Deep Reinforcement Learning

AI602: Recent Advances in Deep Learning

Lecture 12

Slide made by

Hankook Lee and Kimin Lee

KAIST EE

Recall: Course Information

- Assignment: 2 paper summary + 1 presentation
  - Each student should choose two deep learning papers published at NIPS, ICML or ICLR, CVPR, ICCV, ECCV in last 3 years.
  - You can use the authors' codes, but you will receive better grades if (a) the authors do not release their codes or (b) you modify the authors' code for better performance.
  - I will help for deciding which papers to study (e.g., send emails to me or ask after the class)
  - Try reproducing the authors' results (reported in their papers) and applying to other datasets. Or, modify the authors' code or algorithm for better performance.
  - Send the report on the first paper by Oct. 25th and the report on the second paper by Dec. 20th to TA. You also have to send your source-code files with the reports.
  - You have to present one of two papers at the end of this class.

Announcement

- You have to present one of two papers at the end of this class.
  - If you wish to present the first paper, you have to do it at Nov. 21th or 26th during the class.
  - If you wish to present the second paper, you have to do it at Dec. 10th or 12th during the class.
  - Email your choice to TAs in this week.

Announcement

- No class on this Thursday, Nov. 7th
  - Instead, please attend "International Symposium on Chemical Science meets AI", E9, KAIST

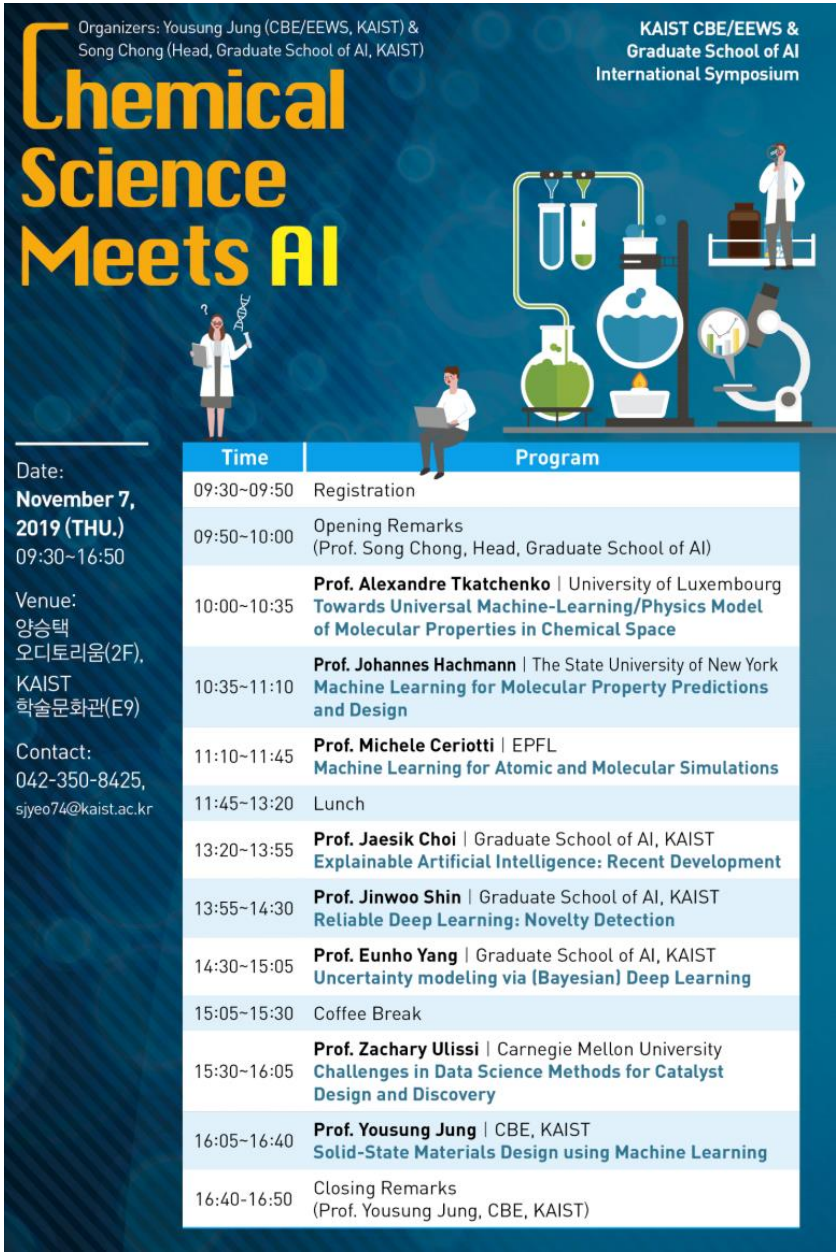


Table of Contents

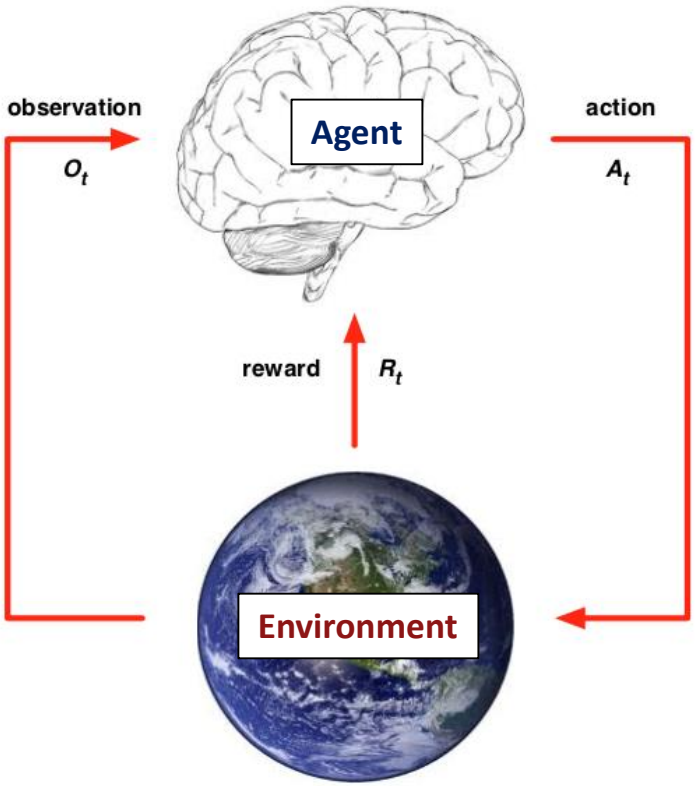
Table of Contents

What is Reinforcement Learning (RL)?

- Reinforcement learning is a **sequential decision making** problem

- **Agent**
  - Receives an observation of the current state
  - Selects an action
  - Receives a reward from the environment

- **Environment**
  - Receives an action from the agent
  - Gives a reward to the agent
  - Changes the environment state



Goal: **Find an optimal strategy (i.e. policy)**
maximizing total future rewards

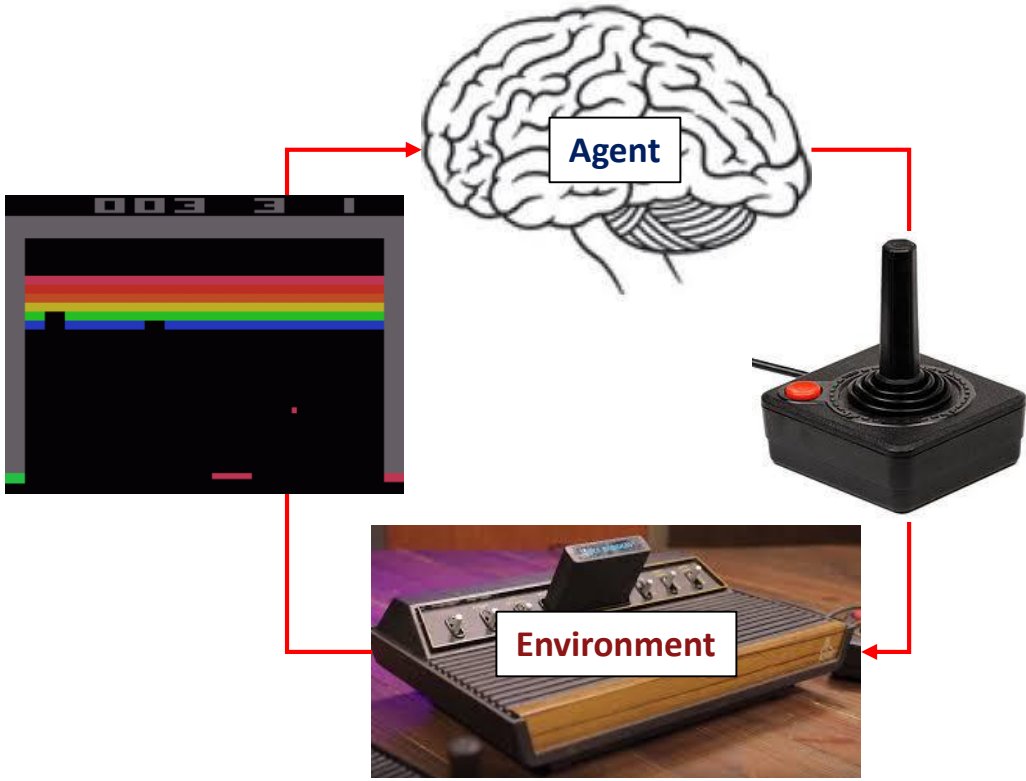
* source : UCL Course on RL (http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html)

Example: Atari Game

- Reinforcement learning is a **sequential decision making** problem

- **Agent (Player)**
  - Receives RGB screen
  - Controls joystick
  - Receives scores

- **Environment (Machine)**
  - Receives the joystick input
  - Gives scores to the player
  - Changes the environment state (e.g., memory, screen, ...)



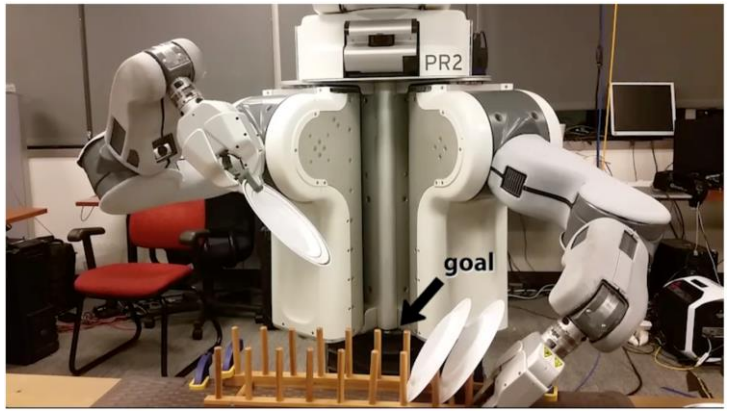
Goal: Find an optimal strategy maximizing game scores

What is Reinforcement Learning (RL)?

- Reinforcement learning *vs.* Other machine learning tasks
  - No supervisor to follow, only a scalar reward signal
  - Feedback can be delayed
  - Agent's behavior affects the subsequent data

  } **makes difficult to learn**

- If defining a reward function is difficult, one can learn from demonstrations



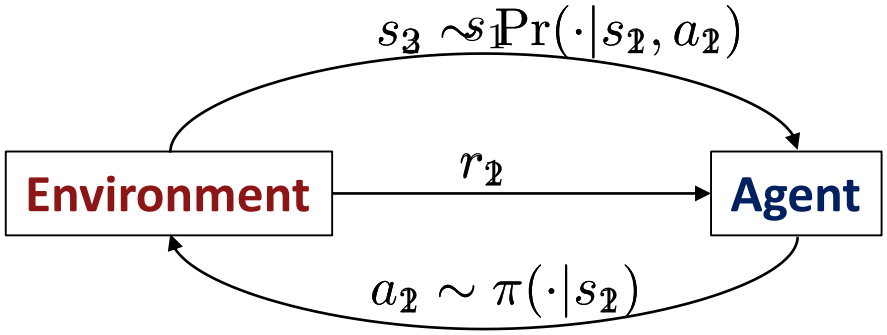
How to define reward?

  - **Imitation Learning**: copying expert's behavior
  - **Inverse RL**: inferring rewards from expert's behavior
  - **Unsupervised RL**: reinforcement learning without rewards
  - But, this lecture only covers the case when the reward oracle/function is available

* source : UC Berkeley Course on DeepRL (http://rail.eecs.berkeley.edu/deeprlcourse/)

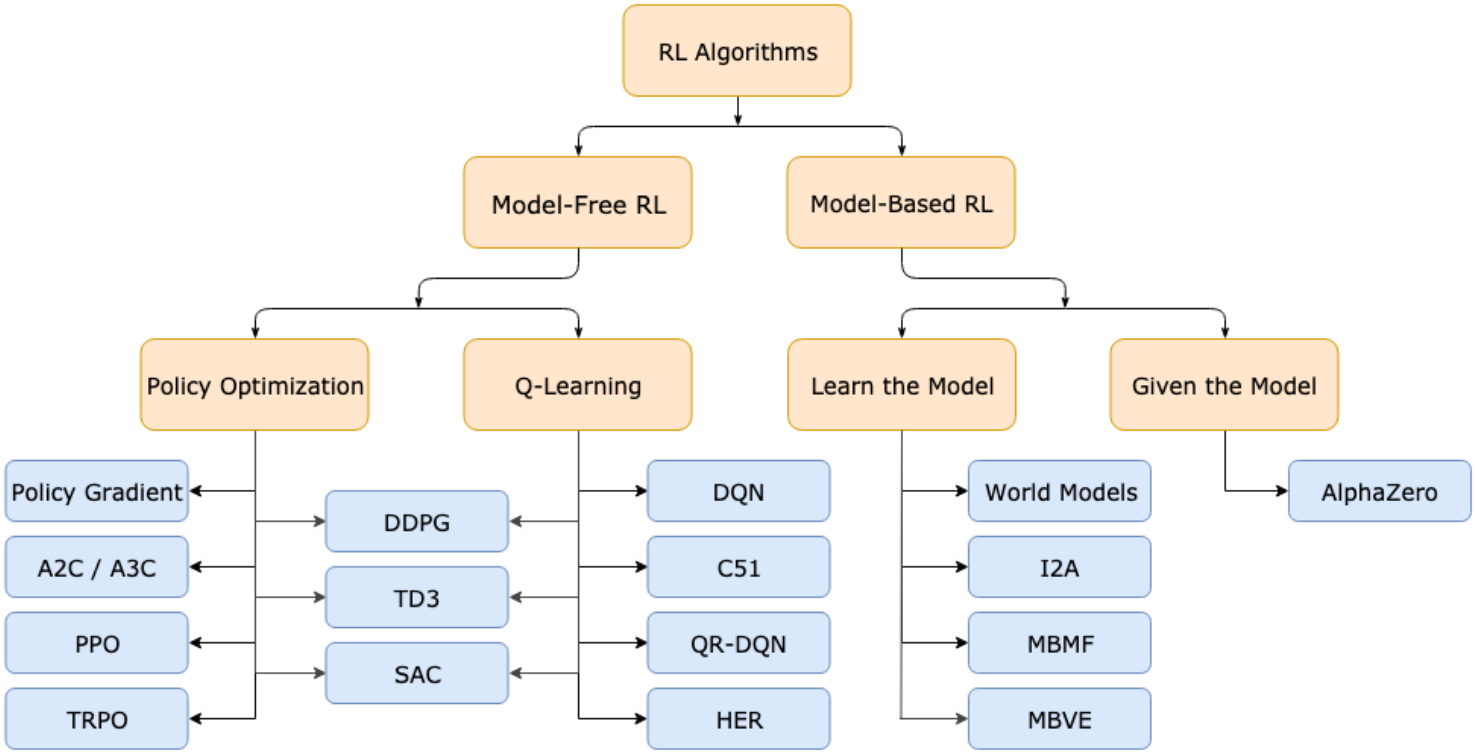
- RL can be formulated by **Markov Decision Process** $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$
  - $\mathcal{S}$: a set of states
  - $\mathcal{A}$: a set of actions
  - $\mathcal{P}$: a conditional state transition probability with Markov property, i.e.,
    $\mathcal{P}(s_t, a_t, s_{t+1}) = \Pr(s_{t+1}|s_t, a_t) = \Pr(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_1, a_1)$
  - $\mathcal{R}$: a reward function, i.e., $r_t = \mathcal{R}(s_t, a_t)$
  - $\gamma \in [0, 1]$: a discount factor

- The agent chooses an action according to $\pi(a|s)$



- **Goal:** find optimal policy $\pi(a|s)$ maximizing **total future reward** $\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t\right]$

A Taxonomy of RL Algorithms



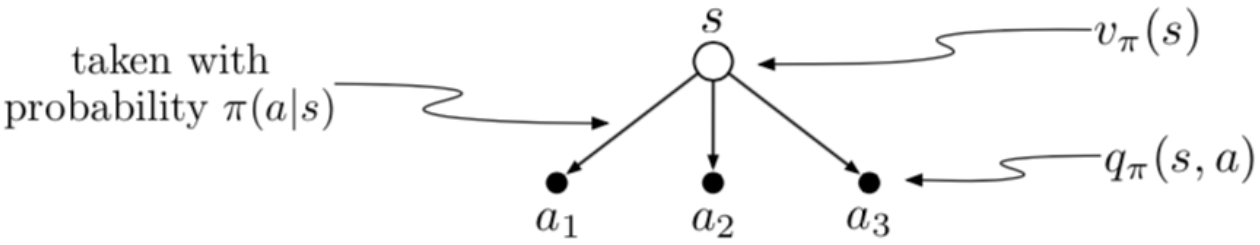
- **Model-Free** *vs.* **Model-Based RL**
  - Whether the agent has access to (or learns) a model of the environment
- **Value-based** *vs.* **policy-based algorithms**
  - Value-based methods approximate the expected total rewards from state
  - Policy-based methods in this family represent a policy explicitly
  - Some methods, e.g., Actor Critic, use both value and policy functions

* source: OpenAI Spinningup (https://spinningup.openai.com)

Table of Contents

Value Functions

- Value functions of a state s under a policy π:
  - State-value function: $v_\pi(s) = \mathbb{E}_{a_1,\ldots\sim\pi}\left[\sum_{t=1}^{\infty}\gamma^{t-1}r_t|s_1 = s\right]$
  - Action-value function: $q_\pi(s,a) = \mathbb{E}_{a_2,\ldots\sim\pi}\left[\sum_{t=1}^{\infty}\gamma^{t-1}r_t|s_1 = s, a_1 = a\right]$



  - Advantage function: $A_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$

- v_π indicates which state is good / q_π, A_π indicate which action is good under π

- Optimal value functions: $v_*(s) = \max_\pi v_\pi(s),\ \ q_*(s,a) = \max_\pi q_\pi(s,a)$
- The optimal policy can be derived from them: $\pi_*(s) = \arg\max_a q_*(s,a)$

* source : Richard Sutton and Andrew Barto, Reinforcement Learning: An Introduction, 2nd edition, 2018

Q-learning algorithm [Watkins, 1989] repeats 1-3 until convergence

1. Choose an action a from the current state s using the ε-greedy policy
   - ε-greedy chooses a random action with probability ε, otherwise $a = \arg\max_a q(s,a)$
2. Observe a reward r, a new state s'
3. Update $q(s,a) \leftarrow q(s,a) + \alpha \left[r + \gamma \max_{a'} q(s',a') - q(s,a) \right]$

Incremental update

- **Intuition:** Q-learning updates the q-value incrementally to satisfy the Bellman equation for the optimal action-value function:

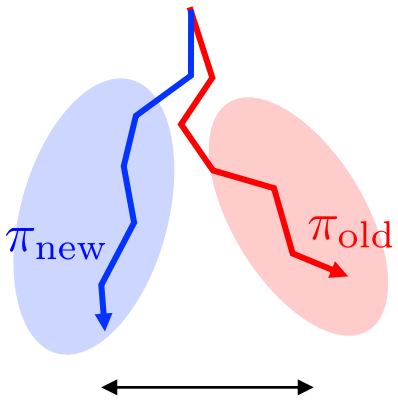
$$q_*(s,a) = \mathbb{E}_{s' \sim \Pr(\cdot|s,a)} \left[r + \gamma \max_{a'} q_*(s',a') \right]$$

- For high-dimensional state and/or action spaces, parameterize $q(s,a) \approx q(s,a;\theta)$
- The update rule for θ :

$$\theta \leftarrow \theta + \alpha \underbrace{\left[r + \gamma \max_{a'} q(s',a';\theta) - q(s,a;\theta) \right]}_{\text{called by Temporal Difference (TD) errors}} \nabla_\theta q(s,a;\theta)$$

- Q-learning is known to be unstable or even to diverge when using nonlinear function approximators such as neural networks

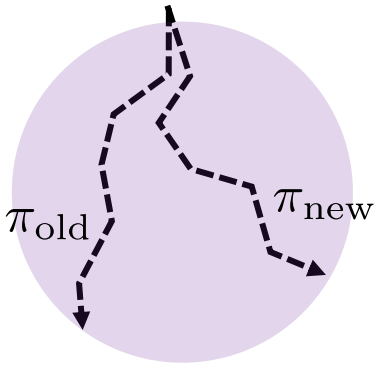
- Because even small updates to q may significantly change …



1. Data distribution
   + high-correlated sequential data

2. Correlations between $q(s, a; \theta)$ and $r + \gamma \max_{a'} q(s', a'; \theta)$

Solution: DQN (Mnih et al., 2015)



1. Experience replay buffer:
- use previous samples
- smoothing data distribution
- remove sequential correlation

2. Slowly updated target network θ^-
- use $r + \gamma \max_{a'} q(s', a'; \theta^-)$
- reducing correlations from target

- Q-learning is known to be unstable or even to diverge when using nonlinear function approximators such as neural networks

- Because even small updates to q may significantly change …

Solution: DQN [Mnih et al., 2015]

Training Deep Q-Network [Minh et al., 2015]

$$\mathcal{L} = \mathbb{E}_{\underbrace{(s,a,r,s')\sim U(D)}_{\text{replay buffer}}}\left[\left(r + \gamma \max_{a'} q(s', a'; \underbrace{\theta^-}_{\text{target network}}) - q(s, a; \theta)\right)^2\right]$$

- Add every observation (s, a, r, s') to replay buffer D
- Update deep Q-network $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}$
- Update target network $\theta^- \leftarrow \theta$ at every C steps

uffer:
es
tribution
correlation

1.

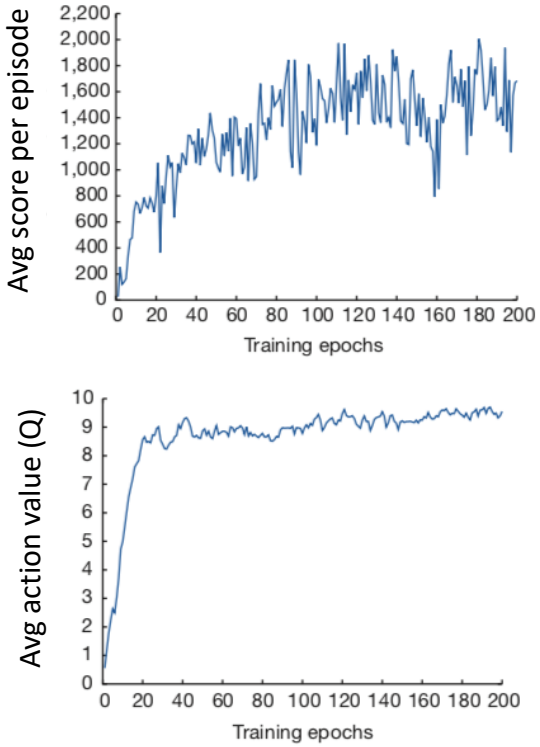
2. Correlations between $q(s, a; \theta)$ and $r + \gamma \max_{a'} q(s', a'; \theta)$

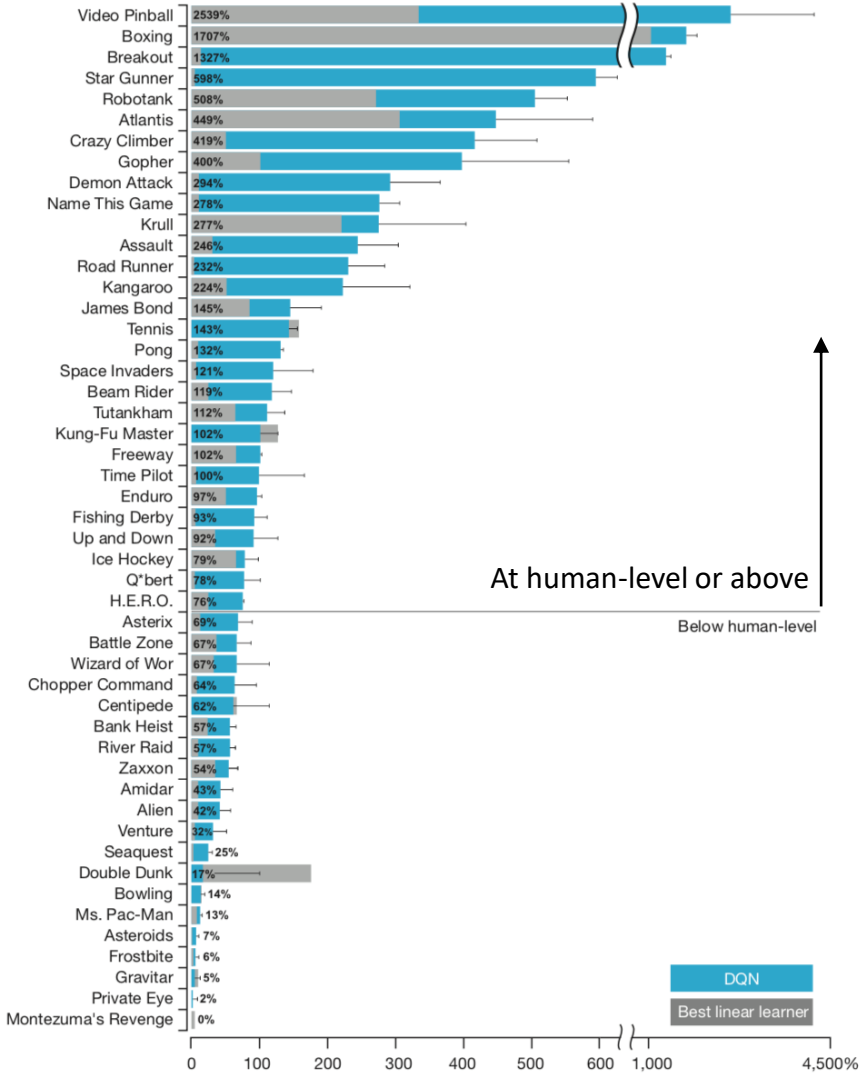
rk θ^-
- use $r + \gamma \max_{a'} q(s', a'; \theta^-)$
- reducing correlations from target

Deep Q-Network for Atari Games

- [Minh et al., 2015] use same architecture/hyper-parameters for all Atari games
  ⇒ Robustness of DQN

- Training curve





- DQN Breakout video

* source : Minh et al., Human-level Control through Deep Reinforcement Learning, Nature 2015

- What is the state-of-the-art DQN???
- Examine six extensions to the DQN algorithm and integrate them into a single integrated agent [Hessel et al., 2018]

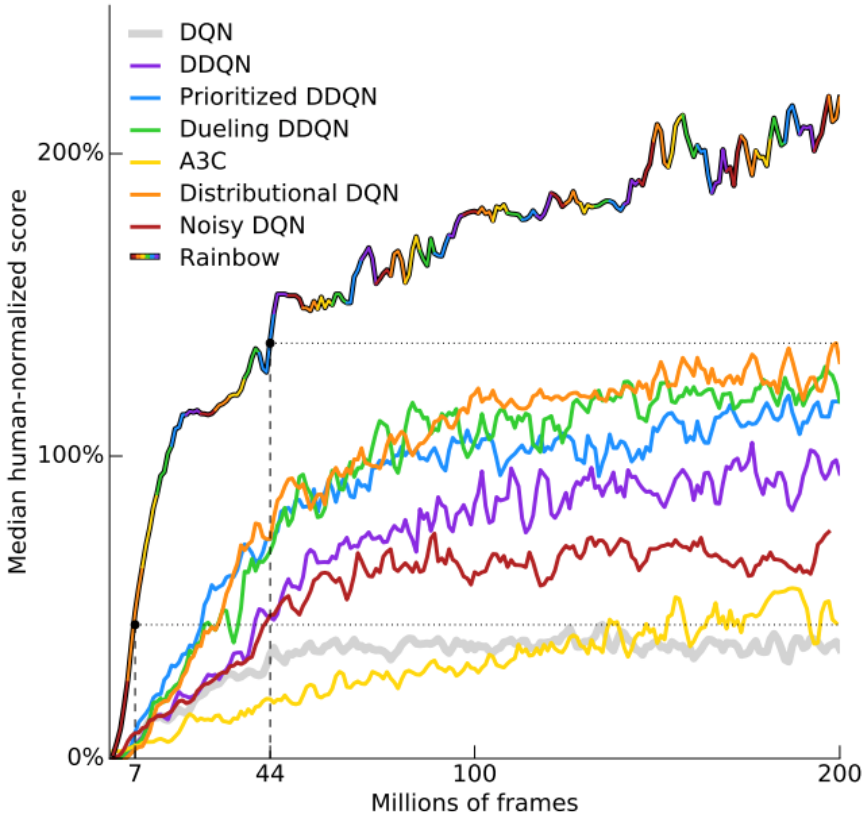


Figure. Median human-normalized performance across 57 Atari games

- In this lecture, DDQN, dueling and prioritized replay will be covered

* source : Hessel et al., Rainbow: Combining Improvements in Deep Reinforcement Learning, AAAI 2018

- Q-learning is known to **overestimate** action values

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

because the max step $\max q(\cdot, \cdot)$ is used to update the same function $q(\cdot, \cdot)$

- In practice, overestimation errors will differ for actions $\Longrightarrow$ poor policy

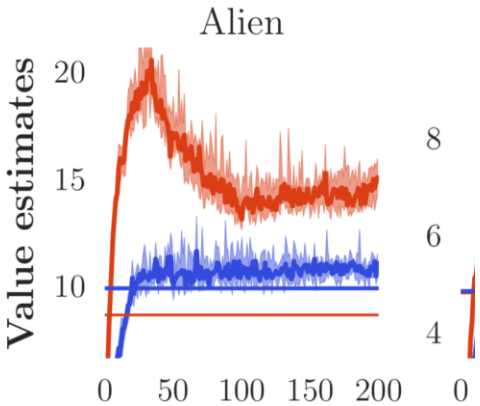
Double Q-learning [van Hasselt, 2010] separates **selection** and **evaluation**:

$$\theta_1 \leftarrow \theta_1 + \alpha \left[r + \gamma q(s', \arg\max_{a'} q(s', a'; \theta_1); \theta_2) - q(s, a; \theta_1) \right] \nabla_{\theta_1} q(s, a; \theta_1)$$

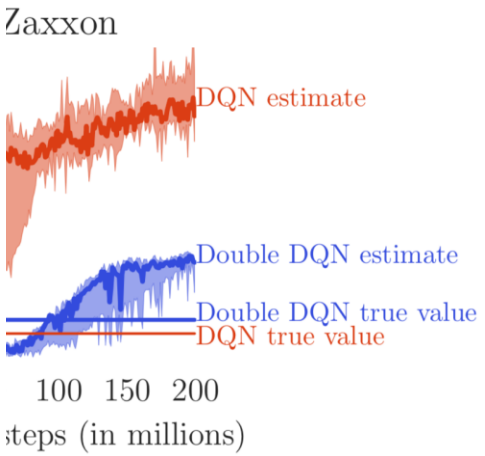
- **Double DQN** [van Hasselt et al., 2015] uses $\theta_1 = \theta$ and $\theta_2 = \theta^-$ (target network)

Advanced Deep Q-learnir

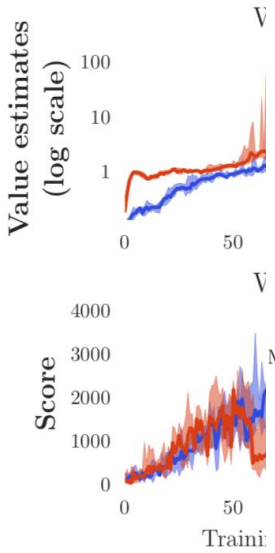
- Value estimations of



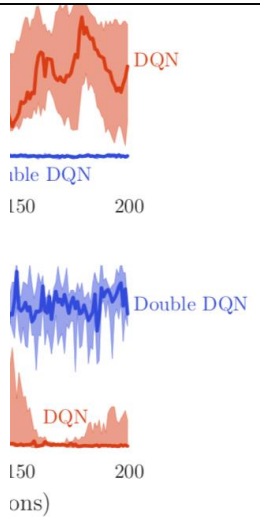
es



- Double DQN learns A



Improve scores over almost games



- DQN samples transitions (s, a, r, s') **uniformly** from **experience replay buffer**

- **Problem:** Unimportant data (e.g., small TD error) might be used with same probability as important ones $\Longrightarrow$ sample inefficiency
- **Solution** [Schaul et al., 2016]: **Prioritize** data and **sample them based on the priority**

Q1) How to prioritize?

$$\theta \leftarrow \theta + \alpha \boxed{r + \gamma \max_{a'} q(s', a'; \theta) - q(s, a; \theta)} \nabla_\theta q(s, a; \theta)$$

$\Rightarrow$ Use TD error δ — measure how much update is required

Q2) How to sample?

- ~~Greedy: sample transitions of maximum TD errors~~ some transitions are never selected
- Stochastically sample with probability $P(i) = p_i^\alpha / \sum_k p_k^\alpha$
  - Proportional: $p_i = |\delta_i| + \epsilon$
  - Rank-based: $p_i = 1/\text{rank}(i)$

sampling probability of ith data

- Prioritized replay $P(D)$ introduces **bias**
  - Because original Q-learning with/without replay buffer uses uniform distribution:

$$\mathbb{E}_{(s,a,r,s')\sim U(D)}[\delta^2] \neq \mathbb{E}_{(s,a,r,s')\sim P(D)}[\delta^2]$$

where $\delta = r + \gamma \max_{a'} q(s', a'; \theta^-) - q(s, a; \theta)$

- To correct this bias, use **importance-sampling** weights $w_i = \left(\frac{1}{N}\frac{1}{P(i)}\right)^{\beta}$
  - In practice, increase β linearly from β_0 to 1

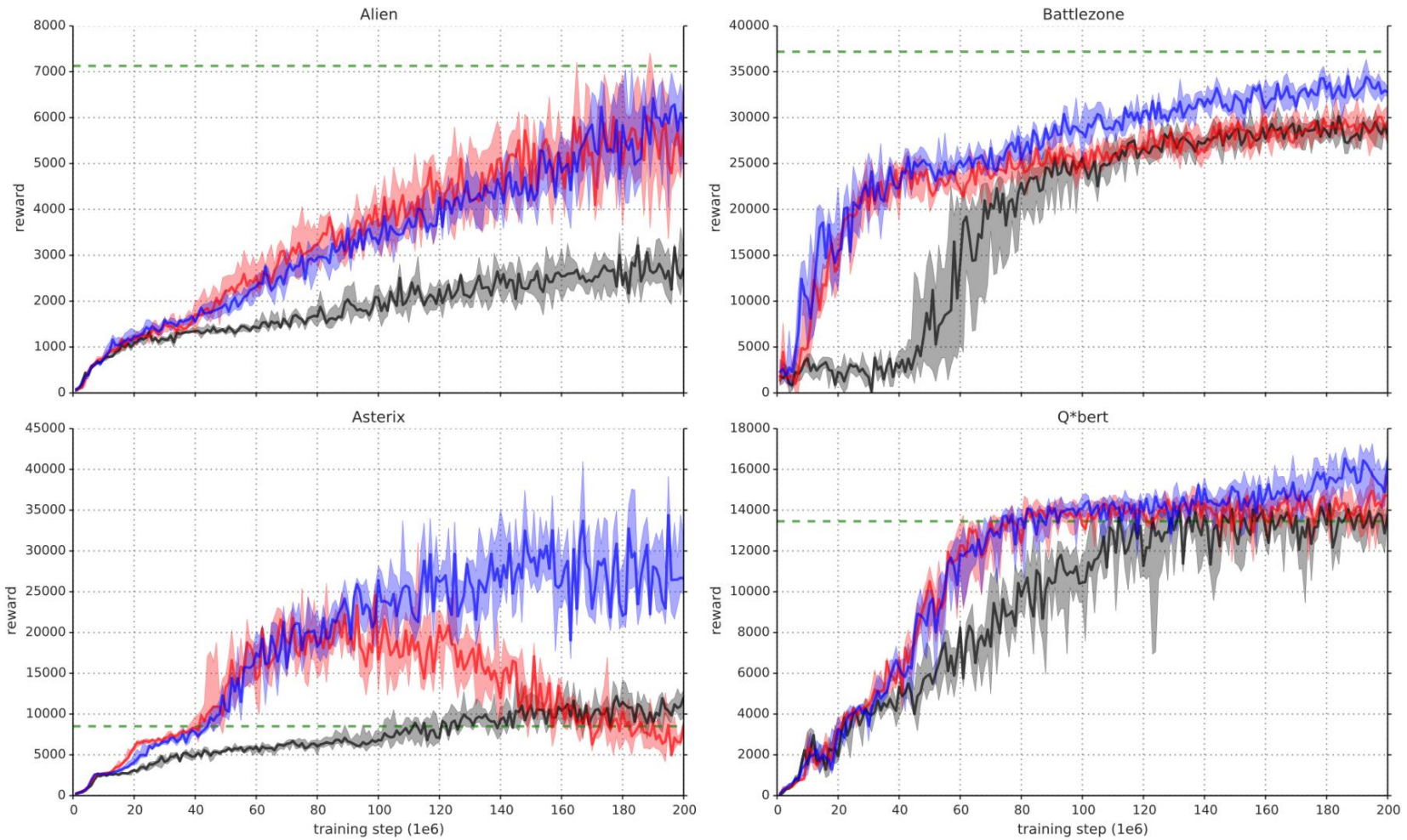
DQN with prioritization [Schaul et al., 2016]

1. Update parameters using $\nabla_\theta \mathcal{L}$ where $\mathcal{L} = \mathbb{E}_{(s,a,r,s')\sim P(D)}[w\delta^2]$
2. Update priorities for sampled transitions $p_i \leftarrow |\delta_i|$

- Prioritized replay buffer can be combined with Double Q-learning

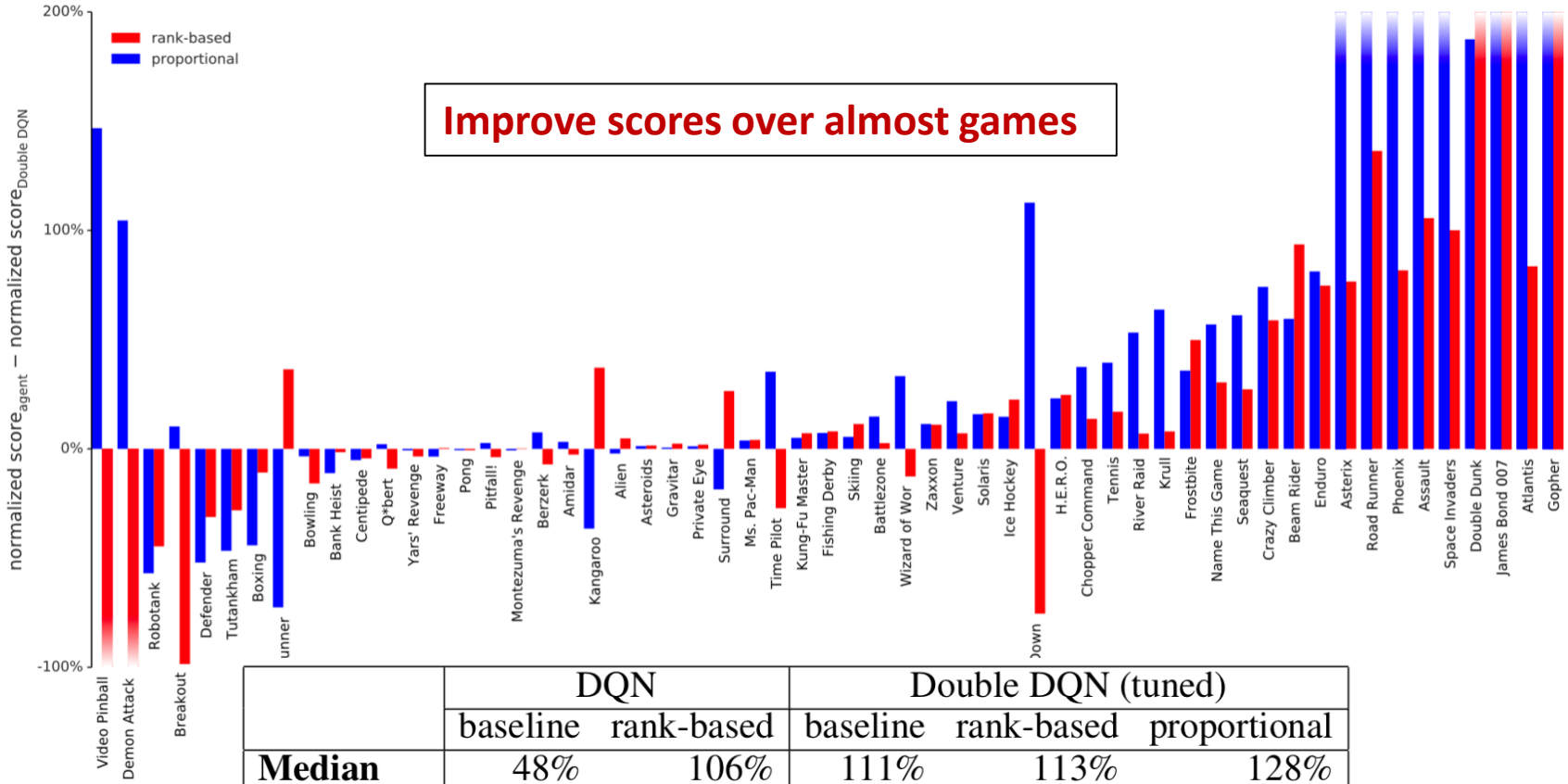
Advanced Deep Q-learning (2) Prioritized Replay

- Learning speed compared to uniform sampling



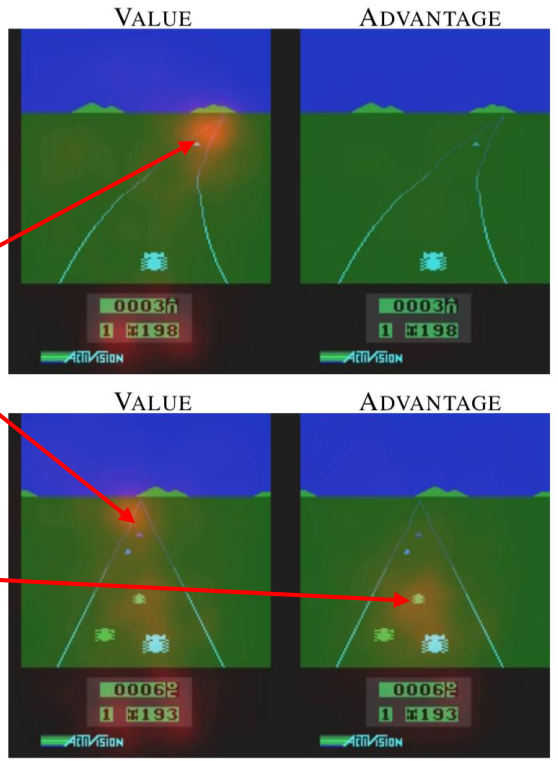
* source : Schaul et al., Prioritized Experience Replay, ICLR 2016

- Comparison scores with Double DQN on Atari games



	DQN		Double DQN (tuned)		
	baseline	rank-based	baseline	rank-based	proportional
Median	48%	106%	111%	113%	128%
Mean	122%	355%	418%	454%	551%
> baseline	–	41	–	38	42
> human	15	25	30	33	33
# games	49	49	57	57	57

* source : Schaul et al., Prioritized Experience Replay, ICLR 2016



Intuition from an example: driving car

- In many states, it is unnecessary to estimate the value of each action choice
  - State-value function pays attention to the road
- In some states, left/right actions should be taken to avoid collision
  - Advantage function pays attention to the front of car when action selection is crucial
- Recall advantage function: $A_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$

Idea [Wang et al., 2016] Decouple action-value q to state-value v and advantage A

$$q(s,a;\theta,\phi_v,\phi_A) = v(s;\theta,\phi_v) + A(s,a;\theta,\phi_A)$$

learn which state is valuable without effect of action

* source: Wang et al., Dueling Network Architectures for Deep Reinforcement Learning, ICML 2016

- In $q = v + A$, v can be arbitrary given an action-value q

Q) How to force v to be the (unique, correct) state-value?

A) Make the maximum of the advantage be zero

$$q(s, a; \theta, \phi_v, \phi_A) = v(s; \theta, \phi_v) + (A(s, a; \theta, \phi_A) - \max_{a'} A(s, a'; \theta, \phi_A))$$

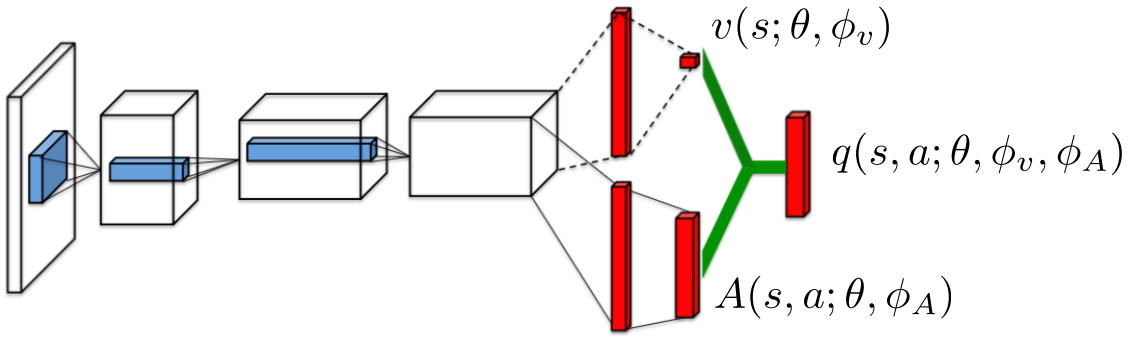
- Then, $q(s, a^*; \theta, \phi_v, \phi_A) = v(s; \theta, \phi_v)$

  this can be derived from $\pi(s) = \arg\max_a q(s, a)$

- In practice, use average instead of maximum for learning stability:

$$q(s, a; \theta, \phi_v, \phi_A) = v(s; \theta, \phi_v) + \left(A(s, a; \theta, \phi_A) - \frac{1}{m} \sum_{a'} A(s, a'; \theta, \phi_A)\right)$$

- **Dueling architecture**
  [Wang et al., 2016]

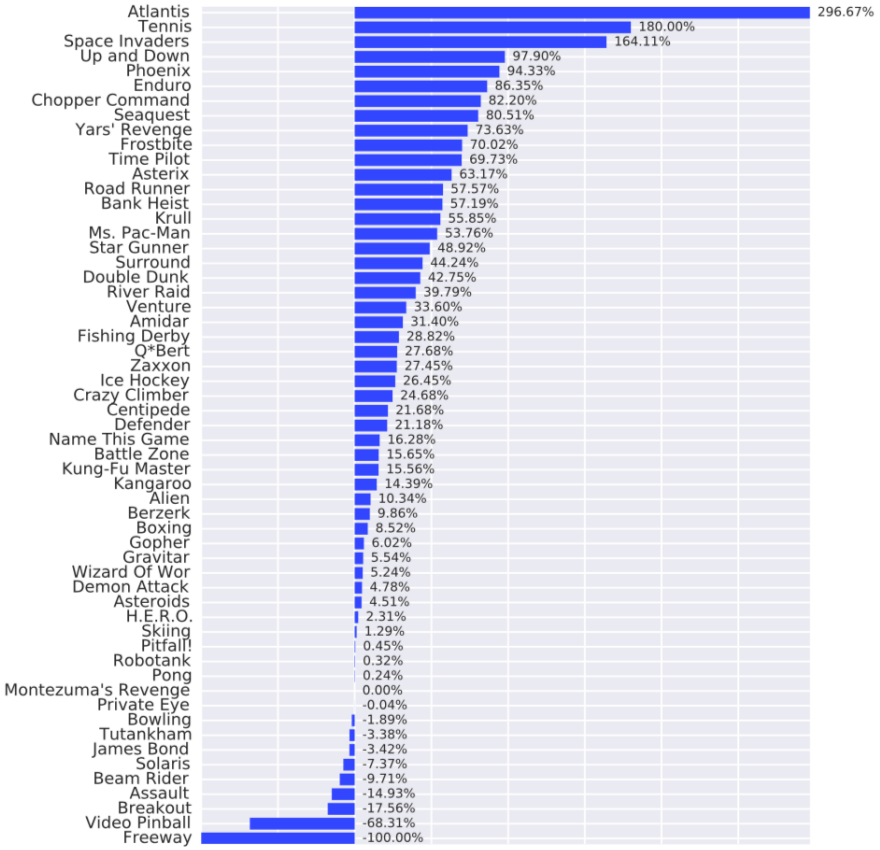


* source: Wang et al., Dueling Network Architectures for Deep Reinforcement Learning, ICML 2016

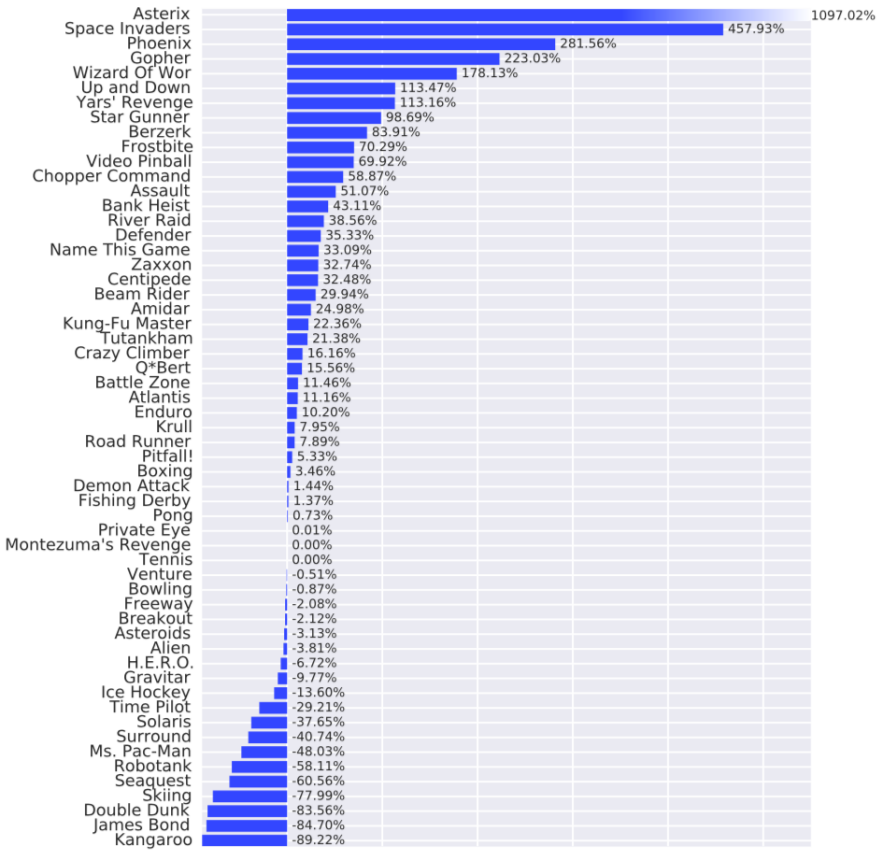
Advanced Deep Q-learning (3) Dueling Architecture

- This dueling architecture also improves DQN performance

vs Double DQN



vs Double DQN + Prioritized replay



* source: Wang et al., Dueling Network Architectures for Deep Reinforcement Learning, ICML 2016

Table of Contents

- Value-based methods (e.g., Q-learning) optimize policies indirectly:

$$\text{Find } q(s,a;\theta) \approx q_*(s,a) \quad \Rightarrow \quad \pi(s;\theta) = \arg\max_a q(s,a;\theta)$$

- **Policy gradient methods** (e.g., REINFORCE, Actor-Critic) **optimize policies directly** via maximizing total reward $\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t\right]$:

$$\arg\max_\theta \mathbb{E}_{a_t \sim \pi(\cdot|s_t;\theta)} \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t\right]$$ where θ is the policy parameters

  - **Approximated value functions might be used** with these methods to resolve optimization issues such as high variance

- **Policy gradient theorem:** If $J(\theta)$ is **the above objective**, then

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi(a|s;\theta) q_{\pi_\theta}(s,a)\right]$$

  - Simply, higher action-value $q_{\pi_\theta}(s,a)$ increases action probability $\pi(a|s;\theta)$
  - Action evaluation & selection should be performed by same policy, i.e., **on-policy**

REINFORCE [Willams, 1992] uses Monte-Carlo estimates of the policy gradient

1. Sample an episode $\{s_1, a_1, r_1, \ldots, s_T, a_T, r_T\} \sim \pi_\theta$
2. Compute $\Delta\theta \leftarrow \sum_{t=1}^{T} \nabla_\theta \log \pi(a_t|s_t;\theta) \left(\sum_{s=t}^{T} \gamma^{s-t} r_s \right)$
3. Update $\theta \leftarrow \theta + \alpha \Delta\theta$

Unbiased estimator of $q_{\pi_\theta}(s_t, a_t)$

- **Issue:** REINFORCE has high variance when estimating gradients
- **Solution:** Use any baseline function $b(s)$ not depending on actions

$$\begin{aligned}
\mathbb{E}_{\pi_\theta}\left[\nabla \log \pi(a|s;\theta) b(s)\right] &= \sum_s \mu(s) \sum_a \pi(a|s;\theta) \frac{\nabla \pi(a|s;\theta)}{\pi(a|s;\theta)} b(s) \\
&= \sum_s \mu(s) b(s) \nabla \sum_a \pi(a|s;\theta) \\
&= \sum_s \mu(s) b(s) \nabla 1 = 0
\end{aligned}$$

REINFORCE [Willams, 1992] uses Monte-Carlo estimates of the policy gradient

1. Sample an episode $\{s_1, a_1, r_1, \ldots, s_T, a_T, r_T\} \sim \pi_\theta$
2. Compute $\Delta\theta \leftarrow \sum_{t=1}^{T} \nabla_\theta \log \pi(a_t|s_t;\theta) \left(\sum_{s=t}^{T} \gamma^{s-t} r_s\right)$
3. Update $\theta \leftarrow \theta + \alpha\Delta\theta$

Unbiased estimator of $q_{\pi_\theta}(s_t, a_t)$

- **Issue:** REINFORCE has high variance when estimating gradients
- **Solution:** Use any baseline function $b(s)$ not depending on actions
  - $\mathbb{E}_{\pi_\theta}[\nabla \log \pi(a|s;\theta) b(s)] = 0$
  - $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \log \pi(a|s;\theta)(q_{\pi_\theta}(s,a) - b(s))]$

This can reduce the variance

- Which $b(s)$ should be used?
  - One natural choice is $b(s) = v_{\pi_\theta}(s)$ since $\mathbb{E}_{a\sim\pi(\cdot|s;\theta)}[q_{\pi_\theta}(s,a) - v_{\pi_\theta}(s)] = 0$
  - In practice, use $b(s) = v(s;w) \approx v_{\pi_\theta}(s)$ with parameters w and learn the function using TD errors such as Q-learning [Sutton et al., 2000]

Issues in ”vanilla” policy gradient methods such as REINFORCE

- Hard to choose step-size α
  - small changes in parameter space can cause poor policy
- Only one gradient step per each sample
  - Sample inefficiency

Solution: formulate an optimization problem on generated data from old policy

- That allows small changes in policy space
- That guarantees improvement of policy performance

Trust Region Policy Optimization [Schulman et al., 2015]: for each iteration, solve

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \mathbb{E}_{\pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A_{\pi_{\theta_{\text{old}}}}(s,a) \right] \\ \text{subject to} \quad & \mathbb{E}_{\pi_{\theta_{\text{old}}}} \left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s) || \pi_\theta(\cdot|s)) \right] \leq \delta \end{aligned}$$

$A_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$ is also approximated by neural networks

Derive TRPO

- Let $\eta(\pi) = \mathbb{E}_\pi[v_\pi(s_1)] = \mathbb{E}_\pi[\sum_{t=1}^{\infty} \gamma^{t-1} r_t]$ be the performance of a policy
- This performance can be written as

$$
\begin{aligned}
\eta(\pi) &= \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \right] \\
&= \eta(\pi_{\text{old}}) + \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t - v_{\pi_{\text{old}}}(s_1) \right] \\
&= \eta(\pi_{\text{old}}) + \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} (\overbrace{r_t + \gamma v_{\pi_{\text{old}}}(s_{t+1})}^{q_{\pi_{\text{old}}}(s,a)} - v_{\pi_{\text{old}}}(s_t)) \right] \\
&= \eta(\pi_{\text{old}}) + \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} A_{\pi_{\text{old}}}(s_t, a_t) \right] \\
&= \eta(\pi_{\text{old}}) + \sum_s \rho_\pi(s) \sum_a \pi(a|s) A_{\pi_{\text{old}}}(s,a)
\end{aligned}
$$

where $\rho_\pi(s) = \sum_{t=1}^{\infty} \gamma^{t-1} \Pr(s_t = s|\pi)$

Derive TRPO

- Let $\eta(\pi) = \mathbb{E}_\pi[v_\pi(s_1)] = \mathbb{E}_\pi[\sum_{t=1}^{\infty} \gamma^{t-1} r_t]$ be the performance of a policy
- This performance can be written as

$$\eta(\pi) = \eta(\pi_{\text{old}}) + \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} A_{\pi_{\text{old}}}(s_t, a_t) \right]$$

$$= \eta(\pi_{\text{old}}) + \sum_s \rho_\pi(s) \sum_a \pi(a|s) A_{\pi_{\text{old}}}(s, a)$$

- Define $\mathcal{L}_{\pi_{\text{old}}}(\pi) = \eta(\pi_{\text{old}}) + \sum_s \rho_{\pi_{\text{old}}}(s) \sum_a \pi(a|s) A_{\pi_{\text{old}}}(s, a)$

- $\mathcal{L}_{\pi_{\theta_{\text{old}}}}(\cdot)$ is a local approximation of $\eta(\cdot)$ at $\theta = \theta_{\text{old}}$:

$$\mathcal{L}_{\pi_{\theta_{\text{old}}}}(\pi_{\theta_{\text{old}}}) = \eta(\pi_{\theta_{\text{old}}})$$

$$\nabla_\theta \mathcal{L}_{\pi_{\theta_{\text{old}}}}(\pi_\theta)\big|_{\theta=\theta_{\text{old}}} = \nabla_\theta \eta(\pi_\theta)\big|_{\theta=\theta_{\text{old}}}$$

- For fixed θ_{old}, we can omit $\eta(\pi_{\theta_{\text{old}}})$: $\mathcal{L}_{\pi_{\theta_{\text{old}}}}(\pi_\theta) = \mathbb{E}_{\pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A_{\pi_{\theta_{\text{old}}}}(s, a) \right]$

Theorem [Schulman et al., 2015]

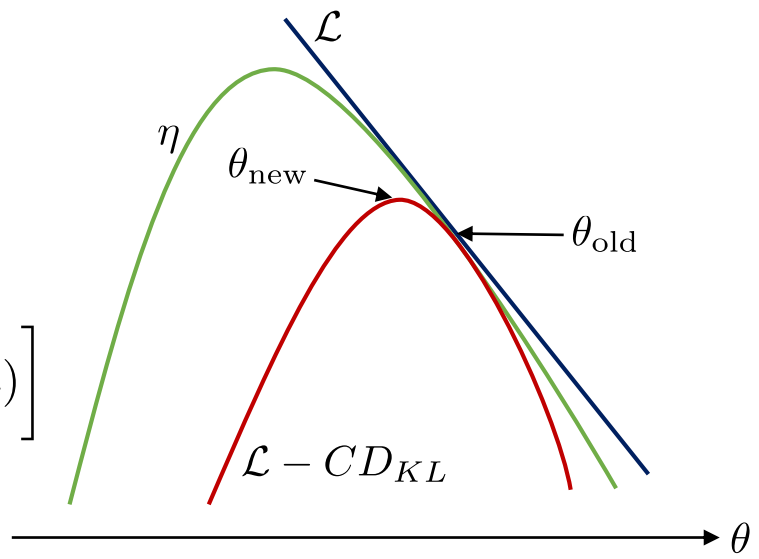
- $\eta(\pi_\theta) \geq \mathcal{L}_{\pi_{\theta_{\text{old}}}}(\pi_\theta) - CD_{\text{KL}}^{\max}(\pi_{\theta_{\text{old}}}, \pi_\theta)$
- C is some constant and $D_{\text{KL}}^{\max}(\pi_{\theta_{\text{old}}}, \pi_\theta) = \max_s D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s)||\pi_\theta(\cdot|s))$

- Policy iteration **guarantees non-decreasing performance**:

$$\theta_{\text{new}} \leftarrow \arg\max_\theta \mathcal{L}_{\pi_{\theta_{\text{old}}}}(\pi_\theta) - CD_{\text{KL}}^{\max}(\pi_{\theta_{\text{old}}}, \pi_\theta)$$

- In practice,
  - Theoretical guaranteed C updates very small steps in policy
  - Use a constraint instead of the penalty
  - Use average instead of maximum

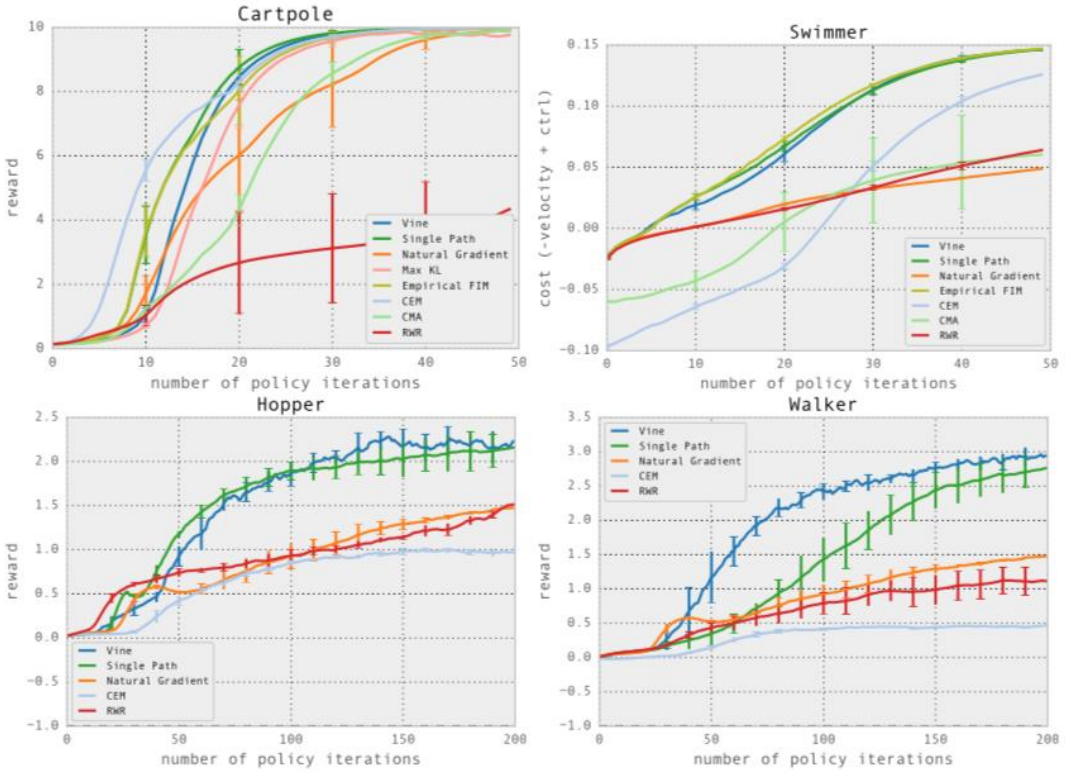
$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \mathcal{L}_{\pi_{\theta_{\text{old}}}}(\pi_\theta) = \mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A_{\pi_{\theta_{\text{old}}}}(s,a)\right] \\ \text{subject to} \quad & \mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s)||\pi_\theta(\cdot|s))\right] \leq \delta \end{aligned}$$



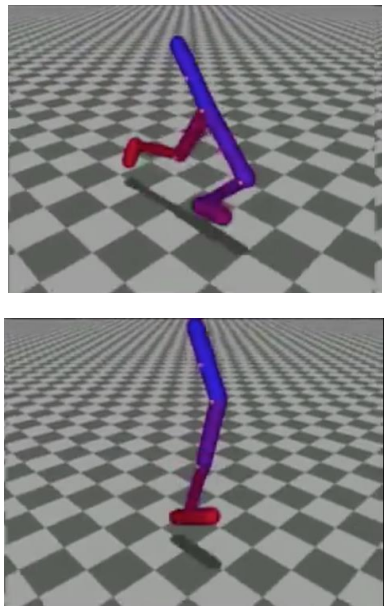
* source : Deep RL Bootcamp Lecture (https://youtu.be/xvRrgxcpaHY)

Trust Region Policy Optimization (TRPO)

- How to solve this optimization? Conjugate gradient algorithm
  - (1) Compute search direction: making a linear approximation to the objective and a quadratic approximation to the constraint
  - (2) Perform a line search in that direction
- Training curves (TRPO: vine & single path)



- TRPO agent video



* source : Schulman et al., Trust Region Policy Optimization, ICML 2015

Issues in TRPO

- To solve the optimization problem, quadratic approximation for the constraint is required
- In some cases, such approach is not possible

Adaptive KL Penalty Coefficient [Schulman et al., 2017]

$$\arg\max_\theta \mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}A(s,a)\right] - \beta\mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[\text{KL}\left(\pi_{\theta_{\text{old}}}(\cdot|s)||\pi_\theta(\cdot|s)\right)\right]$$

- KL divergence is small/large $\Rightarrow$ decrease/increase β, respectively.
- For each iteration, do SGD on the above objective multiple times
- This needs only first-order derivatives

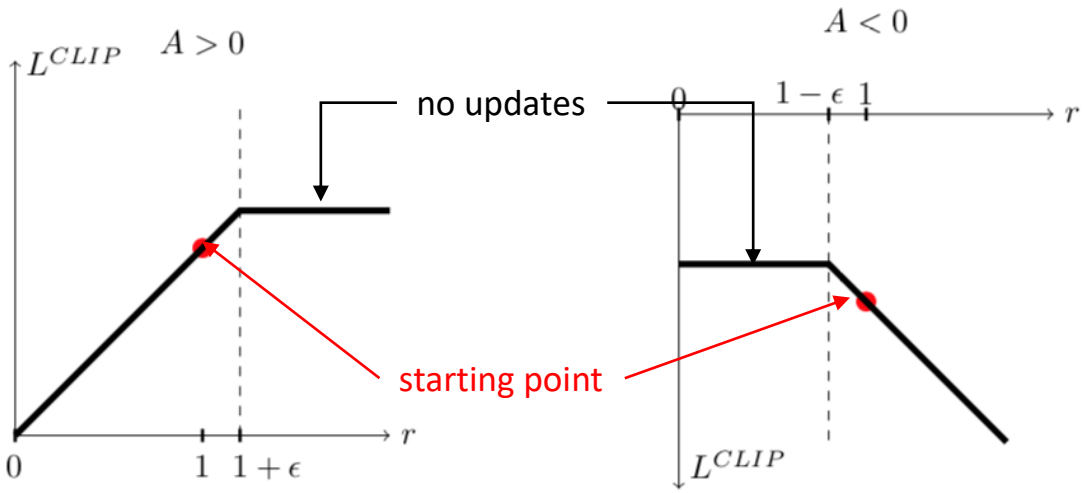
- Still, this has limitations:
  - Hard to use multi-output architectures (e.g., policy & value functions) due to the KL divergence term
  - Empirically poor performance when using deep CNNs / RNNs

Clipped Surrogate Objective [Schulman et al., 2017]

$$\mathcal{L}^{\mathrm{CLIP}}_{\pi_{\theta_{\mathrm{old}}}}(\pi_\theta) = \mathbb{E}_{\pi_{\theta_{\mathrm{old}}}}\left[\min(r(\theta)A, \mathrm{clip}(r(\theta), 1-\epsilon, 1+\epsilon)A)\right]$$

where $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\mathrm{old}}}(a|s)}$

- The objective suppresses changes in policy without KL divergence
- This figure simply shows how $\mathcal{L}^{\mathrm{CLIP}}$ works

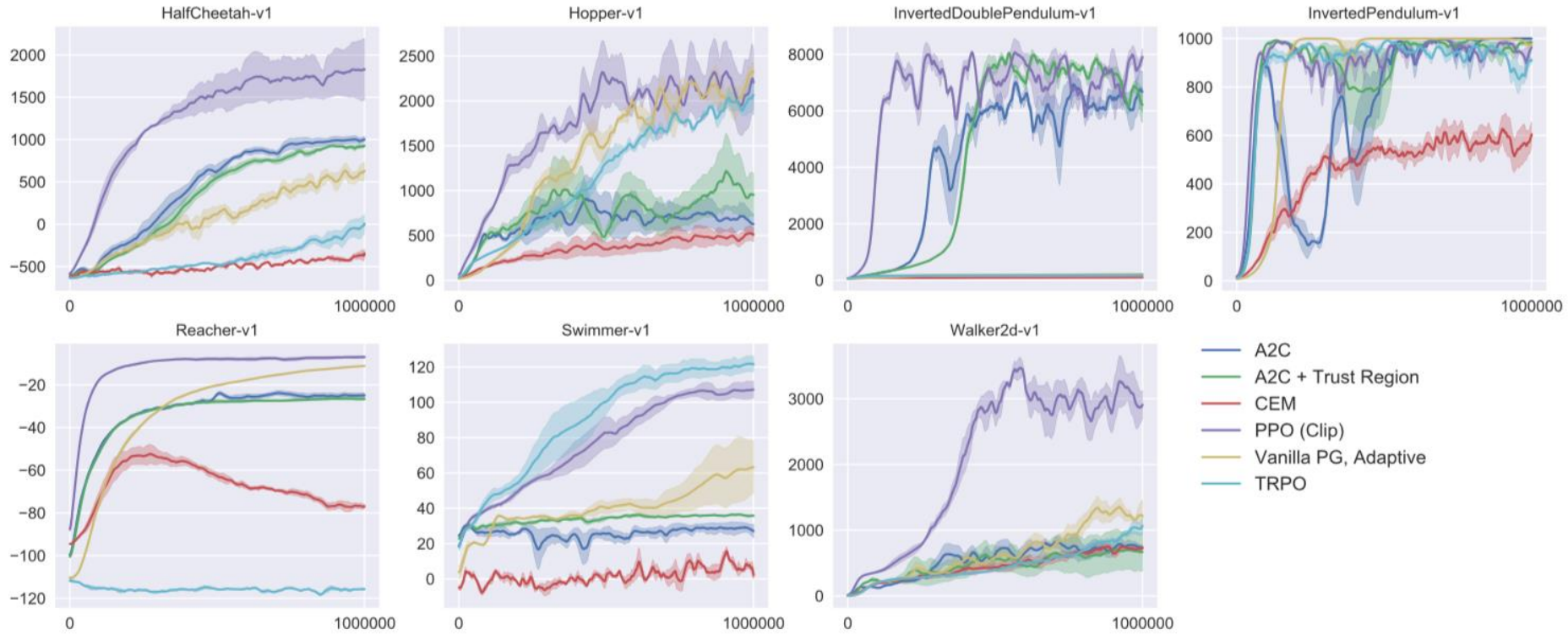


- This objective can be used with multi-output architectures

* source : Schulman et al., Proximal Policy Optimization Algorithms, 2017

Proximal Policy Optimization Algorithms

- On MuJoCo Environments, PPO (clip) outperforms other policy gradient methods



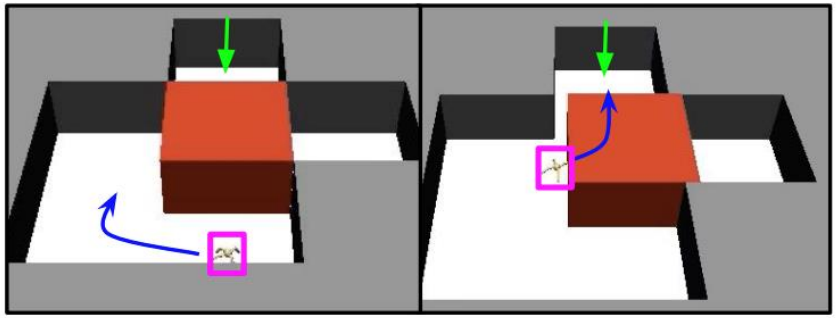
- PPO agent video

* source : Schulman et al., Proximal Policy Optimization Algorithms, 2017

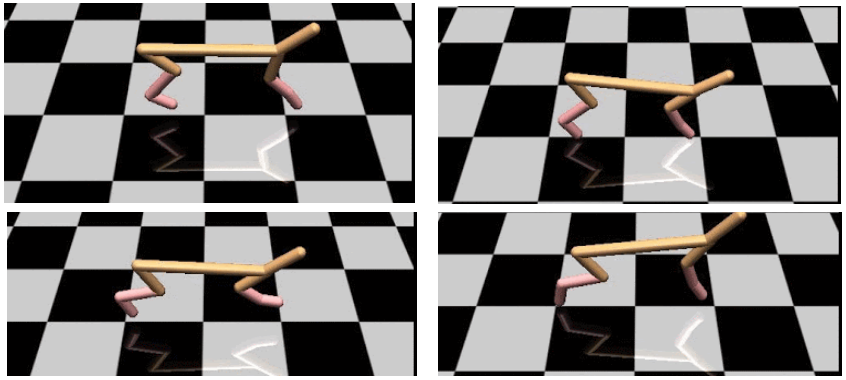
Summary

- Reinforcement learning is another field of machine learning
  - RL agents learn the best strategy using only scalar rewards, no supervision
  - There are many various algorithms: Q-learning, actor-critic, policy optimization

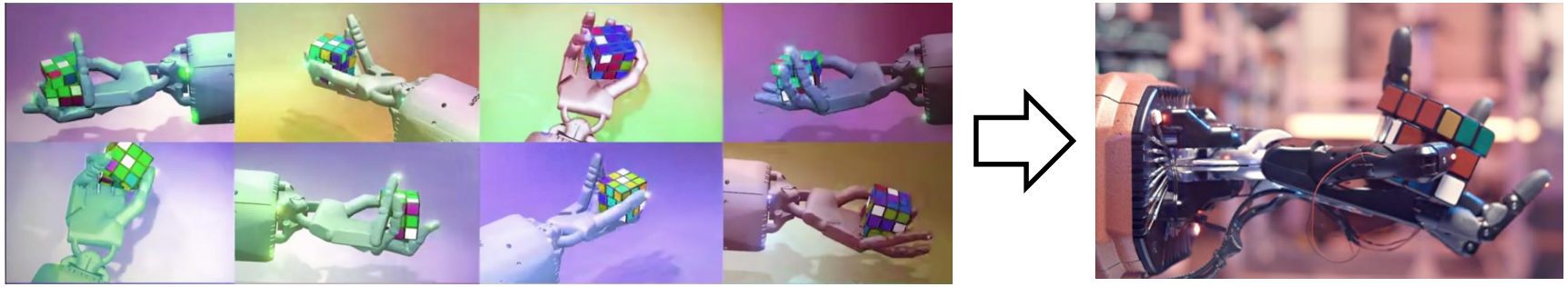
- Other interesting topics



Hierarchical RL [Nachum et al., 2018]



Unsupervised RL [Eysenbach et al., 2018]



Sim-to-Real Transfer [OpenAI et al., 2019]

References

[Watkins, 1989] Learning from Delayed Rewards, Ph.D. thesis, University of Cambridge, 1989
link: http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf

[Watkins, 1992] Q-learning, Machine Learning, 1992
link: https://link.springer.com/article/10.1007/BF00992698

[Willams, 1992] Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning, Machine Learning, 1992
link: https://link.springer.com/article/10.1007/BF00992696

[Sutton et al., 2000] Policy gradient methods for reinforcement learning with function approximation: actor-critic algorithms with value function approximation, NIPS 2000
link: https://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf

[Minh et al., 2015] Human-level Control through Deep Reinforcement Learning, Nature 2015
link: https://www.nature.com/articles/nature14236

[van Hasselt et al., 2016] Deep Reinforcement Learning with Double Q-learning, AAAI 2016
link: https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/download/12389/11847

[van Hasselt, 2010] Double Q-learning, NIPS 2010
link: https://papers.nips.cc/paper/3964-double-q-learning

[Schaul et al., 2016] Prioritized Experience Replay, ICLR 2016
link: https://arxiv.org/abs/1511.05952

[Wang et al., 2016] Dueling Network Architectures for Deep Reinforcement Learning, ICML 2016
link: http://proceedings.mlr.press/v48/wangf16.pdf

References

[Schulman et al., 2015] Trust Region Policy Optimization, ICML 2015
link: http://proceedings.mlr.press/v37/schulman15.pdf

[Schulman et al., 2017] Proximal Policy Optimization Algorithms, 2017
link: https://arxiv.org/abs/1707.06347

[Hessel et al., 2018] Rainbow: Combining Improvements in Deep Reinforcement Learning, AAAI 2018
link: https://arxiv.org/abs/1710.02298

[Nachum et al., 2018] Data-efficient hierarchical reinforcement learning, NeurIPS 2018
link: http://papers.nips.cc/paper/7591-data-efficient-hierarchical-reinforcement-learning

[Eysenbach et al., 2018] Diversity is All You Need: Learning Diverse Skills without a Reward Function, ICLR 2018
link: https://arxiv.org/pdf/1802.06070

[OpenAI et al., 2019] Solving Rubik's Cube with a Robot Hand, 2019
link: https://arxiv.org/abs/1910.07113

Books

Sutton and Barto, Reinforcement Learning: An Introduction, 2nd edition, 2018
link: http://incompleteideas.net/book/the-book-2nd.html

Lectures

UCL Course on Reinforcement Learning
link: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

UC Berkeley Course on Deep Reinforcement Learning
link: http://rail.eecs.berkeley.edu/deeprlcourse/

Deep RL Bootcamp Lectures
link: https://youtu.be/xvRrgxcpaHY