

Map

Part I

DQN

Q-Learning with Neural Networks

Q-Learning:

$$Q(s, a) \leftarrow Q(s, a) + \alpha (r + \gamma \max_{a'} Q(s', a') - Q(s, a))$$

Neural Networks

$$\theta^* = \arg \min_{\theta} \sum_{(x,y) \in \mathcal{D}} l(f_{\theta}(x), y)$$

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} l(f_{\theta}(x), y)$$

Deep Q learning:

- Approximate Q with Q_{θ}
- What should (x, y) be?
- What should l be?

Candidate Algorithm:

loop

$$a \leftarrow \operatorname{argmax} Q(s, a) \text{ w.p. } 1 - \epsilon, \quad \operatorname{rand}(A) \text{ o.w.}$$

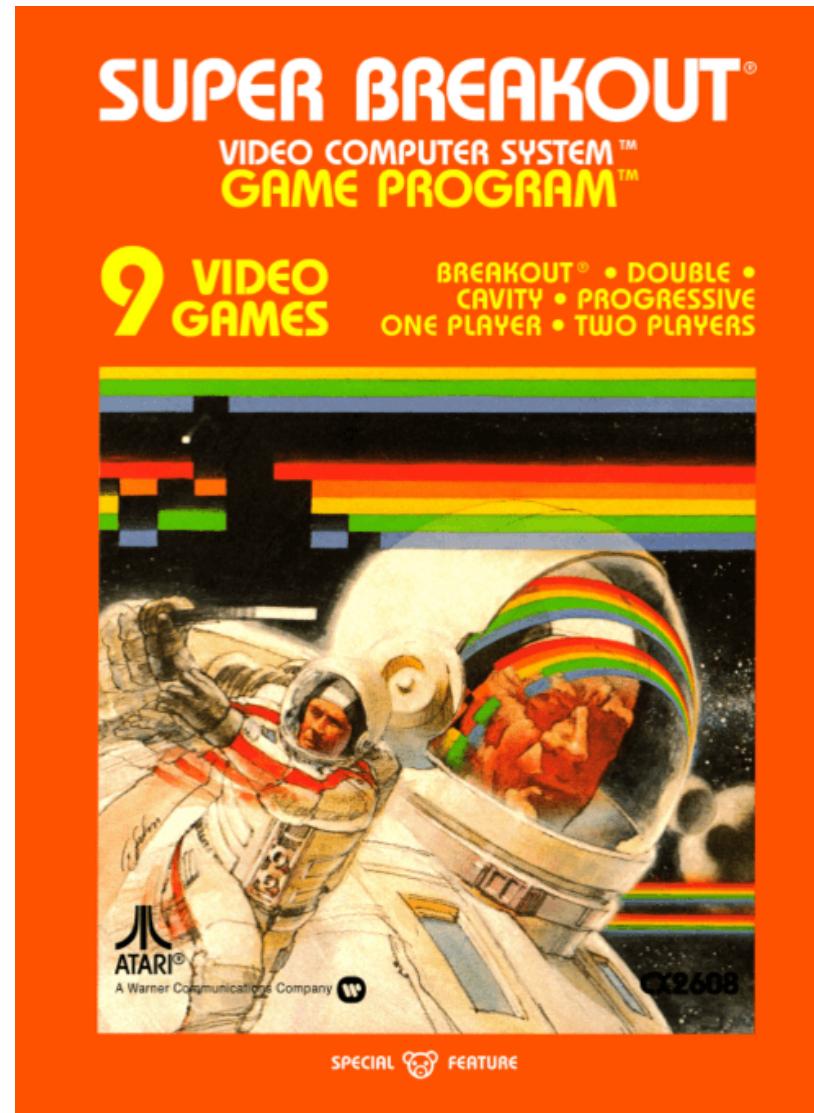
$$r \leftarrow \operatorname{act!}(env, a)$$

$$s' \leftarrow \operatorname{observe}(env)$$

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} (r + \gamma \max_{a'} Q_{\theta}(s', a') - Q_{\theta}(s, a))^2$$

$$s \leftarrow s'$$

DQN: The Atari Benchmark



DQN: Problems with Naive Approach

Candidate Algorithm:

loop

$$a \leftarrow \operatorname{argmax} Q(s, a) \text{ w.p. } 1 - \epsilon, \quad \operatorname{rand}(A) \text{ o.w.}$$

$$r \leftarrow \operatorname{act}!(\text{env}, a)$$

$$s' \leftarrow \operatorname{observe}(\text{env})$$

$$\theta \leftarrow \theta - \alpha \nabla_{\theta} (r + \gamma \max_{a'} Q_{\theta}(s', a') - Q_{\theta}(s, a))^2$$

$$s \leftarrow s'$$

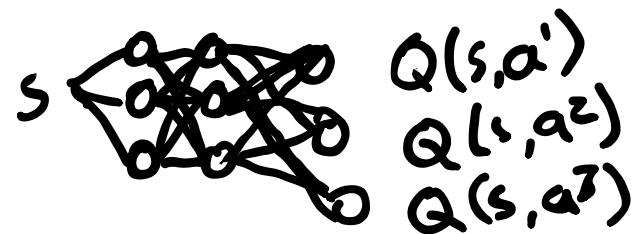
Problems:

1. Samples Highly Correlated
2. Size-1 batches
3. Moving target

} data buffer/experience replay
} periodically freeze target

DQN

Q Network Structure:



Experience Tuple: (s, a, r, s')

Loss:

$$l(s, a, r, s') = \left(r + \gamma \max_{a'} Q_{\theta'}(s', a') - Q_{\theta}(s, a) \right)^2$$

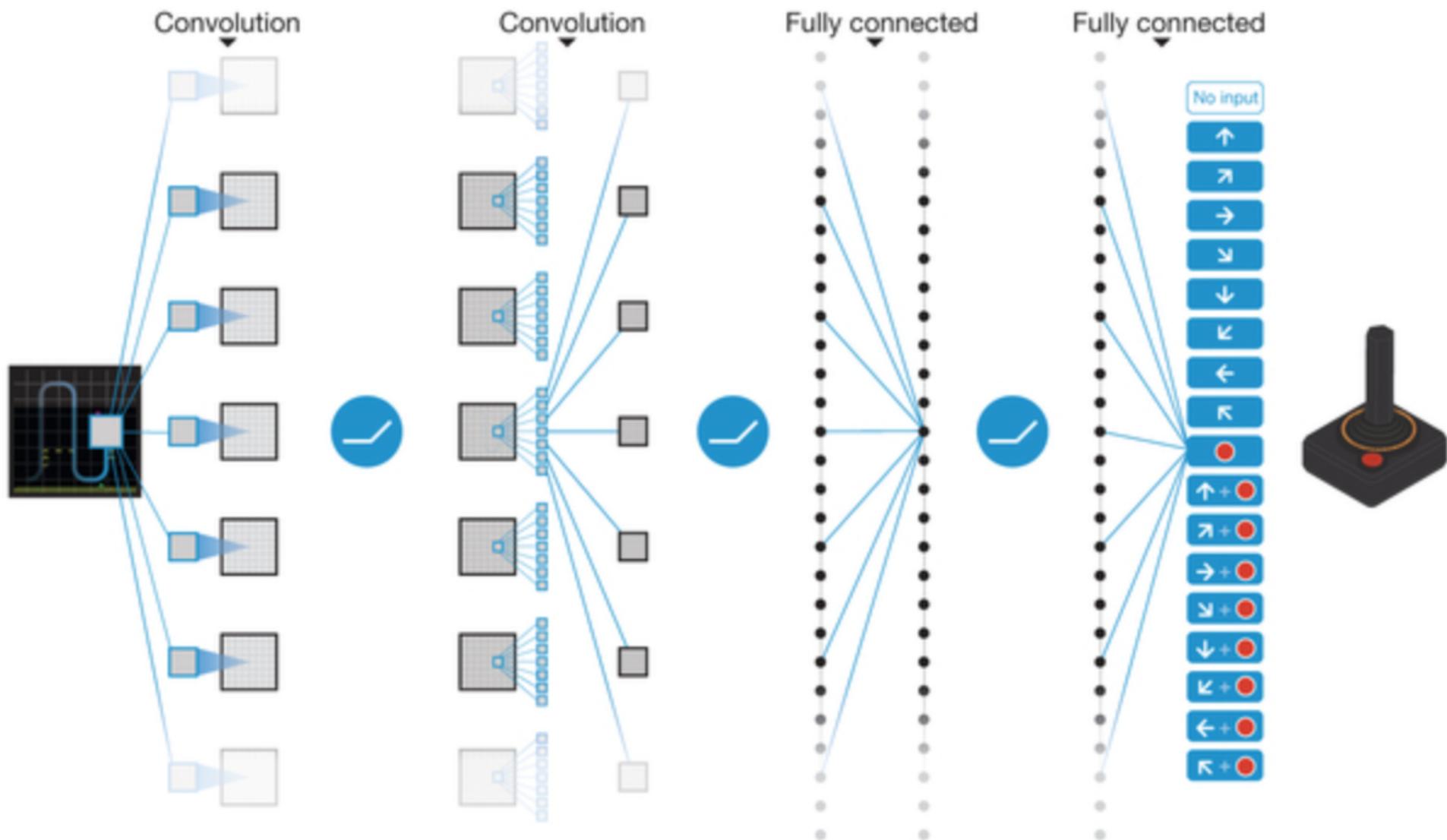
Playing Atari with Deep Reinforcement Learning

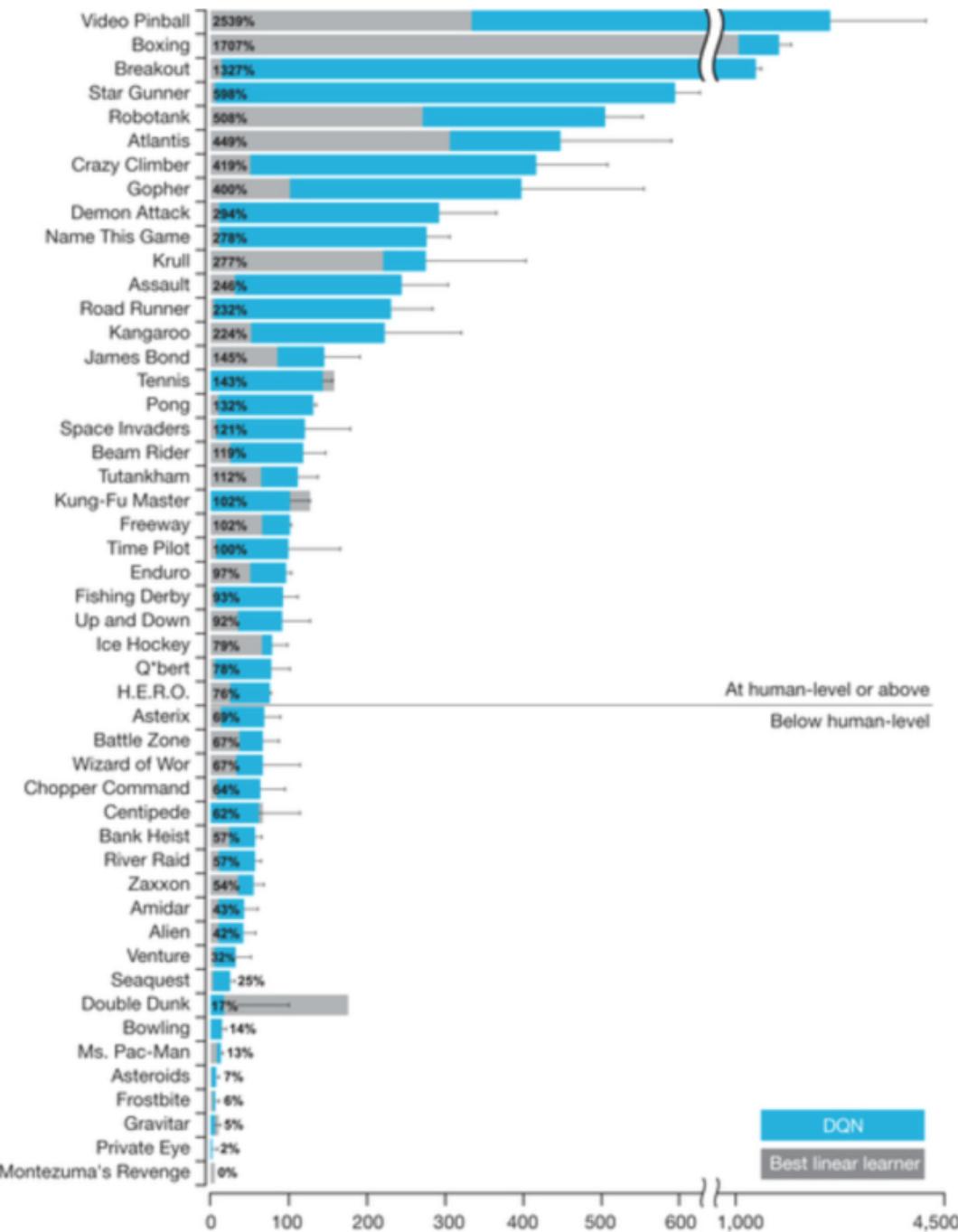
Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou

Daan Wierstra Martin Riedmiller

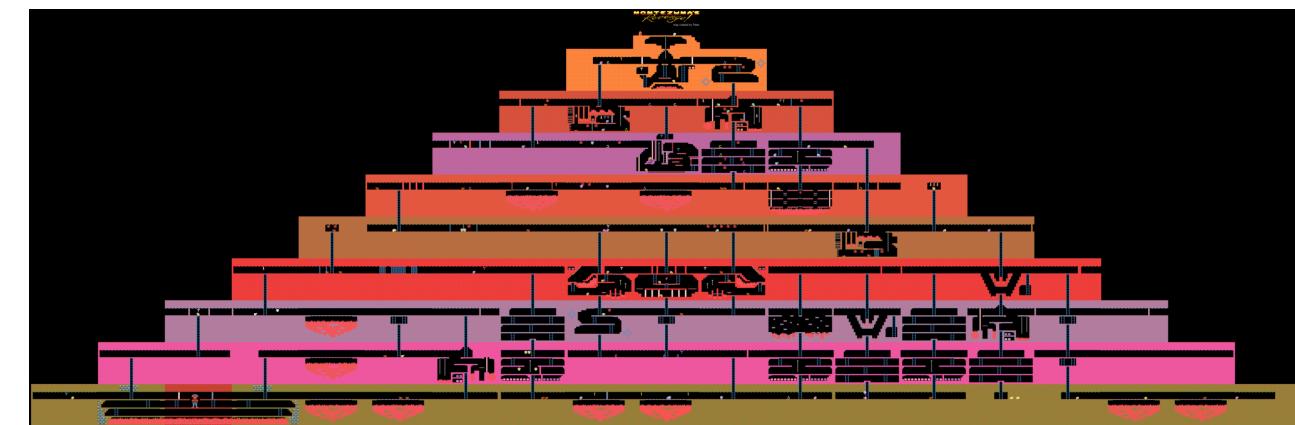
DeepMind Technologies





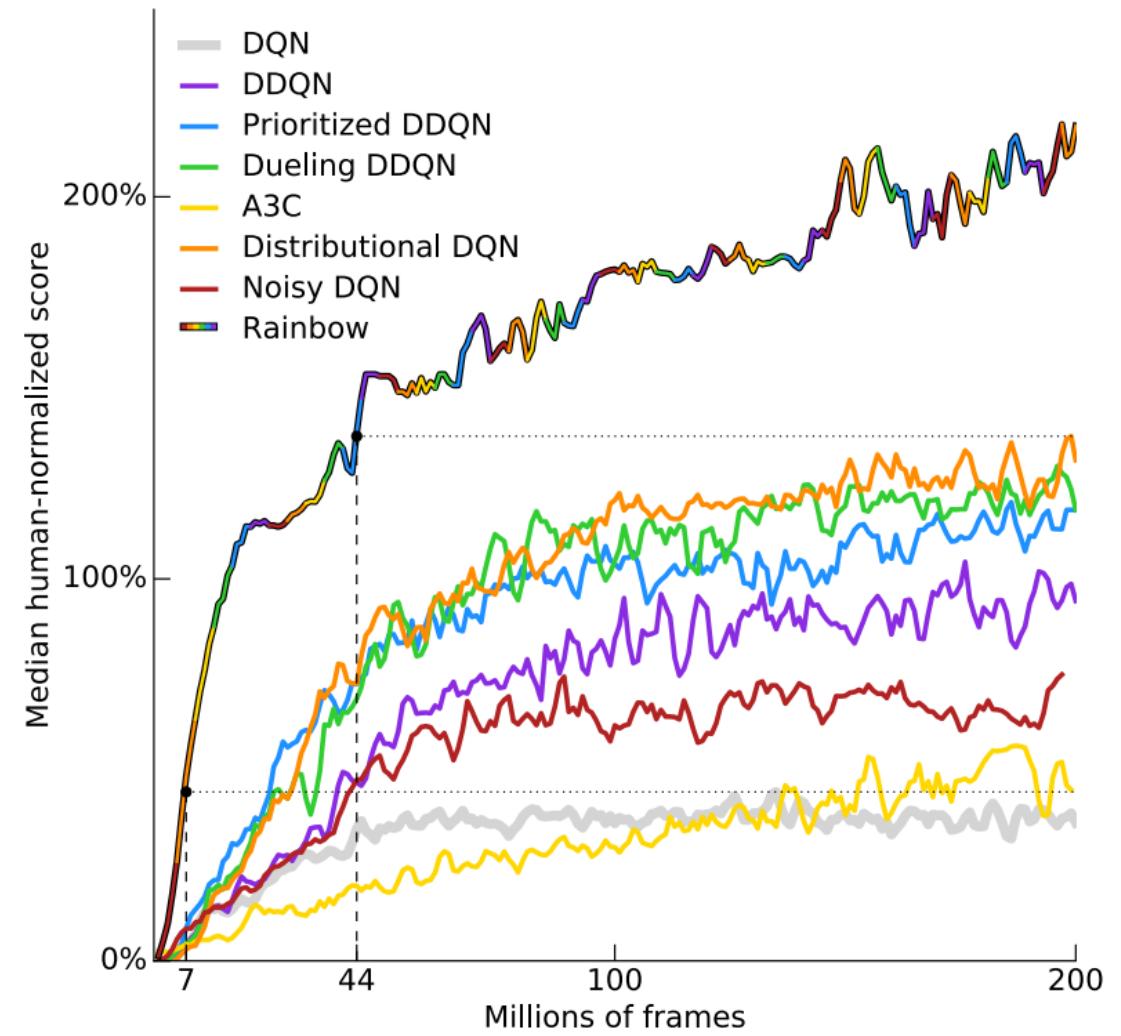


[https://www.youtube.com/watch?
v=SuZVyOlgVek](https://www.youtube.com/watch?v=SuZVyOlgVek)



Rainbow

- Double Q Learning
- Prioritized Replay
(priority proportional to last TD error)
- Dueling networks
Value network + advantage network
$$Q(s, a) = V(s) + A(s, a)$$
- Multi-step learning
$$(r_t + \gamma r_{t+1} + \dots + \gamma^{n-1} r_{t+n-1} + \gamma \max Q_\theta(s_{t+n}, a') - Q_\theta(s_t, a_t))^2$$
- Distributional RL
predict an entire distribution of values instead of just Q
- Noisy Nets



Part II

Improved Policy Gradients

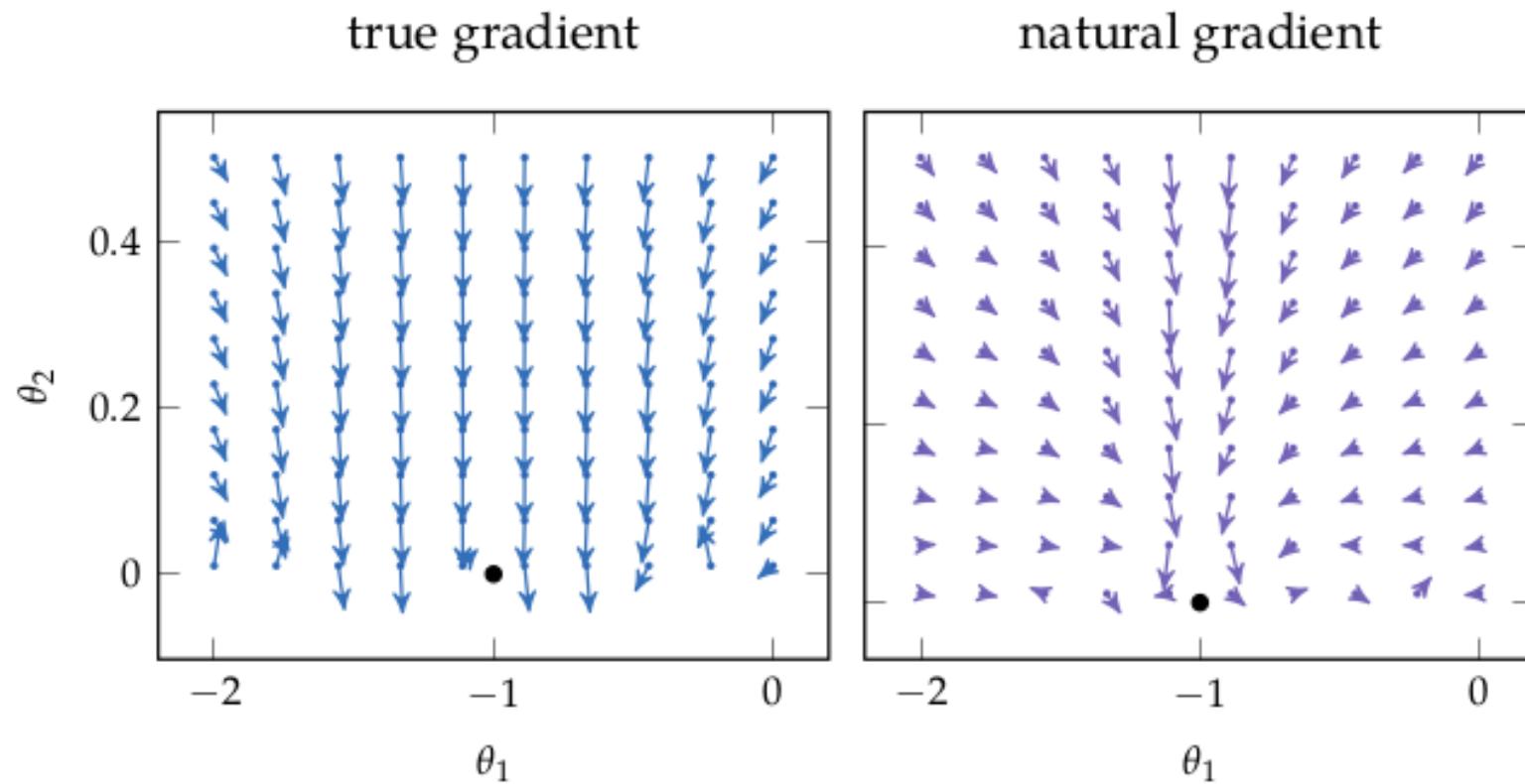
Restricted Gradient Update

$$\widehat{\nabla U}(\theta) = \sum_{k=0}^d \nabla_\theta \log \pi_\theta(a_k \mid s_k) \gamma^k (r_{k,\text{to-go}} - r_{\text{base}}(s_k))$$

$$\theta' = \theta + \alpha \widehat{\nabla U}(\theta)$$

Natural Gradient

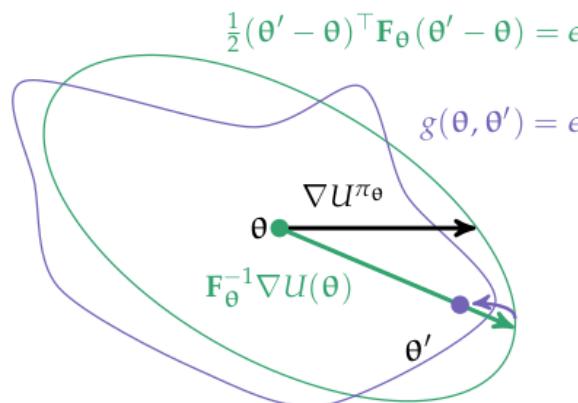
Natural Gradient



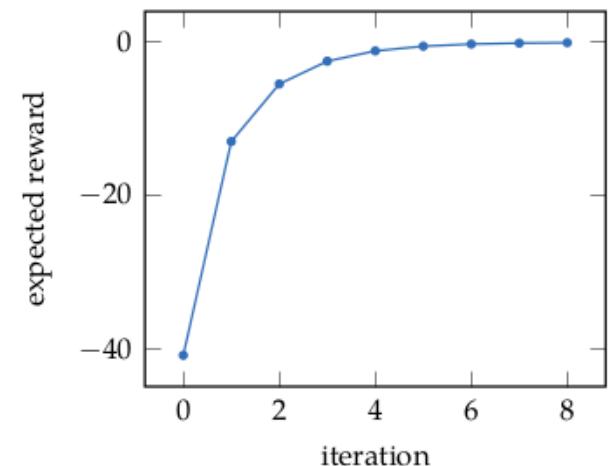
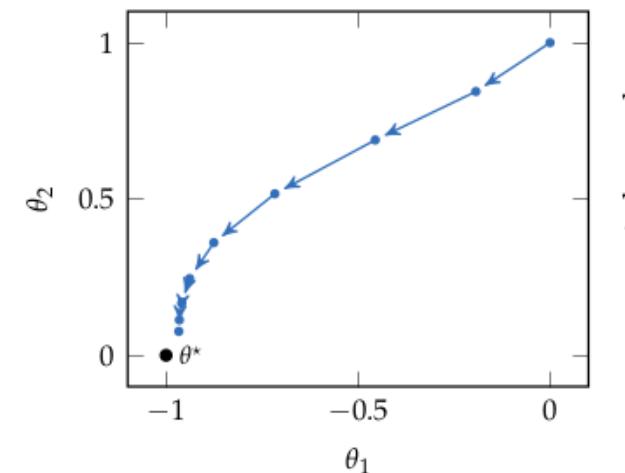
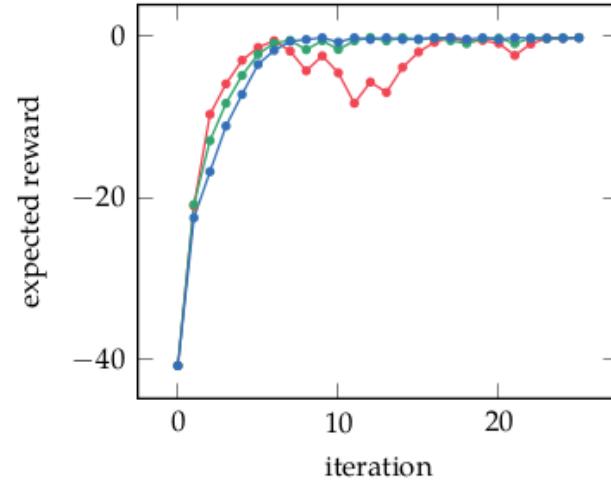
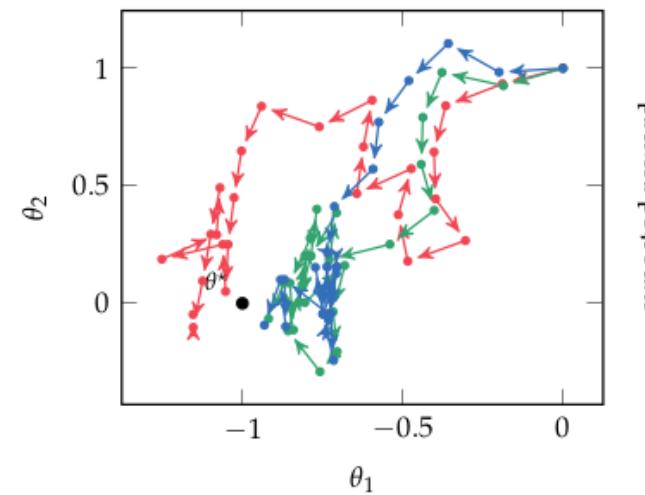
TRPO and PPO

likelihood ratio
reward-to-go
baseline subtraction

TRPO = Trust Region Policy Optimization
(Natural gradient + line search)



PPO = Proximal Policy Optimization
(Use clamped surrogate objective to remove the need for line search)



Part III

Actor-Critic

Actor-Critic

Actor-Critic

Which should we learn? A , Q , or V ?

$$\nabla U(\theta) = E_{\tau} \left[\sum_{k=0}^d \nabla_{\theta} \log \pi_{\theta}(a_k \mid s_k) \gamma^k (r_k + \gamma V_{\phi}(s_{k+1}) - V_{\phi}(s_k)) \right]$$

temporal difference residual

$$l(\phi) = E \left[(V_{\phi}(s) - V^{\pi_{\theta}}(s))^2 \right]$$

*estimate with
reward to go
from sims*

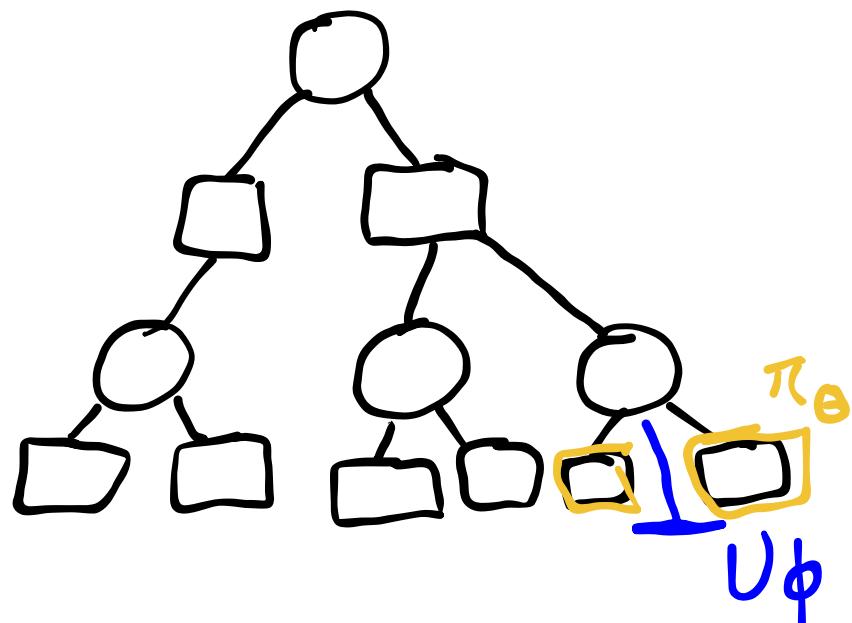
Generalized Advantage Estimation

Recap

Alpha Zero: Actor Critic with MCTS

1. Use π_θ and U_ϕ in MCTS
2. Learn π_θ and U_ϕ from tree

$$\ell(\boldsymbol{\theta}) = -\mathbb{E}_s \left[\sum_a \pi_{\text{MCTS}}(a | s) \log \pi_\theta(a | s) \right]$$
$$\pi_{\text{MCTS}}(a | s) \propto N(s, a)^\eta$$



$$\ell(\boldsymbol{\Phi}) = \frac{1}{2} \mathbb{E}_s \left[(U_\Phi(s) - U_{\text{MCTS}}(s))^2 \right]$$

$$U_{\text{MCTS}}(s) = \max_a Q(s, a)$$

$$a = \arg \max_a Q(s, a) + c \pi_\theta(a | s) \frac{\sqrt{N(s)}}{1 + N(s, a)}$$