DQN and Advanced Policy Gradient

Map

Map

Map

Model Based ⟷ Model Free

learn Q	learn π
SARSA	Policy Gradient

On Policy

Off Policy

MLMBTRL
(learn T, R)

Q-learning

Map

Model Based

Model Free

learn Q

SARSA

learn π

Policy Gradient

On Policy

Off Policy

MLMBTRL
(learn T,R)

Q-learning

Challenges:

1. Exploration vs Exploitation
2. Credit Assignment
3. Generalization

Map

Model Based

Model Free

learn Q

SARSA

learn π

Policy Gradient

On Policy

Off Policy

MLMBTRL
(learn T,R)

Q-learning

Challenges:

1. Exploration vs Exploitation
2. Credit Assignment
3. Generalization

Last Time: Neural Networks

Map

Model Based ⟷ Model Free

learn Q

SARSA

learn π

Policy Gradient

On Policy

Off Policy

MLMBTRL
(learn T,R)

Q-learning

Part 1

Challenges:

1. Exploration vs Exploitation
2. Credit Assignment
3. Generalization

Last Time: Neural Networks

Map

Model Based

Model Free

learn Q

SARSA

learn π

Policy Gradient

Part 2

On Policy

Off Policy

MLMBTRL
(learn T,R)

Q-learning

Part 1

Challenges:

1. Exploration vs Exploitation
2. Credit Assignment
3. Generalization

Last Time: Neural Networks

Map

Model Based

Model Free

learn Q

SARSA

learn π

Policy Gradient

Part 2

On Policy

Off Policy

MLMBTRL (learn T,R)

Q-learning

Part1

Actor-Critic Part3

Challenges:

1. Exploration vs Exploitation
2. Credit Assignment
3. Generalization

Last Time: Neural Networks

Part I

DQN

Q-Learning with Neural Networks

Q-Learning with Neural Networks

Q-Learning:

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

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

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

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

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

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?

Q-Learning with Neural Networks

Q-Learning:

TD → 0

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

temporal difference

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? (s, a, r, s')
- What should l be?

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

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

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

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

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

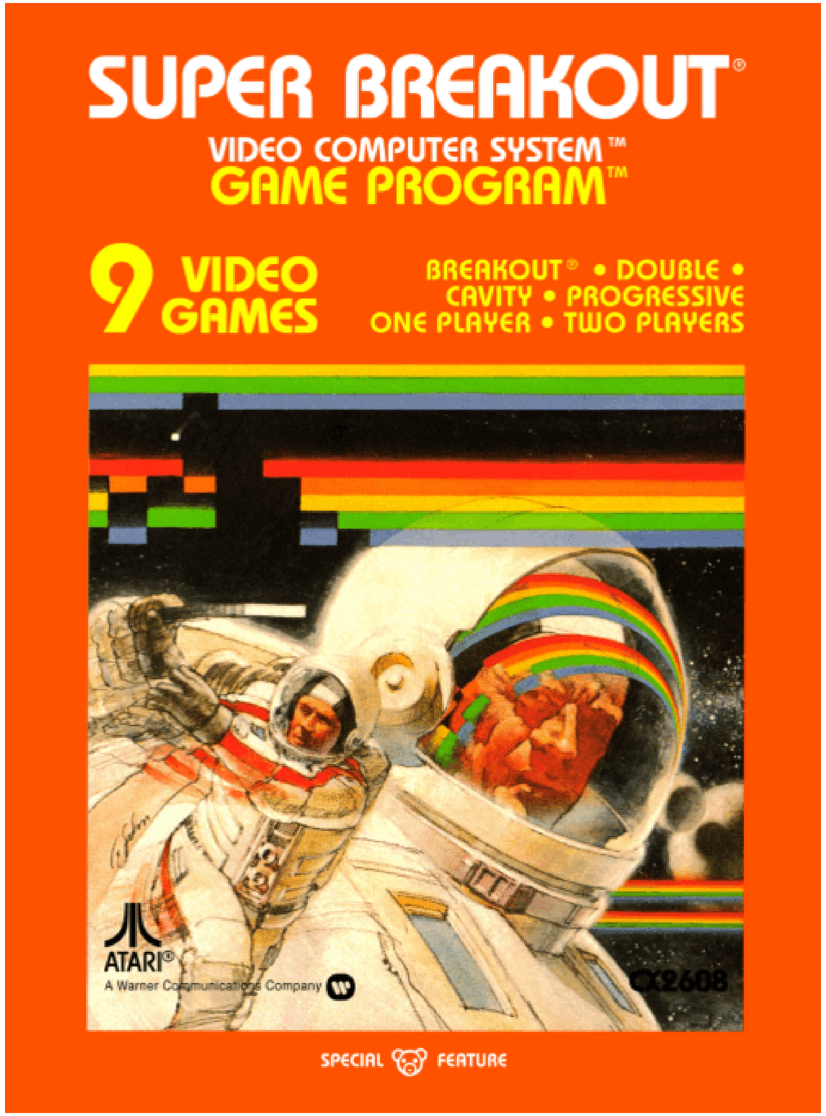
$$\theta \leftarrow \theta - \alpha \nabla_\theta \left(r + \gamma \max_{a'} Q_\theta(s',a') - Q_\theta(s,a)\right)^2$$

$$s \leftarrow s'$$

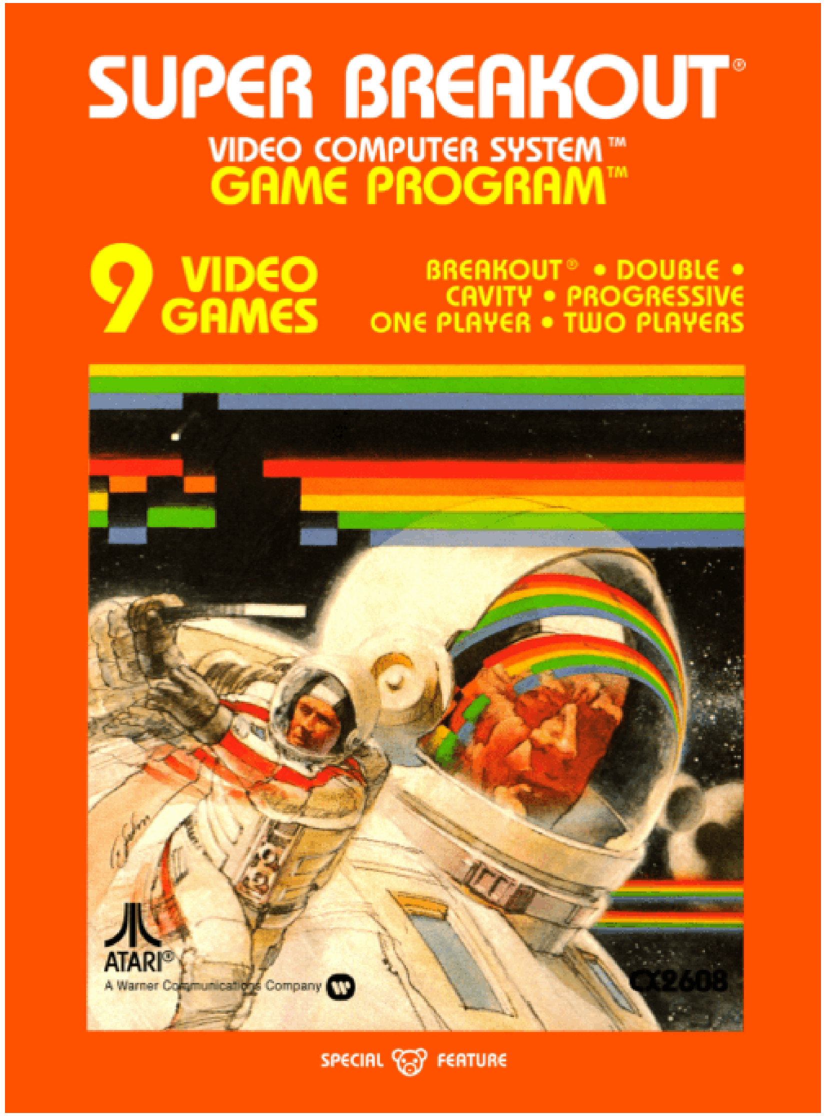
DQN: The Atari Benchmark

DQN: The Atari Benchmark

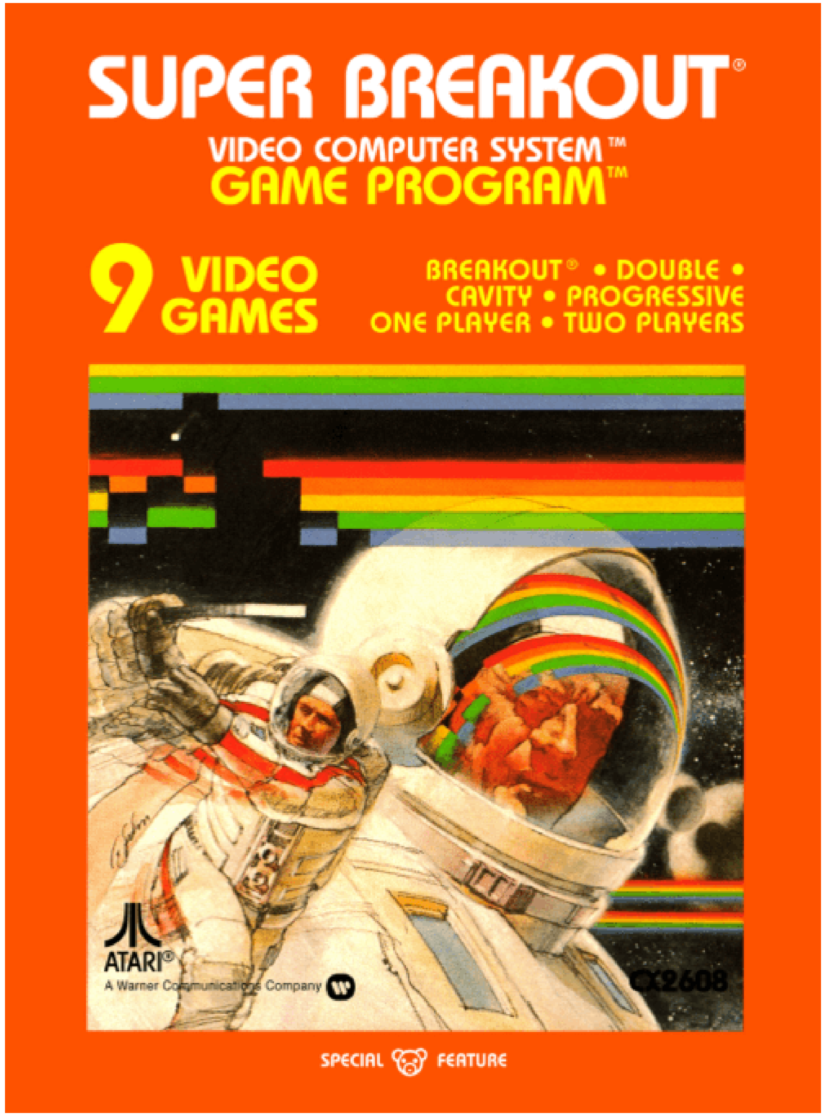
DQN: The Atari Benchmark



DQN: The Atari Benchmark



DQN: The Atari Benchmark



DQN: Problems with Naive Approach

DQN: Problems with Naive Approach

Candidate Algorithm:

loop

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

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

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

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

$$s \leftarrow s'$$

DQN: Problems with Naive Approach

Candidate Algorithm:

loop

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

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

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

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

$$s \leftarrow s'$$

Problems:

1. Samples Highly Correlated

DQN: Problems with Naive Approach

Candidate Algorithm:

loop

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

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

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

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

$$s \leftarrow s'$$

Problems:

1. Samples Highly Correlated
2. Size-1 batches

DQN: Problems with Naive Approach

Candidate Algorithm:

loop

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

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

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

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

$$s \leftarrow s'$$

Problems:

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

DQN: Problems with Naive Approach

Candidate Algorithm:

loop

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

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

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

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

$$s \leftarrow s'$$

Problems:

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

} data buffer/experience replay

} periodically freeze target

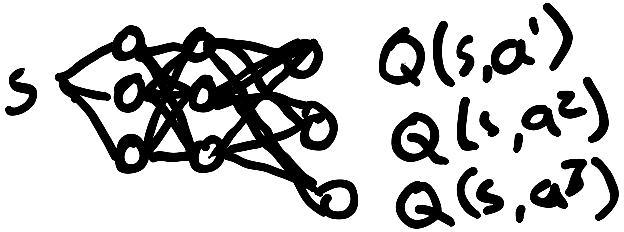
DQN

DQN

Q Network Structure:

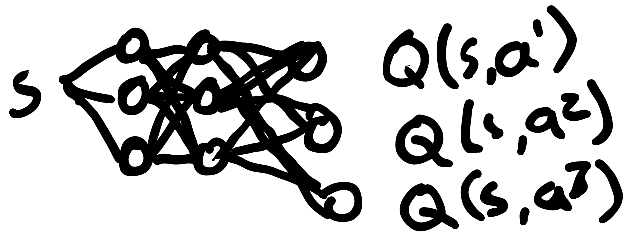
DQN

Q Network Structure:



DQN

Q Network Structure:

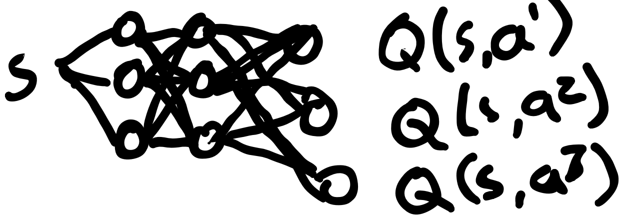


Experience Tuple: (s, a, r, s')

DQN

Trick 0

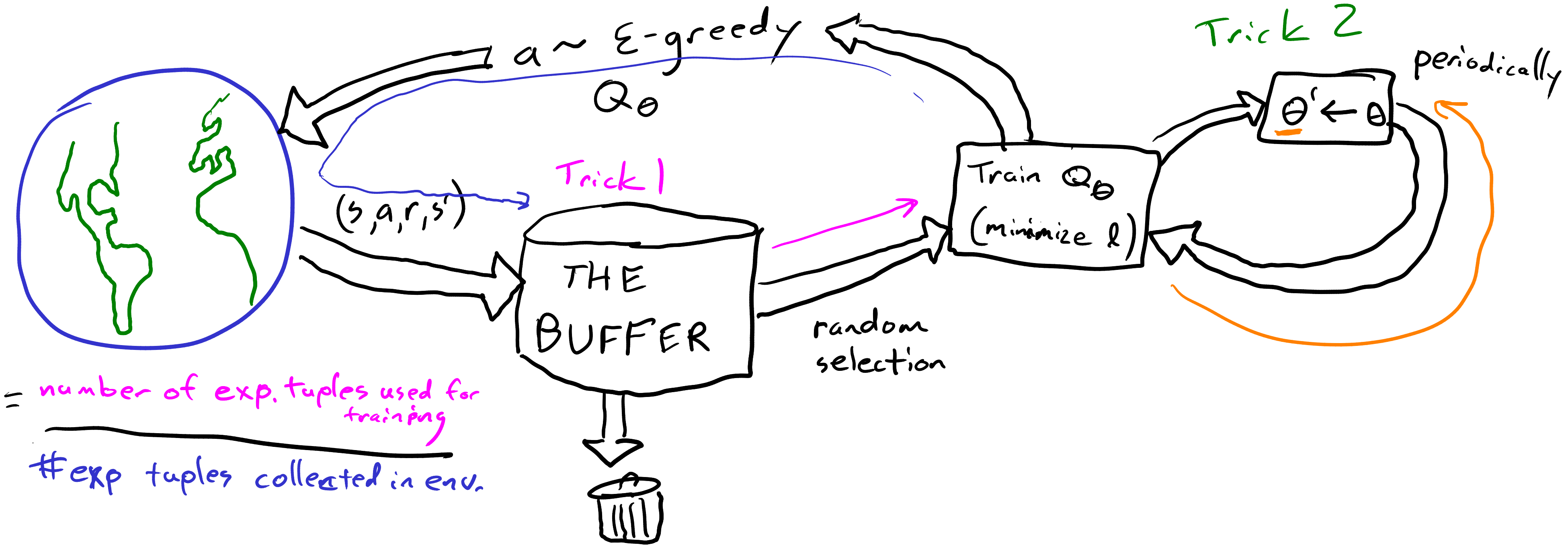
Q Network Structure:



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

Loss:

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



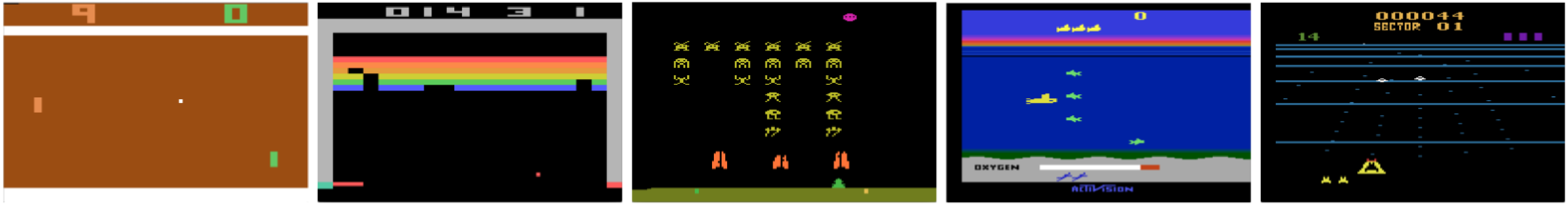
$$\text{Training Intensity} = \frac{\text{number of exp. tuples used for training}}{\text{\# exp tuples collected in env.}}$$

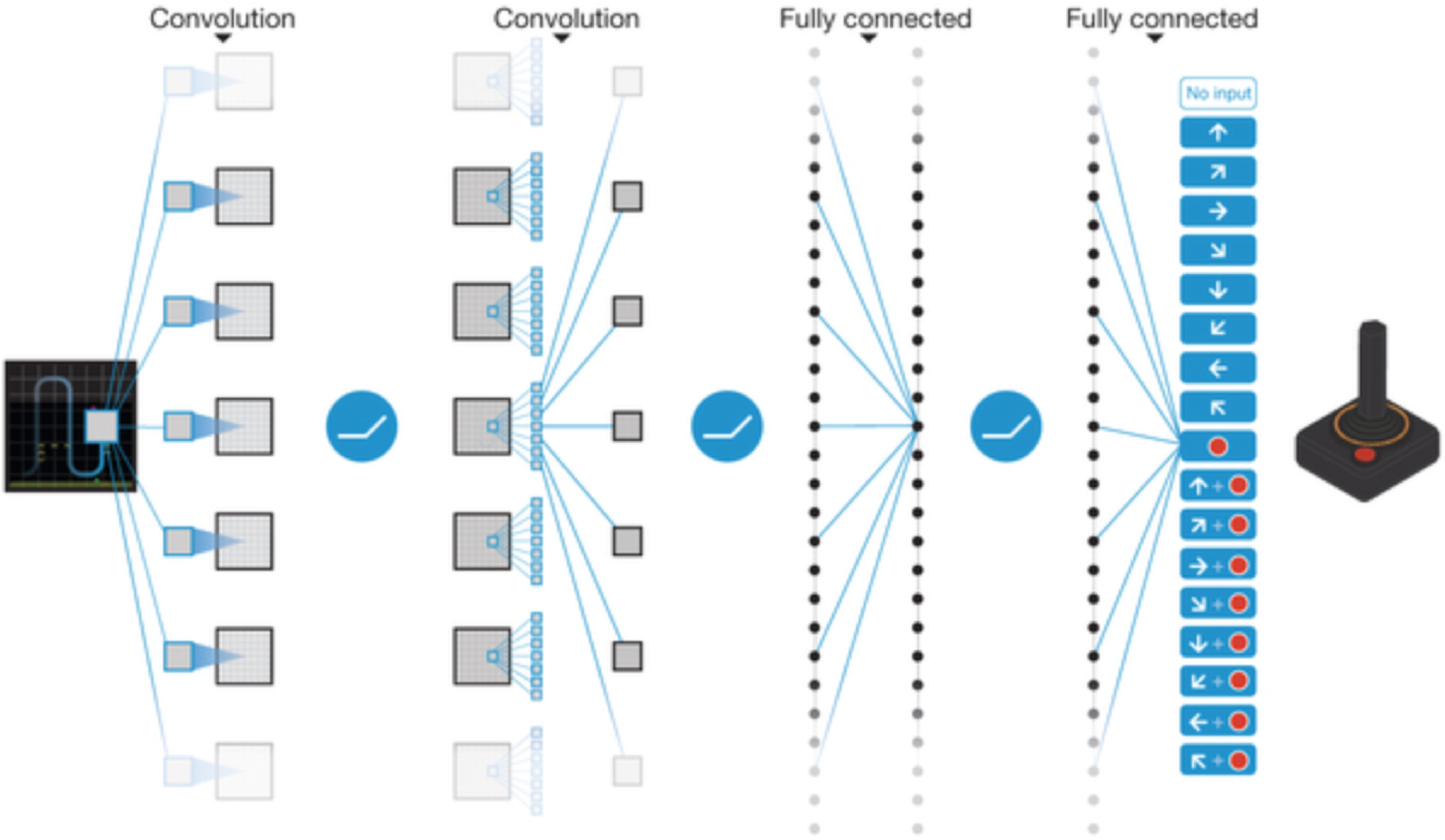
Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou

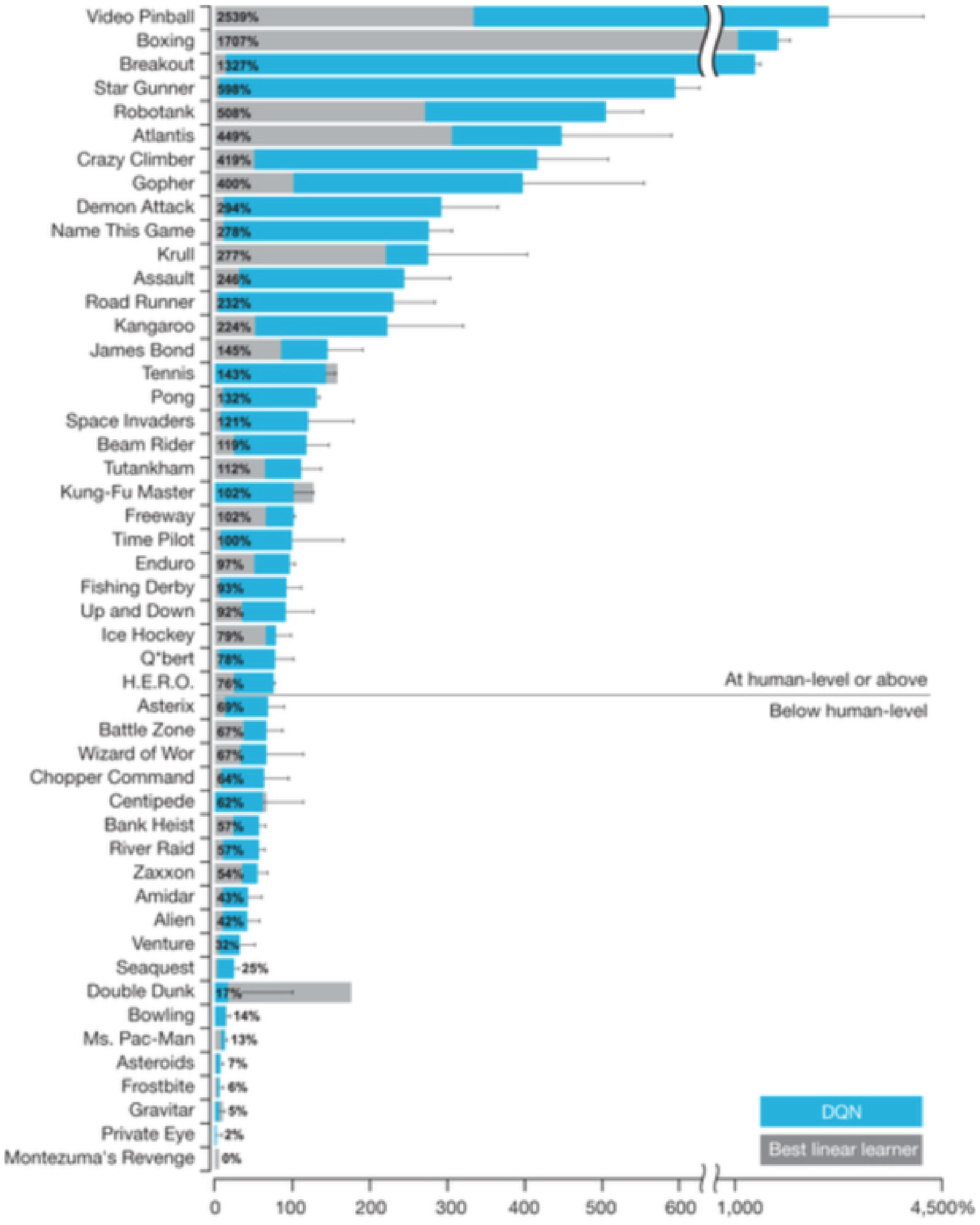
Daan Wierstra Martin Riedmiller

DeepMind Technologies

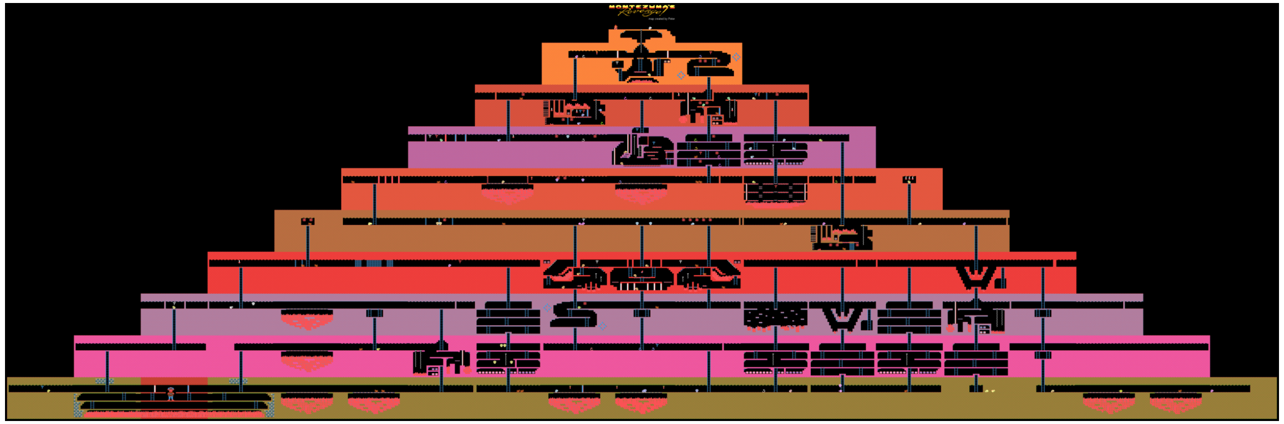




Convolution
Convolution
Fully connected
Fully connected
No input



https://www.youtube.com/watch?v=SuZVyOlgVek



Rainbow

Rainbow

- Double Q Learning

Rainbow

- Double Q Learning
- Prioritized Replay
  (priority proportional to last TD error)

Rainbow

- Double Q Learning
- Prioritized Replay
  (priority proportional to last TD error)
- Dueling networks
  Value network + advantage network
  $Q(s,a) = V(s) + A(s,a)$

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
  $(r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \gamma \max Q_\theta(s_{t+n}, a') - Q_\theta(s_t, a_t))^2$

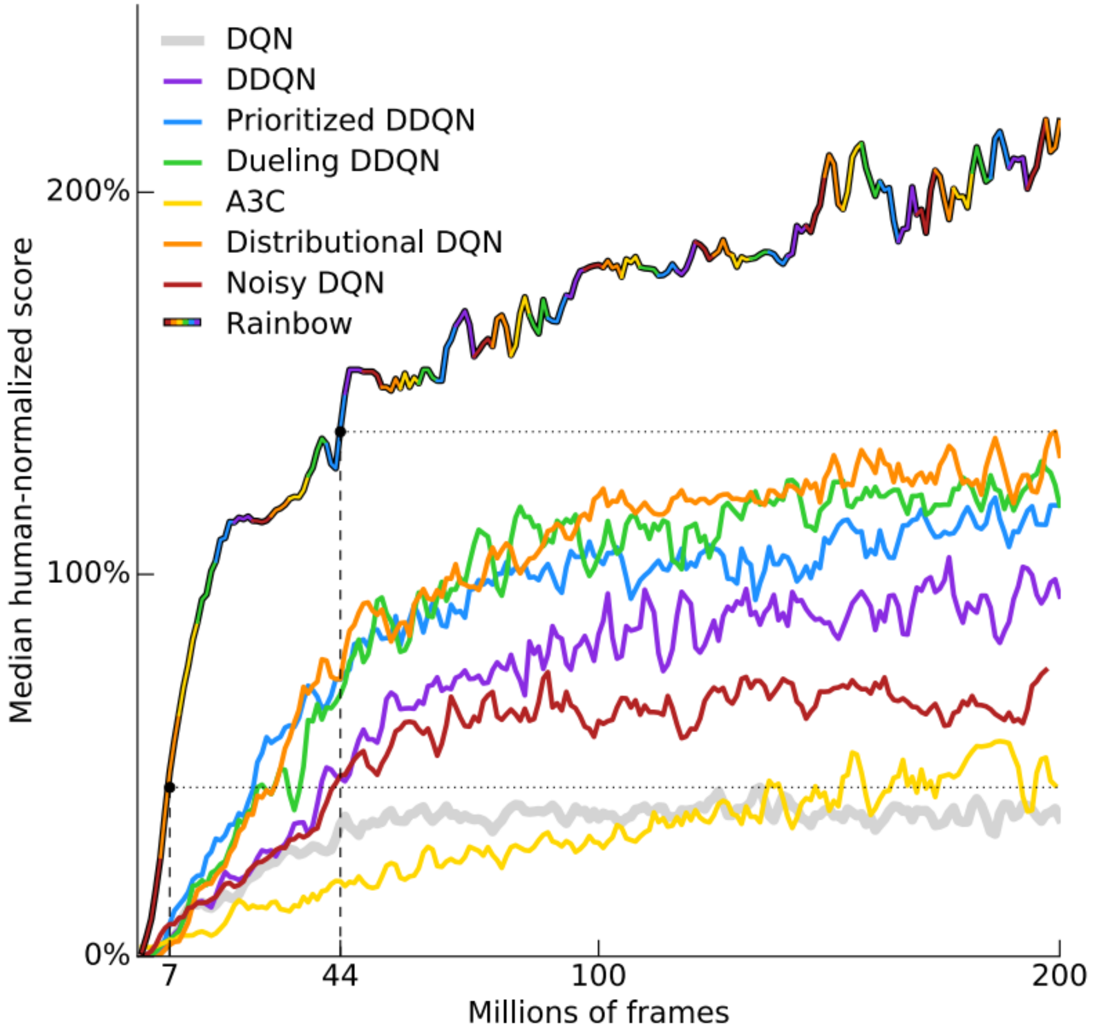
Rainbow

- 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
  $(r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \gamma \max Q_\theta(s_{t+n}, a') - Q_\theta(s_t, a_t))^2$
- Distributional RL
  predict an entire distribution of values instead of just Q

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
  $(r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \gamma \max Q_\theta(s_{t+n}, a') - Q_\theta(s_t, a_t))^2$
- Distributional RL
  predict an entire distribution of values instead of just Q
- Noisy Nets

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
  $(r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \gamma \max Q_\theta(s_{t+n}, a') - Q_\theta(s_t, a_t))^2$
- Distributional RL
  predict an entire distribution of values instead of just Q
- Noisy Nets

Actual Learning Curves

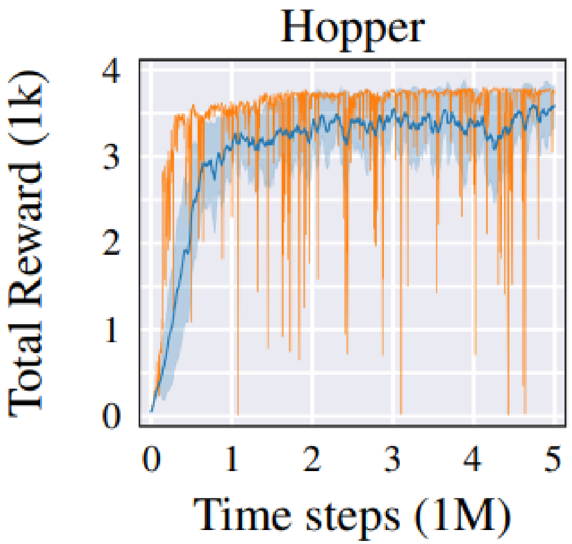


Figure 24: Standard presentation Single seed
The training curve as commonly presented in RL papers and the training curve of a single seed. Both curves are from the TD3 algorithm, trained for 5M time steps. The standard presentation is to evaluate every N_{freq} steps, average scores over N_{episodes} evaluation episodes and N_{seeds} seeds, then to smooth the curve by averaging over a window of N_{window} evaluations. (In our case this corresponds to $N_{\text{freq}} = 5000$, $N_{\text{episodes}} = 10$, $N_{\text{seeds}} = 10$, $N_{\text{window}} = 10$). The learning curve of a single seed has no smoothing over seeds or evaluations ($N_{\text{seeds}} = 1$, $N_{\text{window}} = 1$). By averaging over many seeds and evaluations, the training curves in RL can appear deceptively smooth and stable.

Paper: For SALE: State-Action Representation Learning for Deep Reinforcement Learning

Part II

Improved Policy Gradients

Restricted Gradient Update

Restricted Gradient Update

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

Restricted Gradient Update

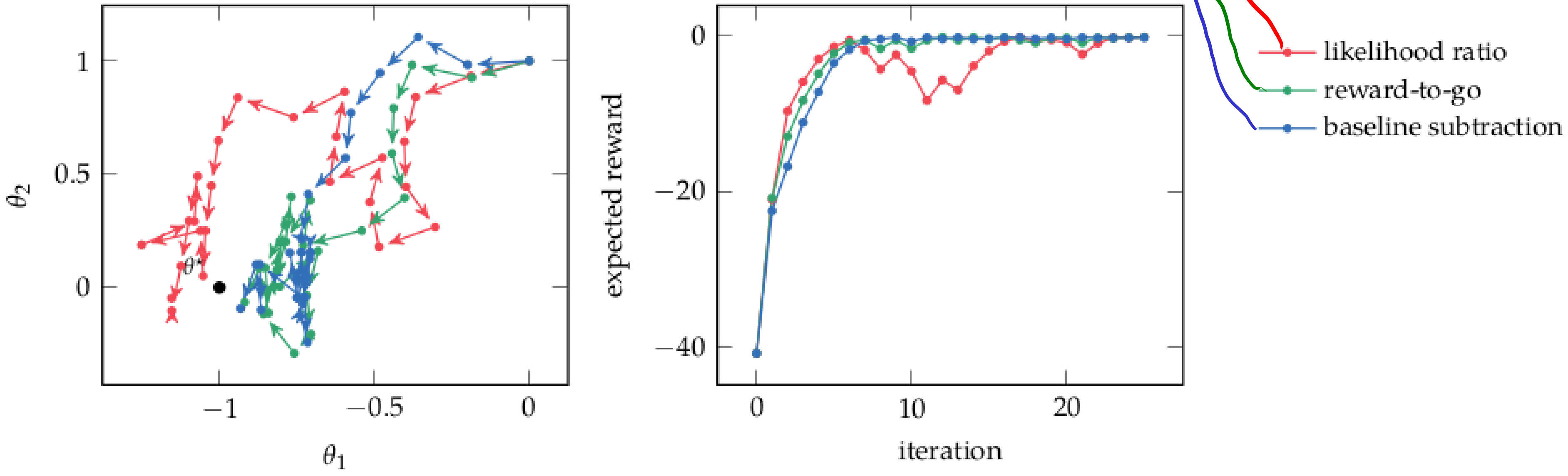
$$\widehat{\nabla U}(\theta) = \sum_{k=0}^{d} \nabla_\theta \log \pi_\theta(a_k \mid s_k)\gamma^k (r_{k,\text{to-go}} - r_{\text{base}}(s_k))$$

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

Restricted Gradient Update

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

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



Restricted Gradient Update

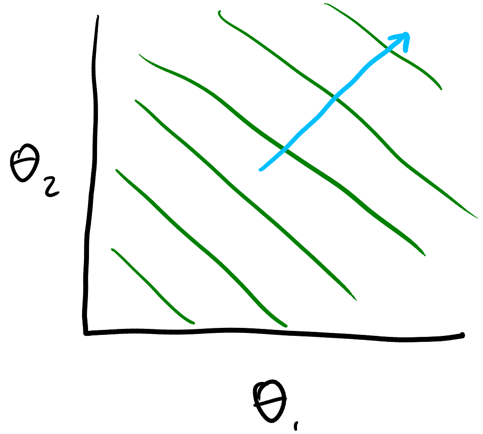
$$\widehat{\nabla U}(\theta) = \sum_{k=0}^{d} \nabla_\theta \log \pi_\theta(a_k \mid s_k)\gamma^k(r_{k,\text{to-go}} - r_{\text{base}}(s_k))$$

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

$$\underset{\theta'}{\text{maximize}} \quad U(\theta') \approx U(\theta) + \nabla U(\theta)^T(\theta' - \theta)$$

$$\text{subject to} \quad g(\theta, \theta') \leq \varepsilon$$

$$g(\theta, \theta') = \|\theta - \theta'\|_2^2 = \frac{1}{2}(\theta' - \theta)^T(\theta' - \theta)$$



$$\theta' = \theta + u\sqrt{\frac{2\varepsilon}{u^T u}} = \theta + \sqrt{2\varepsilon}\frac{u}{\|u\|}$$

$$u = \nabla U(\theta)$$

Natural Gradient

$$g(\theta,\theta') \approx D_{KL}(p(\tau|\theta) \| p(\tau|\theta'))$$

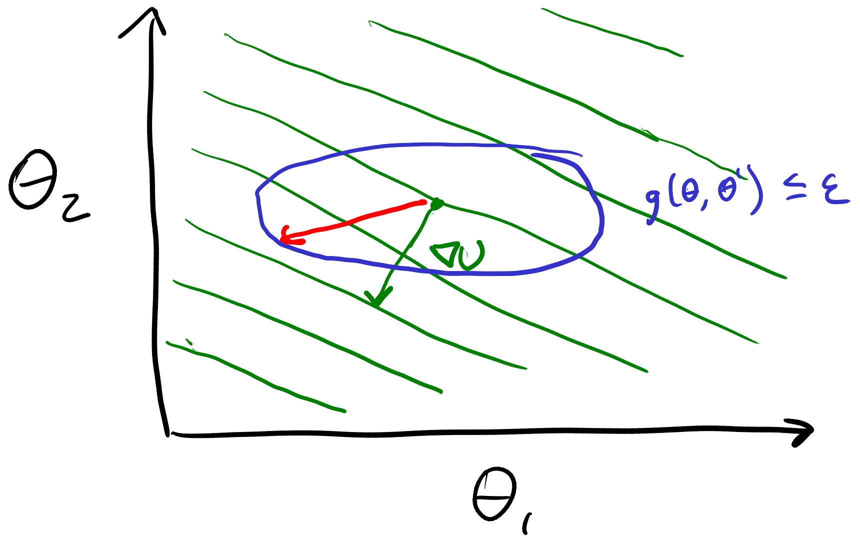
$$D_{KL}(p \| q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

$$\underset{\theta'}{\text{maximize}} \quad U(\theta) + \nabla U(\theta)^T(\theta' - \theta)$$

$$\text{subject to} \quad g(\theta,\theta') = \frac{1}{2}(\theta' - \theta)^T F_\theta (\theta' - \theta) \le \varepsilon$$

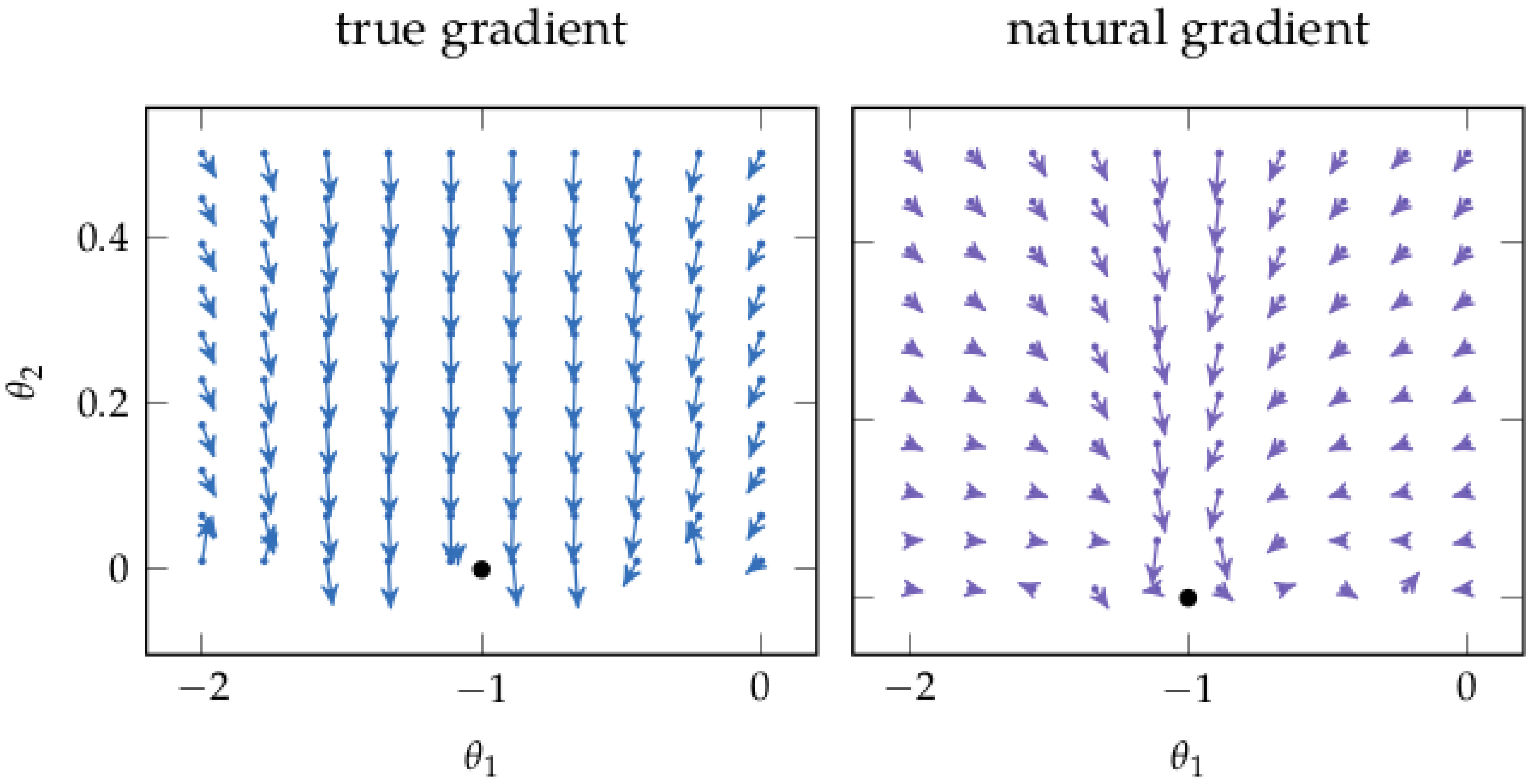
$$F_\theta = \mathop{E}_{\tau}\left[\nabla \log p(\tau|\theta) \nabla \log p(\tau|\theta)^T\right]$$

$$\theta' = \theta + u\sqrt{\frac{2\varepsilon}{\nabla U(\theta)^T u}} \qquad u = F_\theta^{-1} \nabla U(\theta)$$

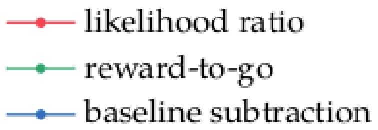


$\vdots$

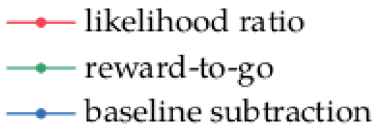
Natural Gradient



TRPO and PPO

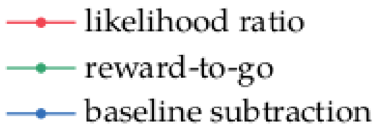


TRPO and PPO



TRPO = Trust Region Policy Optimization

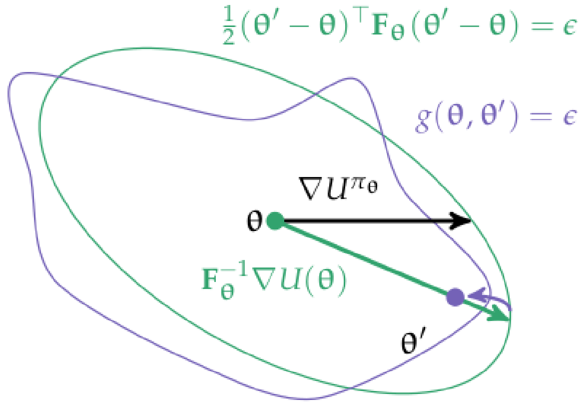
(Natural gradient + line search)



TRPO and PPO

TRPO = Trust Region Policy Optimization

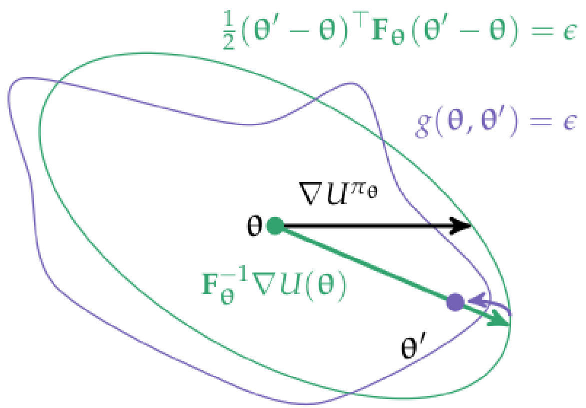
(Natural gradient + line search)

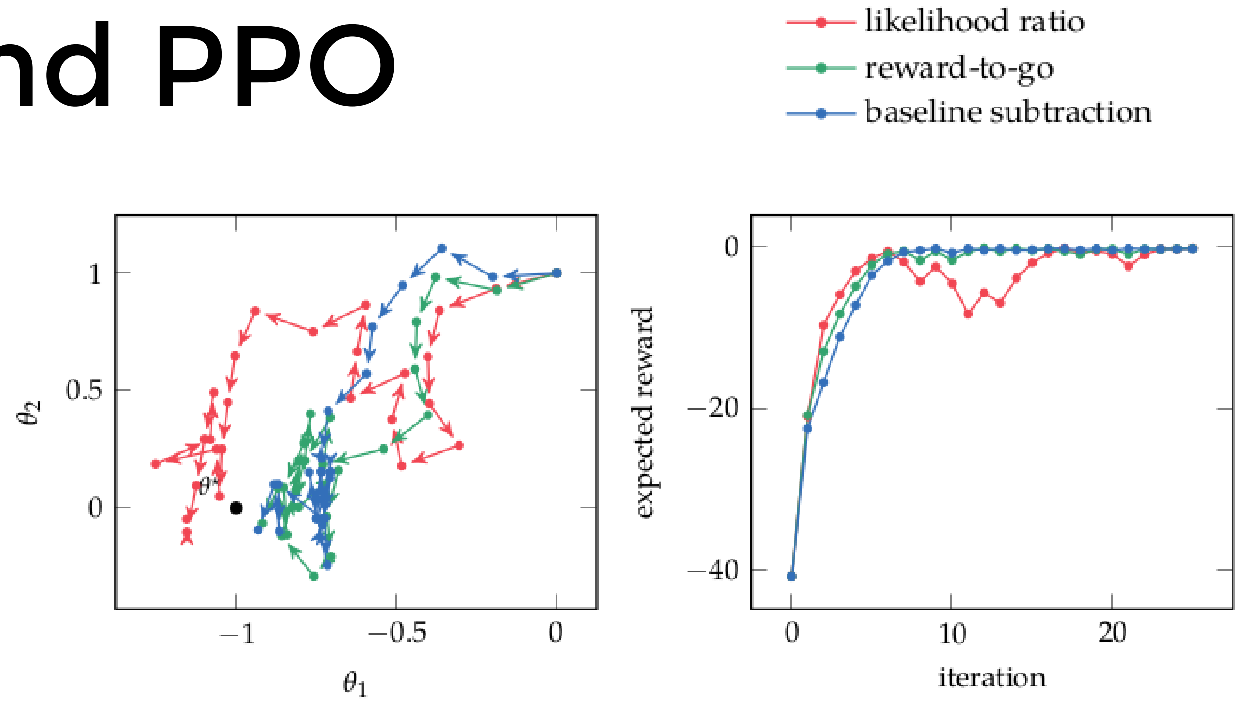


TRPO and PPO

TRPO = Trust Region Policy Optimization

(Natural gradient + line search)

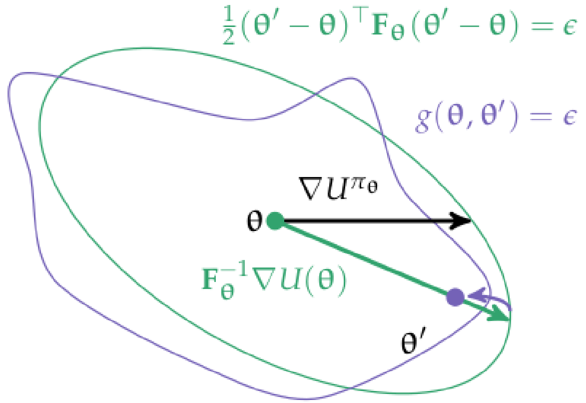


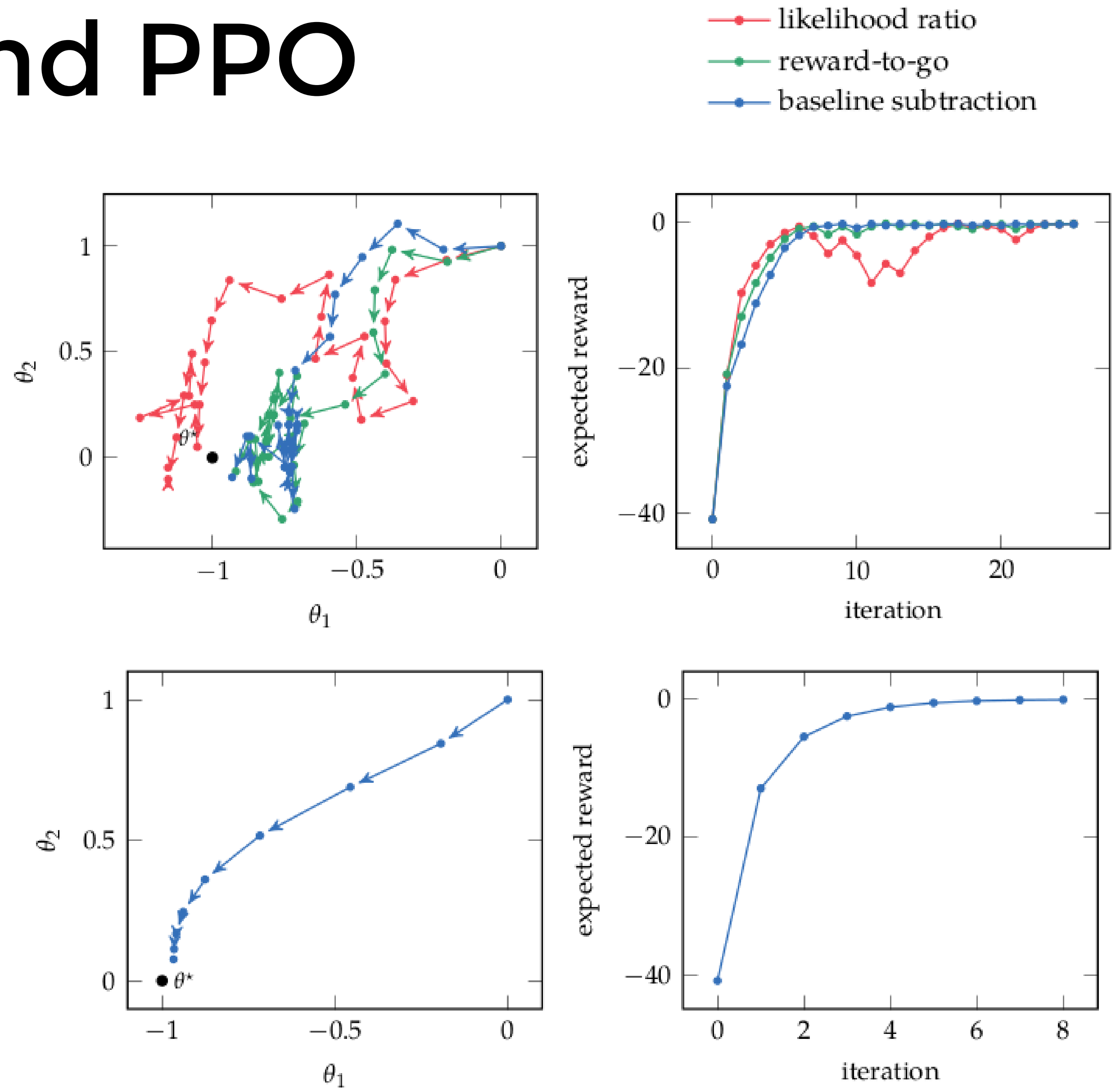


TRPO and PPO

TRPO = Trust Region Policy Optimization

(Natural gradient + line search)

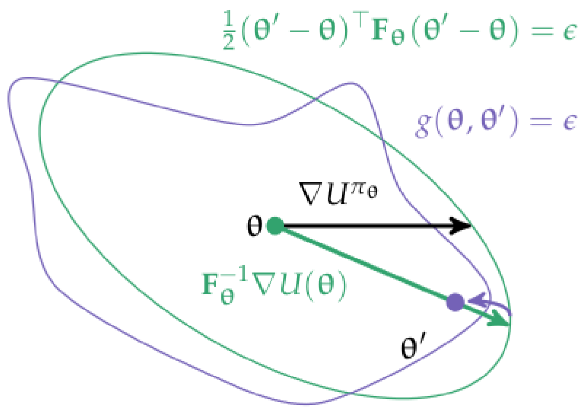


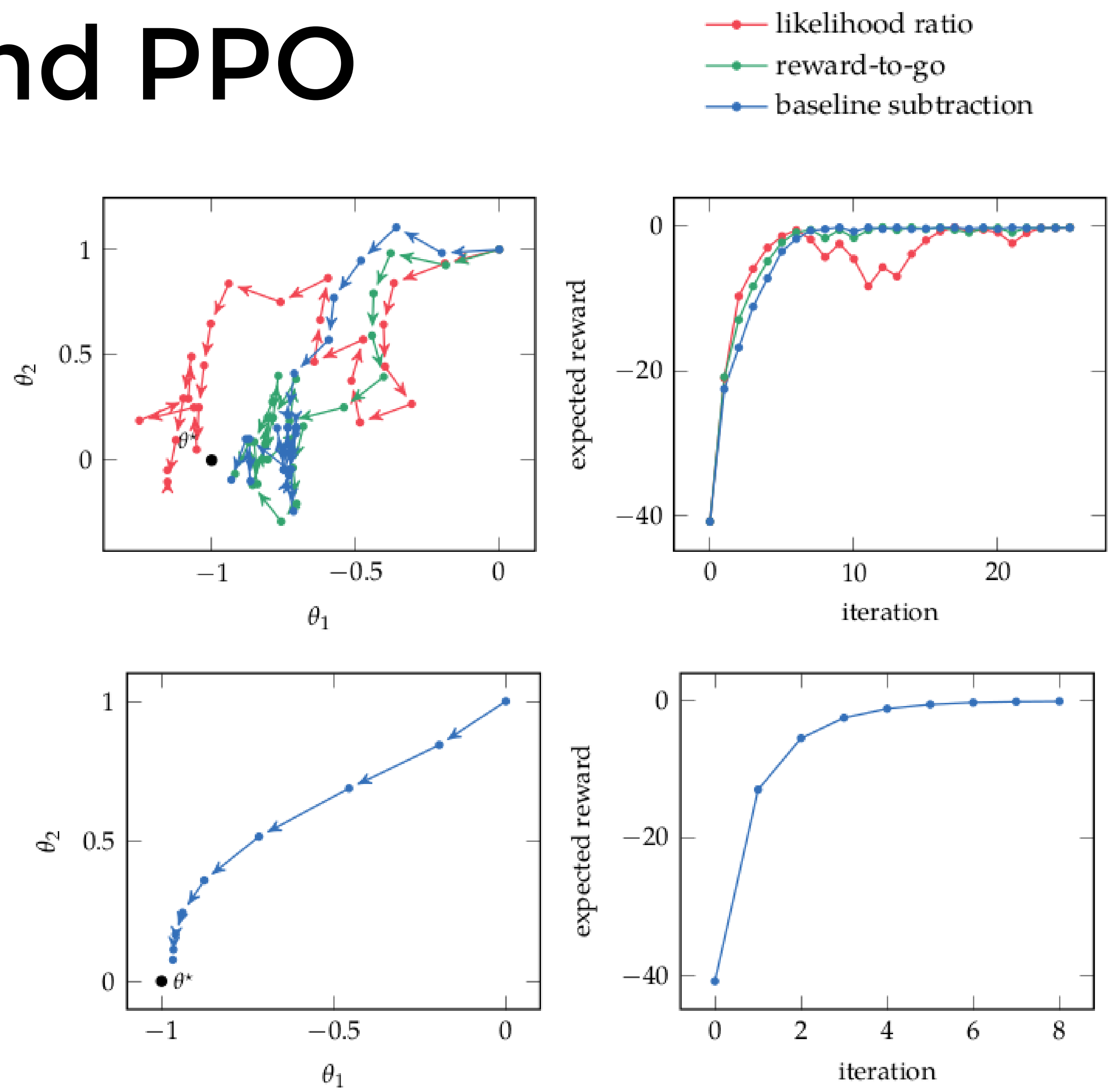


TRPO and PPO

TRPO = Trust Region Policy Optimization

(Natural gradient + line search)

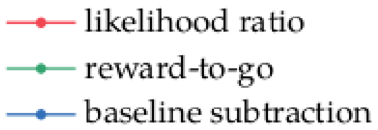




PPO = Proximal Policy Optimization

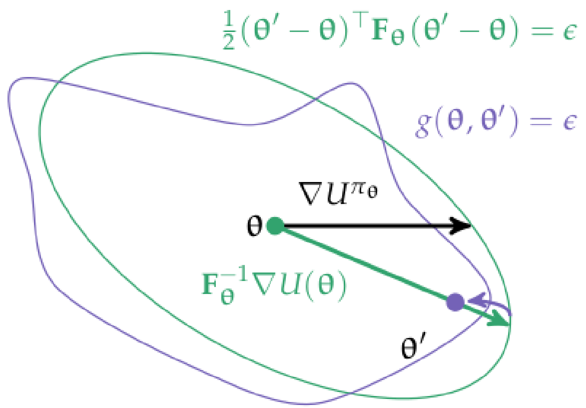
(Use clamped surrogate objective to remove the need for line search)

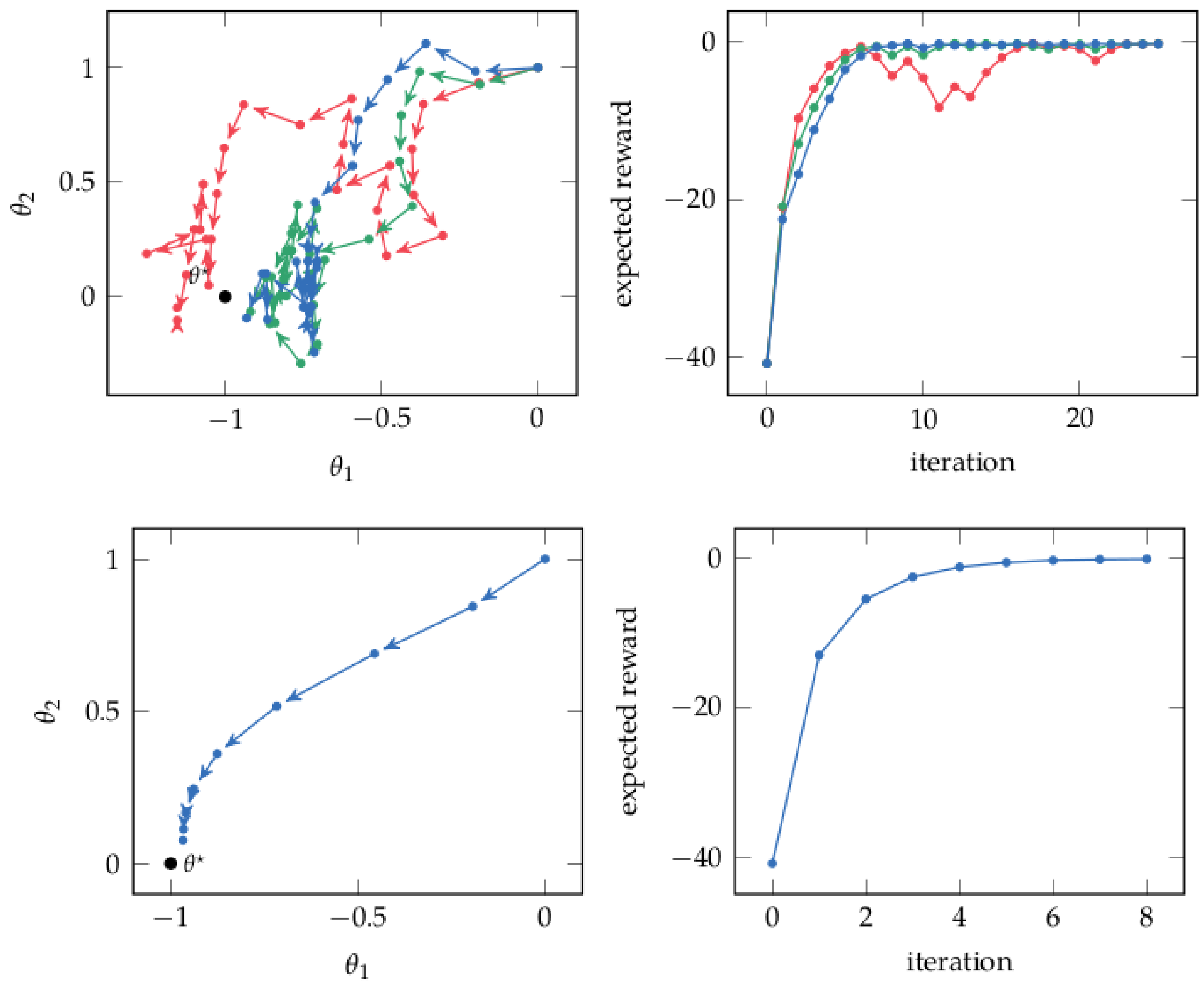
TRPO and PPO



TRPO = Trust Region Policy Optimization

(Natural gradient + line search)





PPO = Proximal Policy Optimization

(Use clamped surrogate objective to remove the need for line search)

https://openai.com/blog/openai-baselines-ppo/

Part III

Actor-Critic

Actor-Critic

Actor-Critic

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

Actor-Critic

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

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

Actor-Critic

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

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

Actor-Critic

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

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

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

Actor-Critic

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

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

$$l(\phi) = E\left[\left(V_\phi(s) - \underbrace{V^{\pi_\theta}(s)}_{\text{estimate with reward to go from sims}} \right)^2 \right]$$

Generalized Advantage Estimation

Recap

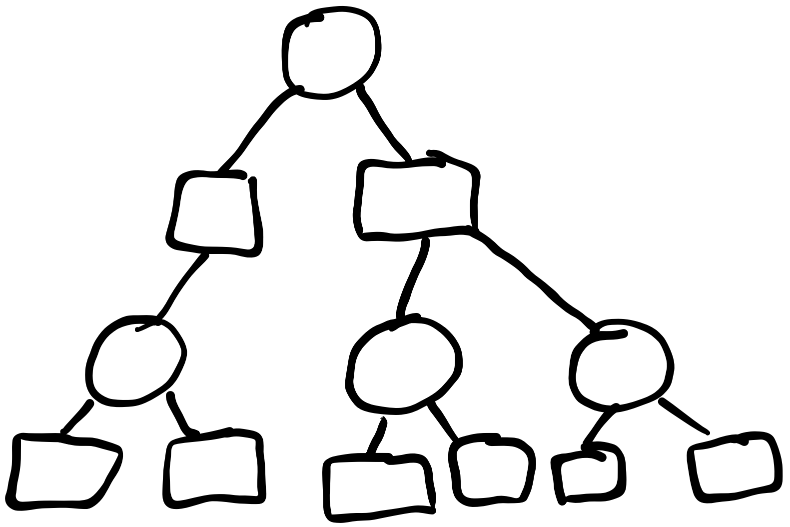
Alpha Zero: Actor Critic with MCTS

Alpha Zero: Actor Critic with MCTS

1. Use π_θ and U_ϕ in MCTS

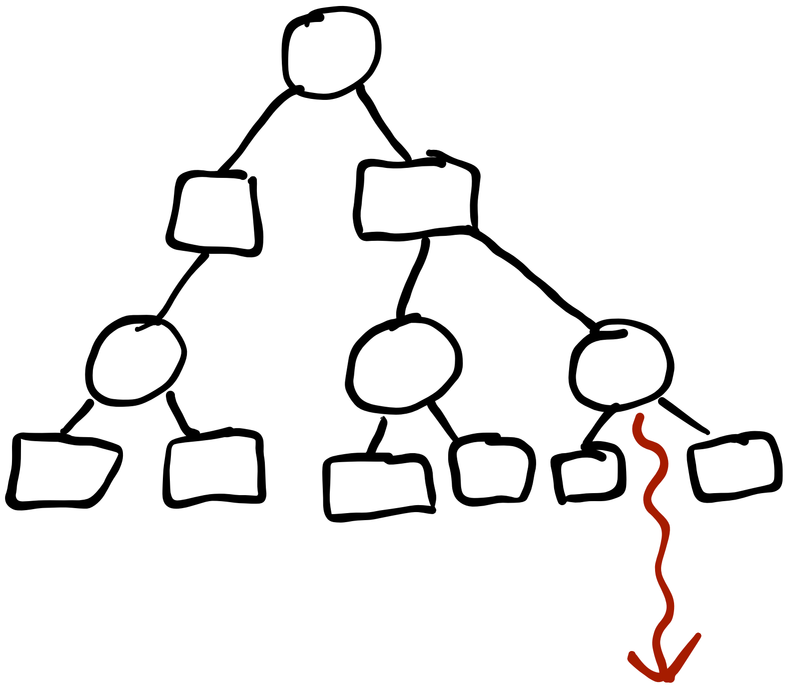
Alpha Zero: Actor Critic with MCTS

1. Use π_θ and U_ϕ in MCTS



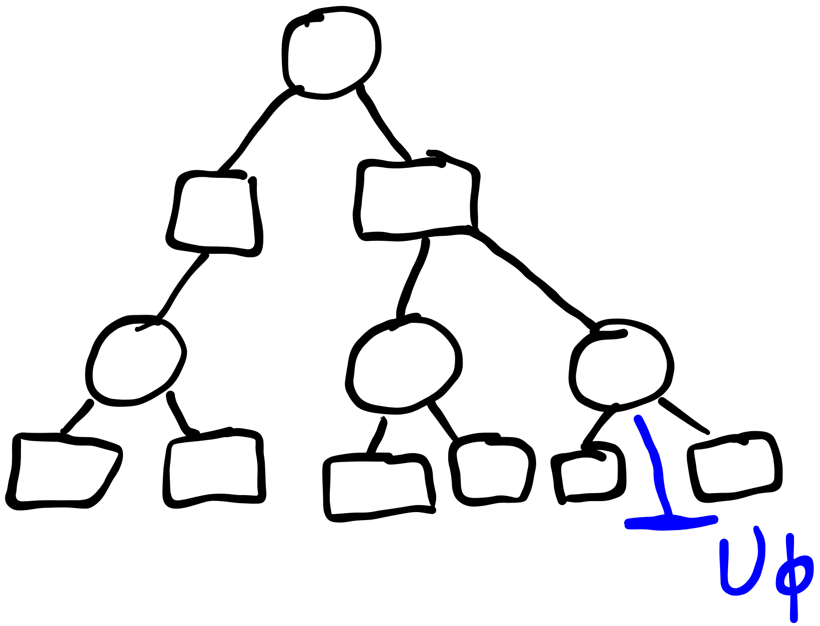
Alpha Zero: Actor Critic with MCTS

1. Use π_θ and U_ϕ in MCTS



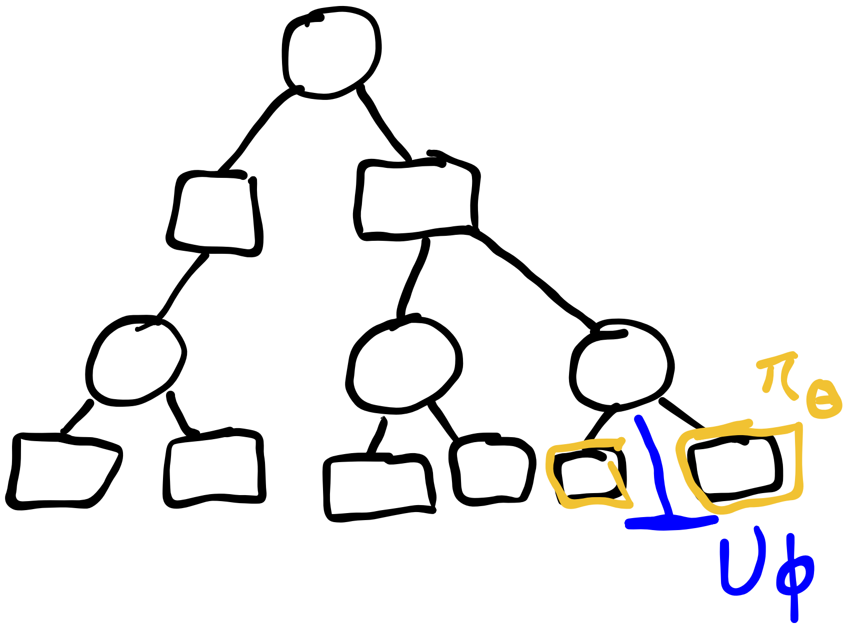
Alpha Zero: Actor Critic with MCTS

1. Use π_θ and U_ϕ in MCTS



Alpha Zero: Actor Critic with MCTS

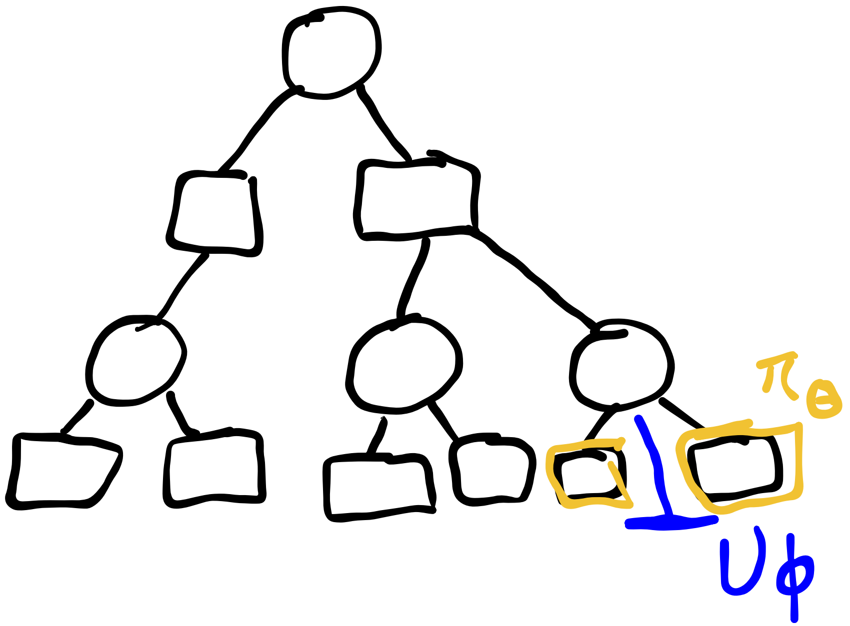
1. Use π_θ and U_ϕ in MCTS



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

Alpha Zero: Actor Critic with MCTS

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



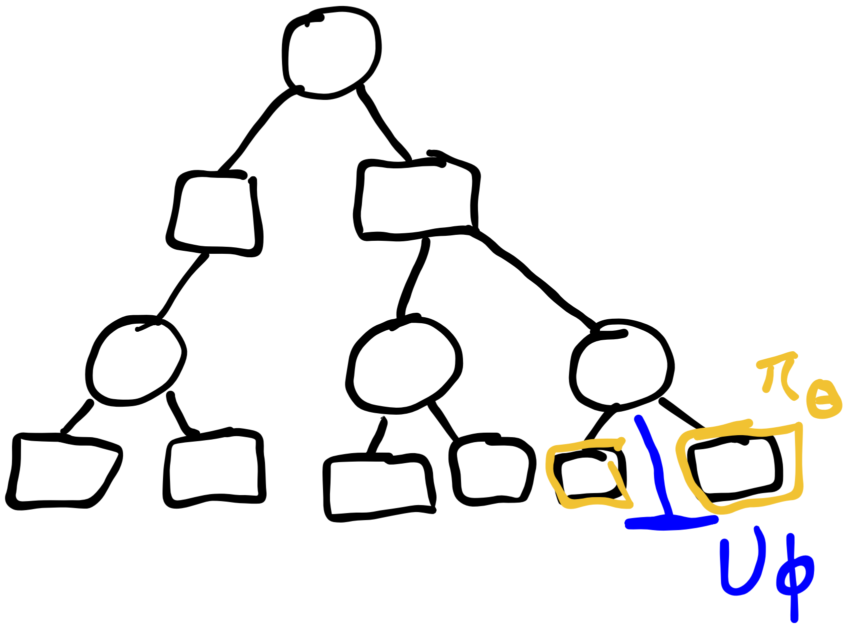
$$a = \arg\max_a Q(s,a) + c\pi_\theta(a \mid s)\frac{\sqrt{N(s)}}{1+N(s,a)}$$

Alpha Zero: Actor Critic with MCTS

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

$$\ell(\theta) = -\mathbb{E}_s\left[\sum_a \pi_{\text{MCTS}}(a \mid s) \log \pi_\theta(a \mid s)\right]$$

$$\pi_{\text{MCTS}}(a \mid s) \propto N(s,a)^\eta$$



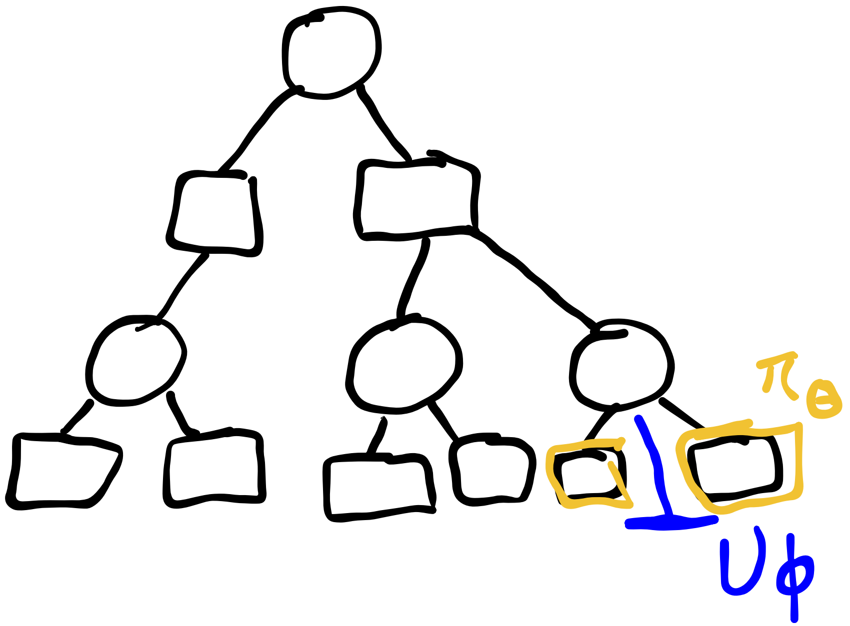
$$a = \arg\max_a Q(s,a) + c\pi_\theta(a \mid s)\frac{\sqrt{N(s)}}{1+N(s,a)}$$

Alpha Zero: Actor Critic with MCTS

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

$$\ell(\theta) = -\mathbb{E}_s\left[\sum_a \pi_{\text{MCTS}}(a \mid s) \log \pi_\theta(a \mid s)\right]$$

$$\pi_{\text{MCTS}}(a \mid s) \propto N(s,a)^\eta$$



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

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

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