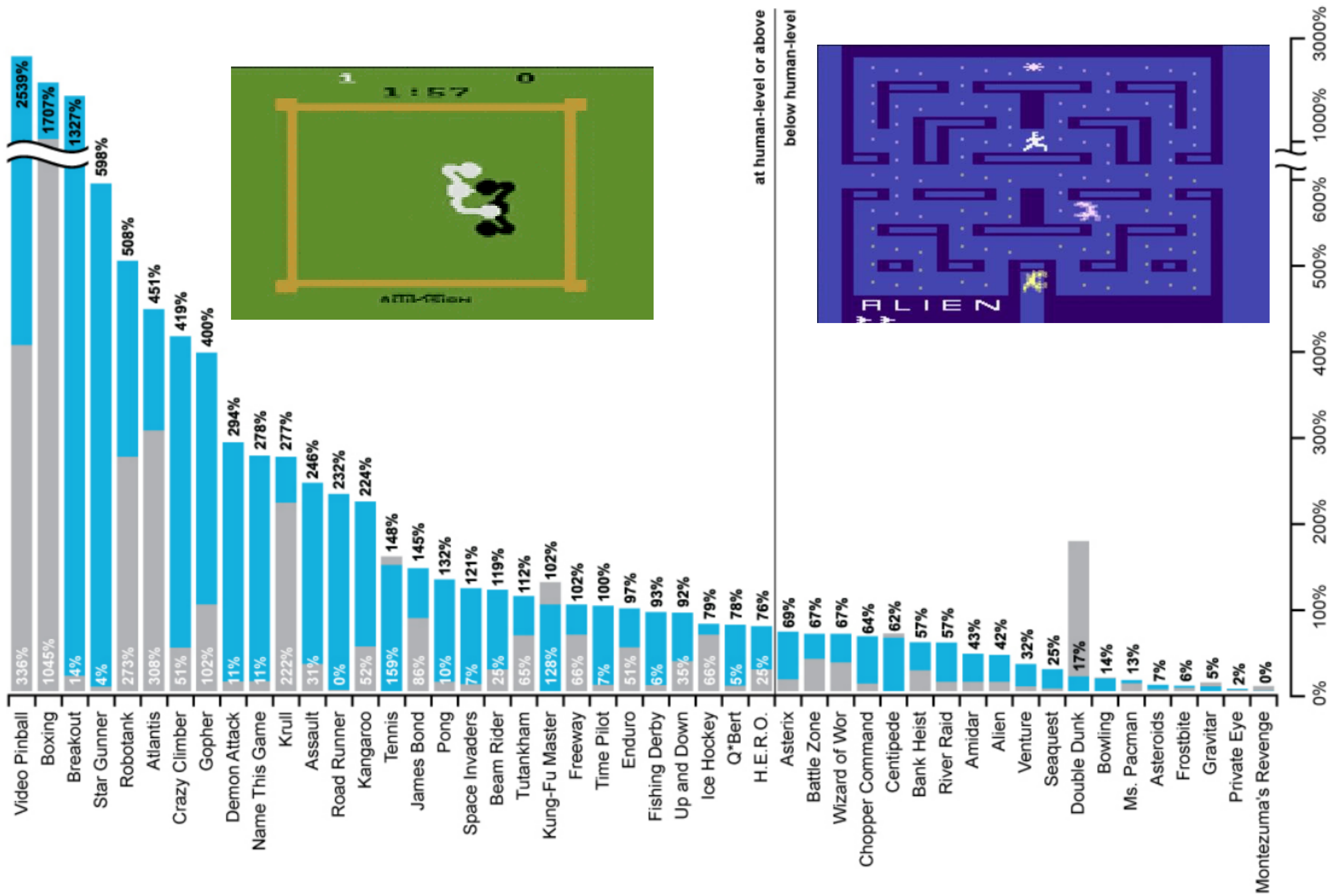
$$\mathcal{L}(w) = \mathbb{E}_{s,a,r,s' \sim D}\left[\left(r + \gamma \max_{a'} Q(s', a', w^{-}) - Q(s, a, w) \right)^2 \right]$$

- Input: state is stack of raw pixels from last 4 frames
- Output: $Q(s, a)$ for all joystick/button positions a
- Reward is the score change for that step



DQN in Atari


nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
LEARNING CURVE
Self-taught AI software attains human-level performance in video games
SHARE DATA IN OUTBREAKS
A GIANT IN THE EARLY UNIVERSE
TELEPORTATION FOR TWO


at human-level or above
below human-level
1 1:57 0
ALIEN
3000%
1000%
600%
500%
400%
300%
200%
100%
0%
Video Pinball 2539% 336%
Boxing 1707% 1045%
Breakout 1327% 14%
Star Gunner 598% 4%
Robotank 508% 273%
Atlantis 451% 308%
Crazy Climber 419% 51%
Gopher 400% 102%
Demon Attack 294% 11%
Name This Game 278% 11%
Krull 277% 222%
Assault 246% 31%
Road Runner 232% 0%
Kangaroo 224% 52%
Tennis 148% 159%
James Bond 145% 86%
Pong 132% 10%
Space Invaders 121% 7%
Beam Rider 119% 25%
Tutankham 112% 65%
Kung-Fu Master 102% 128%
Freeway 102% 66%
Time Pilot 100% 7%
Enduro 97% 51%
Fishing Derby 93% 6%
Up and Down 92% 35%
Ice Hockey 79% 66%
Q*Bert 78% 5%
H.E.R.O. 76% 25%
Asterix 69%
Battle Zone 67%
Wizard of Wor 67%
Chopper Command 64%
Centipede 62%
Bank Heist 57%
River Raid 57%
Amidar 43%
Alien 42%
Venture 32%
Seaquest 25%
Double Dunk 17%
Bowling 14%
Ms. Pacman 13%
Asteroids 7%
Frostbite 6%
Gravitar 5%
Private Eye 2%
Montezuma's Revenge 0%

Concluding Remarks

RL is a general purpose framework for **decision making** under interactions between agent and environment

A value-based RL measures how good each state and/or action is via a value function

- Monte-Carlo (MC) v.s. Temporal-Difference (TD)