2019 CS420, Machine Learning, Lecture 13

# Introduction to Deep Reinforcement Learning

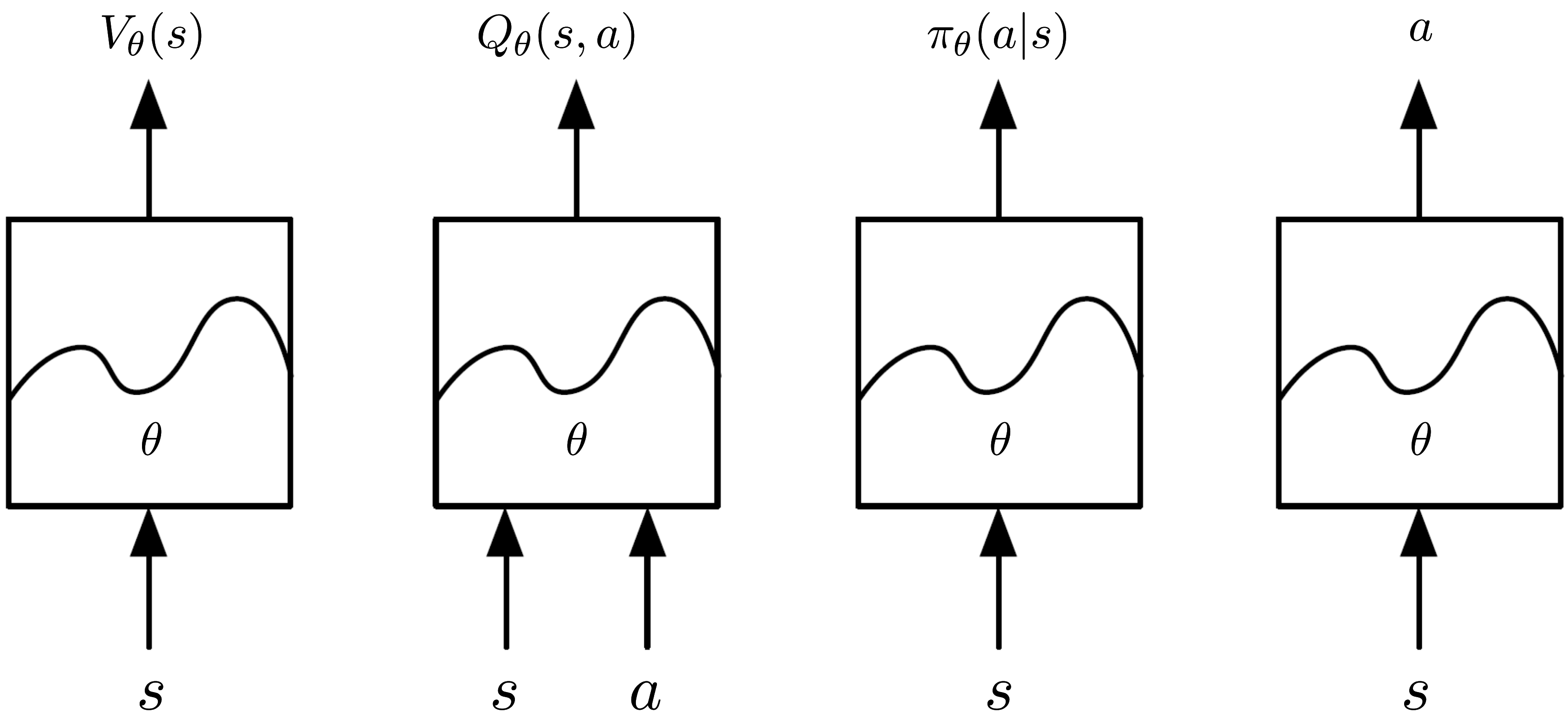
Weinan Zhang

Shanghai Jiao Tong University

http://wnzhang.net

http://wnzhang.net/teaching/cs420/index.html

# Value and Policy Approximation



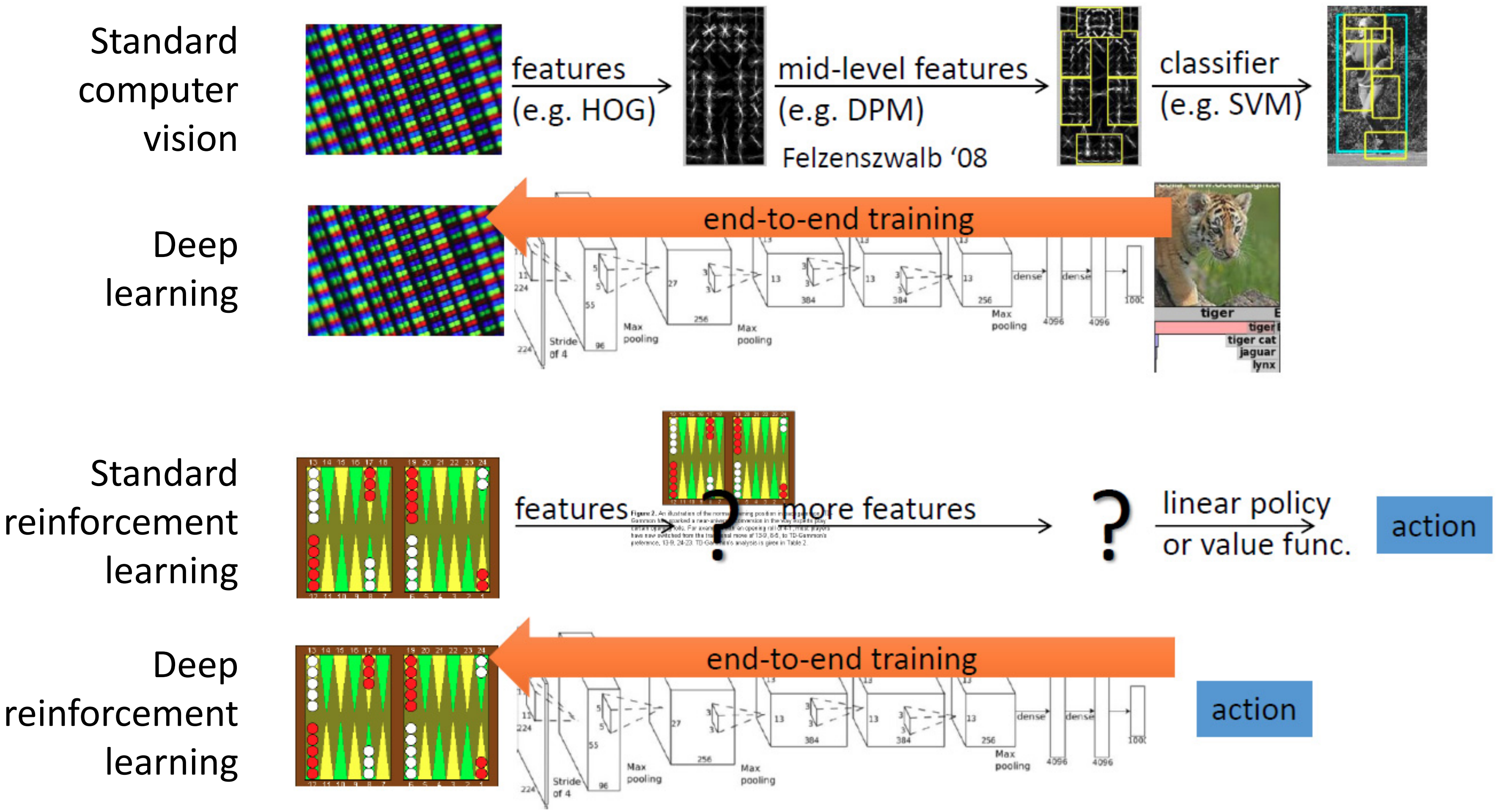

State-value and action-value approx.

Stochastic policy approx.

Deterministic policy approx.

- What if we directly build these approximate function with deep neural networks?

# End-to-End Reinforcement Learning

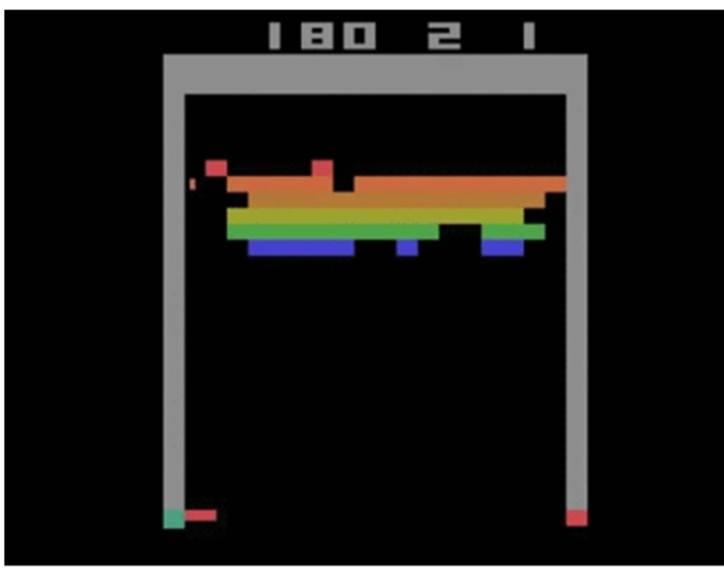


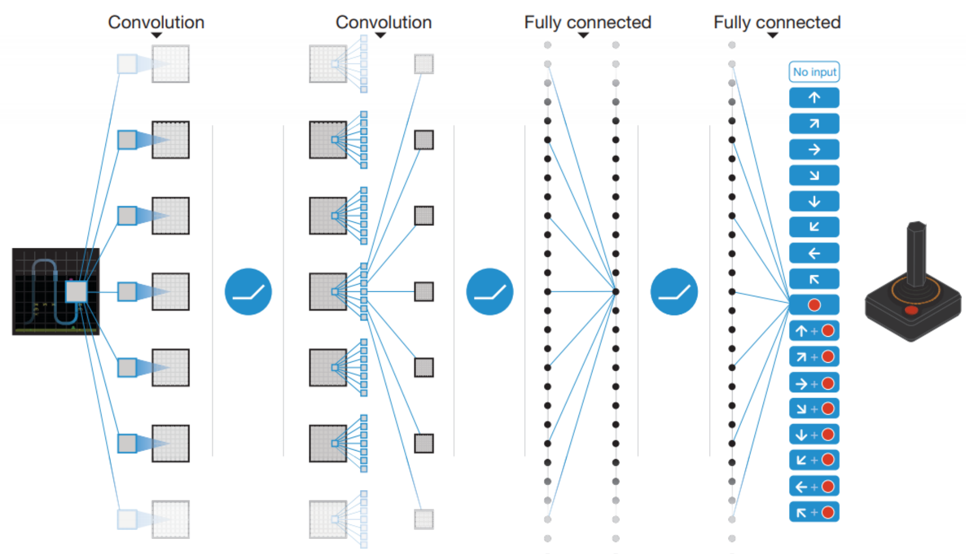

Deep Reinforcement Learning is what allows RL algorithms to solve complex problems in an end-to-end manner.

Slide from Sergey Levine. http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-1.pdf

# Deep Reinforcement Learning

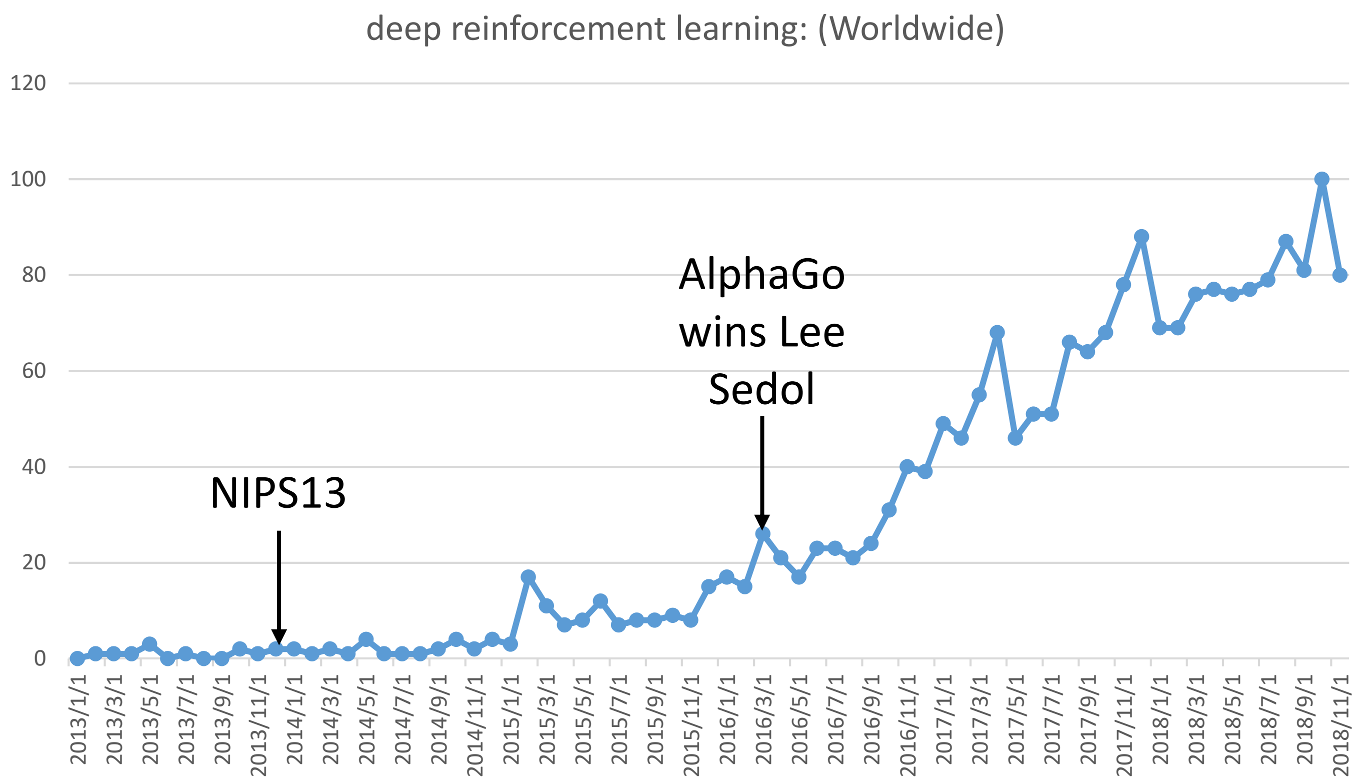
- Deep Reinforcement Learning
  - leverages deep neural networks for value functions and policies approximation
  - so as to allow RL algorithms to solve complex problems in an end-to-end manner.




Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Playing Atari with Deep Reinforcement Learning. NIPS 2013 workshop.

# Deep Reinforcement Learning Trends




- Google search trends of the term ‘deep reinforcement learning’

# Key Changes Brought from DRL

- What will happen when combining DL and RL?
  - Value functions and policies are now deep neural nets
  - Very high-dimensional parameter space
  - Hard to train stably
  - Easy to overfit
  - Need a large amount of data
  - Need high performance computing
  - Balance between CPUs (for collecting experience data) and GPUs (for training neural networks)
  - …
- These new problems motivates novel algorithms for DRL

# Deep Reinforcement Learning Categories

- Value-based methods
  - Deep Q-network and its extensions

- Stochastic policy-based methods
  - Policy gradients with NNs, natural policy gradient, trust-region policy optimization, proximal policy optimization, A3C

- Deterministic policy-based methods
  - Deterministic policy gradient, DDPG

REVIEW

# Q-Learning

- For off-policy learning of action-value $Q(s,a)$
- The next action is chosen using behavior policy $a_{t+1} \sim \mu(\cdot|s_t)$
- But we consider alternative successor action $a \sim \pi(\cdot|s_t)$
- And update $Q(s_t,a_t)$ towards value of alternative action

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma Q(s_{t+1}, a') - Q(s_t, a_t))$$

action from $\pi$ not $\mu$

REVIEW

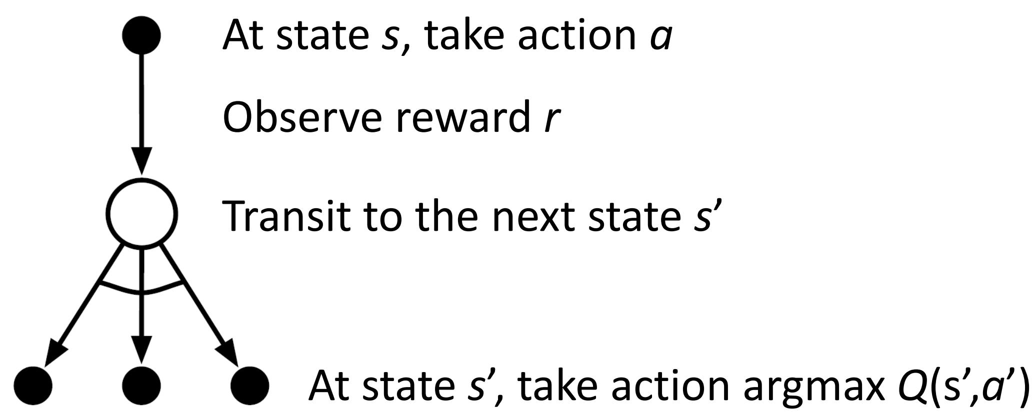
# Off-Policy Control with Q-Learning

- Allow both behavior and target policies to improve
- The target policy $\pi$ is greedy w.r.t. $Q(s,a)$

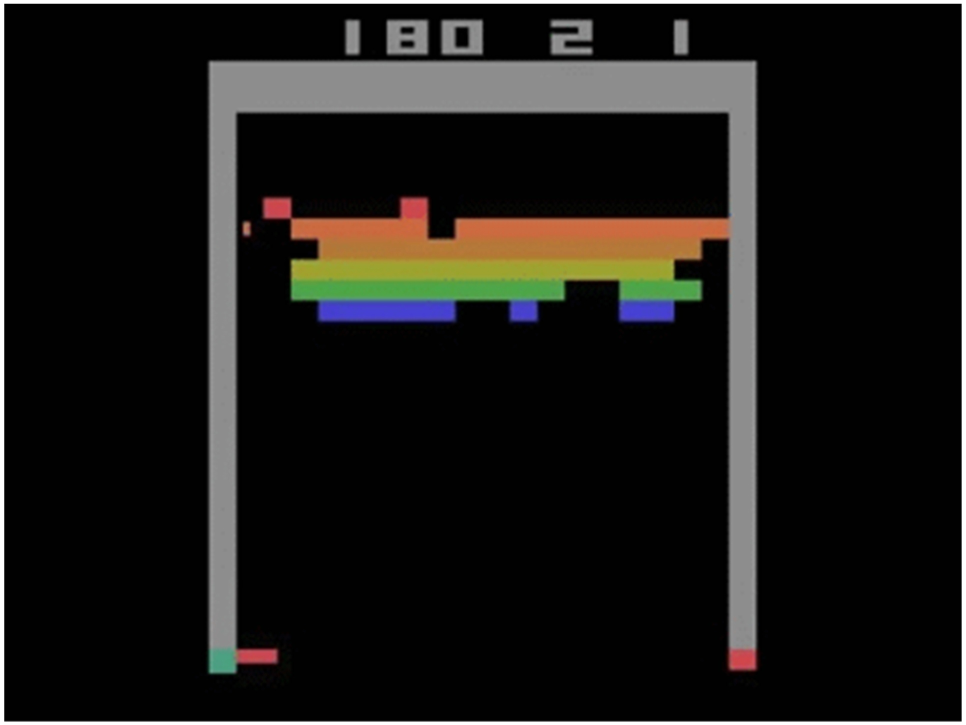
$$\pi(s_{t+1}) = \arg\max_{a'} Q(s_{t+1}, a')$$

- Q-learning update

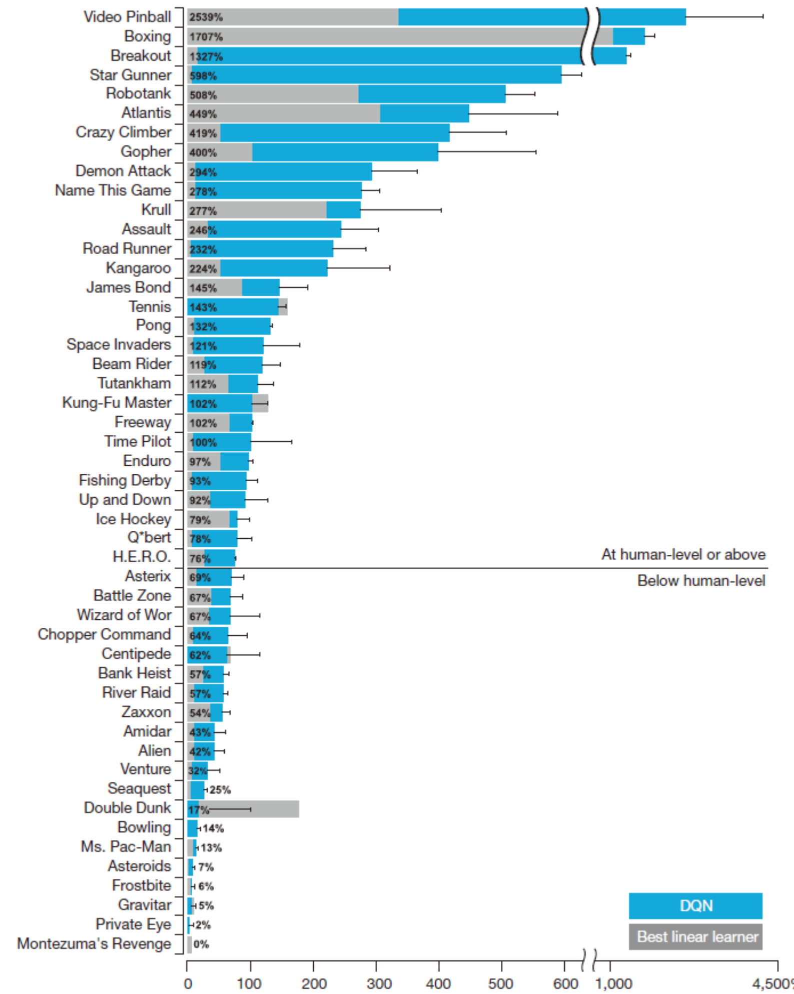
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$



# Deep Q-Network (DQN)



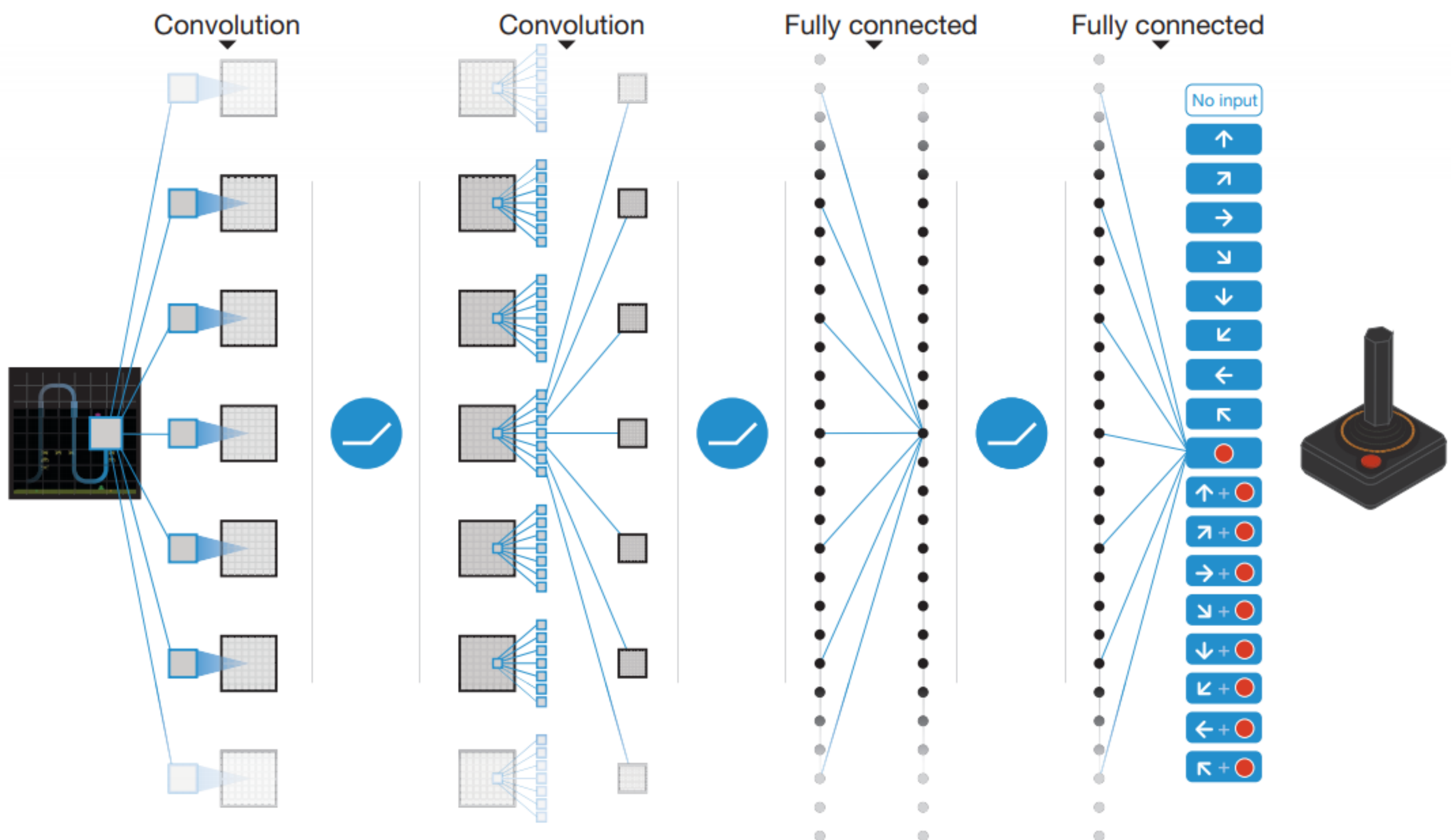

DQN (NIPS 2013) is the beginning of the entire deep reinforcement learning sub-area.




Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Playing Atari with Deep Reinforcement Learning. NIPS 2013 workshop.
Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Human-level control through deep reinforcement learning. Nature 2015.

# Deep Q-Network (DQN)

- Implement Q function with deep neural network
    - Input a state, output Q values for all actions




Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Human-level control through deep reinforcement learning. Nature 2015.

# Deep Q-Network (DQN)

- The loss function of Q-learning update at iteration *i*

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\Big[\big(\underbrace{r + \gamma \max_{a'} Q(s', a'; \theta_i^-)}_{\text{target Q value}} - \underbrace{Q(s, a; \theta_i)}_{\text{estimated Q value}}\big)^2\Big]$$

  - $\vartheta_i$ are the network parameters to be updated at iteration *i*
    - Updated with standard back-propagation algorithms
  - $\vartheta_i^-$ are the target network parameters
    - Only updated with $\vartheta_i$ for every *C* steps
  - (*s*,*a*,*r*,*s'*)~*U*(*D*): the samples are uniformly drawn from the experience pool *D*
    - Thus to avoid the overfitting to the recent experiences

Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Human-level control through deep reinforcement learning. Nature 2015.

# Deep Q-Network (DQN)

- The loss function of Q-learning update at iteration *i*

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\Big[\big(\underbrace{r + \gamma \max_{a'} Q(s',a';\theta_i^-)}_{\text{target Q value}} - \underbrace{Q(s,a;\theta_i)}_{\text{estimated Q value}}\big)^2\Big]$$

- For each experience (*s*,*a*,*r*,*s'*)~*U*(*D*), the gradient is

$$\theta_{i+1} = \theta_i + \eta\big(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i)\big)\underbrace{\nabla_\theta Q(s,a;\theta_i)}_{\text{backpropagation}}$$

Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Human-level control through deep reinforcement learning. Nature 2015.

# DRL with Double Q-Learning

- DQN gradient is

$$\theta_{i+1} = \theta_i + \eta\big(y_i - Q(s, a; \theta_i)\big)\nabla_\theta Q(s, a; \theta_i)$$

target Q value $y_i = r + \gamma \max_{a'} Q(s', a'; \theta_i^-)$

- The target Q value can be rewritten as

$$y_i = r + \gamma Q(s', \arg\max_{a'} Q(s, a'; \theta_i^-); \theta_i^-)$$

uses the same values both to select and to evaluate an action, which makes it more likely to select overestimated values, resulting in overoptimistic value estimates.

Hasselt et al. Deep Reinforcement Learning with Double Q-learning. AAAI 2016.

# DRL with Double Q-Learning

- DQN gradient is

$$\theta_{i+1} = \theta_i + \eta\big(y_i - Q(s,a;\theta_i)\big)\nabla_\theta Q(s,a;\theta_i)$$

target Q value $y_i = r + \gamma \max_{a'} Q(s',a';\theta_i^-)$

- The target Q value can be rewritten as

$$y_i = r + \gamma Q(s', \arg\max_{a'} Q(s,a';\theta_i^-);\theta_i^-)$$

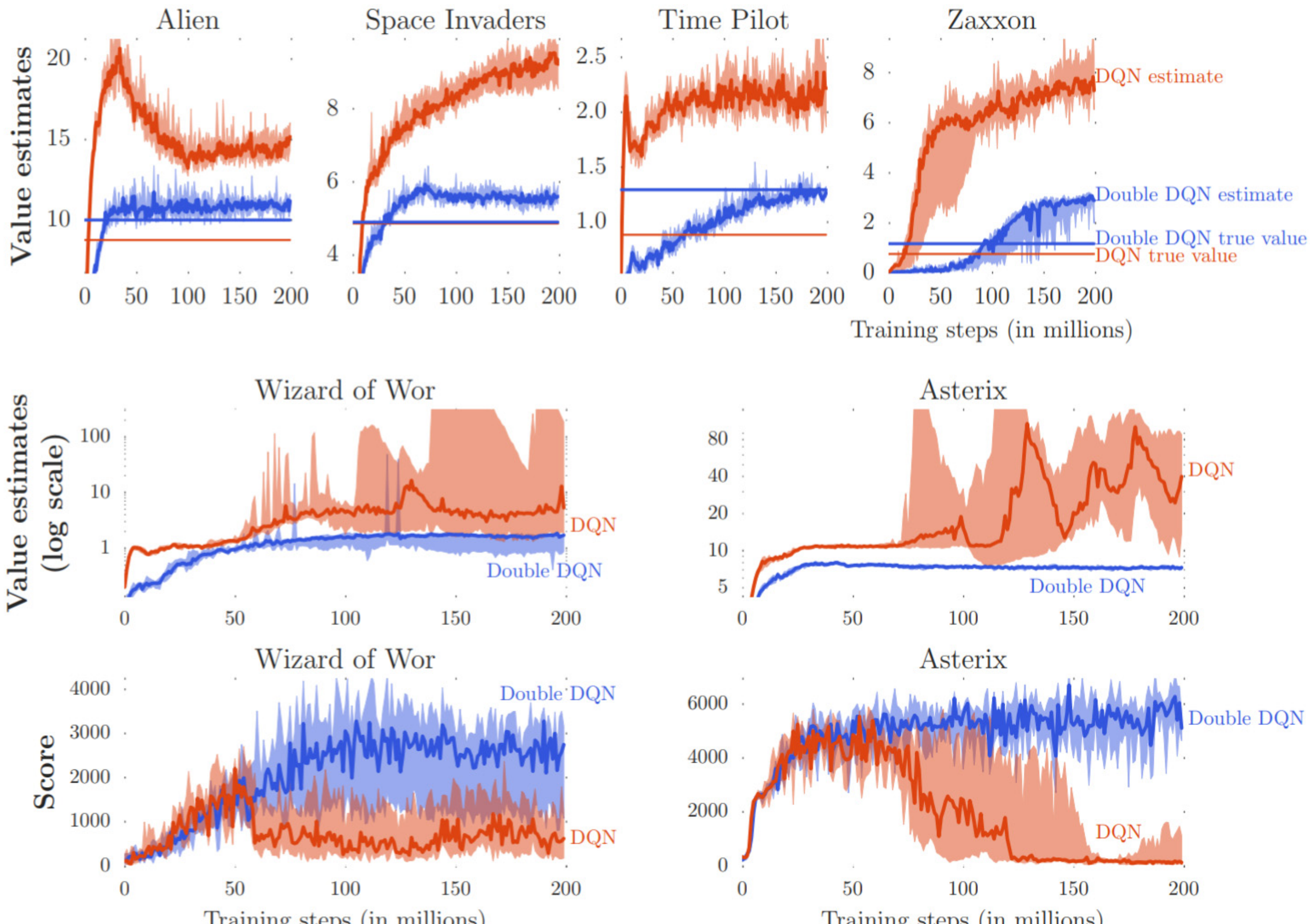
uses the same values both to select and to evaluate an action

- Double Q-learning generalizes using different parameters

$$y_i = r + \gamma Q(s', \arg\max_{a'} Q(s,a';\theta_i);\theta_i')$$

Hasselt et al. Deep Reinforcement Learning with Double Q-learning. AAAI 2016.

# Experiments of DQN vs. Double DQN




Hasselt et al. Deep Reinforcement Learning with Double Q-learning. AAAI 2016.

# Deep Reinforcement Learning Categories

- Value-based methods
  - Deep Q-network and its extensions
- Stochastic policy-based methods
  - Policy gradients with NNs, natural policy gradient, trust-region policy optimization, proximal policy optimization, A3C
- Deterministic policy-based methods
  - Deterministic policy gradient, DDPG

REVIEW

# Policy Gradient Theorem

- The policy gradient theorem generalizes the likelihood ratio approach to multi-step MDPs
  - Replaces instantaneous reward $r_{sa}$ with long-term value $Q^{\pi_\theta}(s,a)$
- Policy gradient theorem applies to
  - start state objective $J_1$, average reward objective $J_{avR}$, and average value objective $J_{avV}$
- Theorem
  - For any differentiable policy $\pi_\theta(a|s)$, for any of policy objective function $J = J_1, J_{avR}, J_{avV}$, the policy gradient is

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} Q^{\pi_\theta}(s,a)\right]$$

# Policy Network Gradients

- For stochastic policy, typically the action probability is defined as a softmax

$$\pi_\theta(a|s) = \frac{e^{f_\theta(s,a)}}{\sum_{a'} e^{f_\theta(s,a')}}$$

  - where $f_\vartheta(s,a)$ is the score function of a state-action pair parametrized by $\vartheta$, which can be implemented with a neural net

- The gradient of its log-form

$$\begin{aligned}\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} &= \frac{\partial f_\theta(s,a)}{\partial \theta} - \frac{1}{\sum_{a'} e^{f_\theta(s,a')}} \sum_{a''} e^{f_\theta(s,a'')} \frac{\partial f_\theta(s,a'')}{\partial \theta} \\ &= \frac{\partial f_\theta(s,a)}{\partial \theta} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)} \Big[ \frac{\partial f_\theta(s,a')}{\partial \theta} \Big]\end{aligned}$$

# Policy Network Gradients

- With the gradient form

$$\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} = \frac{\partial f_\theta(s,a)}{\partial \theta} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)}\left[\frac{\partial f_\theta(s,a')}{\partial \theta}\right]$$

- The policy network gradient is

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta} &= \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} Q^{\pi_\theta}(s,a)\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\left(\underbrace{\frac{\partial f_\theta(s,a)}{\partial \theta}}_{\text{backpropagation}} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)}\left[\underbrace{\frac{\partial f_\theta(s,a')}{\partial \theta}}_{\text{backpropagation}}\right]\right) Q^{\pi_\theta}(s,a)\right]\end{aligned}$$

# Looking into Policy Gradient

- Let $R(\pi)$ denote the expected return of $\pi$

$$R(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a_t \sim \pi(\cdot|s_t)}\Big[\sum_{t=0}^{\infty} \gamma^t r_t\Big]$$

- We collect experience data with another policy $\pi_{\text{old}}$, and want to optimize some objective to get a new better policy $\pi$
- Note that a useful identity

$$R(\pi) = R(\pi_{\text{old}}) + \mathbb{E}_{\tau \sim \pi}\Big[\sum_{t=0}^{\infty} \gamma^t A^{\pi_{\text{old}}}(s_t, a_t)\Big]$$

Trajectories sampled from $\pi$

- Advantage function

$$A^{\pi_{\text{old}}}(s, a) = \mathbb{E}_{s' \sim \rho(s'|s,a)}[r(s) + \gamma V^{\pi_{\text{old}}}(s') - V^{\pi_{\text{old}}}(s)]$$

S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. ICML. 2002.

# Looking into Policy Gradient

- Advantage function

$$A^{\pi_{\text{old}}}(s,a) = \mathbb{E}_{s' \sim \rho(s'|s,a)}[r(s) + \gamma V^{\pi_{\text{old}}}(s') - V^{\pi_{\text{old}}}(s)]$$

- Note that a useful identity

$$R(\pi) = R(\pi_{\text{old}}) + \mathbb{E}_{\tau \sim \pi}\Big[\sum_{t=0}^{\infty} \gamma^t A^{\pi_{\text{old}}}(s_t, a_t)\Big]$$

- Proof:

$$\begin{aligned}
\mathbb{E}_{\tau \sim \pi}\Big[\sum_{t=0}^{\infty} \gamma^t A^{\pi_{\text{old}}}(s_t, a_t)\Big] &= \mathbb{E}_{\tau \sim \pi}\Big[\sum_{t=0}^{\infty} \gamma^t (r(s_t) + \gamma V^{\pi_{\text{old}}}(s_{t+1}) - V^{\pi_{\text{old}}}(s_t))\Big] \\
&= \mathbb{E}_{\tau \sim \pi}\Big[-V^{\pi_{\text{old}}}(s_0) + \sum_{t=0}^{\infty} \gamma^t r(s_t)\Big] \\
&= -\mathbb{E}_{s_0}[V^{\pi_{\text{old}}}(s_0)] + \mathbb{E}_{\tau \sim \pi}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t)\Big] = -R(\pi_{\text{old}}) + R(\pi)
\end{aligned}$$

S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. ICML. 2002.

# More for the Policy Expected Return

- Given the advantage function

$$A^{\pi_{\text{old}}}(s,a) = \mathbb{E}_{s' \sim \rho(s'|s,a)}[r(s) + \gamma V^{\pi_{\text{old}}}(s') - V^{\pi_{\text{old}}}(s)]$$

- Want to manipulate *R*($\pi$) into an objective that can be estimated from data

$$\begin{aligned}
R(\pi) &= R(\pi_{\text{old}}) + \mathbb{E}_{\tau\sim\pi}\Big[\sum_{t=0}^{\infty}\gamma^t A^{\pi_{\text{old}}}(s_t,a_t)\Big] \\
&= R(\pi_{\text{old}}) + \sum_{t=0}^{\infty}\sum_s P(s_t = s|\pi)\sum_a \pi(a|s)\gamma^t A^{\pi_{\text{old}}}(s,a) \\
&= R(\pi_{\text{old}}) + \sum_s \sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi)\sum_a \pi(a|s) A^{\pi_{\text{old}}}(s,a) \\
&= R(\pi_{\text{old}}) + \sum_s \rho_\pi(s)\sum_a \pi(a|s) A^{\pi_{\text{old}}}(s,a)
\end{aligned}$$

# Surrogate Loss Function

- With the importance sampling

$$\begin{aligned} R(\pi) &= R(\pi_{\text{old}}) + \sum_s \rho_\pi(s) \sum_a \pi(a|s) A^{\pi_{\text{old}}}(s,a) \\ &= R(\pi_{\text{old}}) + \mathbb{E}_{s\sim\pi, a\sim\pi}[A^{\pi_{\text{old}}}(s,a)] \\ &= R(\pi_{\text{old}}) + \mathbb{E}_{s\sim\pi, a\sim\pi_{\text{old}}}\left[\frac{\pi(a|s)}{\pi_{\text{old}}(a|s)} A^{\pi_{\text{old}}}(s,a)\right] \end{aligned}$$

- Define a surrogate loss function based on sampled data that ignores change in state distribution

$$L(\pi) = \mathbb{E}_{s\sim\pi_{\text{old}}, a\sim\pi_{\text{old}}}\left[\frac{\pi(a|s)}{\pi_{\text{old}}(a|s)} A^{\pi_{\text{old}}}(s,a)\right]$$

# Surrogate Loss Function

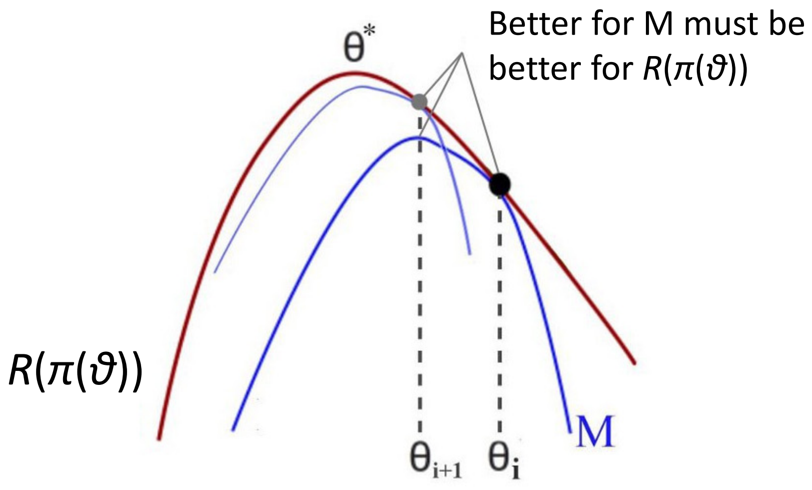
Target function $R(\pi) = R(\pi_{\text{old}}) + \mathbb{E}_{s\sim\pi, a\sim\pi}\left[\pi(a|s)A^{\pi_{\text{old}}}(s,a)\right]$

Surrogate loss $L(\pi) = \mathbb{E}_{s\sim\pi_{\text{old}}, a\sim\pi_{\text{old}}}\left[\frac{\pi(a|s)}{\pi_{\text{old}}(a|s)}A^{\pi_{\text{old}}}(s,a)\right]$

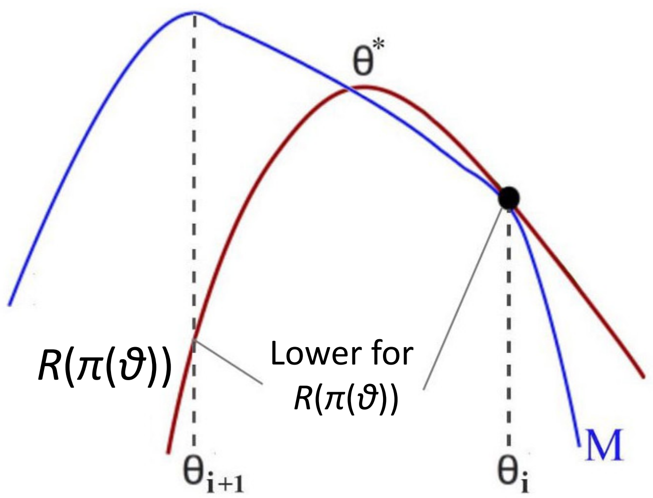
- Matches to first order for parameterized policy

$$
\begin{aligned}
\nabla_\theta L(\pi_\theta)\big|_{\theta_{\text{old}}} &= \mathbb{E}_{s\sim\pi_{\text{old}}, a\sim\pi_{\text{old}}}\left[\frac{\nabla_\theta \pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}A^{\pi_{\text{old}}}(s,a)\right]\bigg|_{\theta_{\text{old}}} \\
&= \mathbb{E}_{s\sim\pi_{\text{old}}, a\sim\pi_{\text{old}}}\left[\frac{\pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a|s)}{\pi_{\text{old}}(a|s)}A^{\pi_{\text{old}}}(s,a)\right]\bigg|_{\theta_{\text{old}}} \\
&= \mathbb{E}_{s\sim\pi_{\text{old}}, a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)A^{\pi_{\text{old}}}(s,a)\right]\bigg|_{\theta_{\text{old}}} \\
&= \nabla_\theta R(\pi_\theta)\big|_{\theta_{\text{old}}}
\end{aligned}
$$

# Trust-Region Policy Optimization




M is the lower bound




M is not the lower bound

- Idea: by optimizing a lower bound function approximating $R(\pi)$ locally, it guarantees policy improvement every time and lead us to the optimal policy eventually.
- How to choose a proper lower bound $M$?

# Trust-Region Policy Optimization

$$R(\pi) = \mathbb{E}_{s_0 \sim \rho_0, a_t \sim \pi(\cdot|s_t)}\Big[\sum_{t=0}^{\infty} \gamma^t r_t\Big]$$

$$= R(\pi_{\text{old}}) + \sum_s \rho_\pi(s) \sum_a \pi(a|s) A^{\pi_{\text{old}}}(s,a)$$

$$L_{\pi_{\text{old}}}(\pi) = \mathbb{E}_{s \sim \pi_{\text{old}}, a \sim \pi_{\text{old}}}\Big[\frac{\pi(a|s)}{\pi_{\text{old}}(a|s)} A^{\pi_{\text{old}}}(s,a)\Big]$$

- The appendix A of the TRPO paper provides a 2-page proof that establishes the following boundary

$$\big|R(\pi) - (R(\pi_{\text{old}}) + L_{\pi_{\text{old}}}(\pi))\big| \le C\sqrt{\mathbb{E}_{s \sim \rho_\pi}[D_{KL}(\pi_{\text{old}}(\cdot|s) \| \pi(\cdot|s))]}$$

Schulman, John, et al. "Trust region policy optimization." *International Conference on Machine Learning*. 2015.

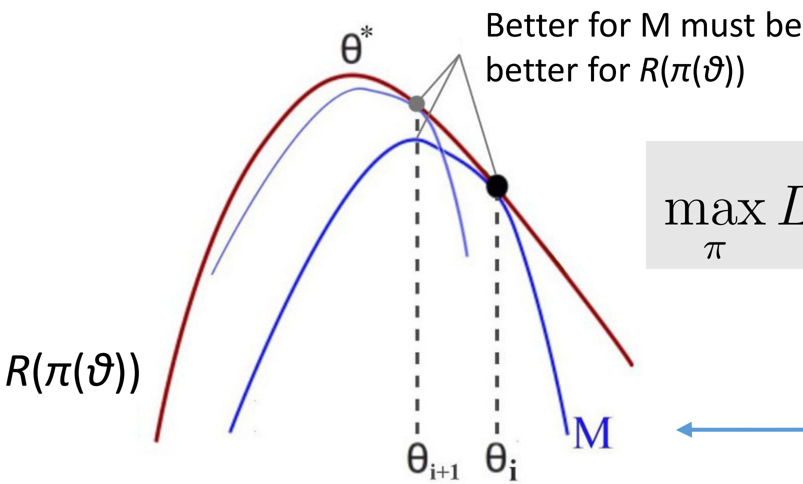
# Trust-Region Policy Optimization

$$\max_{\pi} R(\pi) = \max_{\pi} R(\pi) - R(\pi_{\text{old}})$$

$$\left| R(\pi) - \left(R(\pi_{\text{old}}) + L_{\pi_{\text{old}}}(\pi)\right) \right| \leq C\sqrt{\mathbb{E}_{s\sim\rho_\pi}[D_{KL}(\pi_{\text{old}}(\cdot|s)\|\pi(\cdot|s))]}$$

- With some twitting, this is our final lower bound *M*.

$$R(\pi) - R(\pi_{\text{old}}) \geq L_{\pi_{\text{old}}}(\pi) - C\sqrt{\mathbb{E}_{s\sim\rho_\pi}[D_{KL}(\pi_{\text{old}}(\cdot|s)\|\pi(\cdot|s))]}$$




$$\max_{\pi} L_{\pi_{\text{old}}}(\pi) - C\sqrt{\mathbb{E}_{s\sim\rho_\pi}[D_{KL}(\pi_{\text{old}}(\cdot|s)\|\pi(\cdot|s))]}$$

Schulman, John, et al. "Trust region policy optimization." *International Conference on Machine Learning*. 2015.

# Trust-Region Policy Optimization

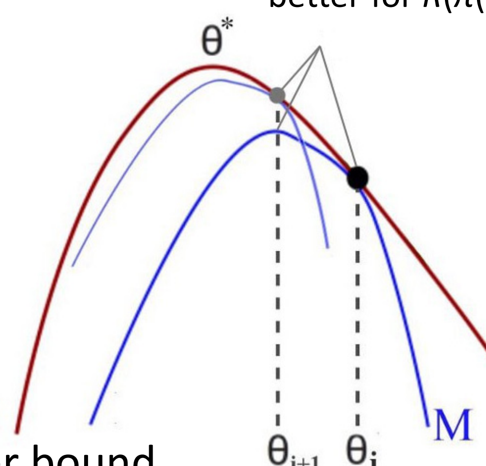
- In fact, with the Lagrangian methods, our objective is mathematically the same as the following using a trust region constraint

$$\max_{\pi} L_{\pi_{\text{old}}}(\pi) - C\sqrt{\mathbb{E}_{s\sim\rho_\pi}[D_{KL}(\pi_{\text{old}}(\cdot|s)\|\pi(\cdot|s)))]}$$

⬍

$$\max_{\pi} \; L_{\pi_{\text{old}}}(\pi)$$
$$\text{s.t.} \; \mathbb{E}_{s\sim\rho_\pi}[D_{KL}(\pi_{\text{old}}(\cdot|s)\|\pi(\cdot|s)))] \leq \delta$$

↑ Trust region




To guarantee the follow inequality to make *M* a valid lower bound

$$\left|R(\pi) - (R(\pi_{\text{old}}) + L_{\pi_{\text{old}}}(\pi))\right| \leq C\sqrt{\mathbb{E}_{s\sim\rho_\pi}[D_{KL}(\pi_{\text{old}}(\cdot|s)\|\pi(\cdot|s)))]}$$


Schulman, John, et al. "Trust region policy optimization." *International Conference on Machine Learning*. 2015.

# Trust-Region Policy Optimization

Line search
(like gradient ascent)

Optimization in Trust Region

https://medium.com/@jonathan_hui/rl-trust-region-policy-optimization-trpo-explained-a6ee04eeeee9

# A3C: Actor Critic Methods

- A3C stands for Asynchronous Advantage Actor Critic
  - Asynchronous: because the algorithm involves executing a set of environments in parallel
  - Advantage: because the policy gradient updates are done using the advantage function
  - Actor Critic: because this is an actor-critic method which involves a policy that updates with the help of learned state-value functions.

$$\nabla_{\theta'} \log \pi(a_t|s_t;\theta') A(s_t, a_t; \theta, \theta_v)$$

$$A(s_t, a_t; \theta, \theta_v) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}; \theta_v) - V(s_t; \theta_v)$$

https://medium.com/@jonathan_hui/rl-proximal-policy-optimization-ppo-explained-77f014ec3f12

**Algorithm S3** Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

*// Assume global shared parameter vectors $\theta$ and $\theta_v$ and global shared counter $T = 0$*
*// Assume thread-specific parameter vectors $\theta'$ and $\theta'_v$*
Initialize thread step counter $t \leftarrow 1$
**repeat**
    Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
    Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
    $t_{start} = t$
    Get state $s_t$
    **repeat**
        Perform $a_t$ according to policy $\pi(a_t|s_t; \theta')$
        Receive reward $r_t$ and new state $s_{t+1}$
        $t \leftarrow t + 1$
        $T \leftarrow T + 1$
    **until** terminal $s_t$ **or** $t - t_{start} == t_{max}$
    $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
    **for** $i \in \{t-1, \ldots, t_{start}\}$ **do**
        $R \leftarrow r_i + \gamma R$
        Accumulate gradients wrt $\theta'$: $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i; \theta')(R - V(s_i; \theta'_v))$
        Accumulate gradients wrt $\theta'_v$: $d\theta_v \leftarrow d\theta_v + \partial \left(R - V(s_i; \theta'_v)\right)^2 / \partial \theta'_v$
    **end for**
    Perform asynchronous update of $\theta$ using $d\theta$ and of $\theta_v$ using $d\theta_v$.
**until** $T > T_{max}$

# Deep Reinforcement Learning Categories

- Value-based methods
  - Deep Q-network and its extensions
- Stochastic policy-based methods
  - Policy gradients with NNs, natural policy gradient, trust-region policy optimization, proximal policy optimization, A3C
- Deterministic policy-based methods
  - Deterministic policy gradient, DDPG

# Stochastic vs. Deterministic Policies

- Stochastic policy

for discrete actions $$\pi(a|s;\theta) = \frac{\exp\{Q_\theta(s,a)\}}{\sum_{a'} \exp\{Q_\theta(s,a')\}}$$

for continuous actions $$\pi(a|s;\theta) \propto \exp\{(a - \mu_\theta(s))^2\}$$

- Deterministic policy

for discrete actions $$\pi(s;\theta) = \arg\max_a Q_\theta(s,a)$$

(non-differentiable)

for continuous actions $$a = \pi_\theta(s)$$ (can be differentiable)

# Deterministic Policy Gradient

- A critic module for state-action value estimation

$$Q^w(s,a) \simeq Q^\pi(s,a)$$

$$L(w) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}\left[(Q^w(s,a) - Q^\pi(s,a))^2\right]$$

- With the differentiable critic, the deterministic continuous-action actor can be updated as
  - Deterministic policy gradient theorem

$$J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}[Q^\pi(s,a)]$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}\left[\nabla_\theta \pi_\theta(s) \nabla_a Q^\pi(s,a)|_{a=\pi_\theta(s)}\right]$$

On-policy Chain rule

D. Silver et al. Deterministic Policy Gradient Algorithms. ICML 2014.

# DDPG: Deep Deterministic Policy Gradient

- For deterministic policy gradient

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s \sim \rho^\pi} [\nabla_\theta \pi_\theta(s) \nabla_a Q^\pi(s,a)|_{a=\pi_\theta(s)}]$$

- In practice, a naive application of this actor-critic method with neural function approximators is unstable for challenging problems
- DDPG solutions over DPG
  - Experience replay (off-policy)
  - Target network
  - Batch normalization on Q network prior to the action input
  - Add noise on continuous

Lillicrap et al. Continuous control with deep reinforcement learning. NIPS 2015.

**Algorithm 1** DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
**for** episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $s_1$
  **for** t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise   (Noise on action)
    Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
    Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$   (Off-policy)
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q)^2)$   (Update critic net)
    Update the actor policy using the sampled gradient:

Target critic network

Target actor network

$$\nabla_{\theta^\mu} \mu|_{s_i} \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

Update actor net

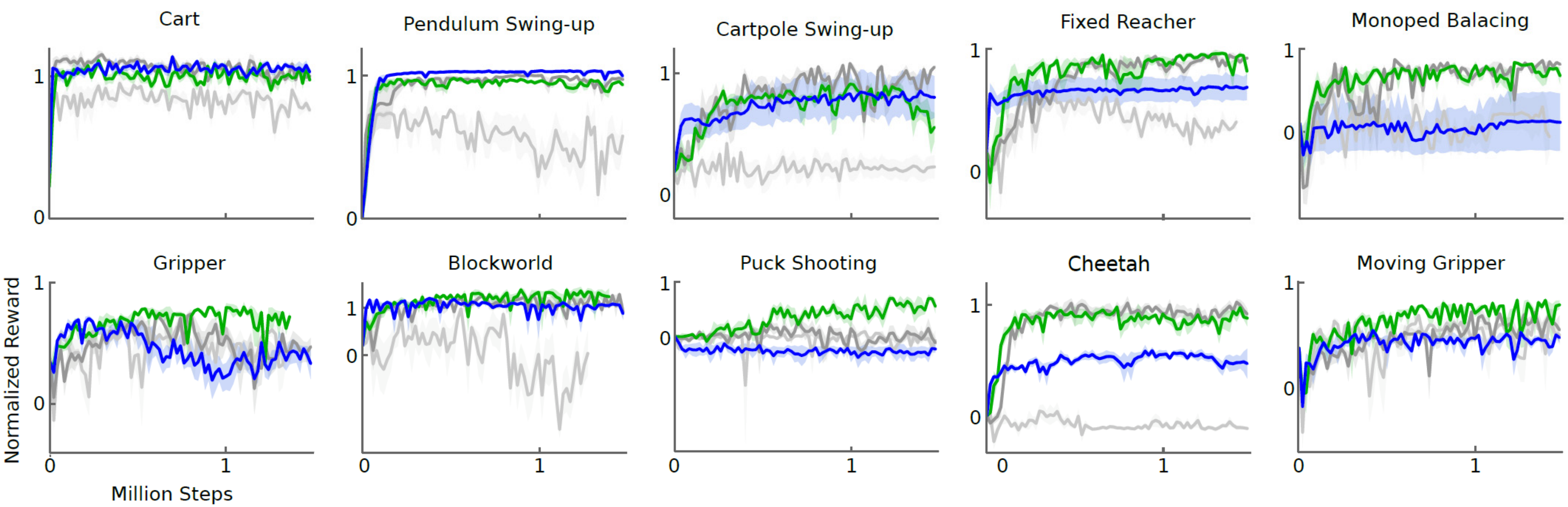
    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  **end for**
**end for**

# DDPG Experiments




- Performance curves for a selection of domains using variants of DPG
  - Light grey: original DPG algorithm with batch normalization
  - Dark grey: with target network
  - Green: with target networks and batch normalization
  - Blue: with target networks from pixel-only inputs.
- Target networks are crucial.

Lillicrap et al. Continuous control with deep reinforcement learning. NIPS 2015.

# Deep Reinforcement Learning Categories

- DRL = RL + DL
  - One of the most challenging problems in machine learning with very fast develop during the recent 5 years
- Value-based methods
  - Deep Q-network and its extensions
- Stochastic policy-based methods
  - Policy gradients with NNs, natural policy gradient, trust-region policy optimization, proximal policy optimization, A3C
- Deterministic policy-based methods
  - Deterministic policy gradient, DDPG