Reinforcement learning

Policy gradient, state-value function, advantage function, actor-critic methods, inverse reinforcement learning

Karel Zimmermann

Problems often formalised as MDP

States: $\mathbf{x} \in \mathcal{R}^n$

Problems often formalised as MDP

States: $\mathbf{x} \in \mathcal{R}^n$

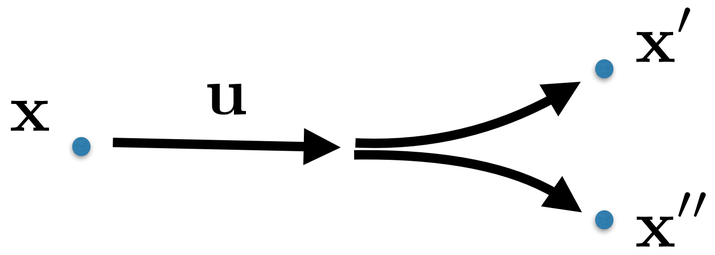
Actions: $\mathbf{u} \in \mathcal{R}^m$

Problems often formalised as MDP

States: $\mathbf{x} \in \mathcal{R}^n$

Actions: $\mathbf{u} \in \mathcal{R}^m$

Model: $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$



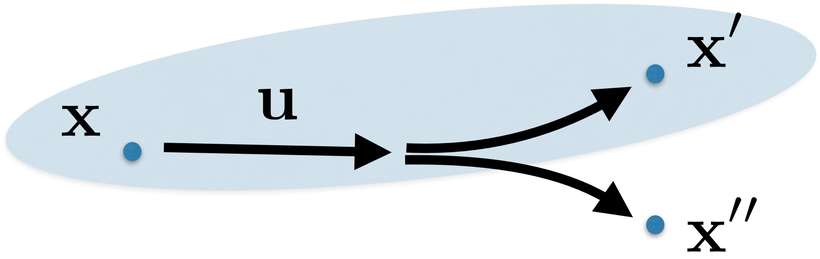
Problems often formalised as MDP

States: $\mathbf{x} \in \mathcal{R}^n$

Actions: $\mathbf{u} \in \mathcal{R}^m$

Model: $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

Rewards: $r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$



Problems often formalised as MDP

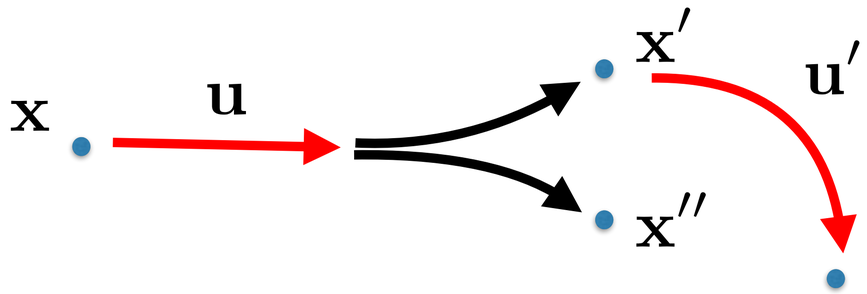
States: $\mathbf{x} \in \mathcal{R}^n$

Actions: $\mathbf{u} \in \mathcal{R}^m$

Model: $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

Rewards: $r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$

Policy: $\pi(\mathbf{u}|\mathbf{x})$



Problems often formalised as MDP

States: $\mathbf{x} \in \mathcal{R}^n$

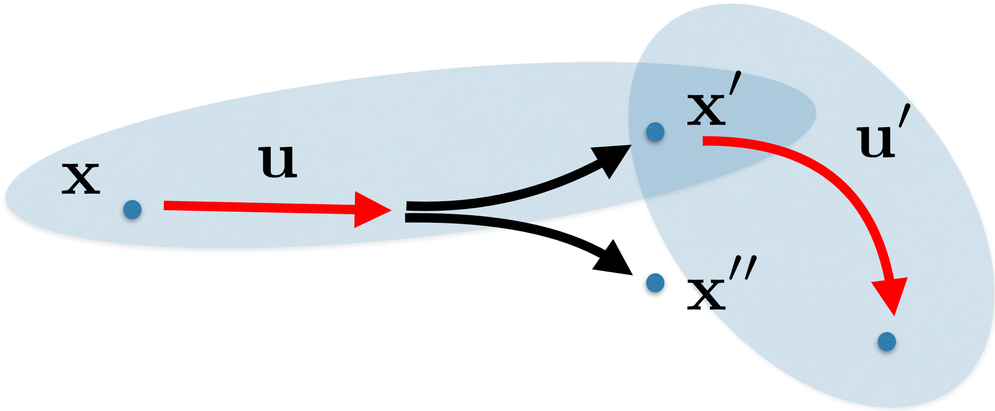
Actions: $\mathbf{u} \in \mathcal{R}^m$

Model: $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

Rewards: $r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$

Policy: $\pi(\mathbf{u}|\mathbf{x})$

Goal: $\pi^* = \arg\max_{\pi} J_\pi$ (e.g. $J_\pi = \mathbb{E}_{\tau\sim\pi}\{\sum_{r_t\sim\tau} \gamma^t r_t\}$)



Problems often formalised as MDP

States: $\mathbf{x} \in \mathcal{R}^n$ incomplete, noisy

Actions: $\mathbf{u} \in \mathcal{R}^m$ continuous high-dimensional

Model: $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$ inaccurate model

Rewards: $r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$ hard to engineer

Policy: $\pi(\mathbf{u}|\mathbf{x})$ execution endanger the robot

Goal: $\pi^* = \arg\max_{\pi} J_\pi$ (e.g. $J_\pi = \mathbb{E}_{\tau\sim\pi}\{\sum_{r_t\sim\tau} \gamma^t r_t\}$)

Typical problems

τ

Model identification:
- given some trajectories
  estimate model

$p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

$p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}')$

Model predictive control / Planning
- given the model and reward
  estimate optimal policy/plan

$\pi^* = \arg\max_{\pi} J_\pi$

τ

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}')$

Reinforcement learning:
- given rewards and trajectories,
  estimate optimal policy

$\pi^* = \arg\max_{\pi} J_\pi$

τ^*

Inverse reinforcement learning:
- given optimal trajectories
  estimate reward function

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$

Typical problems

τ

Model identification:
- given some trajectories
  estimate model

$p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

$p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}')$

Model predictive control / Planning
- given the model and reward
  estimate optimal policy/plan

$\pi^* = \arg\max_{\pi} J_\pi$

τ

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}')$

Reinforcement learning:
- given rewards and trajectories,
  estimate optimal policy

$\pi^* = \arg\max_{\pi} J_\pi$

τ^*

Inverse reinforcement learning:
- given optimal trajectories
  estimate reward function

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$

Model identification:

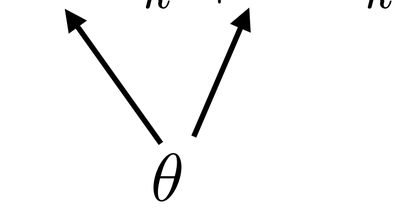
- **Employ physics engine** and experimentally identify its physical quantities

- **Learn network** (deep convolutional) to predict following state $\mathbf{x}_{k+1} = p_\theta(\mathbf{x}_k, \mathbf{u}_k)$.

$$\arg\min_\theta \sum_k \| p_\theta(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_{k+1} \|_2^2$$

- For example fully observable, time-discrete, linear system:

$$\mathbf{x}_{k+1} = p_\theta(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$$



Typical problems

τ

Model identification:
- given some trajectories
  estimate model

$p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

$p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}')$

Model predictive control / Planning
- given the model and reward
  estimate optimal policy/plan

$\pi^* = \arg\max_{\pi} J_\pi$

τ

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}')$

Reinforcement learning:
- given rewards and trajectories,
  estimate optimal policy

$\pi^* = \arg\max_{\pi} J_\pi$

τ^*

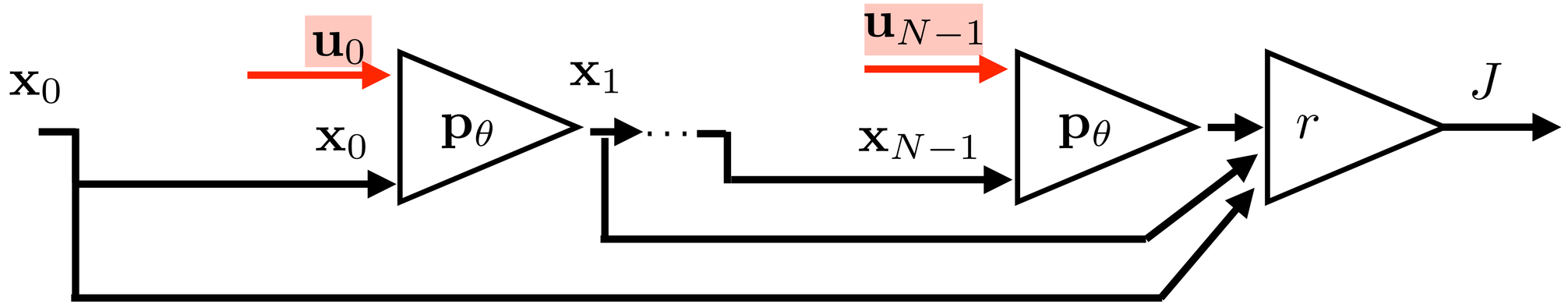
Inverse reinforcement learning:
- given optimal trajectories
  estimate reward function

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$

Planning actions

Plan policy (sequence of actions) maximizing the rewards on model-based trajectories

$$\underset{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}}{\arg\max} \left\{ \sum_k r(\mathbf{x}_k, \mathbf{u}_k) \;\middle|\; \mathbf{x}_{k+1} = p_\theta(\mathbf{x}_k, \mathbf{u}_k) \right\}$$

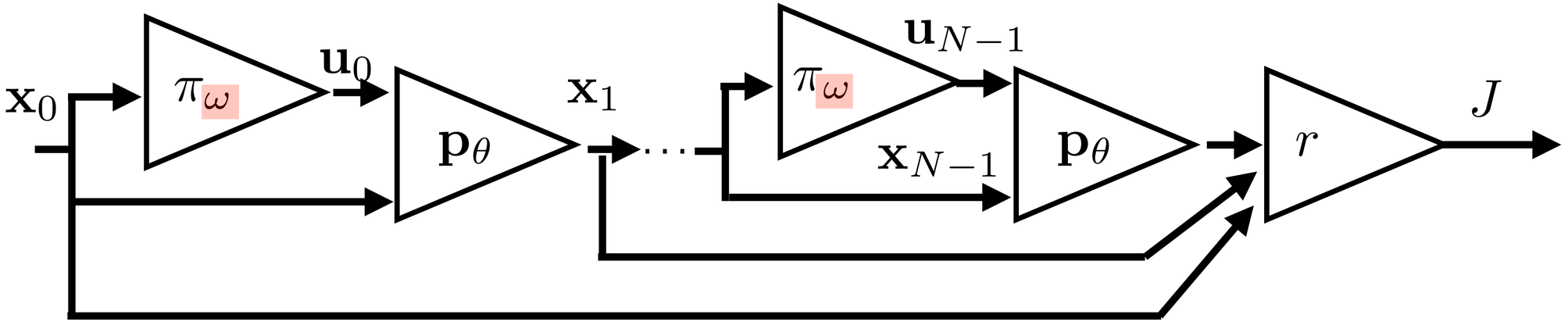


- typically strongly non-convex => gradient optimization inefficient
- exhaustive search (BFS, Dijkstra, A*, RRT, MPC) => **repeated open loop control**

Learning policy

Learn policy (e.g. deep net) maximizing the rewards on model-based trajectories

$$\arg\max_{\omega} \left\{ \sum_{k} r(\mathbf{x}_k, \mathbf{u}_k) \;\middle|\; \mathbf{x}_{k+1} = p_\theta(\mathbf{x}_k, \mathbf{u}_k),\ \mathbf{u}_k = \pi_\omega(\mathbf{x}_k) \right\}$$



- Especially: linear system and policy + quadratic reward function
- LQR has closed form solution => **closed loop control**

Typical problems

τ

Model identification:
- given some trajectories
  estimate model

$p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

$p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}')$

Model predictive control / Planning
- given the model and reward
  estimate optimal policy/plan

$\pi^* = \arg\max_{\pi} J_\pi$

τ

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}')$

Reinforcement learning:
- given rewards and trajectories,
  estimate optimal policy

$\pi^* = \arg\max_{\pi} J_\pi$

τ^*

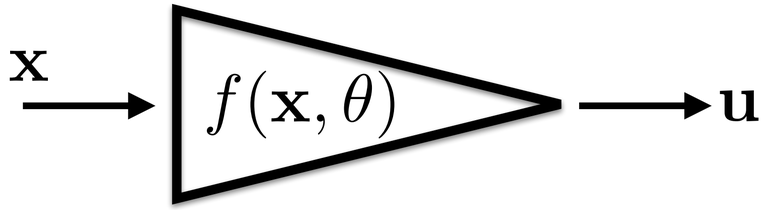
Inverse reinforcement learning:
- given optimal trajectories
  estimate reward function

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$

Deterministic vs stochastic policy

Deterministic policy for continuous control:

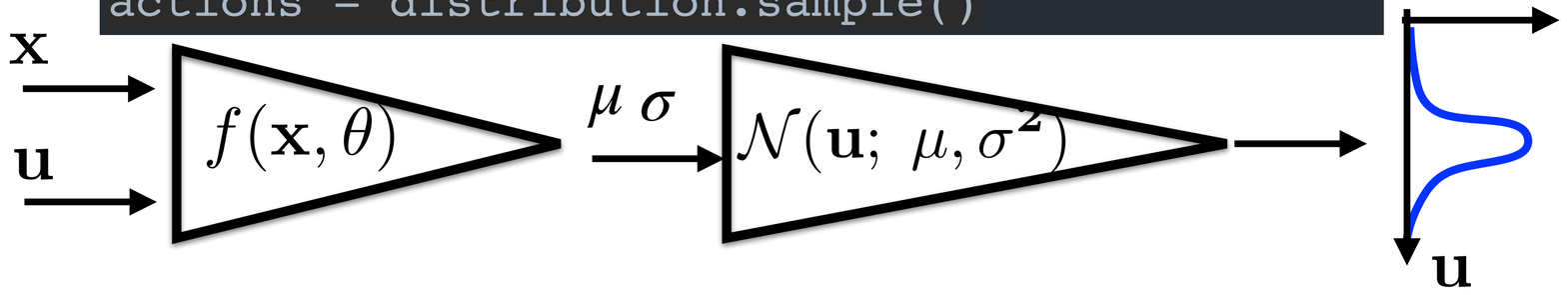
$$\pi_\theta(\mathbf{u}|\mathbf{x}) = f(\mathbf{x}, \theta)$$



Stochastic policy for continuous control:

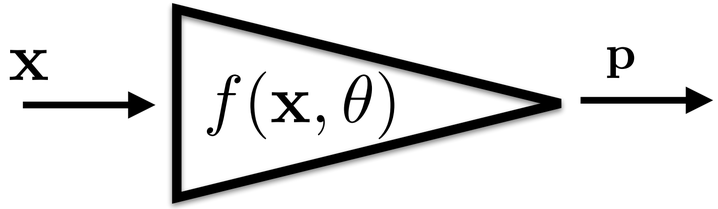
$$\pi_\theta(\mathbf{u}|\mathbf{x}) \sim \mathcal{N}(\mathbf{u};\ \mu, \sigma^2)$$

```
from torch.distributions.normal import Normal
distribution = Normal(mu, sigma)
actions = distribution.sample()
```



Stochastic policy for discrete control:

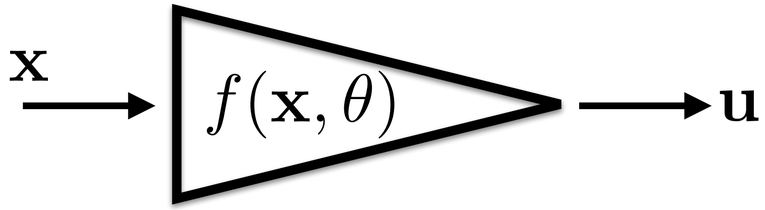
$$\pi_\theta(\mathbf{u}|\mathbf{x}) \sim p(\mathbf{u};\ \mathbf{p})$$



Deterministic vs stochastic policy

Deterministic policy for continuous control:

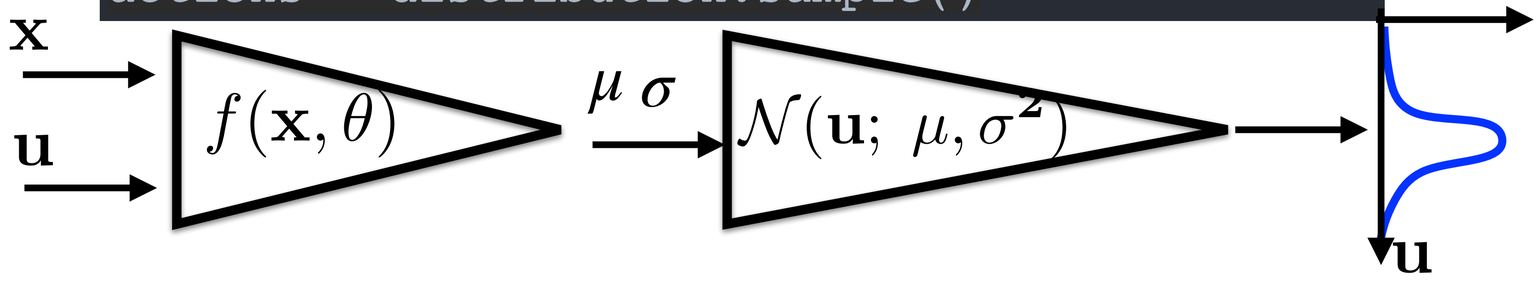
$$\pi_\theta(\mathbf{u}|\mathbf{x}) = f(\mathbf{x}, \theta)$$



Stochastic policy for continuous control:

$$\pi_\theta(\mathbf{u}|\mathbf{x}) \sim \mathcal{N}(\mathbf{u};\ \mu, \sigma^2)$$

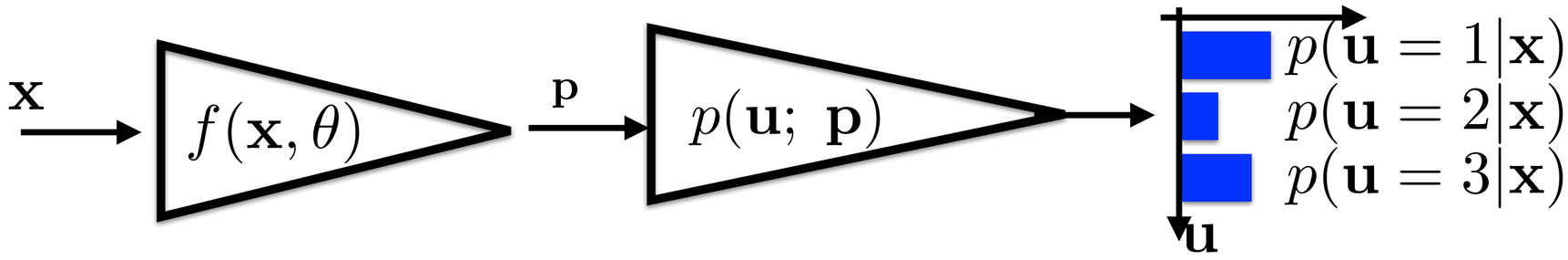
```
from torch.distributions.normal import Normal
distribution = Normal(mu, sigma)
actions = distribution.sample()
```

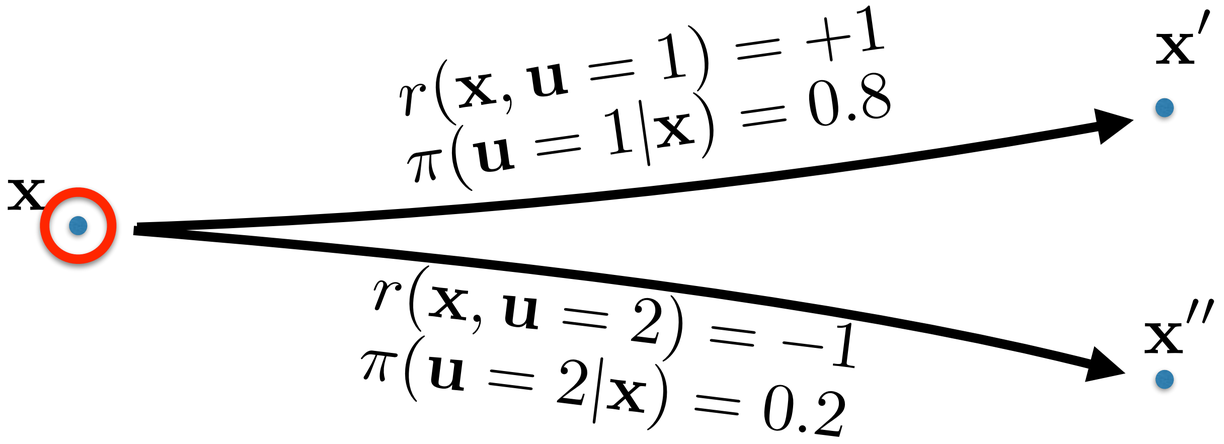


Stochastic policy for discrete control:

$$\pi_\theta(\mathbf{u}|\mathbf{x}) \sim p(\mathbf{u};\ \mathbf{p})$$

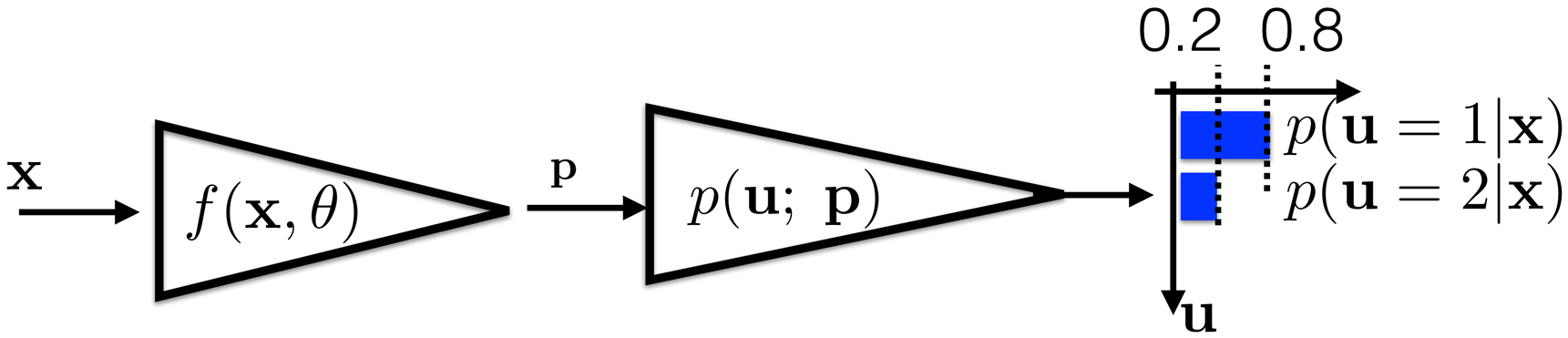
```
from torch.distributions.normal import Categorical
distribution = Categorical(mu, sigma)
actions = distribution.sample()
```

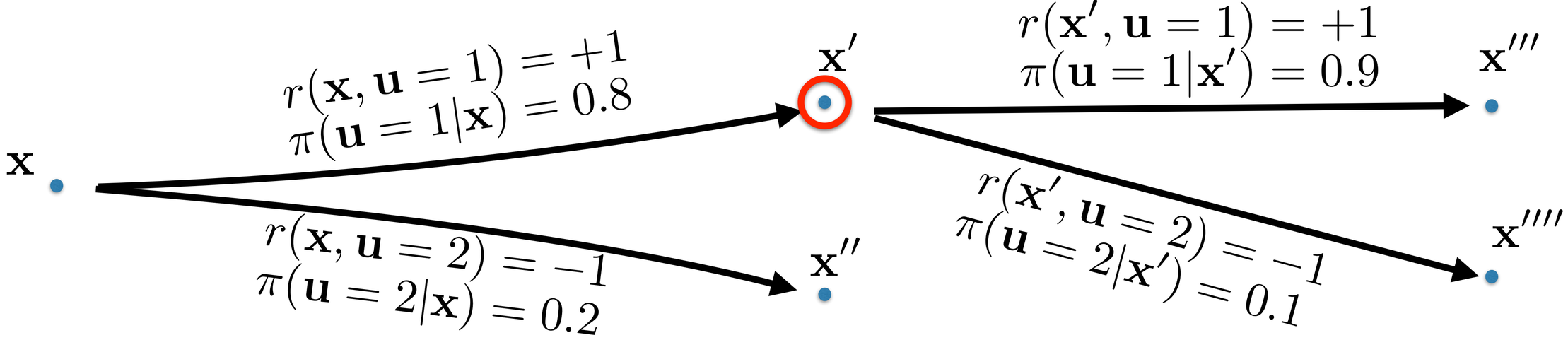




Stochastic policy for discrete control:

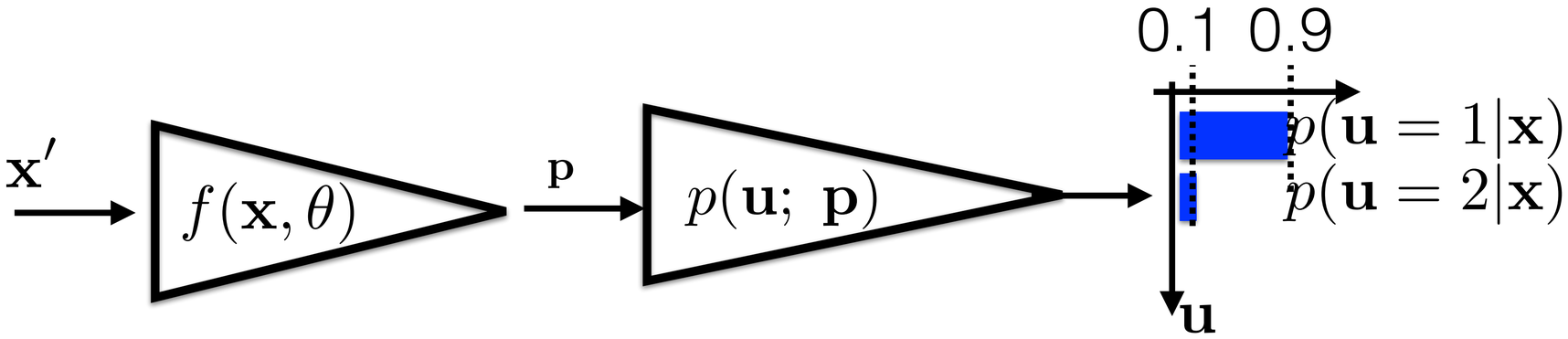
$$\pi_\theta(\mathbf{u}|\mathbf{x}) \sim p(\mathbf{u};\ \mathbf{p})$$

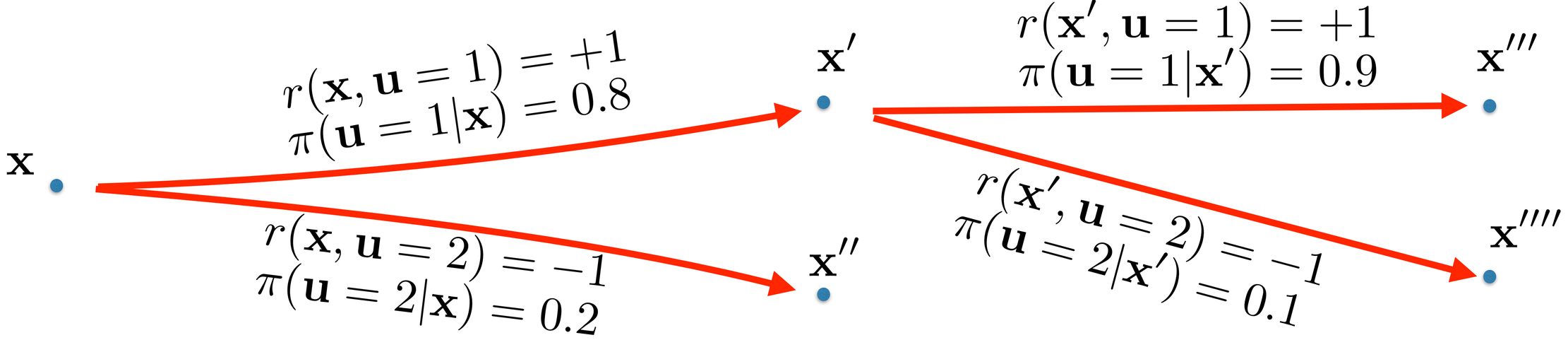




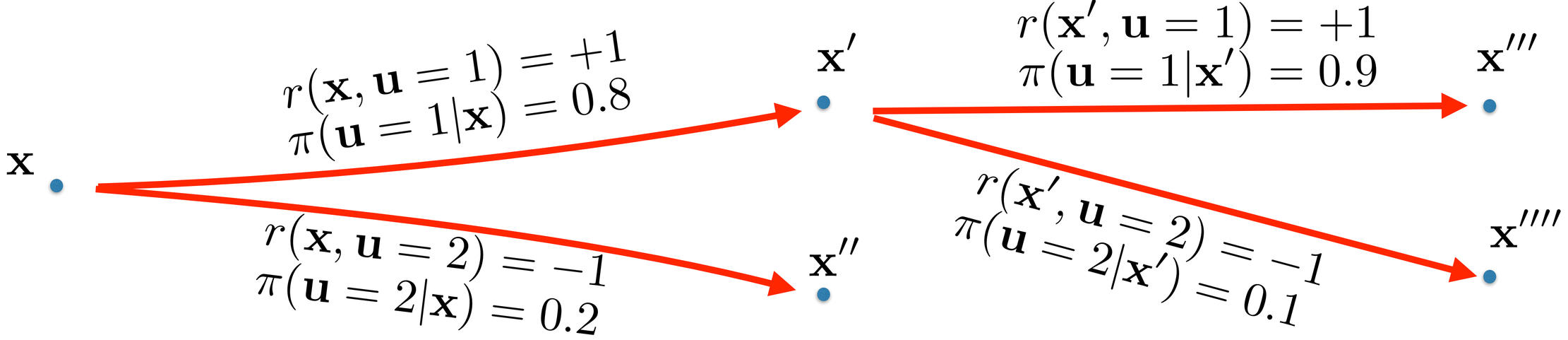
Stochastic policy for discrete control:

$$\pi_\theta(\mathbf{u}|\mathbf{x}) \sim p(\mathbf{u};\ \mathbf{p})$$

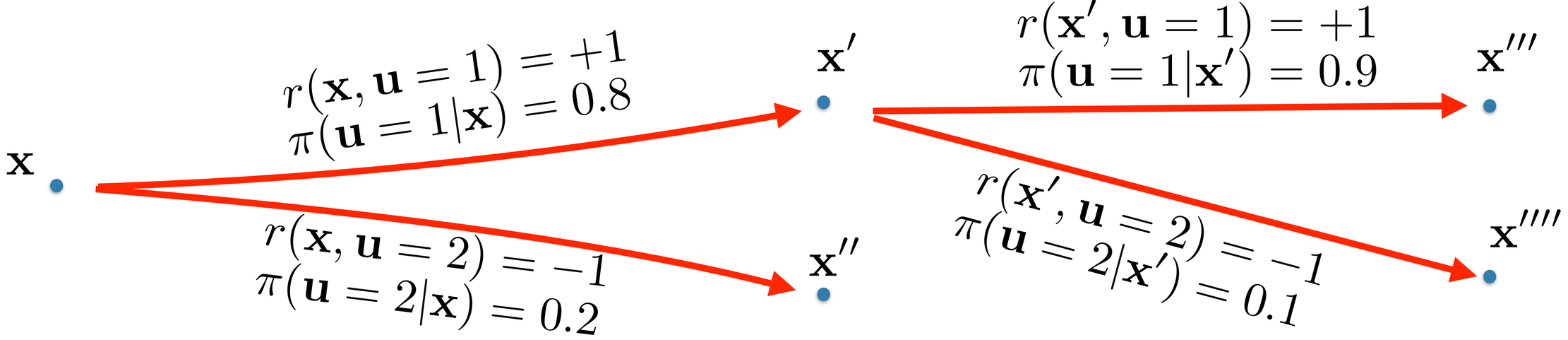




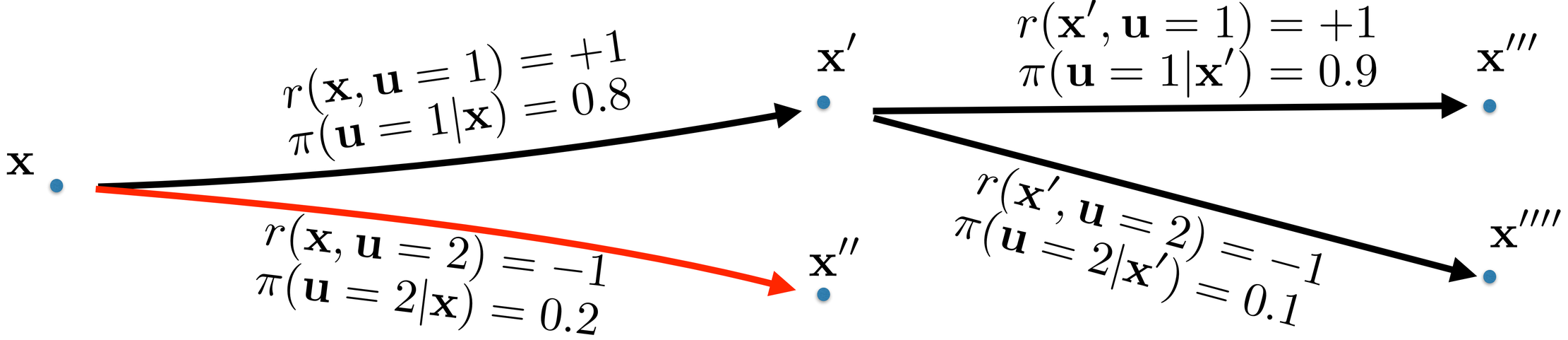
$$V^{\pi}(\mathbf{x}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}} [\sum_{t=0}^{\infty} \gamma^t r_{t+1}]$$



$$V^{\pi}(\mathbf{x}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}} [\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}} [r(\tau)]$$

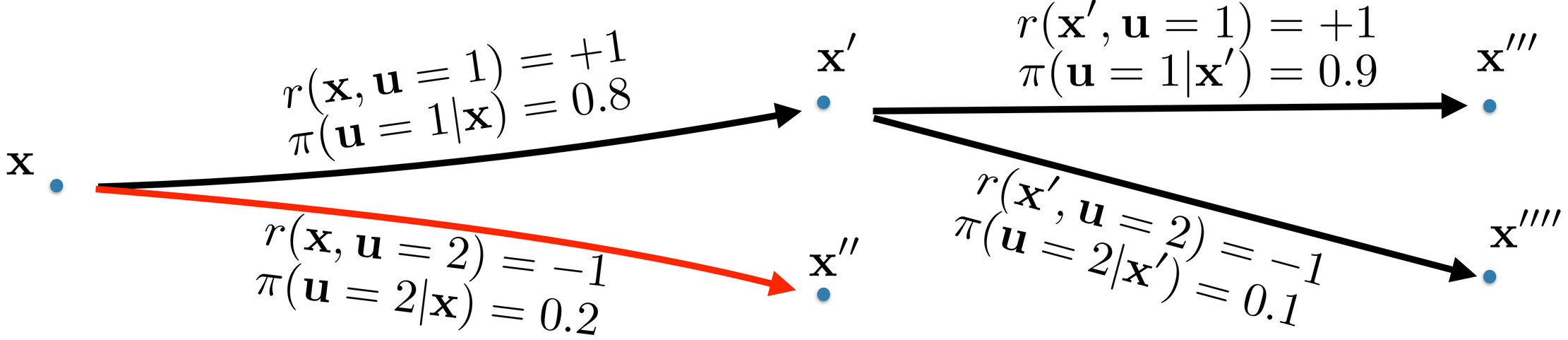


$$V^{\pi}(\mathbf{x}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[r(\tau)] = \int_{\tau : \mathbf{x}_0 = \mathbf{x}} p(\tau|\pi) r(\tau)$$



$$V^{\pi}(\mathbf{x}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[r(\tau)] = \int_{\tau : \mathbf{x}_0 = \mathbf{x}} p(\tau|\pi) r(\tau)$$

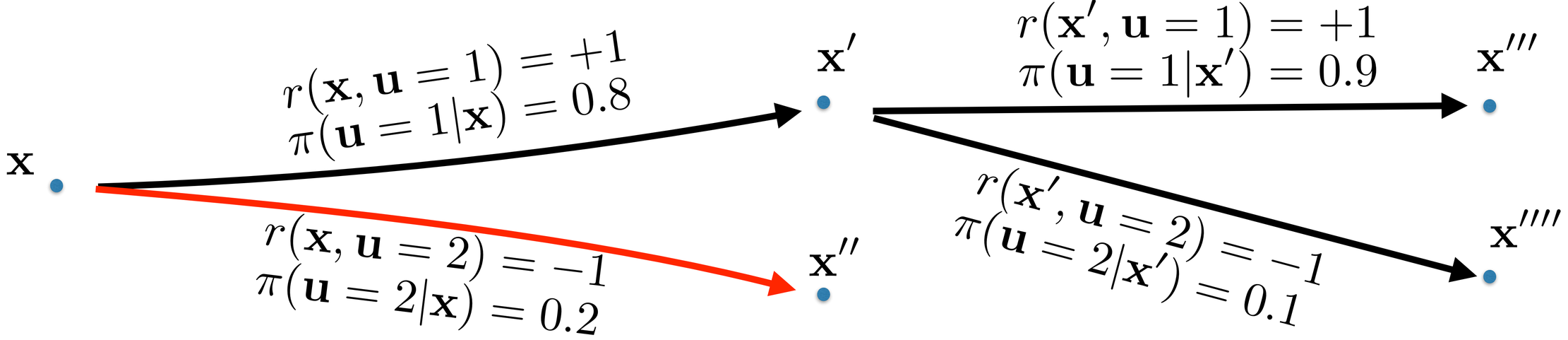
$$Q^{\pi}(\mathbf{x}, \mathbf{u}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[r(\tau)] = \int_{\substack{\tau : \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}} p(\tau|\pi) r(\tau)$$



$$V^\pi(\mathbf{x}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[r(\tau)] = \int_{\tau : \mathbf{x}_0 = \mathbf{x}} p(\tau|\pi) r(\tau)$$

$$Q^\pi(\mathbf{x}, \mathbf{u}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[r(\tau)] = \int_{\substack{\tau : \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}} p(\tau|\pi) r(\tau)$$

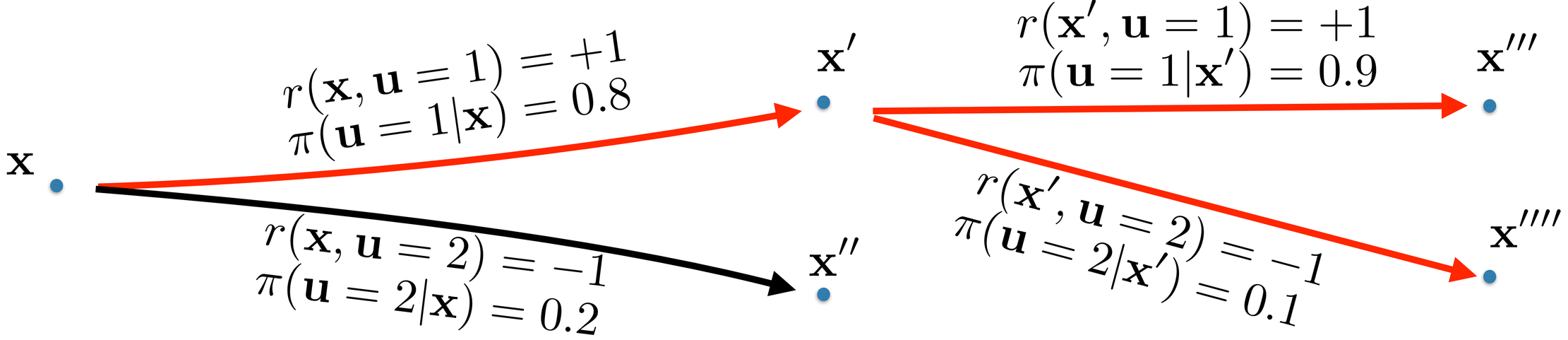
$$Q^\pi(\mathbf{x}, \mathbf{u} = 2) = \text{???}$$



$$V^\pi(\mathbf{x}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[r(\tau)] = \int_{\tau : \mathbf{x}_0 = \mathbf{x}} p(\tau|\pi) r(\tau)$$

$$Q^\pi(\mathbf{x}, \mathbf{u}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[r(\tau)] = \int_{\substack{\tau : \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}} p(\tau|\pi) r(\tau)$$

$$Q^\pi(\mathbf{x}, \mathbf{u} = 2) = -1$$

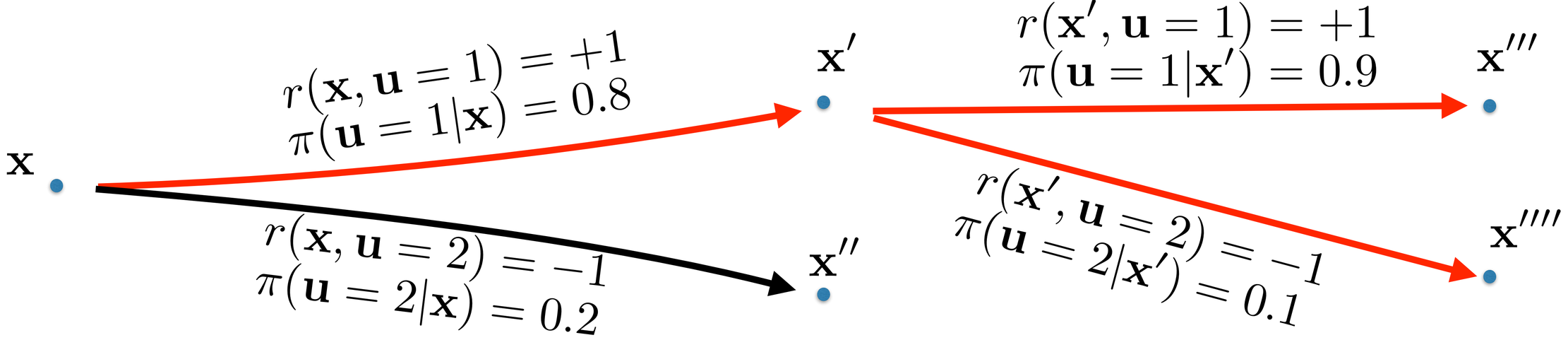


$$V^{\pi}(\mathbf{x}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[r(\tau)] = \int_{\tau : \mathbf{x}_0 = \mathbf{x}} p(\tau|\pi) r(\tau)$$

$$Q^{\pi}(\mathbf{x}, \mathbf{u}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[r(\tau)] = \int_{\substack{\tau : \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}} p(\tau|\pi) r(\tau)$$

$$Q^{\pi}(\mathbf{x}, \mathbf{u} = 2) = -1$$

$$Q^{\pi}(\mathbf{x}, \mathbf{u} = 1) = 1 + 0.9 * 1 + 0.1 * (-1) = \textbf{???}$$

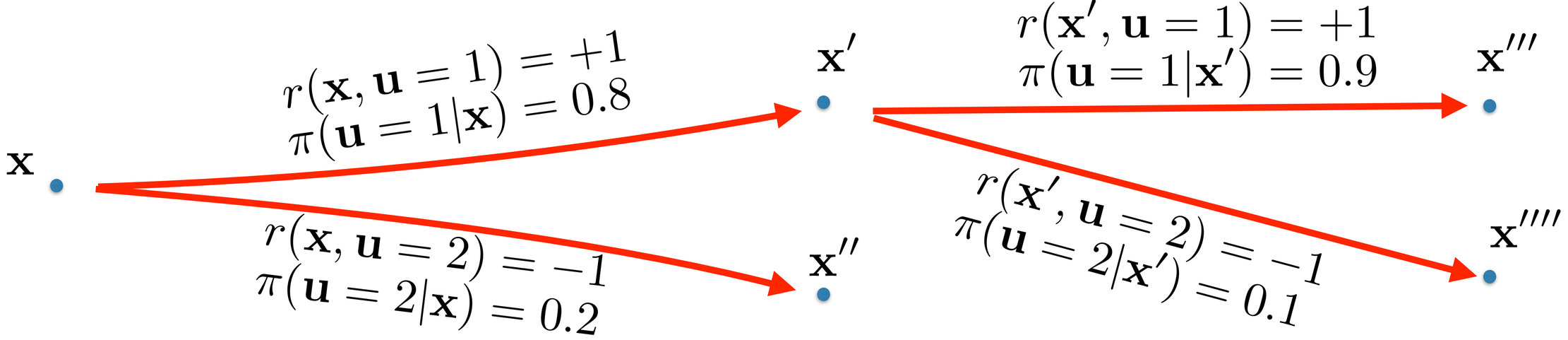


$$V^{\pi}(\mathbf{x}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[r(\tau)] = \int_{\tau : \mathbf{x}_0 = \mathbf{x}} p(\tau|\pi) r(\tau)$$

$$Q^{\pi}(\mathbf{x}, \mathbf{u}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[r(\tau)] = \int_{\substack{\tau : \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}} p(\tau|\pi) r(\tau)$$

$$Q^{\pi}(\mathbf{x}, \mathbf{u} = 2) = -1$$

$$Q^{\pi}(\mathbf{x}, \mathbf{u} = 1) = 1 + 0.9 * 1 + 0.1 * (-1) = 1.8$$



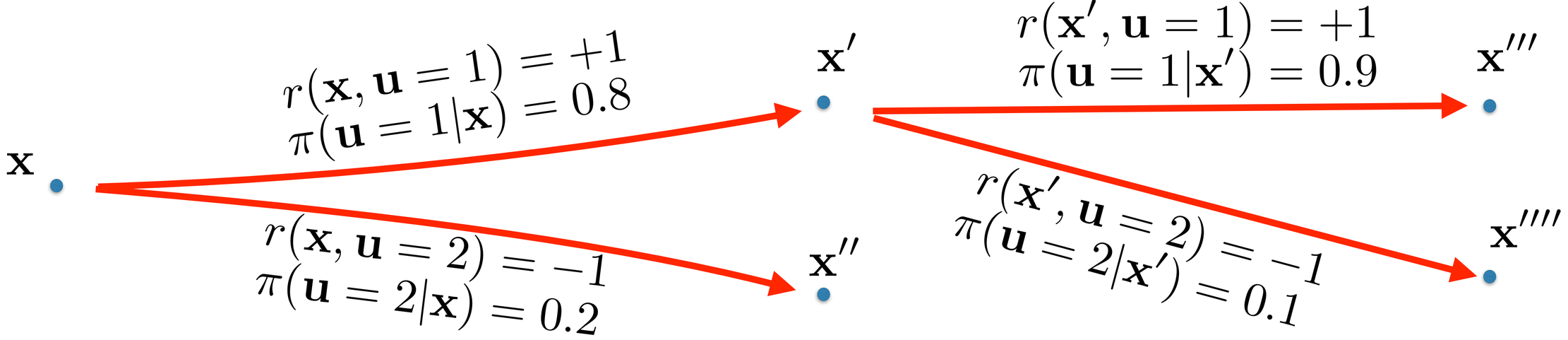
$$V^\pi(\mathbf{x}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[r(\tau)] = \int_{\tau : \mathbf{x}_0 = \mathbf{x}} p(\tau|\pi) r(\tau)$$

$$Q^\pi(\mathbf{x}, \mathbf{u}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[r(\tau)] = \int_{\substack{\tau : \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}} p(\tau|\pi) r(\tau)$$

$$Q^\pi(\mathbf{x}, \mathbf{u} = 2) = -1$$

$$Q^\pi(\mathbf{x}, \mathbf{u} = 1) = 1 + 0.9 * 1 + 0.1 * (-1) = 1.8$$

$$V^\pi(\mathbf{x}) = 0.8 * (1 + 0.9 * 1 + 0.1 * (-1)) + 0.2 * (-1) = \textbf{???}$$



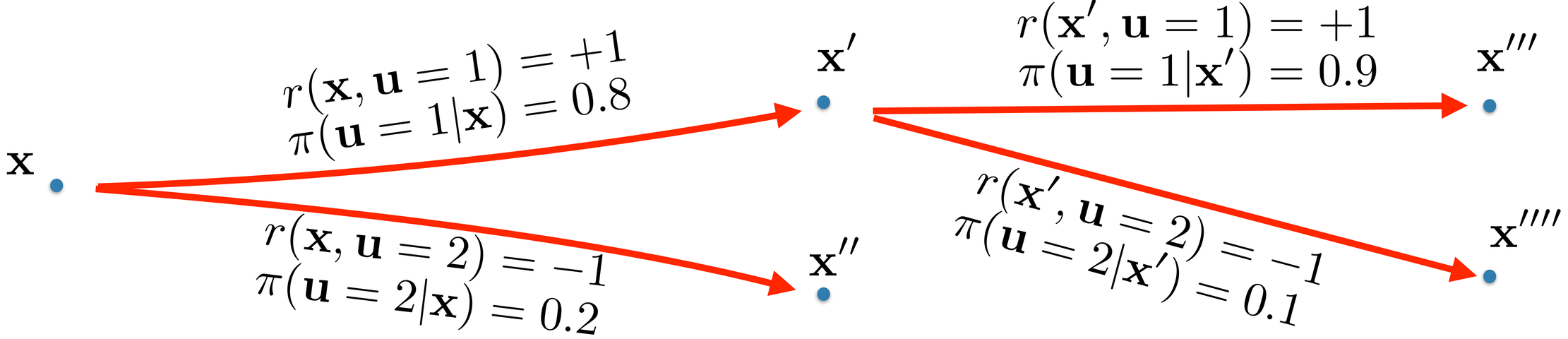
$$V^{\pi}(\mathbf{x}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[r(\tau)] = \int_{\tau : \mathbf{x}_0 = \mathbf{x}} p(\tau|\pi) r(\tau)$$

$$Q^{\pi}(\mathbf{x}, \mathbf{u}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[r(\tau)] = \int_{\substack{\tau : \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}} p(\tau|\pi) r(\tau)$$

$$Q^{\pi}(\mathbf{x}, \mathbf{u} = 2) = -1$$

$$Q^{\pi}(\mathbf{x}, \mathbf{u} = 1) = 1 + 0.9 * 1 + 0.1 * (-1) = 1.8$$

$$V^{\pi}(\mathbf{x}) = 0.8 * (1 + 0.9 * 1 + 0.1 * (-1)) + 0.2 * (-1) = 1.24$$



$$V^\pi(\mathbf{x}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x}}}[r(\tau)] = \int_{\tau : \mathbf{x}_0 = \mathbf{x}} p(\tau|\pi) r(\tau)$$

$$Q^\pi(\mathbf{x}, \mathbf{u}) = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[\sum_{t=0}^{\infty} \gamma^t r_{t+1}] = \mathop{\mathbb{E}}_{\substack{\tau \sim \pi \\ \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}}[r(\tau)] = \int_{\substack{\tau : \mathbf{x}_0 = \mathbf{x} \\ \mathbf{u}_0 = \mathbf{u}}} p(\tau|\pi) r(\tau)$$

$$Q^\pi(\mathbf{x}, \mathbf{u} = 2) = -1$$

$$Q^\pi(\mathbf{x}, \mathbf{u} = 1) = 1 + 0.9 * 1 + 0.1 * (-1) = 1.8$$

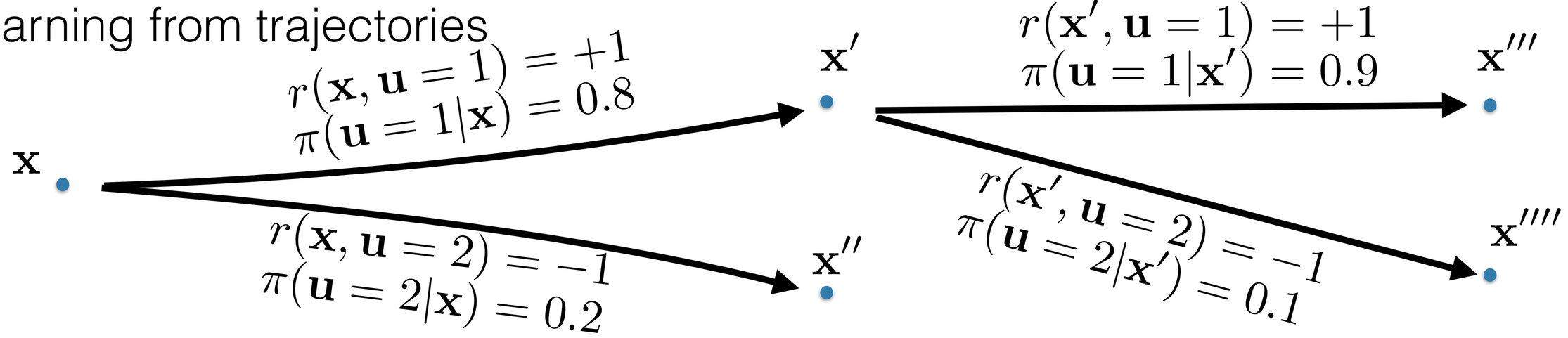
$$V^\pi(\mathbf{x}) = 0.8 * (1 + 0.9 * 1 + 0.1 * (-1)) + 0.2 * (-1) = 1.24$$

$$A^\pi(\mathbf{x}, \mathbf{u} = 1) = Q^\pi(\mathbf{x}, \mathbf{u} = 1) - V^\pi(\mathbf{x}) = 1.8 - 1.24 = 0.56$$

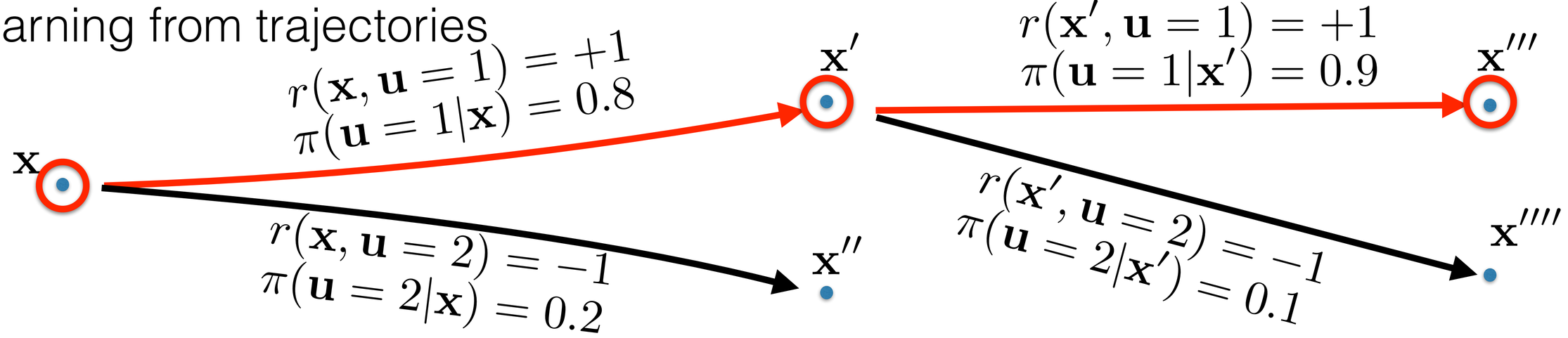
$$A^\pi(\mathbf{x}, \mathbf{u} = 2) = Q^\pi(\mathbf{x}, \mathbf{u} = 2) - V^\pi(\mathbf{x}) = -1 - 1.24 = -2.24$$

Can you interpret it?

Learning from trajectories



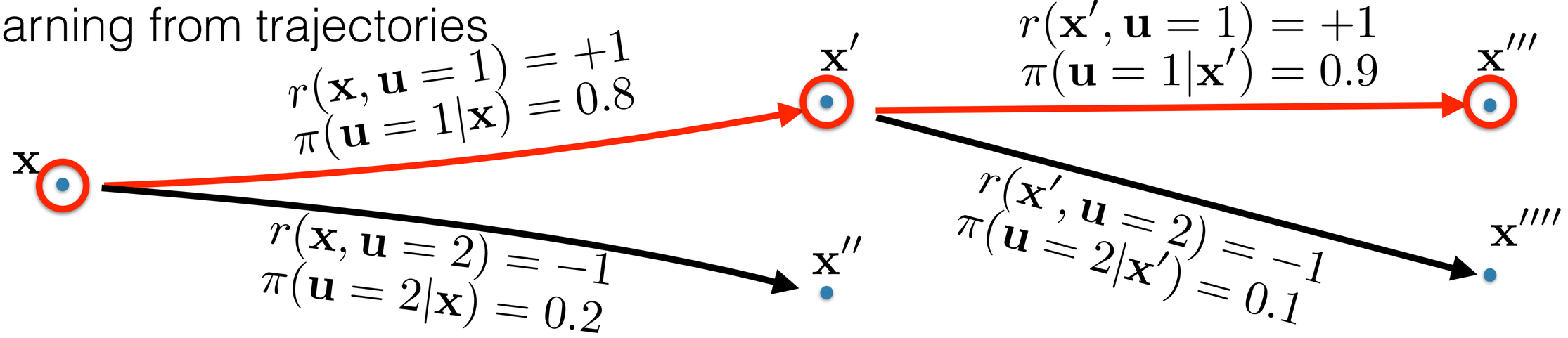
Learning from trajectories



$$\tau_1 : \quad \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_1 = 1, r_1 = +1, \quad \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_1 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}'',$$

$$Q^{\pi}(\mathbf{x}, \mathbf{u} = 1) \approx \text{???}$$

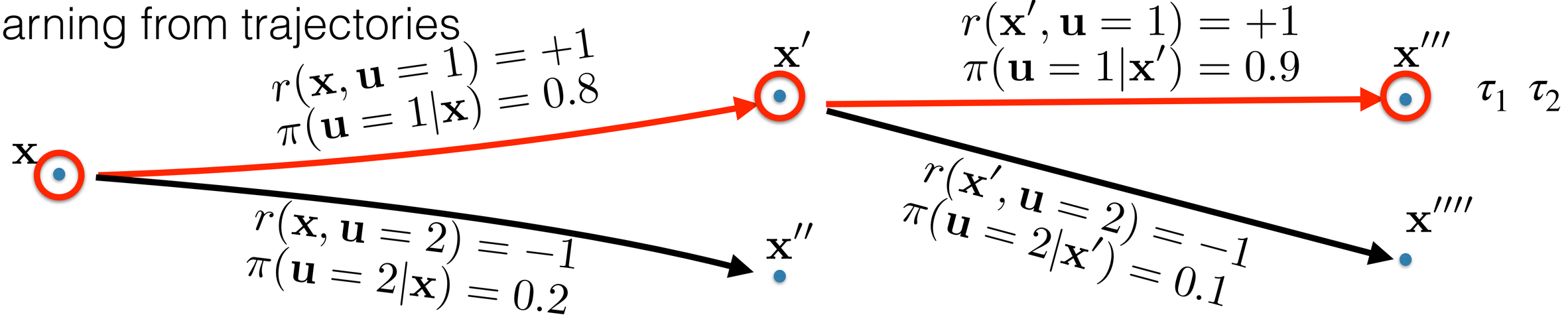
Learning from trajectories



$$\tau_1 : \quad \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_1 = 1, r_1 = +1, \quad \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_2 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}''',$$

$$Q^{\pi}(\mathbf{x}, \mathbf{u} = 1) \approx r_1 + \gamma r_2$$

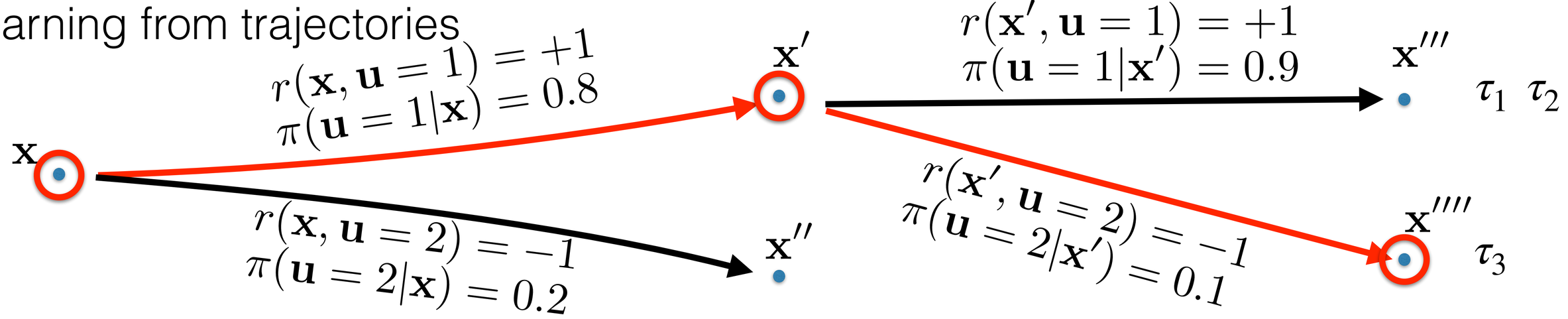
Learning from trajectories



$$\tau_1: \quad \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_1 = 1, r_1 = +1, \quad \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_2 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}''',$$

$$\tau_2: \quad \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_1 = 1, r_1 = +1, \quad \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_2 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}''',$$

Learning from trajectories



$$\tau_1: \quad \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_1 = 1, r_1 = +1, \quad \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_2 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}''',$$

$$\tau_2: \quad \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_1 = 1, r_1 = +1, \quad \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_2 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}''',$$

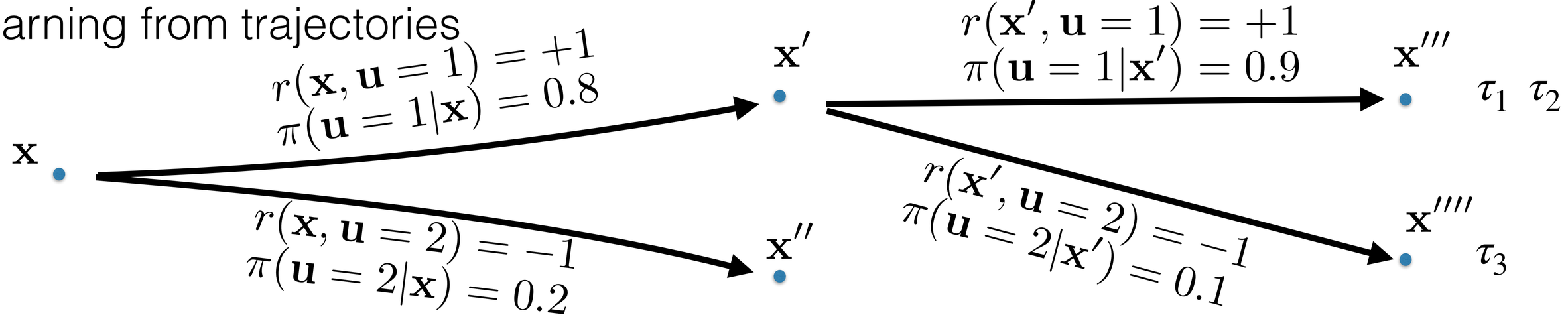
$$\tau_3: \quad \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_1 = 1, r_1 = +1, \quad \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_2 = 2, r_2 = -1, \mathbf{x}_2 = \mathbf{x}'''',$$

$$Q^\pi(\mathbf{x}, \mathbf{u} = 1) \approx \frac{1}{N} \sum_\tau r_1 + \gamma r_2 = \frac{1}{3}(1+1 + 1+1 + 1+(-1)) = 1.33 \quad \text{vs} \quad Q^\pi(\mathbf{x}, \mathbf{u} = 1) = 1.8$$

If you are given true value function $V^\pi(\mathbf{x}') = 0.8$ then

$$Q^\pi(\mathbf{x}, \mathbf{u} = 1) \approx \frac{1}{N} \sum_\tau r_1 + \gamma V^\pi(\mathbf{x}') = \frac{1}{3}(1+0.8 + 1+0.8 + 1+0.8) = 1.8$$

Learning from trajectories



There are many possible approximations of $Q^\pi(\mathbf{x}, \mathbf{u})$ when $V^\pi(\mathbf{x})$ is available:

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \dots$$

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 V^\pi(\mathbf{x}_4)$$

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 V^\pi(\mathbf{x}_3)$$

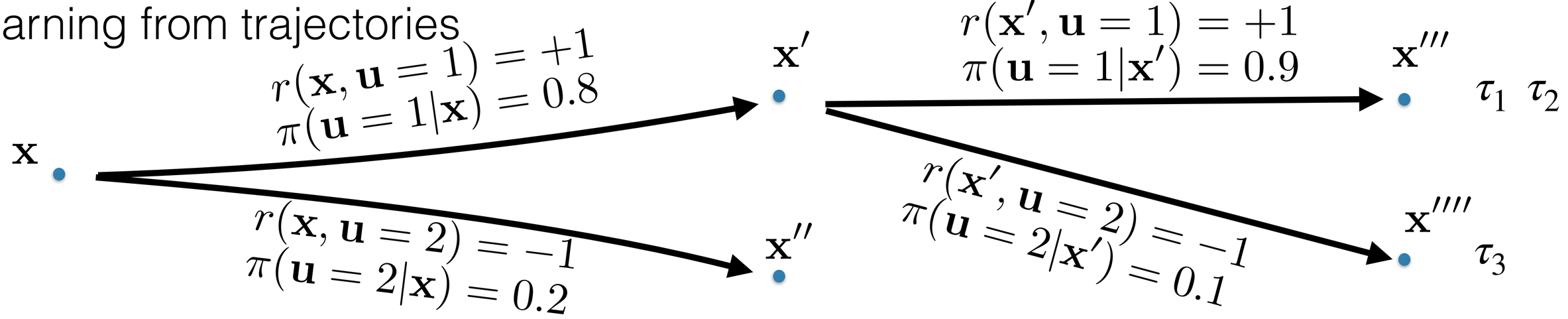
$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma V^\pi(\mathbf{x}_2)$$

Which I should use?

TD(0) estimate (if reliable $V^\pi(\mathbf{x})$ available):

- small variance,
- strong bias

Learning from trajectories



There are many possible approximations of $Q^\pi(\mathbf{x}, \mathbf{u})$ when $V^\pi(\mathbf{x})$ is available:

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \dots$$

TD(1) estimate (if no reliable $V^\pi(\mathbf{x})$ available):

- high variance,
- no bias

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 V^\pi(\mathbf{x}_4)$$

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 V^\pi(\mathbf{x}_3)$$

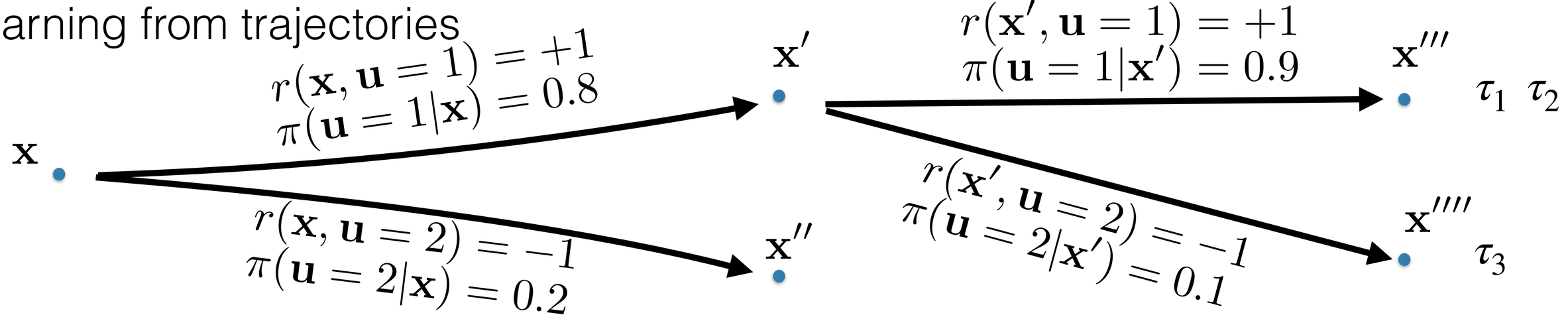
$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma V^\pi(\mathbf{x}_2)$$

TD(0) estimate (if reliable $V^\pi(\mathbf{x})$ available):

- small variance,
- strong bias

Which I should use?

Learning from trajectories



There are many possible approximations of $Q^\pi(\mathbf{x}, \mathbf{u})$ when $V^\pi(\mathbf{x})$ is available:

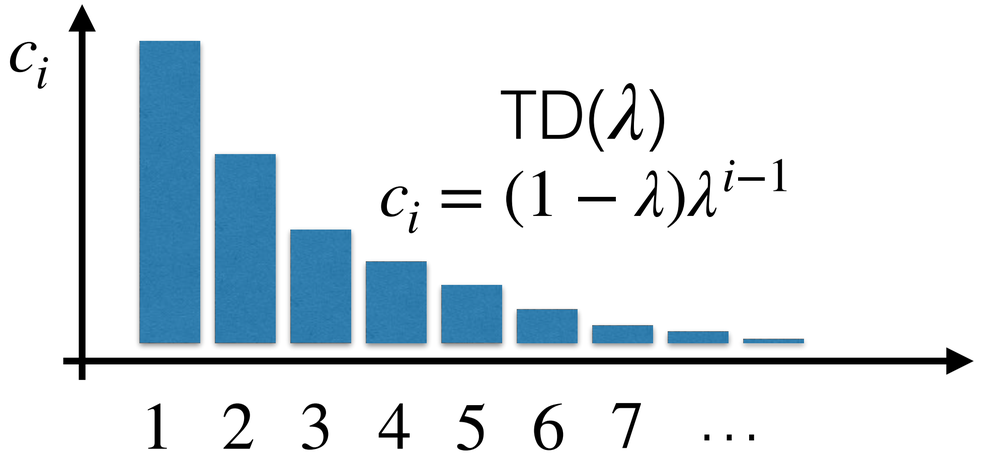
$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \ldots = G^{(4)}$$

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 V^\pi(\mathbf{x}_4) = G^{(3)}$$

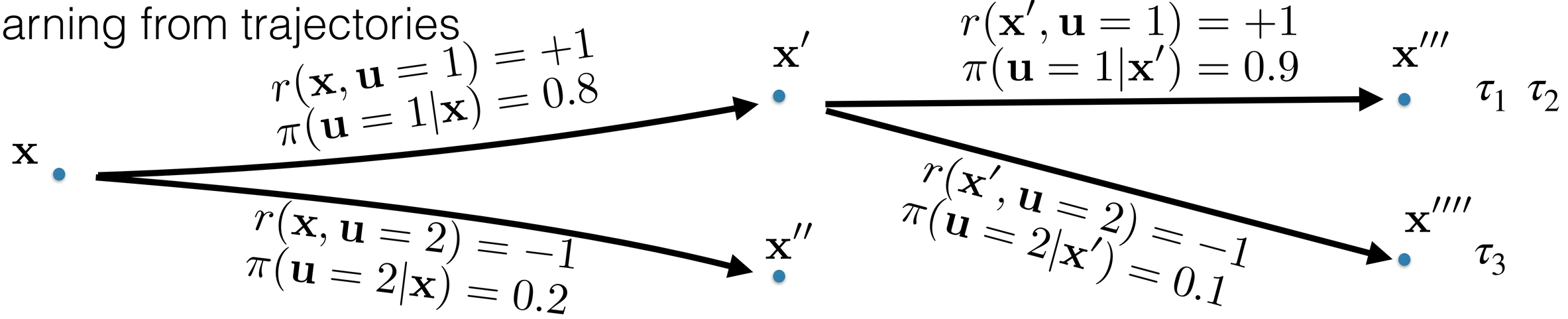
$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 V^\pi(\mathbf{x}_3) = G^{(2)}$$

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma V^\pi(\mathbf{x}_2) = G^{(1)}$$

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx c_1 G^{(1)} + c_2 G^{(2)} + c_3 G^{(3)} + c_4 G^{(4)} + \ldots$$



Learning from trajectories



There are many possible approximations of $Q^\pi(\mathbf{x}, \mathbf{u})$ when $V^\pi(\mathbf{x})$ is available:

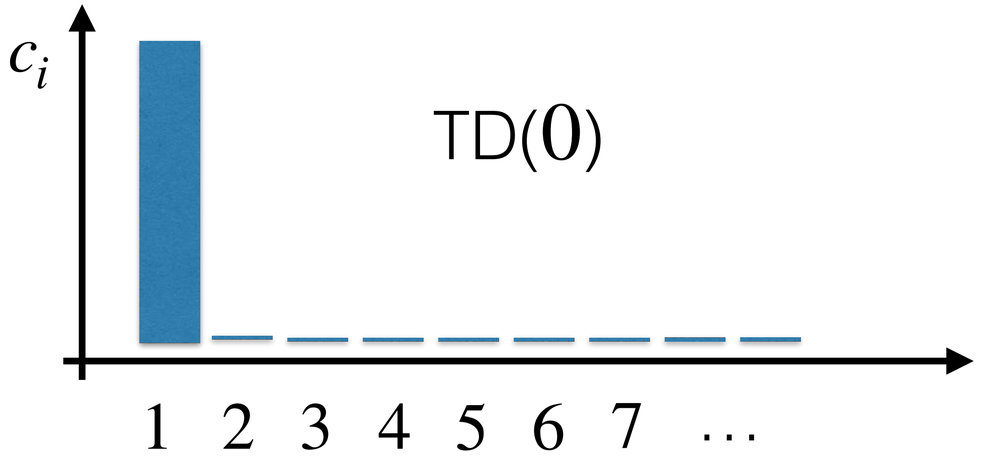
$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \ldots = G^{(4)}$$

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 V^\pi(\mathbf{x}_4) = G^{(3)}$$

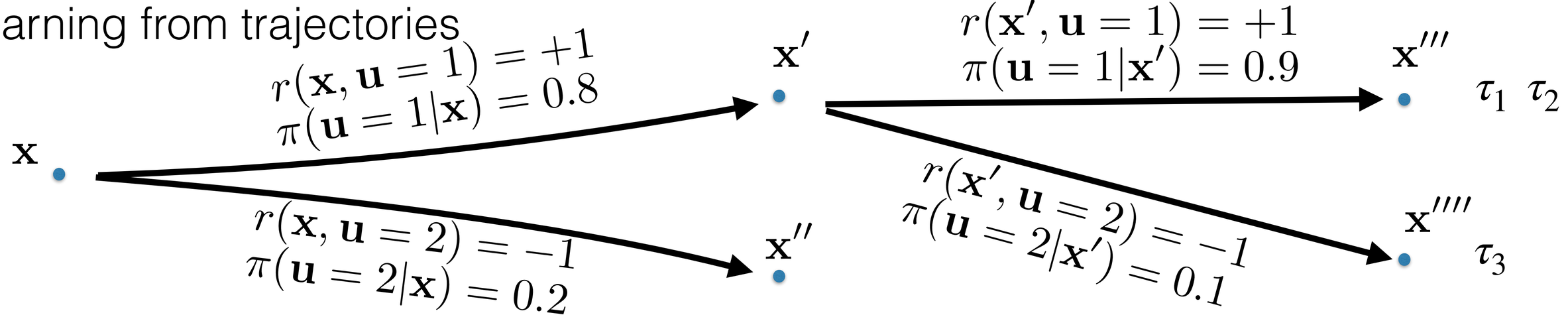
$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 V^\pi(\mathbf{x}_3) = G^{(2)}$$

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma V^\pi(\mathbf{x}_2) = G^{(1)}$$

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx c_1 G^{(1)} + c_2 G^{(2)} + c_3 G^{(3)} + c_4 G^{(4)} + \ldots$$



Learning from trajectories



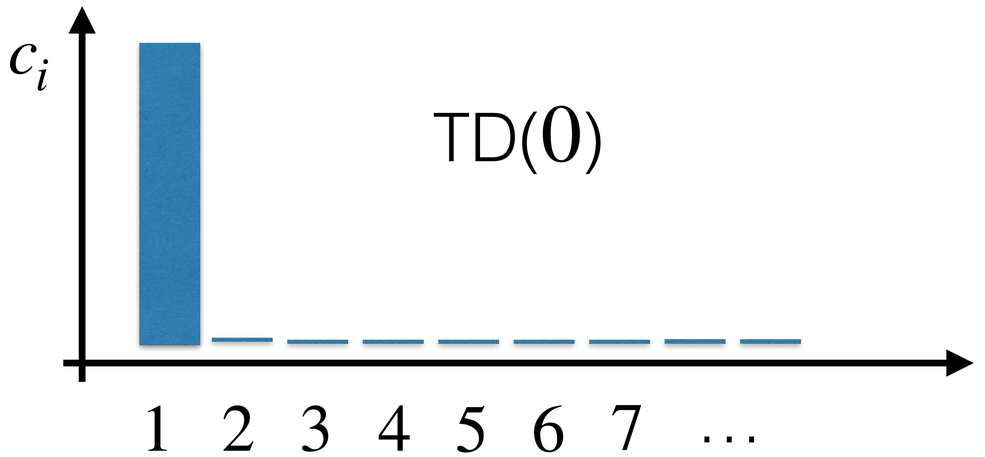
There are many possible approximations of $Q^\pi(\mathbf{x}, \mathbf{u})$ when $V^\pi(\mathbf{x})$ is available:

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \ldots \quad = G^{(4)}$$

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 V^\pi(\mathbf{x}_4) \quad = G^{(3)}$$

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma r_2 + \gamma^2 V^\pi(\mathbf{x}_3) \quad = G^{(2)}$$

$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx r_1 + \gamma V^\pi(\mathbf{x}_2) \quad = G^{(1)}$$



$$Q^\pi(\mathbf{x}_1, \mathbf{u}_1) \approx c_1 G^{(1)} + c_2 G^{(2)} + c_3 G^{(3)} + c_4 G^{(4)} + \ldots$$

$$A^\pi(\mathbf{x}_1, \mathbf{u}_1) = Q^\pi(\mathbf{x}_1, \mathbf{u}_1) - V^\pi(\mathbf{x}_1) \approx G^{(1)} - V^\pi(\mathbf{x}_1) = r_1 + \gamma V^\pi(\mathbf{x}_2) - V^\pi(\mathbf{x}_1)$$

This determines $Q^\pi(\mathbf{x}, \mathbf{u})$, $V^\pi(\mathbf{x})$, $A^\pi(\mathbf{x}, \mathbf{u})$ for a given policy

How do I find $\pi^* = \arg\max_\pi J_\pi$?

How do I find $\pi^* = \arg\max_{\pi} J_\pi$?

There are two ways:

Straightforward (primal) way:

- Gradient optimization in parameter space of the criterion

$$\theta^* = \arg\max_{\theta} J_{\pi_\theta}$$

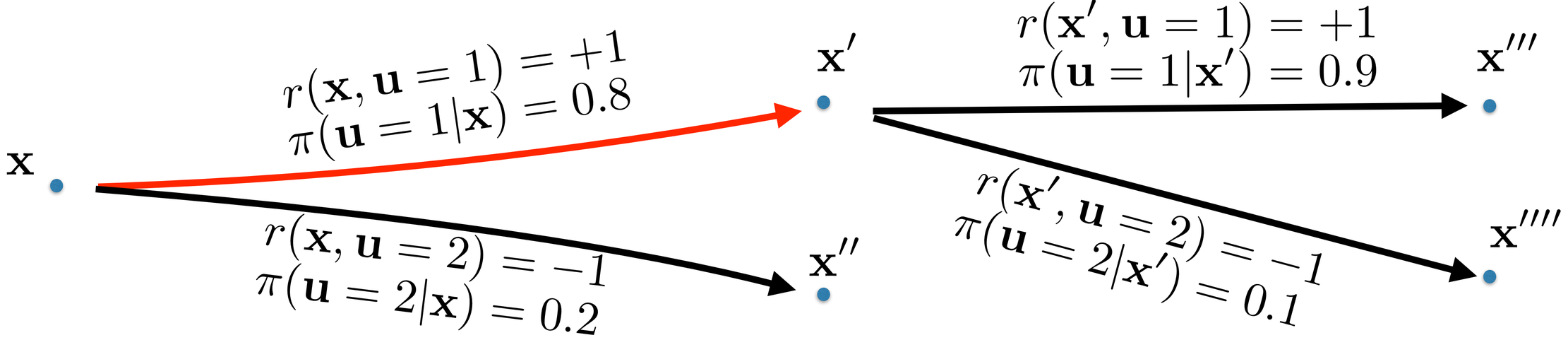
Dual way:

- Search for Q satisfying Bellman equation (for every transition):

  $Q^\pi(\mathbf{x}, \mathbf{u}) = r(\mathbf{x}, \mathbf{u}) + \gamma \max_{\mathbf{u}'} Q^\pi(\mathbf{x}', \mathbf{u}')$

- Once we find it, the optimal policy is:

  $\pi^*(\mathbf{x}) = \arg\max_{\mathbf{u}} Q^\pi(\mathbf{x}, \mathbf{u}) = \arg\max_{\pi} J_\pi$

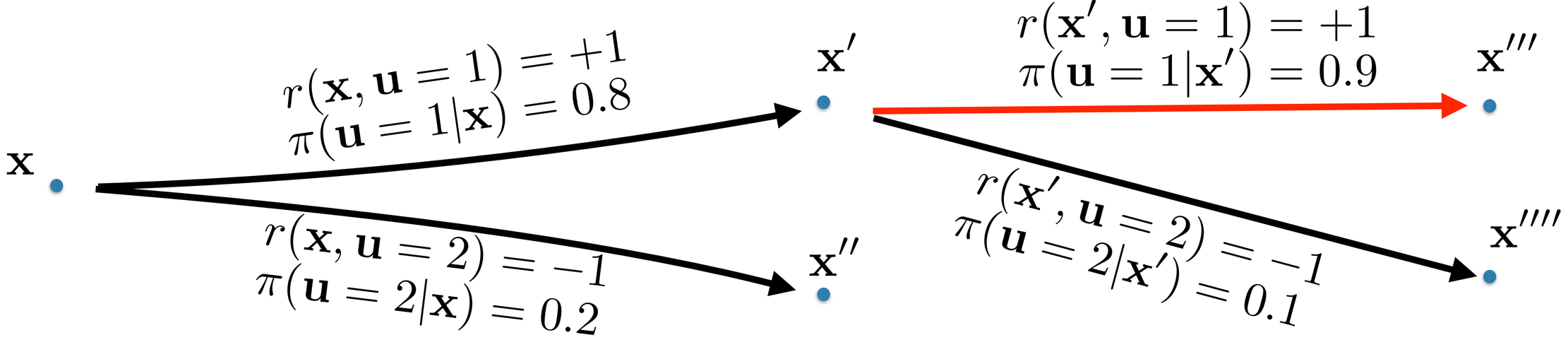


$$\tau_1 : \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1, r_1 = +1, \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_1 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}'',$$

state $\mathbf{x}_1$ action $\mathbf{u}_1$ reward r_1 next state $\mathbf{x}_2$

$$Q(\mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}) \quad = +1$$

Q	u=1	u=2
x	0	0
x’	0	0
x’’	0	0
x’’’	0	0
x’’’’	0	0



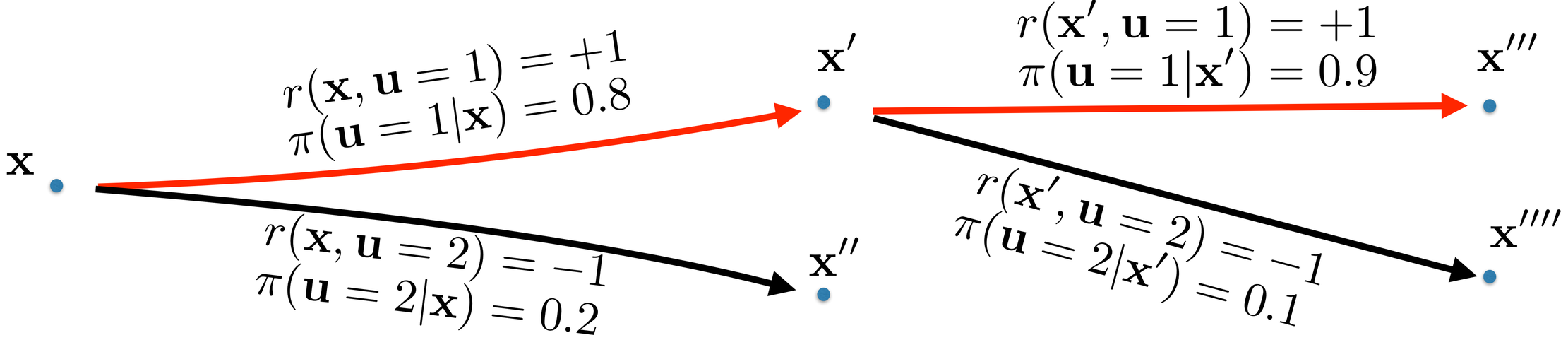
$$\tau_1 : \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1, r_1 = +1, \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_1 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}'',$$

state $\mathbf{x}_1$ | action $\mathbf{u}_1$ | reward r_1 | next state $\mathbf{x}_2$

$$Q(\mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}) \quad = +1$$

$$Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_2 = \mathbf{x}'', \mathbf{u}) \quad = +1$$

Q	u=1	u=2
x	0	0
x’	0	0
x’’	0	0
x’’’	0	0
x’’’’	0	0



$$\tau_1 : \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1, r_1 = +1, \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_1 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}'',$$

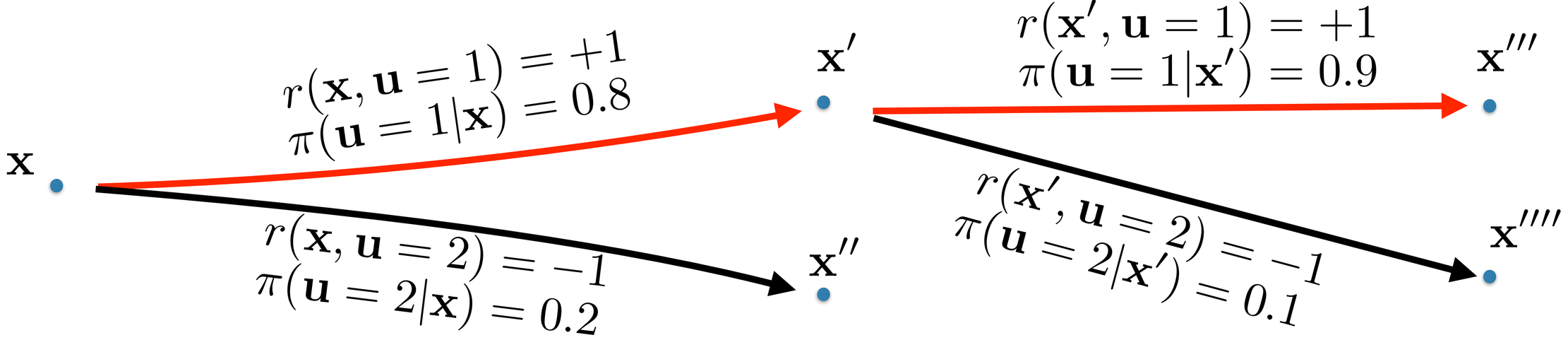
state $\mathbf{x}_1$ action $\mathbf{u}_1$ reward r_1 next state $\mathbf{x}_2$

$$Q(\mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}) \quad = +1$$

$$Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_2 = \mathbf{x}'', \mathbf{u}) \quad = +1$$

Bellman equation is not satisfied

Q	u=1	u=2
x	0	0
x’	0	0
x’’	0	0
x’’’	0	0
x’’’’	0	0



$$\tau_1 : \ \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1, r_1 = +1, \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_1 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}'',$$

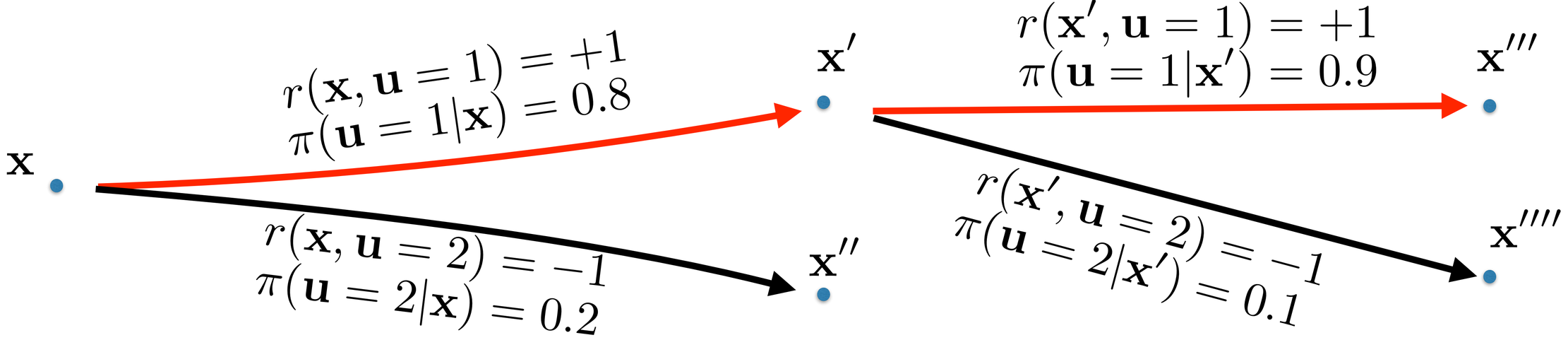
state $\mathbf{x}_1$ | action $\mathbf{u}_1$ | reward r_1 | next state $\mathbf{x}_2$

$$Q(\mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}) \quad = +1$$

$$Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_2 = \mathbf{x}'', \mathbf{u}) \quad = +1$$

Search for solution by successive subst. of RHS to LHS.

Q	u=1	u=2
x	1	0
x’	0	0
x’’	0	0
x’’’	0	0
x’’’’	0	0



$$\tau_1 : \ \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1, r_1 = +1, \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_1 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}'',$$

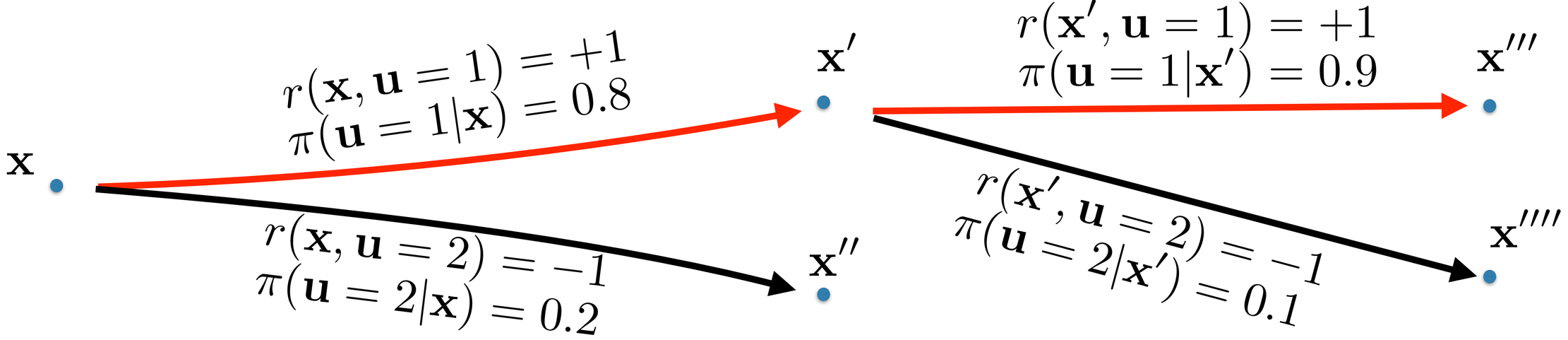
state $\mathbf{x}_1$ — action $\mathbf{u}_1$ — reward r_1 — next state $\mathbf{x}_2$

$$Q(\mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}) \quad = +1$$

$$Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_2 = \mathbf{x}'', \mathbf{u}) \quad = +1$$

Search for solution by successive subst. of RHS to LHS.

Q	u=1	u=2
x	1	0
x’	1	0
x’’	0	0
x’’’	0	0
x’’’’	0	0



$$\tau_1 : \ \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1, r_1 = +1, \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_1 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}'',$$

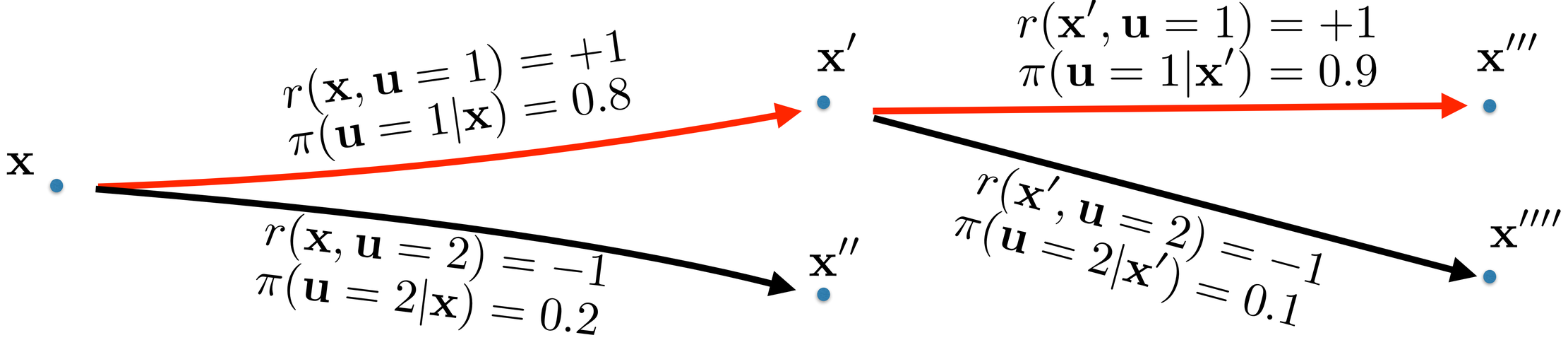
state $\mathbf{x}_1$ action $\mathbf{u}_1$ reward r_1 next state $\mathbf{x}_2$

$$Q(\mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}) \quad = +1.9$$

$$Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_2 = \mathbf{x}'', \mathbf{u}) \quad = +1$$

Substitute of RHS to LHS.

Q	u=1	u=2
x	1.9	0
x’	1	0
x’’	0	0
x’’’	0	0
x’’’’	0	0



$$\tau_1 : \ \mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1, r_1 = +1, \mathbf{x}_1 = \mathbf{x}', \mathbf{u}_1 = 1, r_2 = +1, \mathbf{x}_2 = \mathbf{x}'',$$

state $\mathbf{x}_1$ action $\mathbf{u}_1$ reward r_1 next state $\mathbf{x}_2$

$$Q(\mathbf{x}_0 = \mathbf{x}, \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}) \quad = +1.9$$

$$Q(\mathbf{x}_1 = \mathbf{x}', \mathbf{u}_0 = 1) = +1 + \gamma \max_{\mathbf{u}} Q(\mathbf{x}_2 = \mathbf{x}'', \mathbf{u}) \quad = +1$$

If Q is table, the mapping is contraction and iterations always converge to a fixed point of Bellman operator.

Q	u=1	u=2
x	1.9	0
x’	1	0
x”	0	0
x”’	0	0
x””	0	0

Q-learning

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

Q-learning

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality

Q-learning

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality
- Replace table $Q(\mathbf{x}, \mathbf{u})$ by function $Q_\theta(\mathbf{x}, \mathbf{u})$

Q-learning

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality
- Replace table $Q(\mathbf{x}, \mathbf{u})$ by function $Q_\theta(\mathbf{x}, \mathbf{u})$

Approximate Q-learning (DQN)

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Estimate $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_\theta(\mathbf{x}', \mathbf{u}')$
3. Update parameters by learning

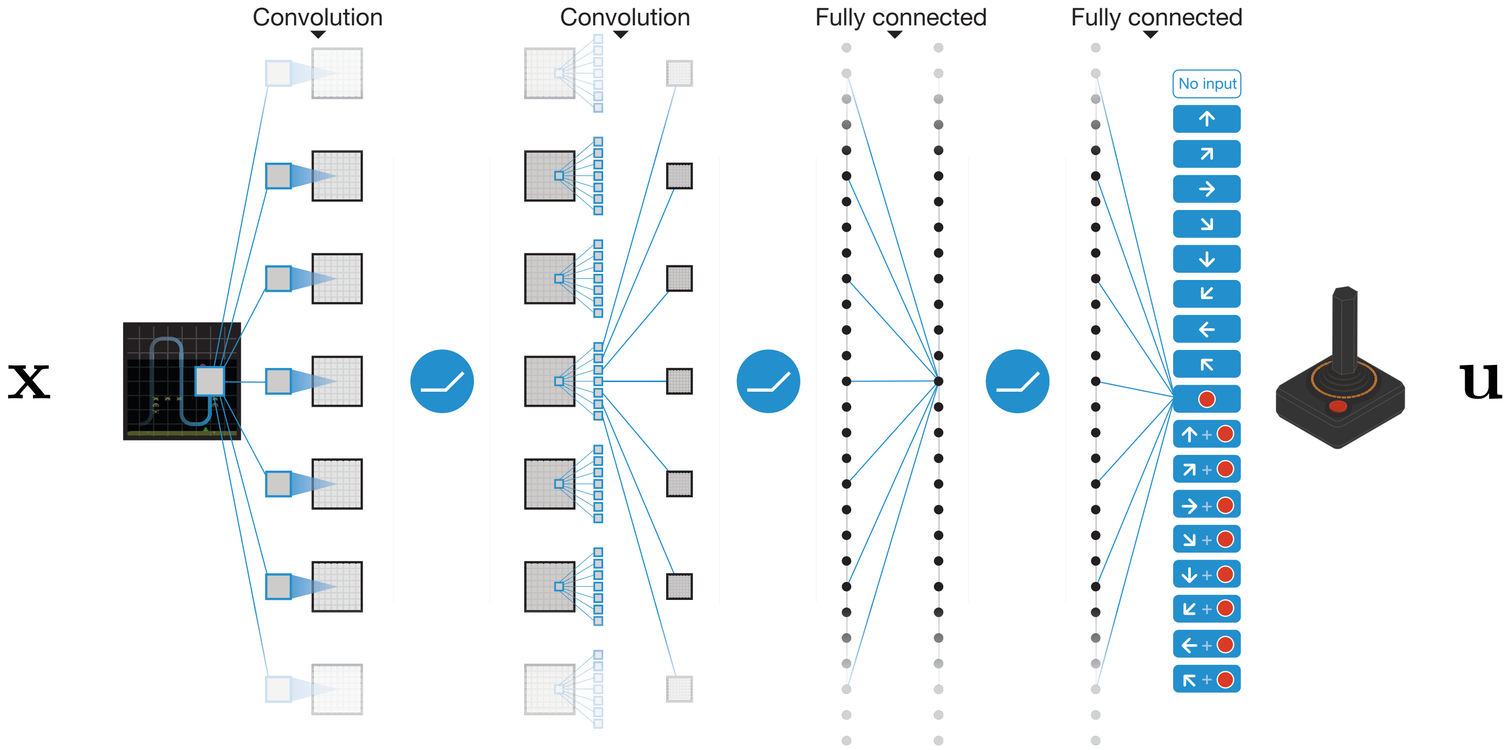
$$\arg\min_\theta \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_\theta(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$$

4. Repeat from 1

Mnih et al. Nature 2015

- 2600 atari games
- **state space** $\mathbf{x}$**:** last four frames to capture dynamics
  (e.g. RGB images in VGA resolution)
- **action space** $\mathbf{u}$**:** 18 discrete joystic actions
  (8 direction + 8 direction with button + neutral action + neutral with button)

$$Q_\theta(\mathbf{x}, \mathbf{u})$$



Q-learning

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

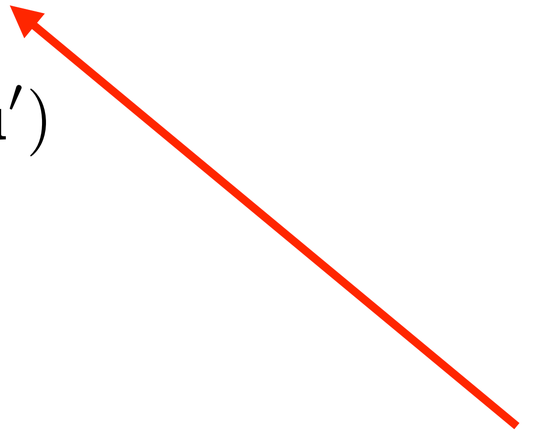
- Curse of dimensionality
- Replace table $Q(\mathbf{x}, \mathbf{u})$ by function $Q_\theta(\mathbf{x}, \mathbf{u})$

Approximate Q-learning (DQN)

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Estimate target $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_\theta(\mathbf{x}', \mathbf{u}')$
3. Update parameters by learning

$$\arg\min_\theta \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_\theta(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$$

4. Repeat from 1



There are 2 wtf issues in this algorithm !
Do you see them?

Q-learning

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality
- Replace table $Q(\mathbf{x}, \mathbf{u})$ by function $Q_\theta(\mathbf{x}, \mathbf{u})$

Approximate Q-learning (DQN)

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Estimate target $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_\theta(\mathbf{x}', \mathbf{u}')$
3. Update parameters by learning (assumes i.i.d+n.n.)

$$\arg\min_\theta \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_\theta(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$$

4. Repeat from 1

★ Transitions are strongly correlated !

Q-learning

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality
- Replace table $Q(\mathbf{x}, \mathbf{u})$ by function $Q_\theta(\mathbf{x}, \mathbf{u})$

Approximate Q-learning (DQN)

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_\theta(\mathbf{x}', \mathbf{u}')$
4. Update parameters by learning (assumes i.i.d+n.n.)

$$\arg\min_\theta \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_\theta(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$$

★ Transitions are strongly correlated !

5. Repeat from 1

★ Transitions are strongly correlated !

Solution: ReplayMemory => minibatch sampled at random
(decorrelates samples to be “more i.i.d”)

```
Transition = namedtuple( 'Transition',
                                ('state', 'action', 'next_state', 'reward'))

class ReplayMemory(object):
    def push(self, *args):
        if len(self.memory) < self.capacity:
            self.memory.append(None)
        self.memory[self.position] = Transition(*args)
        self.position = (self.position + 1) % self.capacity

    def sample(self, batch_size):
        return random.sample(self.memory, batch_size)
```

https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

Q-learning

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality
- Replace table $Q(\mathbf{x}, \mathbf{u})$ by function $Q_\theta(\mathbf{x}, \mathbf{u})$

Approximate Q-learning (DQN)

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_\theta(\mathbf{x}', \mathbf{u}')$
4. Update parameters by learning (assumes i.i.d+n.n.)

$$\arg\min_\theta \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_\theta(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$$

★ Transitions are strongly correlated !

5. Repeat from 1

Q-learning

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality
- Replace table $Q(\mathbf{x}, \mathbf{u})$ by function $Q_\theta(\mathbf{x}, \mathbf{u})$

Approximate Q-learning (DQN)

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_\theta(\mathbf{x}', \mathbf{u}')$
4. Update parameters by learning (assumes i.i.d+n.n.)

$$\arg\min_\theta \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \| Q_\theta(\mathbf{x}, \mathbf{u}) - \mathbf{y} \|$$

5. Repeat from 1

★ **Transitions are strongly correlated !**

★ **Training/Testing distribution mismatch**

Q-learning

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality
- Replace table $Q(\mathbf{x}, \mathbf{u})$ by function $Q_\theta(\mathbf{x}, \mathbf{u})$

Approximate Q-learning (DQN)

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_{\bar{\theta}}(\mathbf{x}', \mathbf{u}')$
4. Update parameters by learning (assumes i.i.d+n.n.)

$$\arg\min_\theta \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \| Q_\theta(\mathbf{x}, \mathbf{u}) - \mathbf{y} \|$$

Target net (slowly upd.)
encourage stability

Policy net (regularly upd.)
encourage exploration

5. Repeat from 1

Q-learning

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$
2. Solve $Q(\mathbf{x}, \mathbf{u}) = r + \gamma \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}')$
3. Repeat from 1

- Curse of dimensionality
- Replace table $Q(\mathbf{x}, \mathbf{u})$ by function $Q_\theta(\mathbf{x}, \mathbf{u})$

Approximate Q-learning (DQN)

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_{\overline{\theta}}(\mathbf{x}', \mathbf{u}')$
4. Update parameters by learning (assumes i.i.d+n.n.)

$$\arg\min_{\theta} \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \| Q_\theta(\mathbf{x}, \mathbf{u}) - \mathbf{y} \|$$

Target net (slowly upd.)
encourage stability

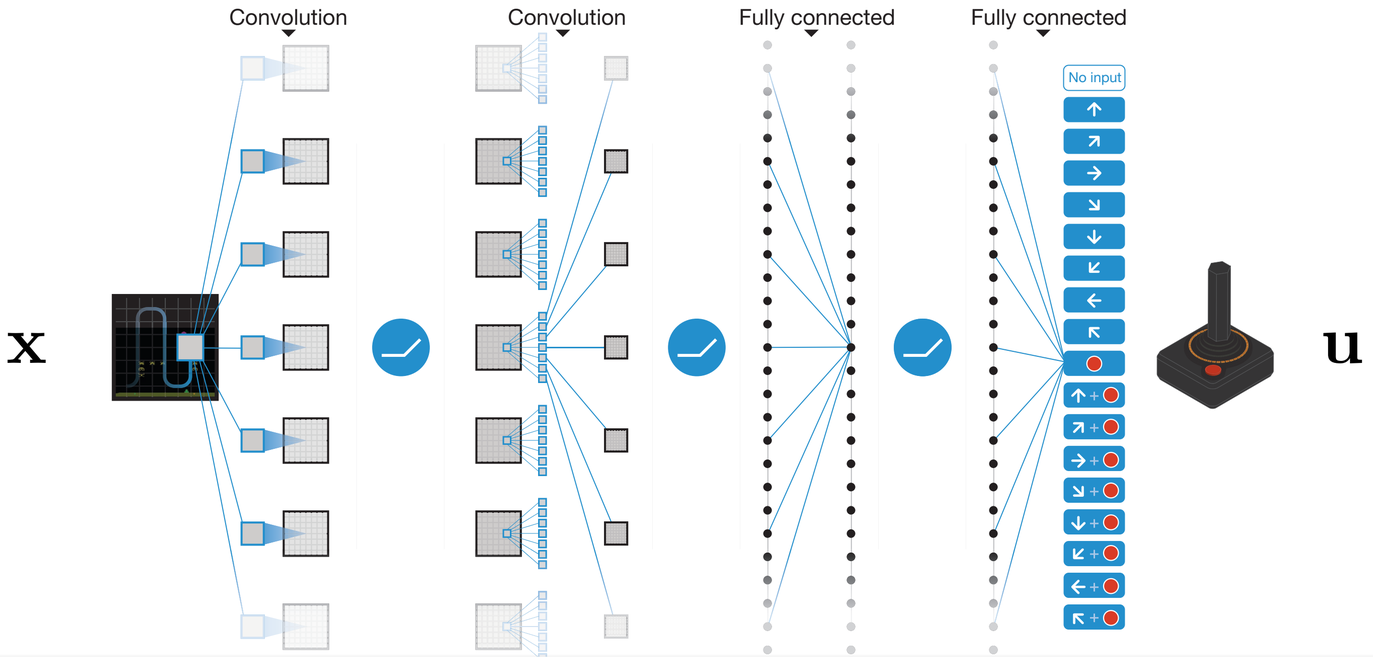
Policy net (regularly upd.)
encourage exploration

5. Update target network $\overline{\theta} := \alpha\theta + (1 - \alpha)\overline{\theta}$
6. Repeat from 1

Mnih et al. Nature 2015

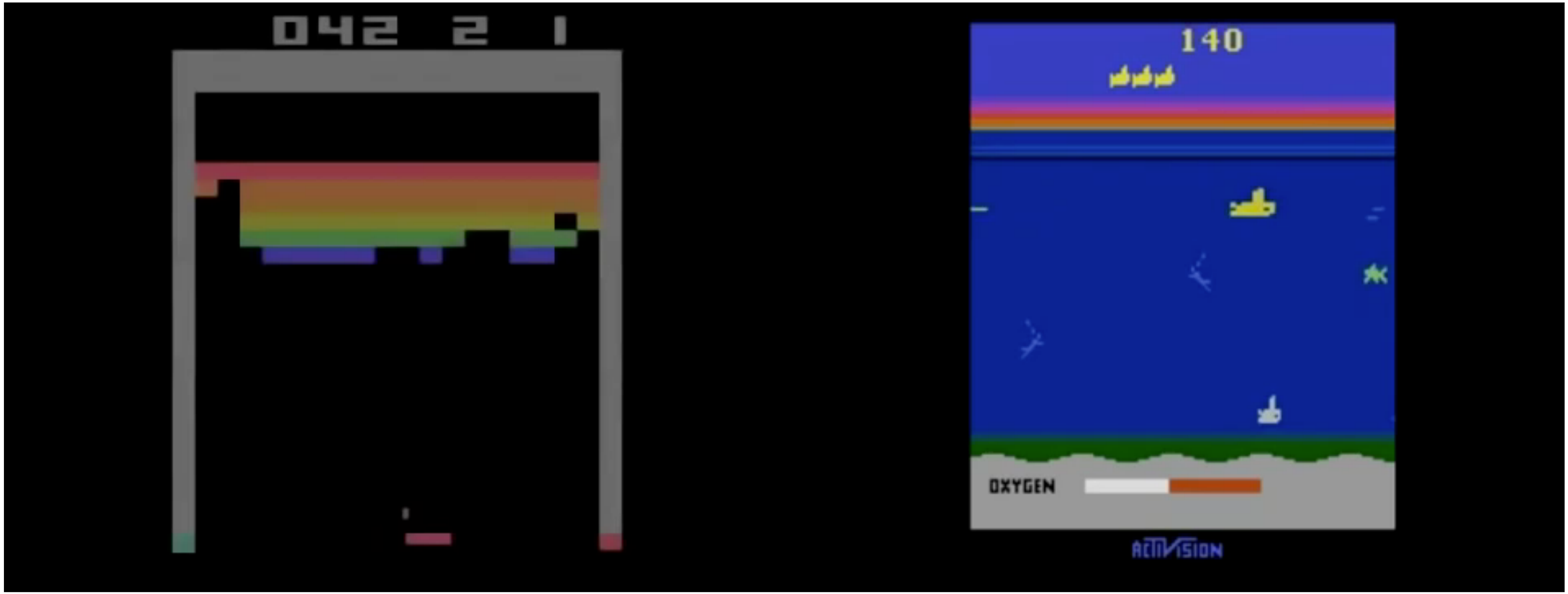
- 2600 atari games
- **state space $\mathbf{x}$:** last four frames to capture dynamics
  (e.g. RGB images in VGA resolution)
- **action space $\mathbf{u}$:** 18 discrete joystic actions
  (8 direction + 8 direction with button + neutral action + neutral with button)

$$Q_\theta(\mathbf{x}, \mathbf{u})$$



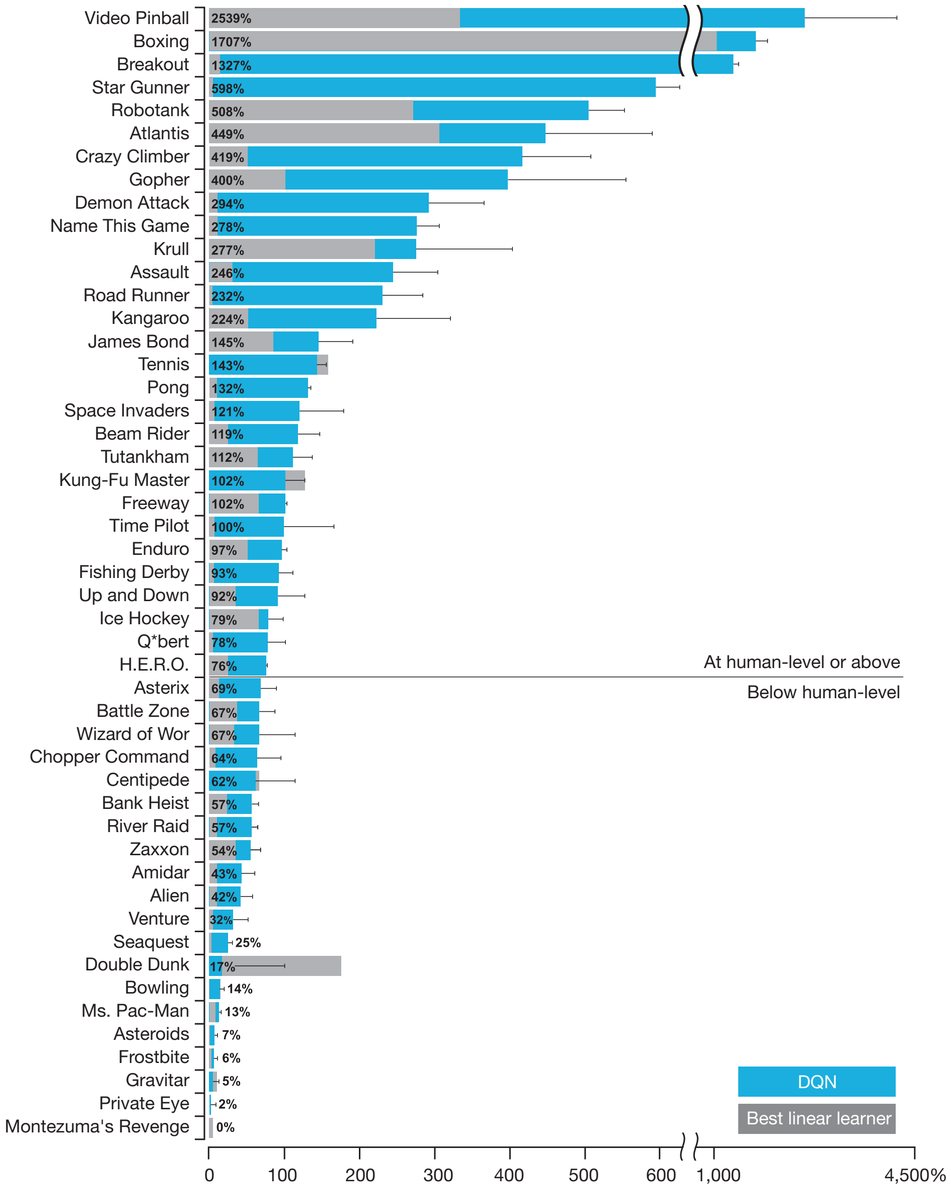
Mnih et al. Nature 2015

- **replay buffer:** decorrelates samples to be “more i.i.d” through
- **two Q-networks:** suppress trn/tst mismatch (oscilations)



- collection of control tasks: https://gym.openai.com

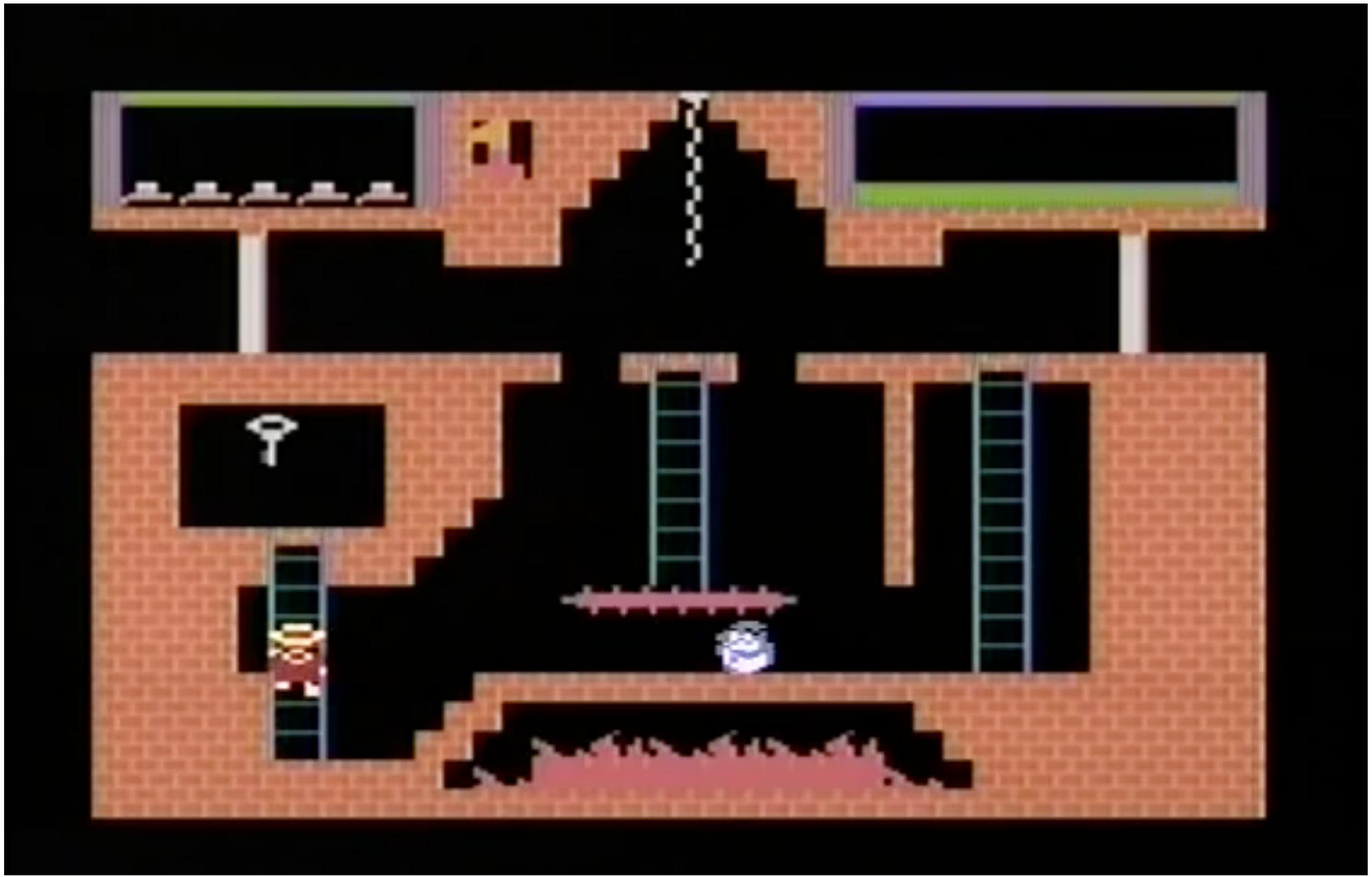
Mnih et al. Nature 2015



Best linear learner

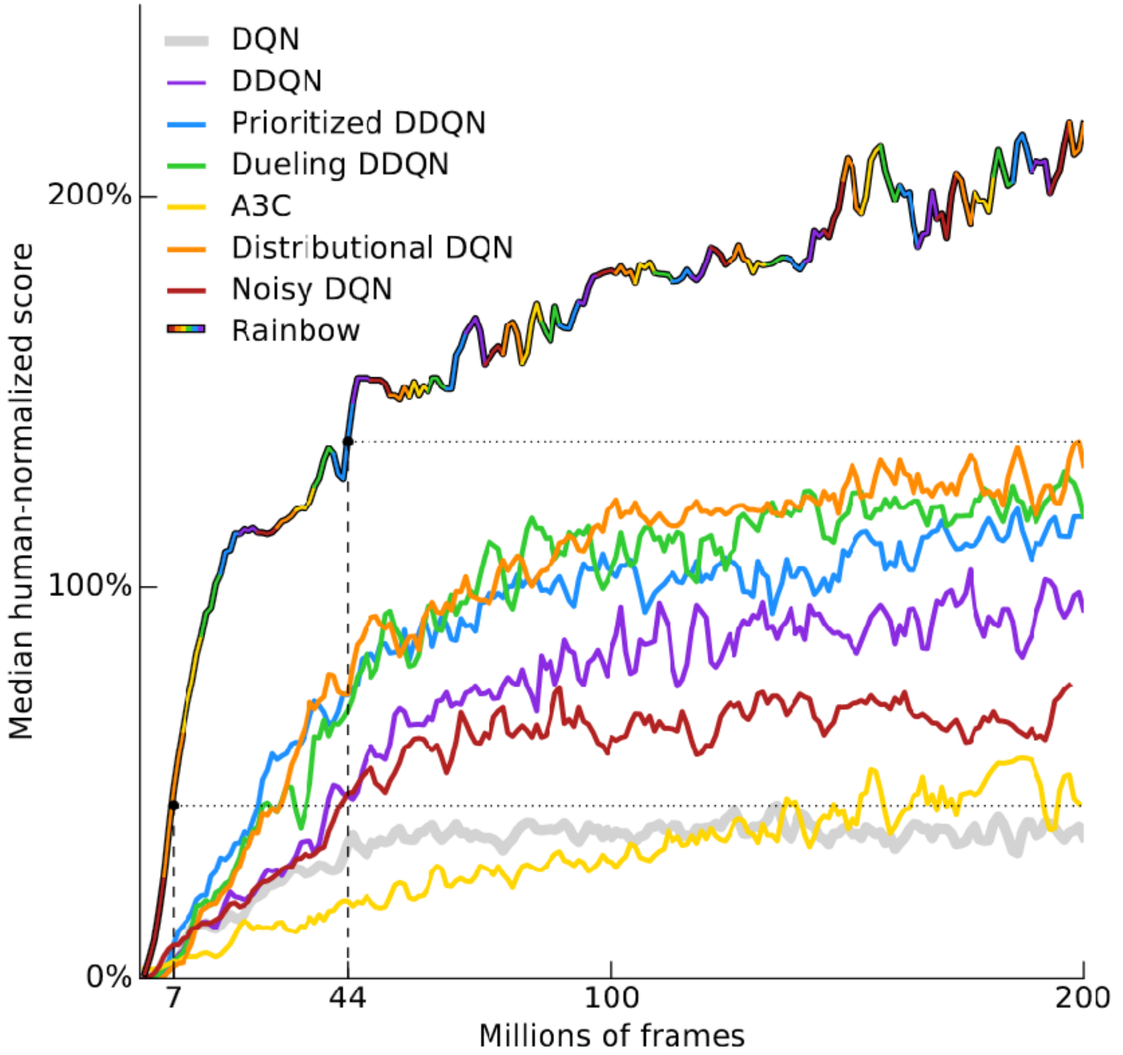
Montezuma's Revenge 0%

0 100 200 300 400 500 600 1,000 4,500%



Hessel et. al Rainbow DQN, 2017

Ensemble of different RL methods



Main bottleneck of approximate Q-learning (DQN)

continuous
high-dimensional

$\mathbf{x}$ → $Q_\theta(\mathbf{x}, \mathbf{u})$ →

$Q_\theta(\mathbf{x}, \mathbf{u}_1)$
$Q_\theta(\mathbf{x}, \mathbf{u}_2)$
...
...
$Q_\theta(\mathbf{x}, \mathbf{u}_K)$

6 legs, 3 motors/leg,
10 input levels/motor

$\dim(\mathbf{u}) =$ **???**

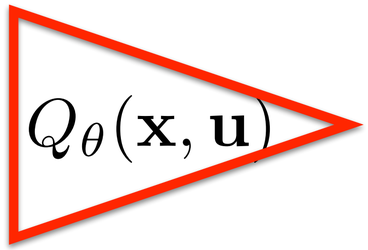
1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_{\overline{\theta}}(\mathbf{x}', \mathbf{u}')$
4. Update critic $\arg\min_{\theta^Q} \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_{\theta^Q}(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$

6. Update target network $\overline{\theta} := \alpha\theta + (1 - \alpha)\overline{\theta}$
7. Repeat from 1

Main bottleneck of approximate Q-learning (DQN)

continuous
high-dimensional

$\mathbf{x}$



$Q_\theta(\mathbf{x}, \mathbf{u}_1)$
$Q_\theta(\mathbf{x}, \mathbf{u}_2)$
...
...
$Q_\theta(\mathbf{x}, \mathbf{u}_K)$

6 legs, 3 motors/leg,
10 input levels/motor

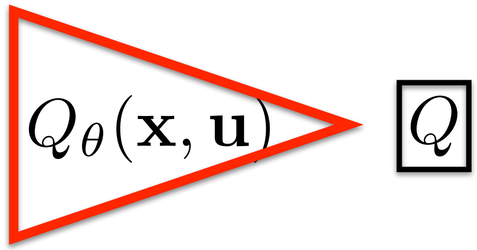
$\dim(\mathbf{u}) =$ **10^(3*6)**

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_{\overline{\theta}}(\mathbf{x}', \mathbf{u}')$
4. Update critic $\arg\min_{\theta^Q} \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_{\theta^Q}(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$

6. Update target network $\overline{\theta} := \alpha\theta + (1 - \alpha)\overline{\theta}$
7. Repeat from 1

Main bottleneck of approximate Q-learning (DQN)

continuous high-dimensional

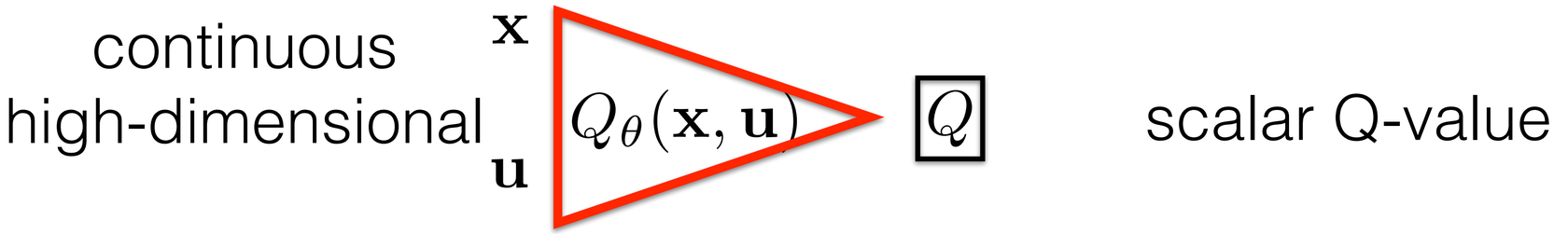


scalar Q-value

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_{\overline{\theta}}(\mathbf{x}', \mathbf{u}')$
4. Update critic $\arg\min_{\theta^Q} \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_{\theta^Q}(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$

6. Update target network $\overline{\theta} := \alpha\theta + (1 - \alpha)\overline{\theta}$
7. Repeat from 1

Main bottleneck of approximate Q-learning (DQN)

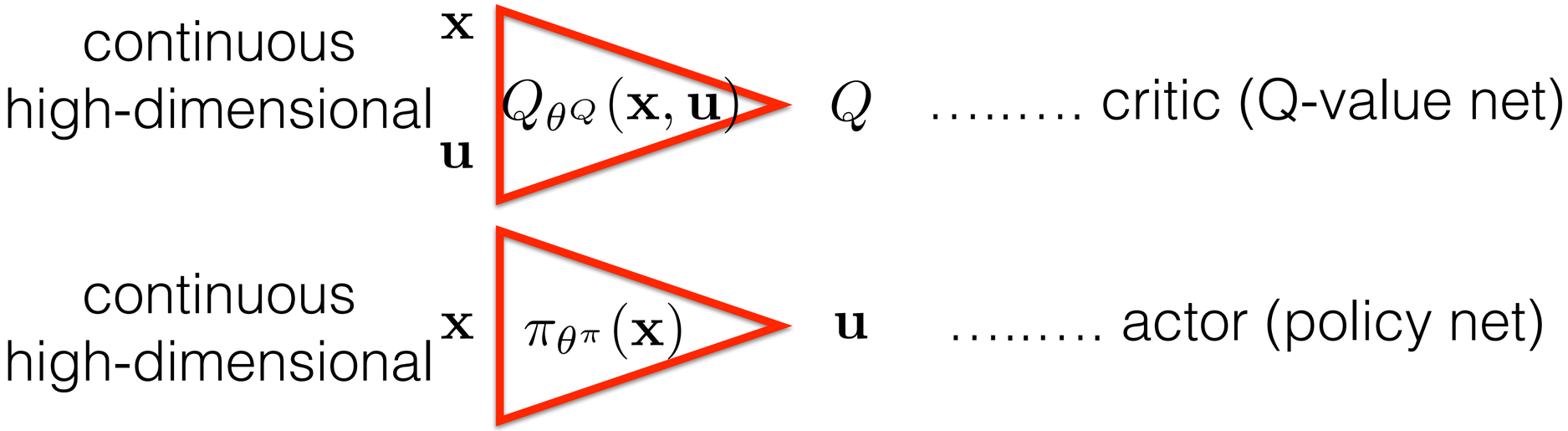


You cannot exhaustively maximize

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_{\overline{\theta}}(\mathbf{x}', \mathbf{u}')$
4. Update critic $\arg\min_{\theta^Q} \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_{\theta^Q}(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$

6. Update target network $\overline{\theta} := \alpha\theta + (1 - \alpha)\overline{\theta}$
7. Repeat from 1

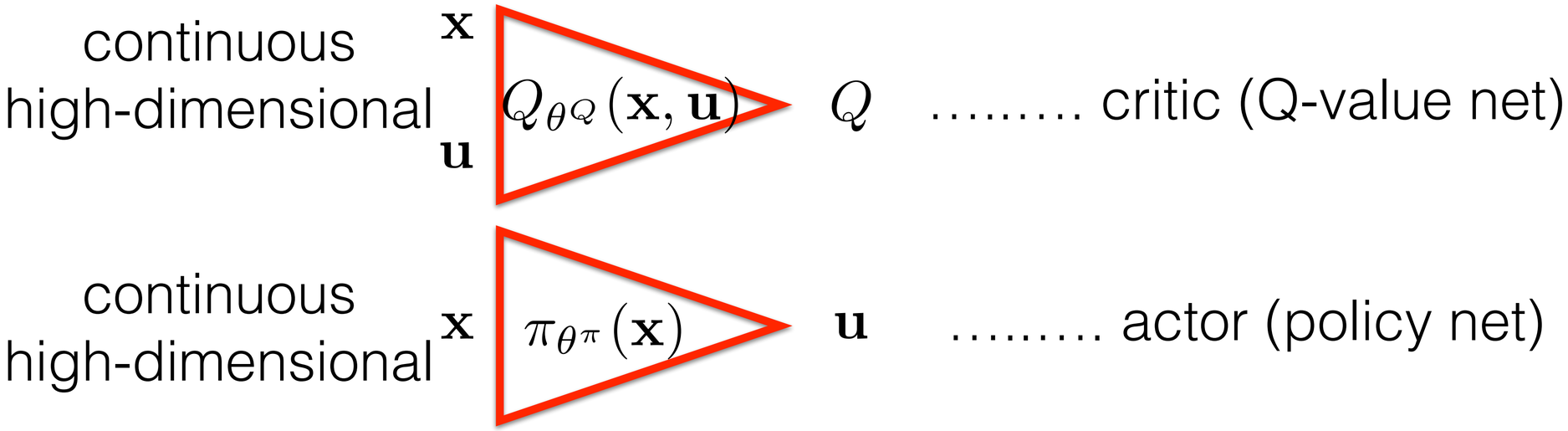
Deep Deterministic Policy Gradient (DDPG)



1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_{\overline{\theta^Q}}(\mathbf{x}', \mathbf{u}')$
4. Update critic $\arg\min_{\theta^Q} \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_{\theta^Q}(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$

5. Update target network $\overline{\theta^Q} := \alpha\theta^Q + (1-\alpha)\overline{\theta^Q}$
6. Repeat from 1

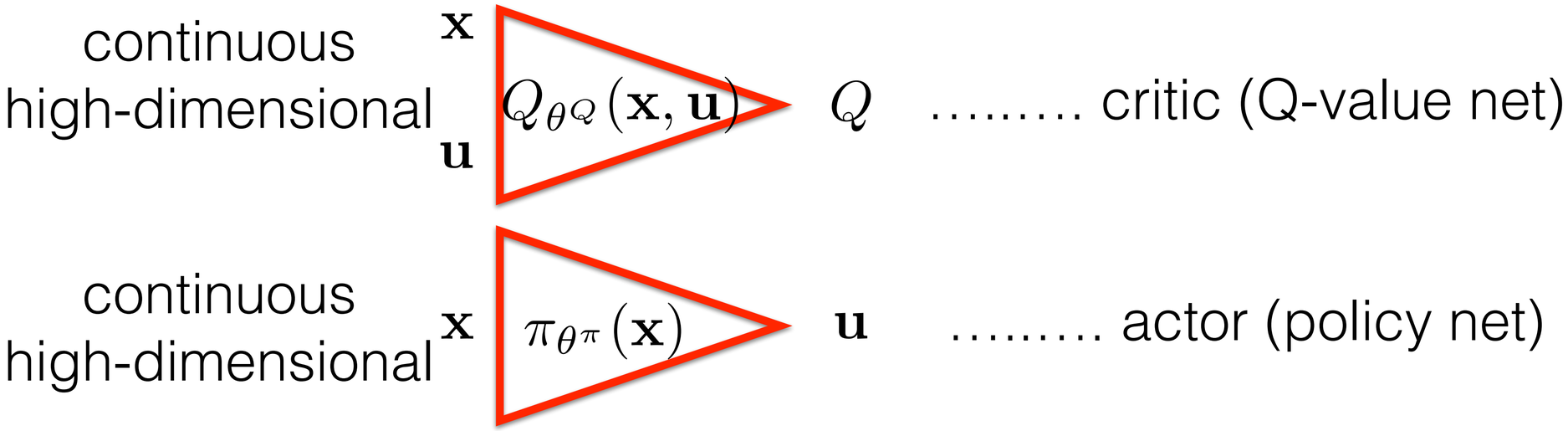
Deep Deterministic Policy Gradient (DDPG)



1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma \max_{\mathbf{u}'} Q_{\overline{\theta^Q}}(\mathbf{x}', \mathbf{u}')$
4. Update critic $\arg\min_{\theta^Q} \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_{\theta^Q}(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$

5. Update target network $\overline{\theta^Q} := \alpha\theta^Q + (1-\alpha)\overline{\theta^Q}$
6. Repeat from 1

Deep Deterministic Policy Gradient (DDPG)



1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma Q_{\overline{\theta^Q}}(\mathbf{x}', \pi_{\overline{\theta^\pi}}(\mathbf{x}'))$
4. Update critic $\arg\min_{\theta^Q} \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_{\theta^Q}(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$
5. Update target network $\overline{\theta^Q} := \alpha\theta^Q + (1-\alpha)\overline{\theta^Q}$
6. Repeat from 1

Deep Deterministic Policy Gradient (DDPG)

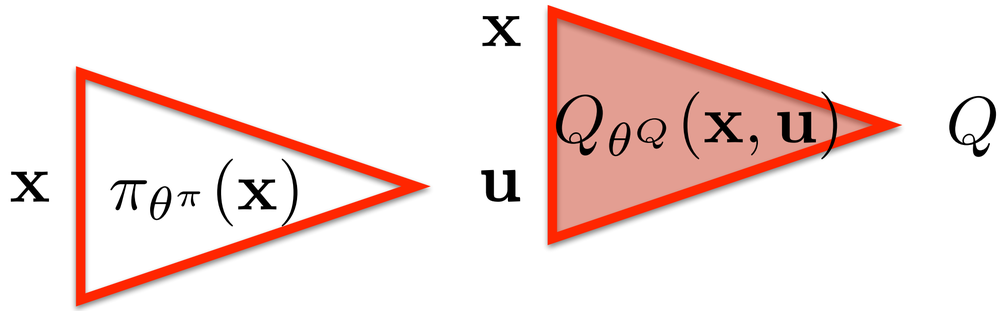
continuous high-dimensional $\mathbf{x}$, $\mathbf{u}$ → $Q_{\theta^Q}(\mathbf{x}, \mathbf{u})$ → Q ..... critic (Q-value net)

continuous high-dimensional $\mathbf{x}$ → $\pi_{\theta^\pi}(\mathbf{x})$ → $\mathbf{u}$ ..... actor (policy net)

1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma Q_{\overline{\theta^Q}}(\mathbf{x}', \pi_{\overline{\theta^\pi}}(\mathbf{x}'))$
4. Update critic $\arg\min_{\theta^Q} \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_{\theta^Q}(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$
5. Update actor $\arg\max_{\theta^\pi} \sum_{\mathbf{x}} Q_{\theta^Q}(\mathbf{x}, \pi_{\theta^\pi}(\mathbf{x}))$
6. Update target network $\overline{\theta^Q} := \alpha\theta^Q + (1-\alpha)\overline{\theta^Q}$, $\overline{\theta^\pi} := \alpha\theta^\pi + (1-\alpha)\overline{\theta^\pi}$
7. Repeat from 1

Deep Deterministic Policy Gradient (DDPG)

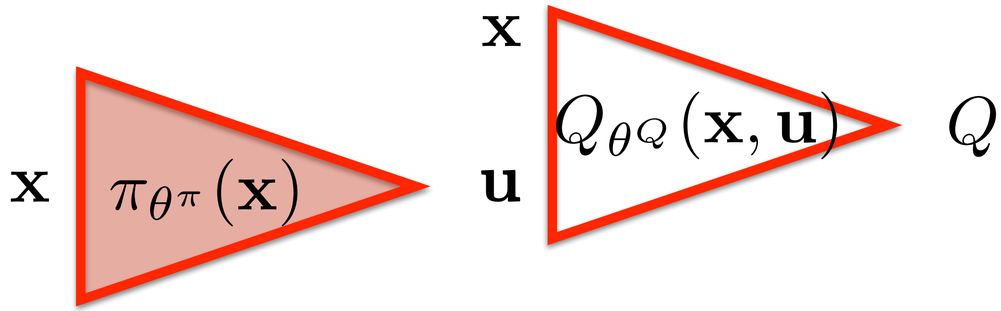
Update critic



1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma Q_{\overline{\theta^Q}}(\mathbf{x}', \pi_{\overline{\theta^\pi}}(\mathbf{x}'))$
4. Update critic $\arg\min_{\theta^Q} \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_{\theta^Q}(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$
5. Update actor $\arg\max_{\theta^\pi} \sum_{\mathbf{x}} Q_{\theta^Q}(\mathbf{x}, \pi_{\theta^\pi}(\mathbf{x}))$
6. Update target network $\overline{\theta^Q} := \alpha\theta^Q + (1-\alpha)\overline{\theta^Q}$
   $\overline{\theta^\pi} := \alpha\theta^\pi + (1-\alpha)\overline{\theta^\pi}$
7. Repeat from 1

Deep Deterministic Policy Gradient (DDPG)

Update actor



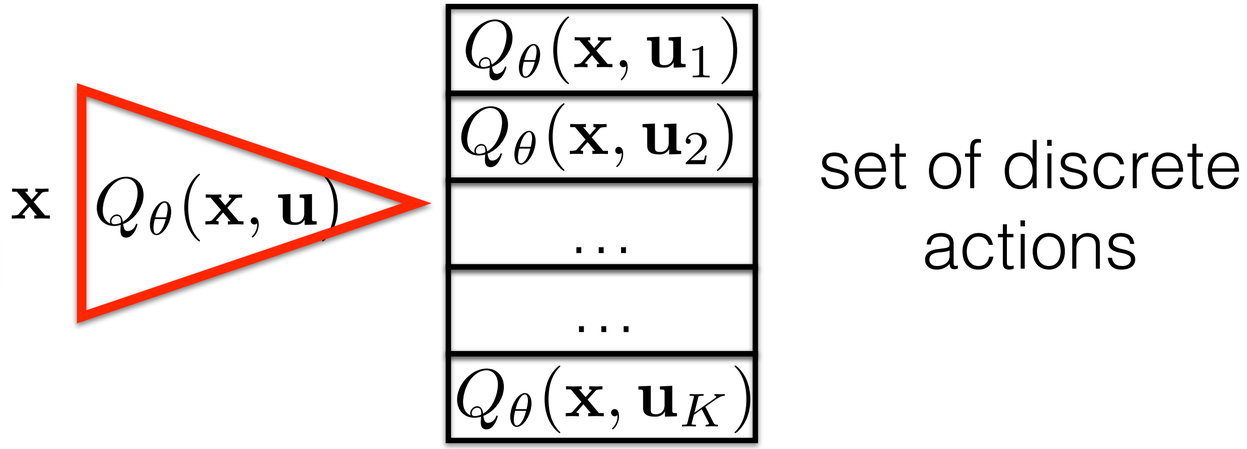
1. Collect transition $[\mathbf{x}, \mathbf{u}, r, \mathbf{x}']$=> ReplayMemory
2. Sample transition(s) at random from ReplayMemory
3. Estimate target(s) $\mathbf{y} = r + \gamma Q_{\overline{\theta^Q}}(\mathbf{x}', \pi_{\overline{\theta^\pi}}(\mathbf{x}'))$
4. Update critic $\arg\min_{\theta^Q} \sum_{\mathbf{x},\mathbf{u},\mathbf{y}} \|Q_{\theta^Q}(\mathbf{x}, \mathbf{u}) - \mathbf{y}\|$
5. Update actor $\arg\max_{\theta^\pi} \sum_{\mathbf{x}} Q_{\theta^Q}(\mathbf{x}, \pi_{\theta^\pi}(\mathbf{x}))$
6. Update target network $\overline{\theta^Q} := \alpha\theta^Q + (1-\alpha)\overline{\theta^Q}$
   $\overline{\theta^\pi} := \alpha\theta^\pi + (1-\alpha)\overline{\theta^\pi}$
7. Repeat from 1

Summary

- DQN and DDPG are off-policy algorithms
  (can learn from transitions collected by a different policy)

DQN:

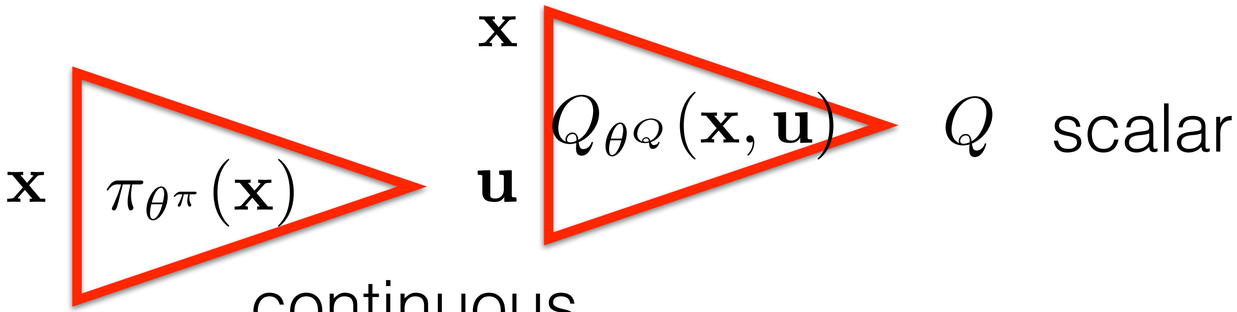
continuous
high-dimensional



set of discrete
actions

DDPG:

continuous
high-dimensional



Q scalar

continuous
high-dimensional

Summary

- **off-policy** algorithms can learn from transitions collected by a different policy
- **on-policy** algorithms can learn only from transitions collected by the current policy
- DQN and DDPG are **off-policy** algorithms
  - => Can use ReplayMemory (which includes outdated transitions)
  - => Can learn deterministic policy (while using synth.noise for exploration)

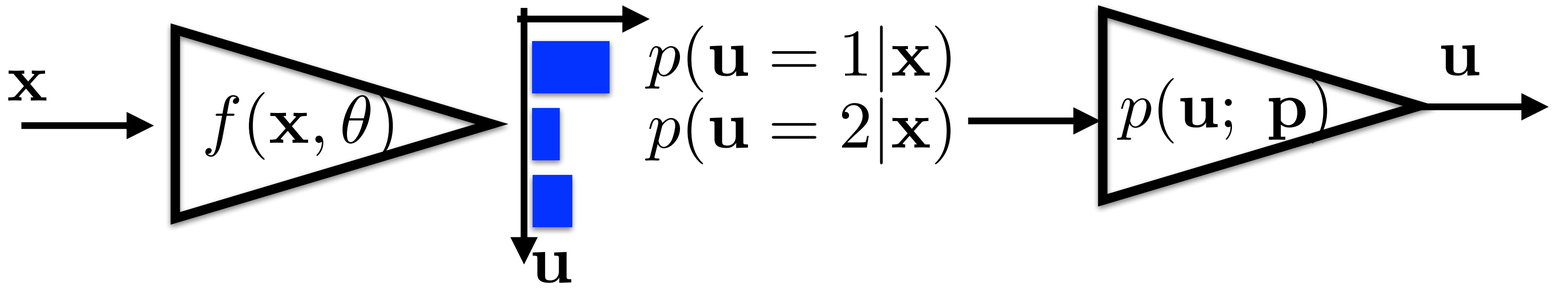
- **Replay memory** helps to decorrelate samples.
- **Slowly updated network** suppresses trn/tst distribution mismatch (and conesequent learning oscillations).
- **Ensemble** of different algorithms helps a lot.

- Next: **On-policy** methods with stochastic gradient

REINFORCE

Stochastic policy
discrete control:

$\pi_\theta(\mathbf{u}|\mathbf{x}) \sim p(\mathbf{u};\ \mathbf{p})$



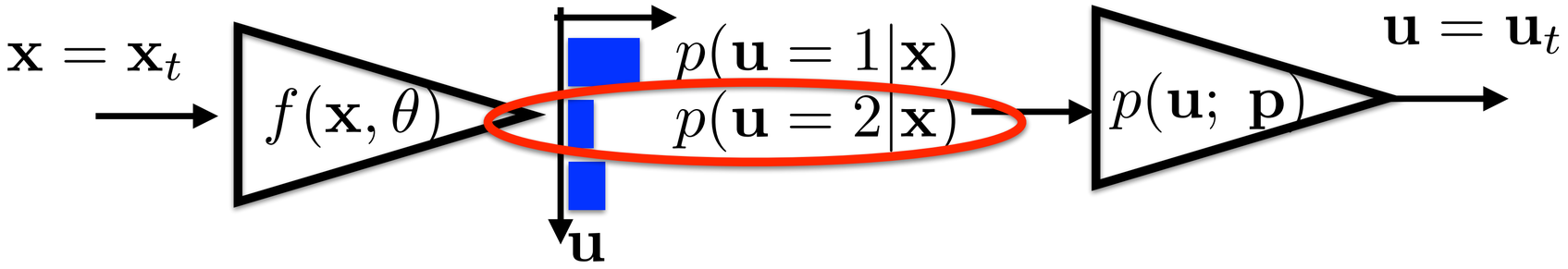
1. Initialize policy $\pi_\theta(\mathbf{u}|\mathbf{x}) = f(\mathbf{x},\theta)$
2. Collect trajectories τ with policy π_θ
3. Optimize criterion: $\theta := \theta + \alpha \dfrac{\partial J(\theta)}{\partial \theta}$
4. Repeat from 2

What is the gradient???

You control robot with $\mathbf{u} = 2$ and sum of rewards $r(\tau)$ is **high**, how to change policy?

You control robot with $\mathbf{u} = 1$ and sum of rewards $r(\tau)$ is **small**, how to change policy?

What is the gradient???



Assume that you performed action $\mathbf{u}_t$ in state $\mathbf{x}_t$

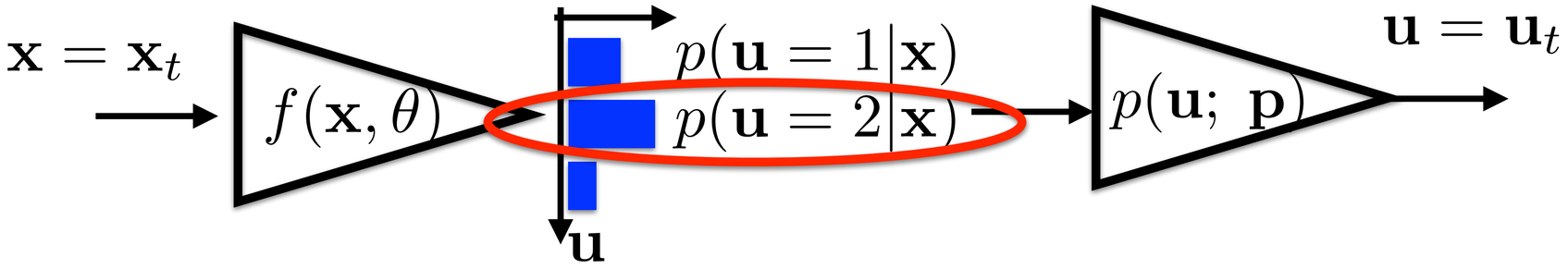
$$\frac{\partial J(\theta)}{\partial \theta} \approx \sum_{t=0}^{T} \frac{\partial \log(\pi_\theta(\mathbf{u}_t|\mathbf{x}_t))}{\partial \theta} \cdot r(\tau)$$

$r(\tau)$ =+100

Sum of rewards along the resulting trajectory.

Direction (in θ-space) that increases probability of performed action.

What is the gradient???



Assume that you performed action $\mathbf{u}_t$ in state $\mathbf{x}_t$

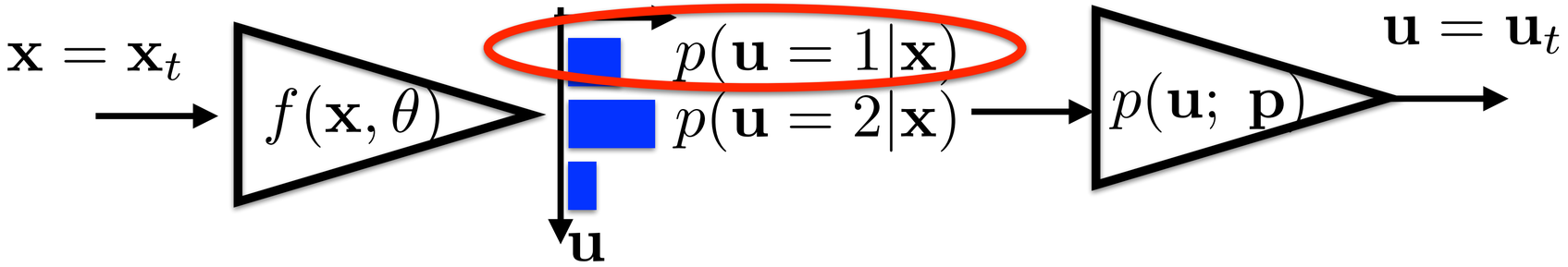
$$\frac{\partial J(\theta)}{\partial \theta} \approx \sum_{t=0}^{T} \frac{\partial \log(\pi_\theta(\mathbf{u}_t|\mathbf{x}_t))}{\partial \theta} \cdot r(\tau)$$

$r(\tau) = +100$

Sum of rewards along the resulting trajectory.

Direction (in θ-space) that increases probability of performed action.

What is the gradient???



Assume that you performed action $\mathbf{u}_t$ in state $\mathbf{x}_t$

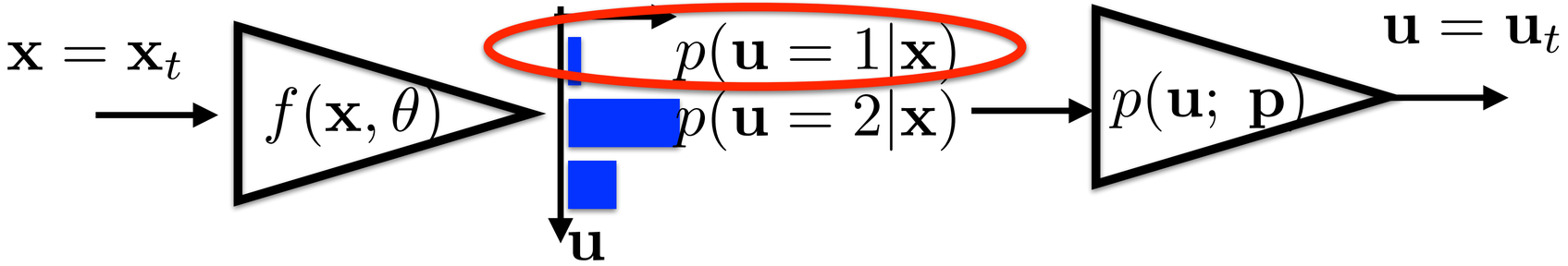
$$\frac{\partial J(\theta)}{\partial \theta} \approx \sum_{t=0}^{T} \frac{\partial \log(\pi_\theta(\mathbf{u}_t|\mathbf{x}_t))}{\partial \theta} \cdot r(\tau)$$

$r(\tau)$ =-100

Sum of rewards along the resulting trajectory.

Direction (in θ-space) that increases probability of performed action.

What is the gradient???



Assume that you performed action $\mathbf{u}_t$ in state $\mathbf{x}_t$

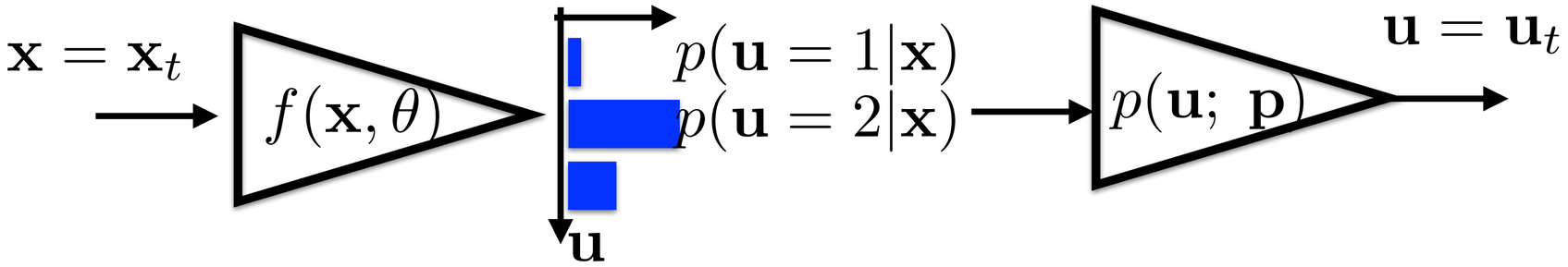
$$\frac{\partial J(\theta)}{\partial \theta} \approx \sum_{t=0}^{T} \frac{\partial \log(\pi_\theta(\mathbf{u}_t|\mathbf{x}_t))}{\partial \theta} \cdot r(\tau)$$

$r(\tau)$ =-100

Sum of rewards along the resulting trajectory.

Direction (in θ-space) that increases probability of performed action.

REINFORCE



1. Initialize policy $\pi_\theta(\mathbf{u}|\mathbf{x}) = f(\mathbf{x}, \theta)$
2. Collect trajectories τ with policy $\pi_\theta(\mathbf{u}|\mathbf{x})$
3. Update policy (actor):

$$\frac{\partial J(\theta)}{\partial \theta} \approx \sum_{t=0}^{T} \frac{\partial \log(\pi_\theta(\mathbf{u}_t|\mathbf{x}_t))}{\partial \theta} \cdot r(\tau)$$

$$\theta := \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}$$

4. Repeat from 2

Policy gradient derivation

$$J(\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} [\underbrace{\sum_{t=0}^{T} \gamma^t r_{t+1}}_{r(\tau)}] = \int_T p(\tau|\pi_\theta) r(\tau) d\tau$$

$$\frac{\partial J(\theta)}{\partial \theta} = \int_T \frac{\partial p(\tau|\pi_\theta)}{\partial \theta} r(\tau) d\tau = \int_T p(\tau|\pi_\theta) \frac{\partial \log p(\tau|\pi_\theta)}{\partial \theta} r(\tau) d\tau =$$

$$\frac{\partial p(\tau|\pi_\theta)}{\partial \theta} = p(\tau|\pi_\theta) \frac{\partial \log p(\tau|\pi_\theta)}{\partial \theta}$$

Policy gradient derivation

$$J(\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[\underbrace{\sum_{t=0}^{T}\gamma^t r_{t+1}}_{r(\tau)}] = \int_T p(\tau|\pi_\theta)r(\tau)d\tau$$

$$\frac{\partial J(\theta)}{\partial\theta} = \int_T \frac{\partial p(\tau|\pi_\theta)}{\partial\theta}r(\tau)d\tau = \int_T p(\tau|\pi_\theta)\frac{\partial \log p(\tau|\pi_\theta)}{\partial\theta}r(\tau)d\tau = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[\frac{\partial \log p(\tau|\pi_\theta)}{\partial\theta}r(\tau)]$$

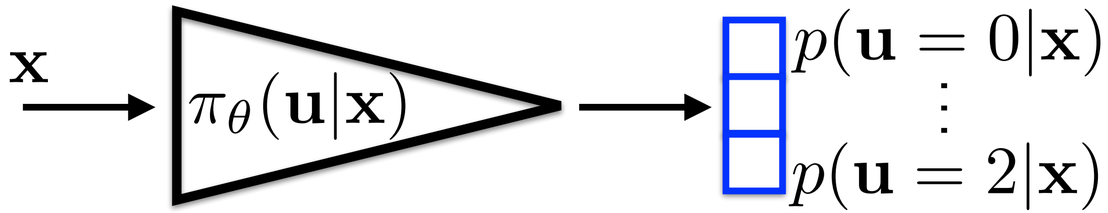
$$= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T}\frac{\partial \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}{\partial\theta}r(\tau)] \approx \frac{1}{N}\sum_{\tau\in\mathcal{T}}\sum_{t=0}^{T}\frac{\partial \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}{\partial\theta}r(\tau)$$

$$p(\tau|\pi_\theta) = p(\mathbf{x}_0)\prod_{t=0}^{T} p(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x_t}) \quad \text{.... assuming MDP}$$

$$\frac{\partial \log p(\tau|\pi_\theta)}{\partial\theta} = \frac{\partial}{\partial\theta}[\log p(\mathbf{x}_0) + \sum_{t=0}^{T}\log(p(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t)) + \sum_{t=0}^{T}\log(\pi_\theta(\mathbf{u}_t|\mathbf{x_t})]$$

REINFORCE

Stochastic policy for discrete control:



1. Initialize policy $\pi_\theta(\mathbf{u}|\mathbf{x}) = f(\mathbf{x}, \theta)$
2. Collect trajectories τ with policy $\pi_\theta(\mathbf{u}|\mathbf{x})$
4. **Actor:** Update policy:

$$\frac{\partial J(\theta)}{\partial \theta} \approx \frac{1}{N} \sum_{\tau \in \mathcal{T}} \sum_{t=0}^{T} \frac{\partial \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}{\partial \theta} r(\tau)$$

$$\theta := \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}$$

5. Repeat from 2

Policy gradients for stochastic policy [Schulman et al 2016]

- temporal coherence

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{(\mathbf{u}_t, \mathbf{x}_t) \in \tau} \frac{\partial \log \pi_\theta(\mathbf{u}_t | \mathbf{x}_t)}{\partial \theta} \cdot r(\tau)$$

Policy gradients for stochastic policy [Schulman et al 2016]

- temporal coherence

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{(\mathbf{u}_t, \mathbf{x}_t) \in \tau} \frac{\partial \log \pi_\theta(\mathbf{u}_t | \mathbf{x}_t)}{\partial \theta} \cdot \left(\sum_{k=1}^{T} \gamma^{k-1} r(\mathbf{u}_k, \mathbf{x}_k) \right)$$

Policy gradients for stochastic policy [Schulman et al 2016]

- temporal coherence

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{(\mathbf{u}_t, \mathbf{x}_t) \in \tau} \frac{\partial \log \pi_\theta(\mathbf{u}_t | \mathbf{x}_t)}{\partial \theta} \cdot \left(\sum_{k=1}^{T} \gamma^{k-1} r(\mathbf{u}_k, \mathbf{x}_k) \right)$$

Policy gradients for stochastic policy [Schulman et al 2016]

- temporal coherence

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{(\mathbf{u}_t, \mathbf{x}_t) \in \tau} \frac{\partial \log \pi_\theta(\mathbf{u}_t | \mathbf{x}_t)}{\partial \theta} \cdot \left(\sum_{k=t}^{T} \gamma^{k-1} r(\mathbf{u}_k, \mathbf{x}_k) \right)$$

Policy gradients for stochastic policy [Schulman et al 2016]

- temporal coherence

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{(\mathbf{u}_t, \mathbf{x}_t) \in \tau} \frac{\partial \log \pi_\theta(\mathbf{u}_t | \mathbf{x}_t)}{\partial \theta} \cdot \left(\sum_{k=t}^{T} \gamma^{k-1} r(\mathbf{u}_k, \mathbf{x}_k) \right)$$

- state-action function (policy gradient theorem):

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{(\mathbf{u}_t, \mathbf{x}_t) \in \tau} \frac{\partial \log \pi_\theta(\mathbf{u}_t | \mathbf{x}_t)}{\partial \theta} \cdot Q(\mathbf{u}_t, \mathbf{x}_t)$$

Policy gradients for stochastic policy [Schulman et al 2016]

- temporal coherence

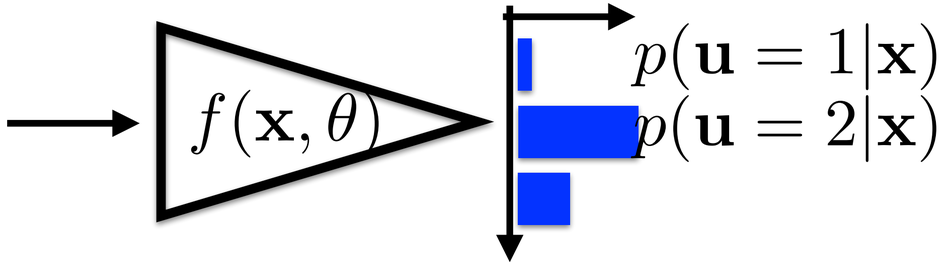
$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{(\mathbf{u}_t, \mathbf{x}_t) \in \tau} \frac{\partial \log \pi_\theta(\mathbf{u}_t | \mathbf{x}_t)}{\partial \theta} \cdot \left(\sum_{k=t}^{T} \gamma^{k-1} r(\mathbf{u}_k, \mathbf{x}_k) \right)$$

- state-action function (policy gradient theorem):

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{(\mathbf{u}_t, \mathbf{x}_t) \in \tau} \frac{\partial \log \pi_\theta(\mathbf{u}_t | \mathbf{x}_t)}{\partial \theta} \cdot Q(\mathbf{u}_t, \mathbf{x}_t)$$

- arbitrary baseline can be subtracted (Q-function => A-function)

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{(\mathbf{u}_t, \mathbf{x}_t) \in \tau} \frac{\partial \log \pi_\theta(\mathbf{u}_t | \mathbf{x}_t)}{\partial \theta} \cdot A(\mathbf{u}_t, \mathbf{x}_t)$$



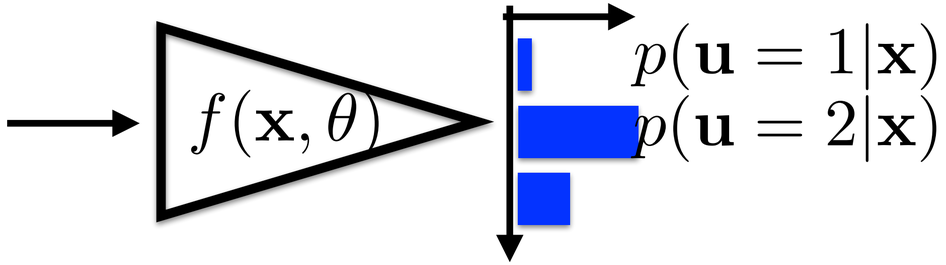
1. Initialize policy $\pi_\theta(\mathbf{u}|\mathbf{x})$
2. Collect trajectories τ with policy $\pi_\theta(\mathbf{u}|\mathbf{x})$
4. **Actor:** Update policy by policy gradient:

$$\frac{\partial \mathcal{L}_{\text{actor}}(\theta)}{\partial \theta} = \sum_{(\mathbf{u}_t, \mathbf{x}_t) \in \tau} \frac{\partial \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}{\partial \theta} \cdot A(\mathbf{u}_t, \mathbf{x}_t)$$

replace by approximation

$$\theta := \theta + \alpha \frac{\partial \mathcal{L}_{\text{actor}}(\theta)}{\partial \theta}$$

5. Repeat from 2



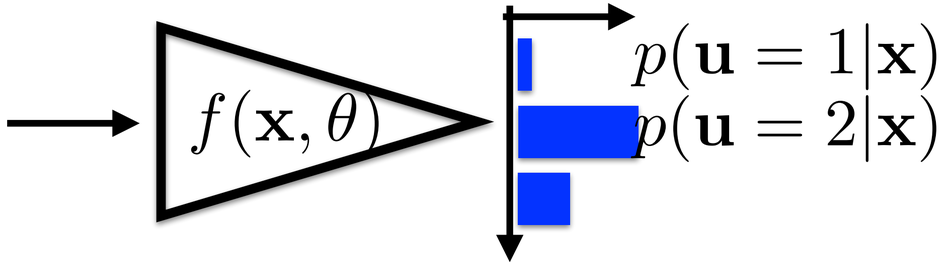
1. Initialize policy $\pi_\theta(\mathbf{u}|\mathbf{x})$
2. Collect trajectories τ with policy $\pi_\theta(\mathbf{u}|\mathbf{x})$
4. **Actor:** Update policy by policy gradient:

$$\frac{\partial \mathcal{L}_{\text{actor}}(\theta)}{\partial \theta} = \sum_{(\mathbf{u}_t, \mathbf{x}_t) \in \tau} \frac{\partial \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}{\partial \theta} \cdot \underbrace{\Big(r + \gamma V_\omega(\mathbf{x}') - V_\omega(\mathbf{x})\Big)}_{A_\omega = Q - V}$$

replace by approximation

$$\theta := \theta + \alpha \frac{\partial \mathcal{L}_{\text{actor}}(\theta)}{\partial \theta}$$

5. Repeat from 2



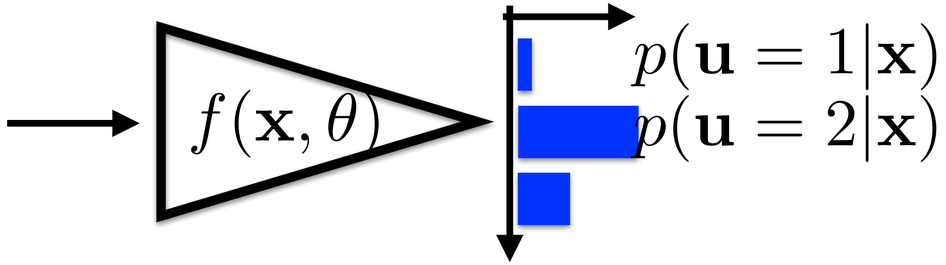
1. Initialize policy $\pi_\theta(\mathbf{u}|\mathbf{x})$
2. Collect trajectories τ with policy $\pi_\theta(\mathbf{u}|\mathbf{x})$

4. **Actor:** Update policy by policy gradient:
replace by criterion

$$\frac{\partial \mathcal{L}_{\text{actor}}(\theta)}{\partial \theta} = \sum_{(\mathbf{u}_t, \mathbf{x}_t) \in \tau} \frac{\partial \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}{\partial \theta} \cdot \underbrace{\Big(r + \gamma V_\omega(\mathbf{x}') - V_\omega(\mathbf{x})\Big)}_{A_\omega = Q - V}$$

$$\theta := \theta + \alpha \frac{\partial \mathcal{L}_{\text{actor}}(\theta)}{\partial \theta}$$

5. Repeat from 2

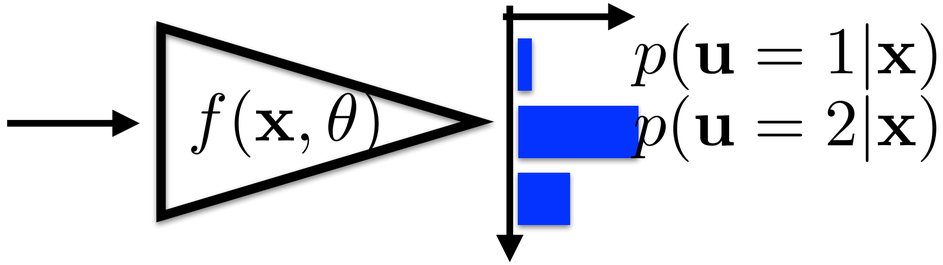


1. Initialize policy $\pi_\theta(\mathbf{u}|\mathbf{x})$
2. Collect trajectories τ with policy $\pi_\theta(\mathbf{u}|\mathbf{x})$
4. **Actor:** Update policy by policy gradient:

$$\mathcal{L}_{\text{actor}}(\theta) = \sum_{(\mathbf{u},\mathbf{x},\mathbf{x}')\in\tau} \log \pi_\theta(\mathbf{u}|\mathbf{x}) \cdot \underbrace{\Big(r + \gamma V_\omega(\mathbf{x}') - V_\omega(\mathbf{x})\Big)}_{A_\omega = Q - V}$$

$$\theta := \theta + \alpha \frac{\partial \mathcal{L}_{\text{actor}}(\theta)}{\partial \theta}$$

5. Repeat from 2



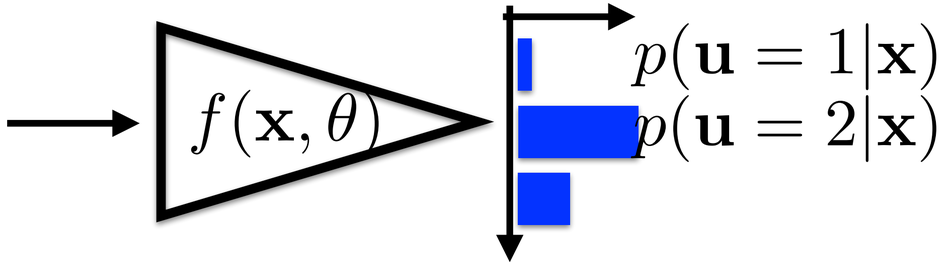
1. Initialize policy $\pi_\theta(\mathbf{u}|\mathbf{x})$, $V_\omega(\mathbf{x})$
2. Collect trajectories τ with policy $\pi_\theta(\mathbf{u}|\mathbf{x})$

4. **Actor:** Update policy by policy gradient:

$$\mathcal{L}_{\text{actor}}(\theta) = \sum_{(\mathbf{u},\mathbf{x},\mathbf{x}')\in\tau} \log \pi_\theta(\mathbf{u}|\mathbf{x}) \cdot \underbrace{\Big(r + \gamma V_\omega(\mathbf{x}') - V_\omega(\mathbf{x})\Big)}_{A_\omega = Q - V}$$

$$\theta := \theta + \alpha \frac{\partial \mathcal{L}_{\text{actor}}(\theta)}{\partial \theta}$$

5. Repeat from 2



1. Initialize policy $\pi_\theta(\mathbf{u}|\mathbf{x})$, $V_\omega(\mathbf{x})$
2. Collect trajectories τ with policy $\pi_\theta(\mathbf{u}|\mathbf{x})$
3. **Critic:** Update value function to predict observed values: $V_\omega(\mathbf{x}) \leftarrow r + \gamma V_\omega(\mathbf{x}')$

$$\mathcal{L}_{\text{critic}}(\omega) = \Big(\underbrace{r + \gamma V_\omega(\mathbf{x}') - V_\omega(\mathbf{x})}_{A_\omega} \Big)^2$$

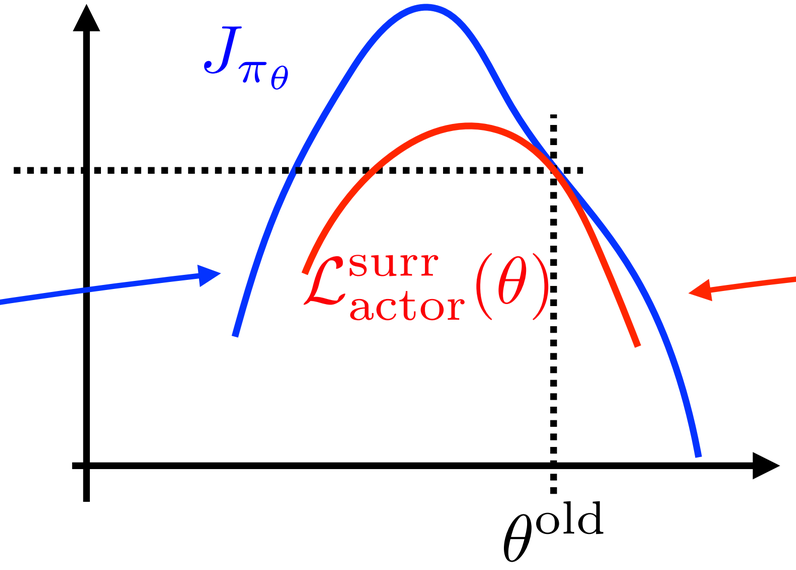
4. **Actor:** Update policy by policy gradient:

$$\mathcal{L}_{\text{actor}}(\theta) = \sum_{(\mathbf{u},\mathbf{x},\mathbf{x}') \in \tau} \log \pi_\theta(\mathbf{u}|\mathbf{x}) \cdot \underbrace{\Big(r + \gamma V_\omega(\mathbf{x}') - V_\omega(\mathbf{x}) \Big)}_{A_\omega = Q - V}$$

$$\theta := \theta + \alpha \frac{\partial \mathcal{L}_{\text{actor}}(\theta)}{\partial \theta} \qquad \omega := \omega + \beta \frac{\partial \mathcal{L}_{\text{critic}}(\omega)}{\partial \omega}$$

5. Repeat from 2

Proximal Policy Optimization PPO



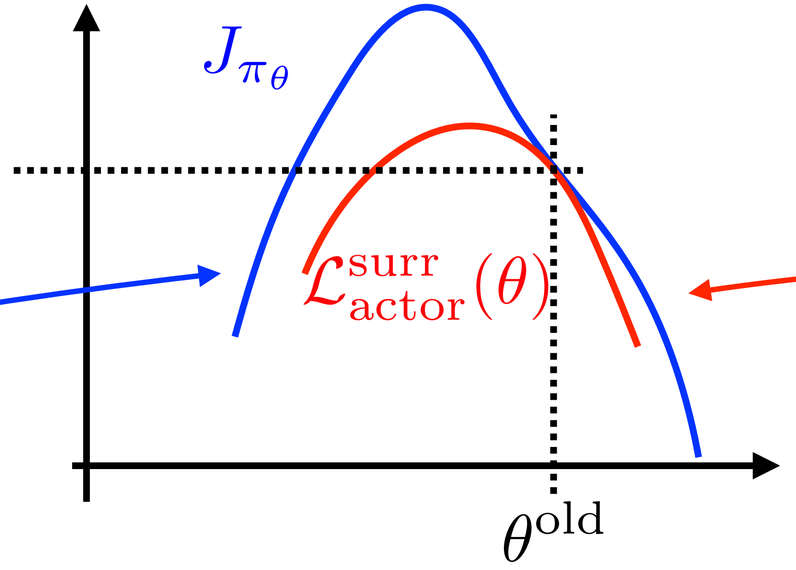
$$J_{\pi_\theta} = \mathbb{E}_{\tau\sim\pi_\theta}\{\sum_{r_t\sim\tau}\gamma^t r_t\} \geq \sum_{(\mathbf{u},\mathbf{x},\mathbf{x}')\in\tau} \frac{\pi_\theta(\mathbf{u}|\mathbf{x})}{\pi_{\theta^{\text{old}}}(\mathbf{u}|\mathbf{x})} \cdot \hat{A}_\omega(\mathbf{u},\mathbf{x}) = \mathcal{L}^{\text{surr}}_{\text{actor}}(\theta)$$

Lower bound surrogate loss that **locally approximates** the true criterion

You can also understand it as importance-sampling re-weighted policy gradient

https://jonathan-hui.medium.com/rl-the-math-behind-trpo-ppo-d12f6c745f33

Trust Region Policy Optimization (TRPO)

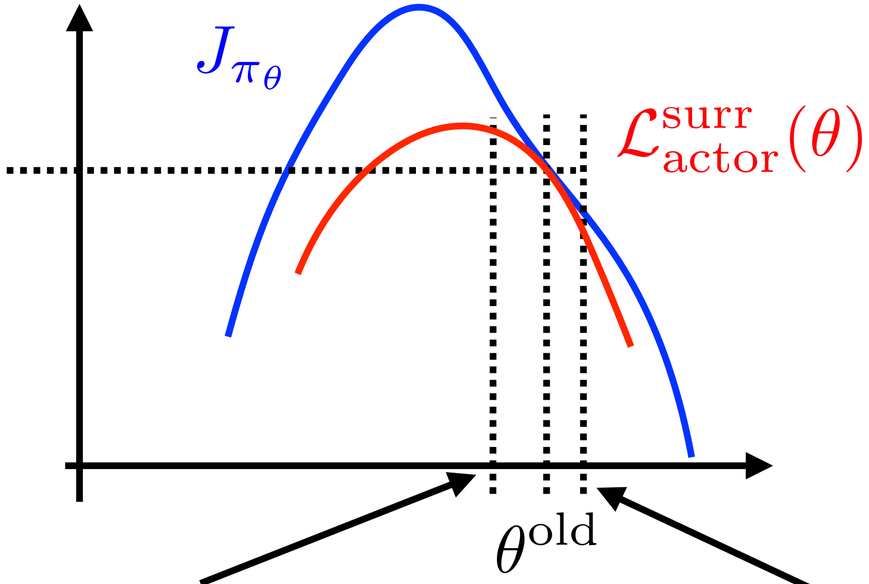


$$J_{\pi_\theta} = \mathbb{E}_{\tau\sim\pi_\theta}\{\sum_{r_t\sim\tau}\gamma^t r_t\} \geq \sum_{(\mathbf{u},\mathbf{x},\mathbf{x}')\in\tau} \frac{\pi_\theta(\mathbf{u}|\mathbf{x})}{\pi_{\theta^{\text{old}}}(\mathbf{u}|\mathbf{x})} \cdot \hat{A}_\omega(\mathbf{u},\mathbf{x}) = \mathcal{L}^{\text{surr}}_{\text{actor}}(\theta)$$

TRPO: $$\arg\max_\theta \; \mathbb{E}_{\tau\sim\pi_\theta}\left[\frac{\pi_\theta(\mathbf{u}|\mathbf{x})}{\pi_{\theta^{\text{old}}}(\mathbf{u}|\mathbf{x})} \cdot \hat{A}_\omega(\mathbf{u},\mathbf{x})\right]$$

subject to: $$\mathbb{E}_{\tau\sim\pi_\theta}\left[D_{\text{KL}}\left(\pi_\theta(\mathbf{u}|\mathbf{x}) \,\|\pi_{\theta^{\text{old}}}(\mathbf{u}|\mathbf{x})\right)\right] \leq \epsilon$$

Proximal Policy Optimization PPO

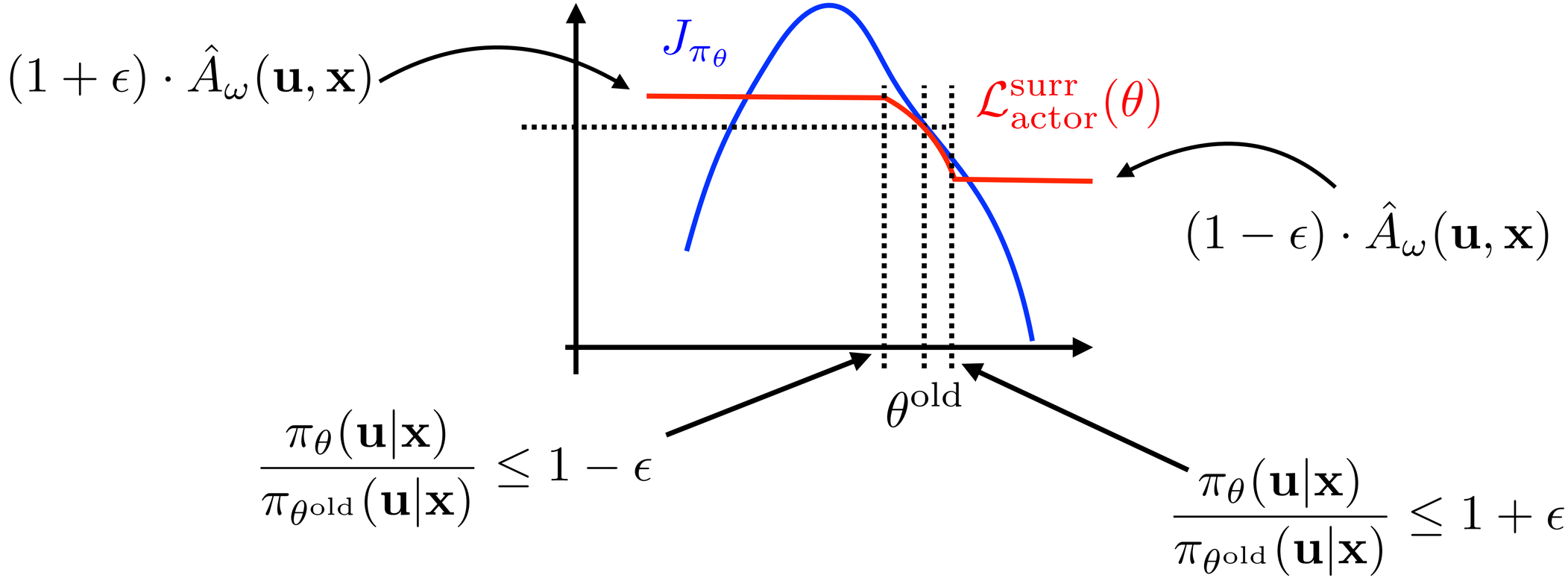


$$\frac{\pi_\theta(\mathbf{u}|\mathbf{x})}{\pi_{\theta^{\text{old}}}(\mathbf{u}|\mathbf{x})} \leq 1-\epsilon$$

$$\frac{\pi_\theta(\mathbf{u}|\mathbf{x})}{\pi_{\theta^{\text{old}}}(\mathbf{u}|\mathbf{x})} \leq 1+\epsilon$$

TRPO: $$\arg\max_\theta \; \mathbb{E}_{\tau\sim\pi_\theta}\left[\frac{\pi_\theta(\mathbf{u}|\mathbf{x})}{\pi_{\theta^{\text{old}}}(\mathbf{u}|\mathbf{x})}\cdot \hat{A}_\omega(\mathbf{u},\mathbf{x})\right]$$

Proximal Policy Optimization PPO



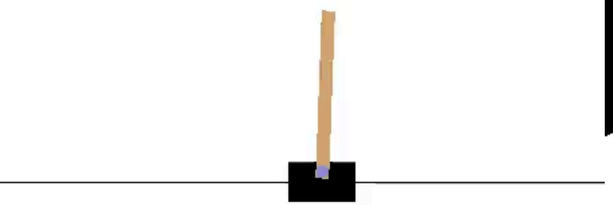
$$\arg\max_\theta \ \mathbb{E}_{\tau\sim\pi_\theta}\left[\min\left\{\frac{\pi_\theta(\mathbf{u}|\mathbf{x})}{\pi_{\theta^{\text{old}}}(\mathbf{u}|\mathbf{x})}\cdot\hat{A}_\omega(\mathbf{u},\mathbf{x}),\ \ \text{clip}\left(\frac{\pi_\theta(\mathbf{u}|\mathbf{x})}{\pi_{\theta^{\text{old}}}(\mathbf{u}|\mathbf{x})},1-\epsilon,1+\epsilon\right)\cdot\hat{A}_\omega(\mathbf{u},\mathbf{x})\right\}\right]$$

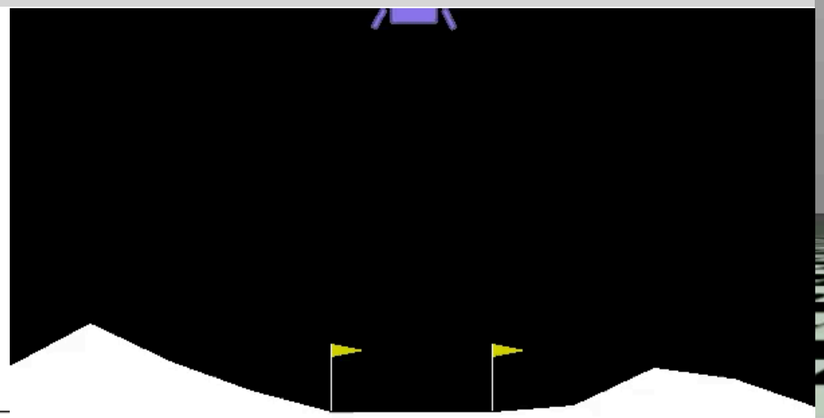
Reinforcement learning baselines
https://gym.openai.com/

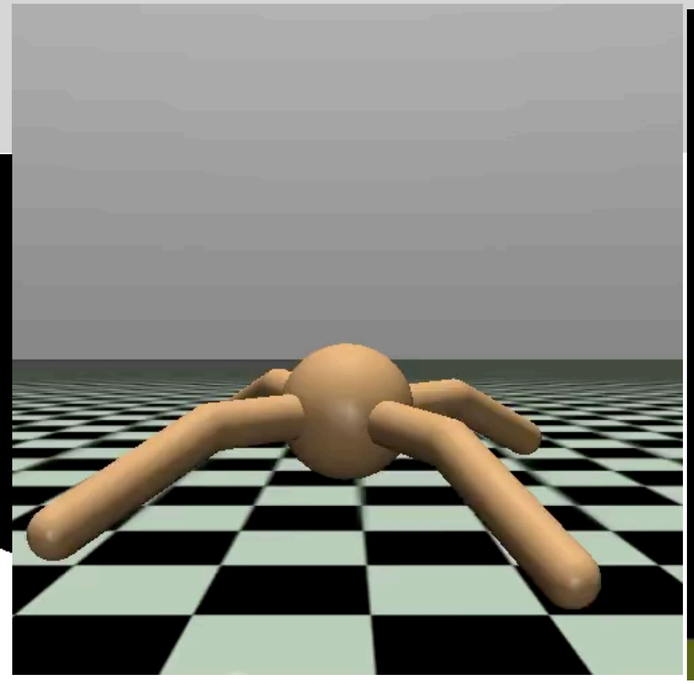
```
import gym

env = gym.make('CartPole-v1')

obs = env.reset()
for i in range(1000):
    action, _state = model.predict(obs, deterministic=True)
    obs, reward, done, info = env.step(action)
    env.render()
    if done:
      obs = env.reset()
```







Reinforcement learning baselines

https://stable-baselines3.readthedocs.io/

```
import gym

from stable_baselines3 import A2C

env = gym.make('CartPole-v1')

model = A2C('MlpPolicy', env, verbose=1)
model.learn(total_timesteps=10000)
```

Known successes of RL

- Computer games controlled from pixel inputs
  - Starcraft II (AlphaStar)
  - Atari 2D platformers (DQN)
  - Doom 2 - VizDoom [Wydemuch 2018] https://arxiv.org/abs/1809.03470
  - Quake III - Arena capture the flag
  - DOTA 2 openAI+ bot https://blog.openai.com/dota-2/

Known successes of RL - Starcraft II

- Starcraft II (Deepmind AlphaStart beaten top-end professional human gamers 5:0)

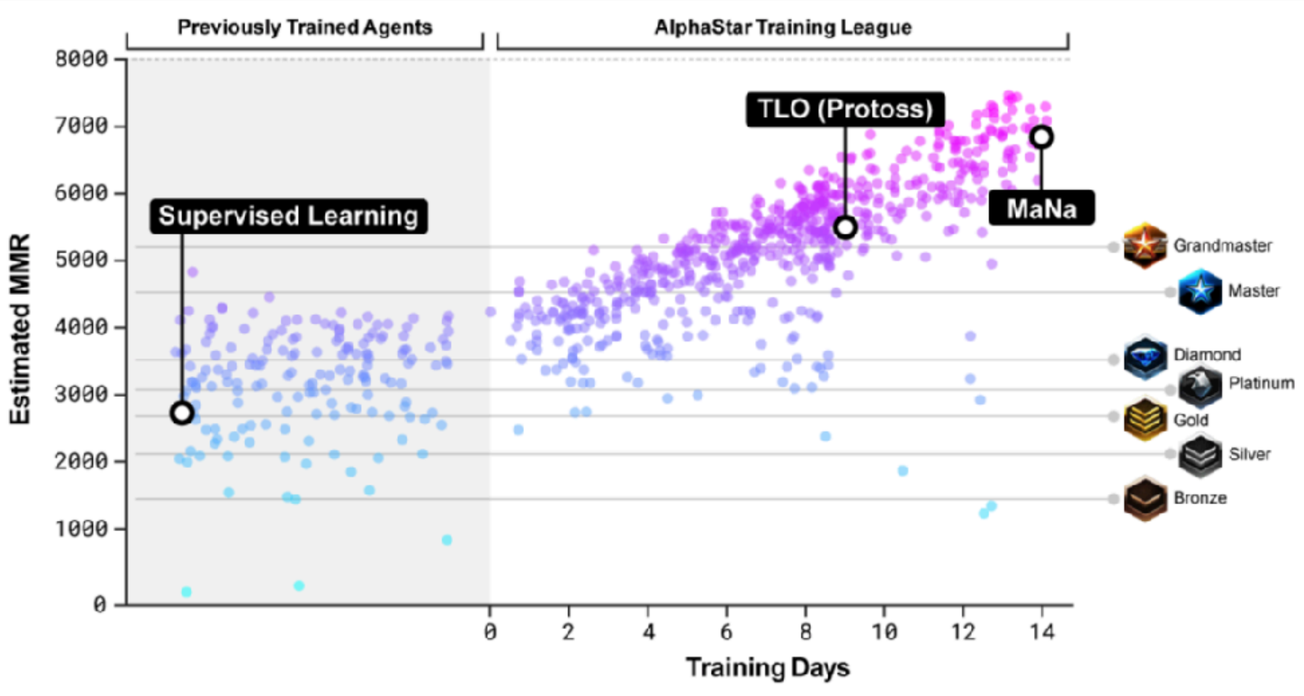
- no single best strategy
- partially observable
- longterm planning (delayed rewards from upgrades)
- realtime
- large action space
- human gameplay dataset
- more-than-realtime implementation of the game for RL

https://medium.com/mlmemoirs/deepminds-ai-alphastar-showcases-significant-progress-towards-agi-93810c94fbe9

Known successes of RL - Starcraft II
Minigames allows for traing small RL agents

Learning Starcraft II

- **Supervised learning** from human games dataset (performance: (i) humans - gold level, (ii) AI - elite level
- **Reinforcement learning**:
- minigames (collect minerals, build marines, …)
- self-play
- 14 days playing against two grand masters (TLO, MaNa)



Known successes of RL - Starcraft II

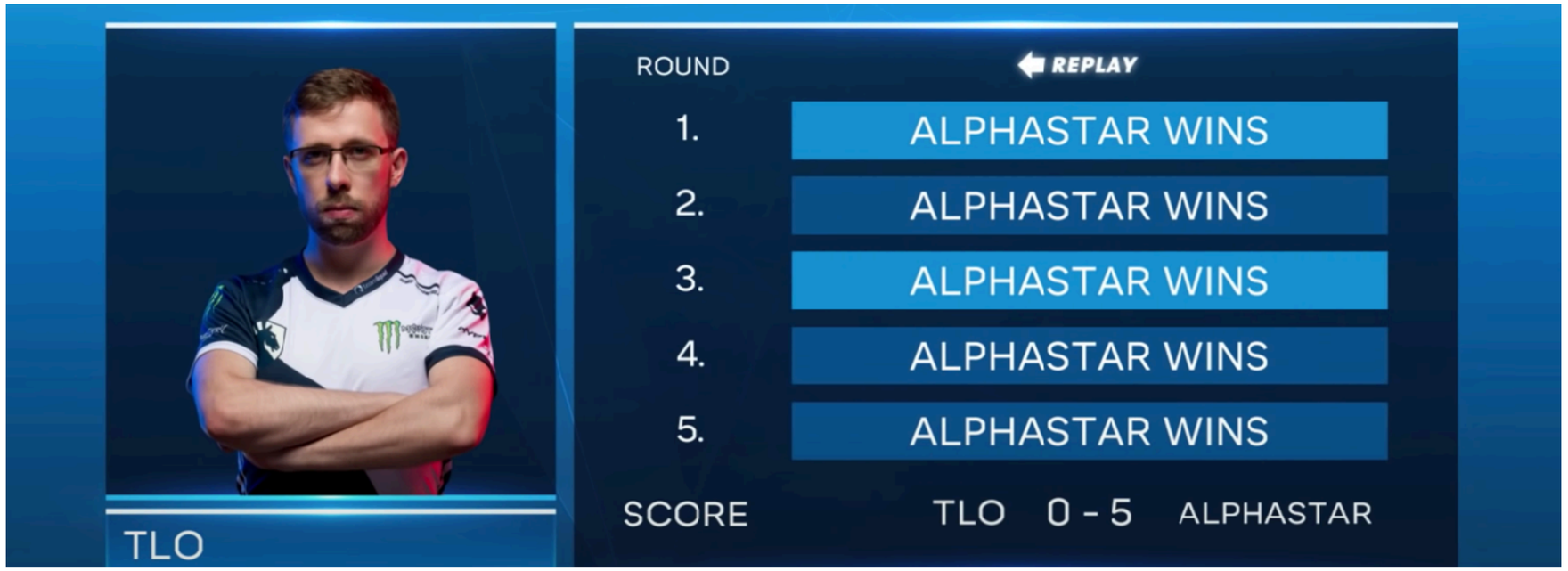
- **Reinforcement learning**: 14 days playing against two grand masters (TLO, MaNa)
  - Distributed Actor-Critic method IMPALA: https://arxiv.org/pdf/1802.01561.pdf
  - $TD(\lambda)$learning of $Q_\theta(\mathbf{x}, \mathbf{u})$
  - stochastic policy gradient:

$$\mathbb{E}_{\tau \sim p(\tau|\pi_\omega)} \left[\sum_{(\mathbf{x},\mathbf{u}) \in \tau} \frac{\partial \log \pi_\omega(\mathbf{u}|\mathbf{x})}{\partial \omega} \cdot Q_\theta(\mathbf{x}, \mathbf{u}) \right] \approx$$

$$\approx \sum_k \frac{\partial \log \pi_\omega(\mathbf{u}_k|\mathbf{x}_k)}{\partial \omega} \cdot Q_\theta(\mathbf{x}_k, \mathbf{u}_k)$$

  - recurrent policy architecture with LSTM blocks
  - paralelized learning

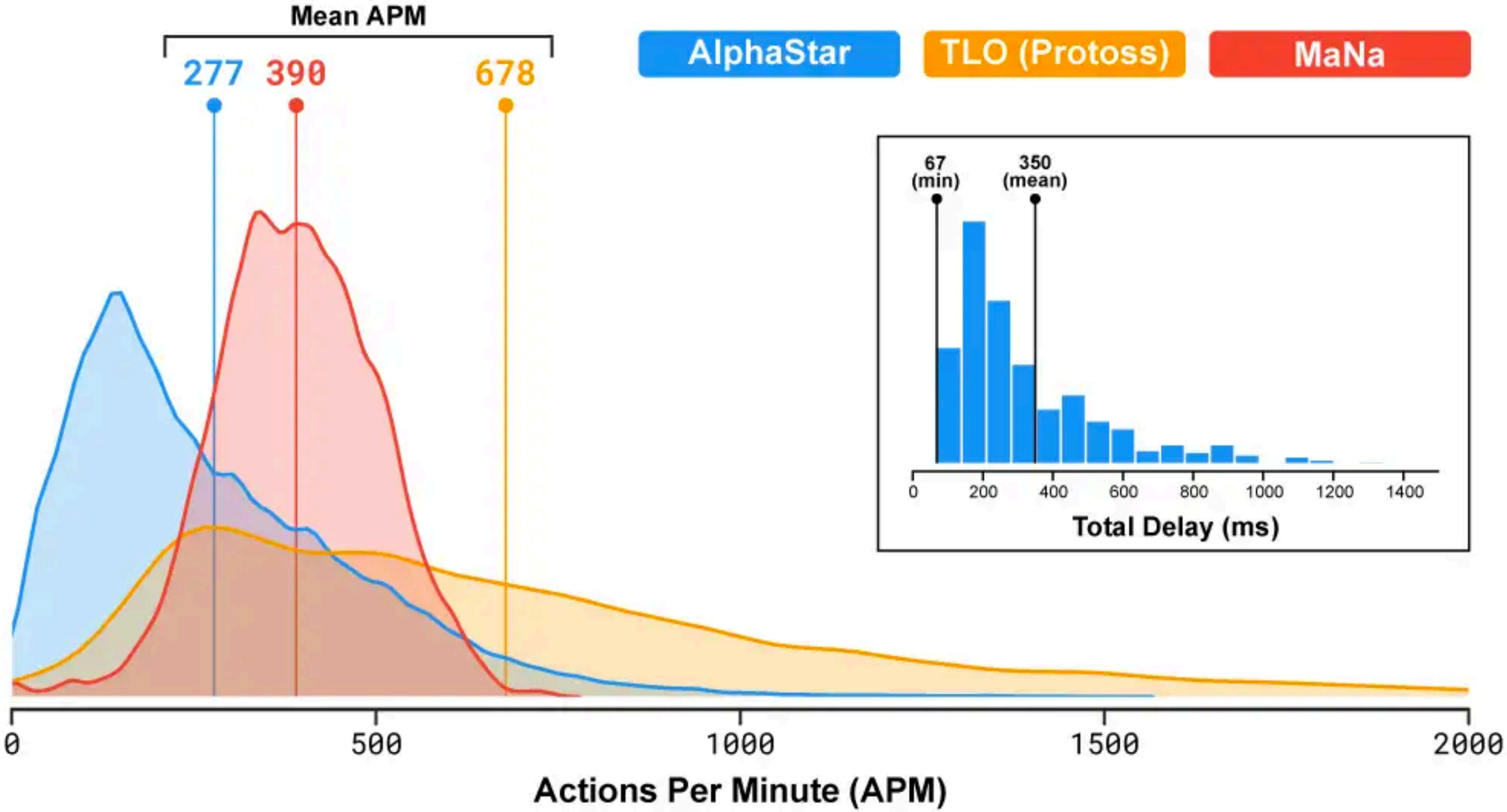
Results:



Suggested explanations:

- higher APM
- smaller delays in counter-actions
- does not have to move camera
- fog of war

- ~~higher APM~~

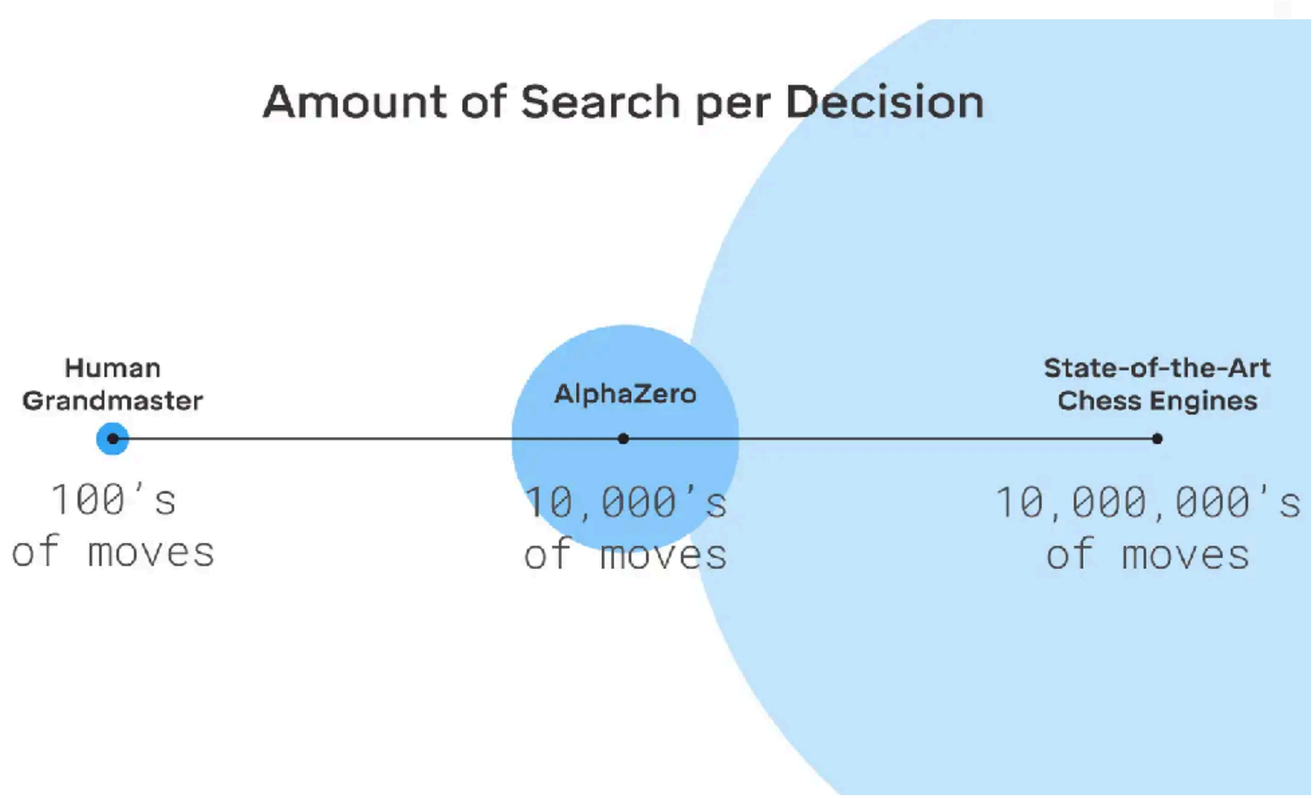

Mean APM
277
390
678
AlphaStar
TLO (Protoss)
MaNa
67
(min)
350
(mean)
0
200
400
600
800
1000
1200
1400
Total Delay (ms)
0
500
1000
1500
2000
Actions Per Minute (APM)

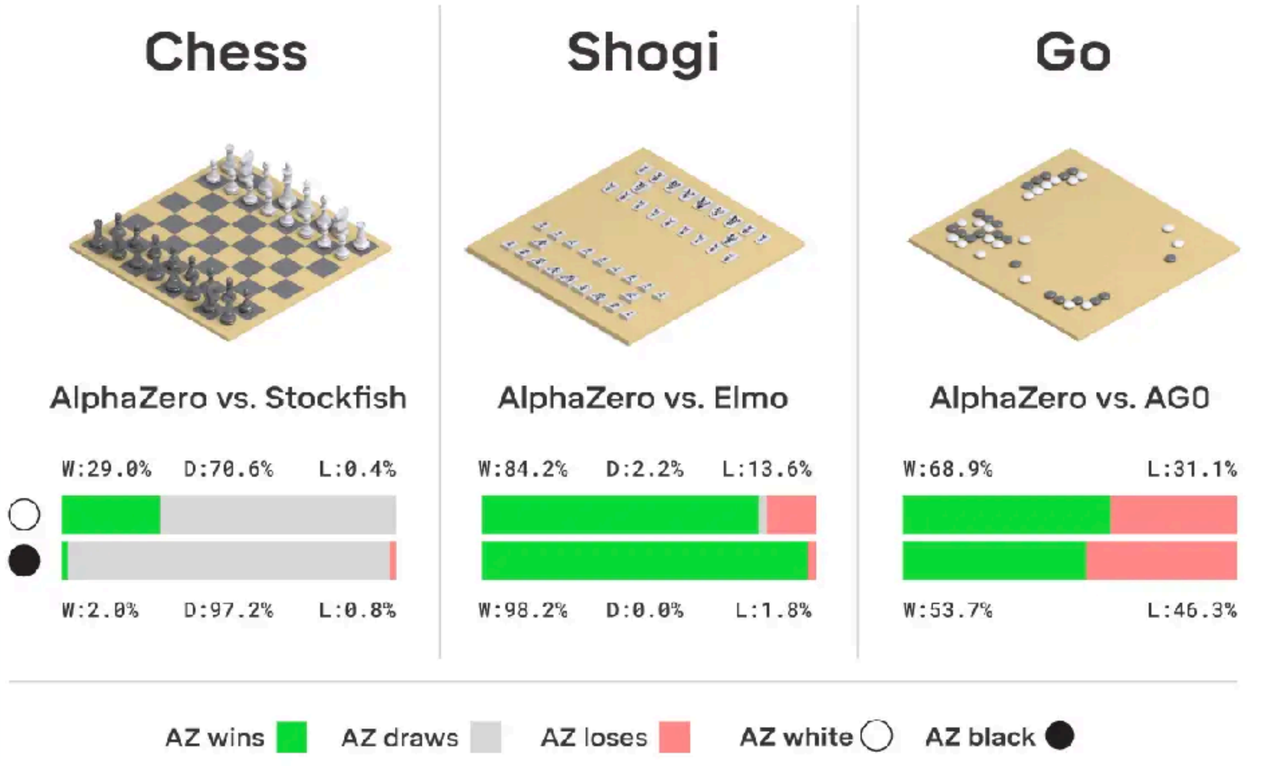
- ~~moving interface~~

- Of course, haze of war is used.

Known successes of RL - board games

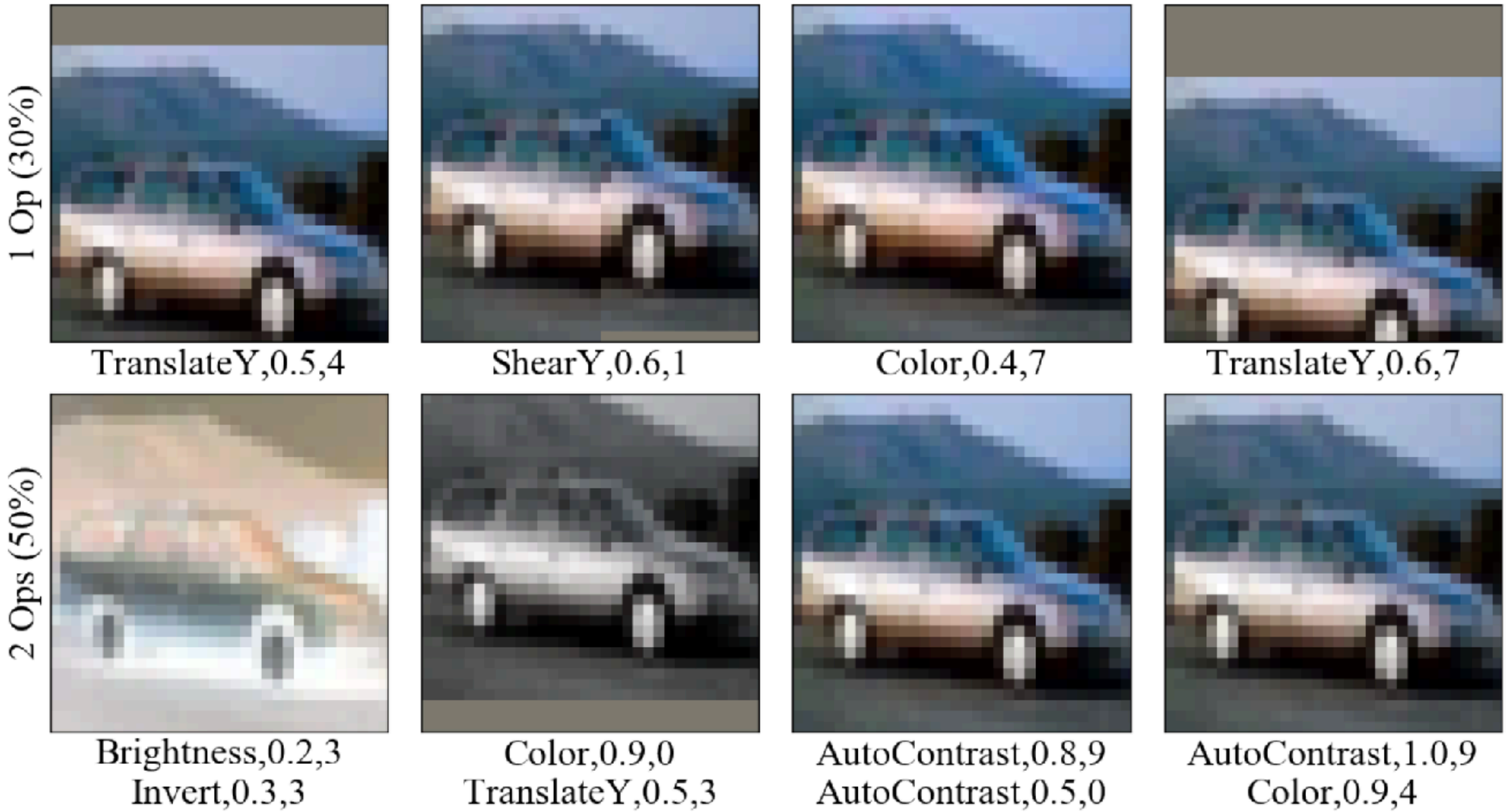
- Brute-force search-based algorithms has no chance in huge state-action spaces => trained net guides the search efficiently





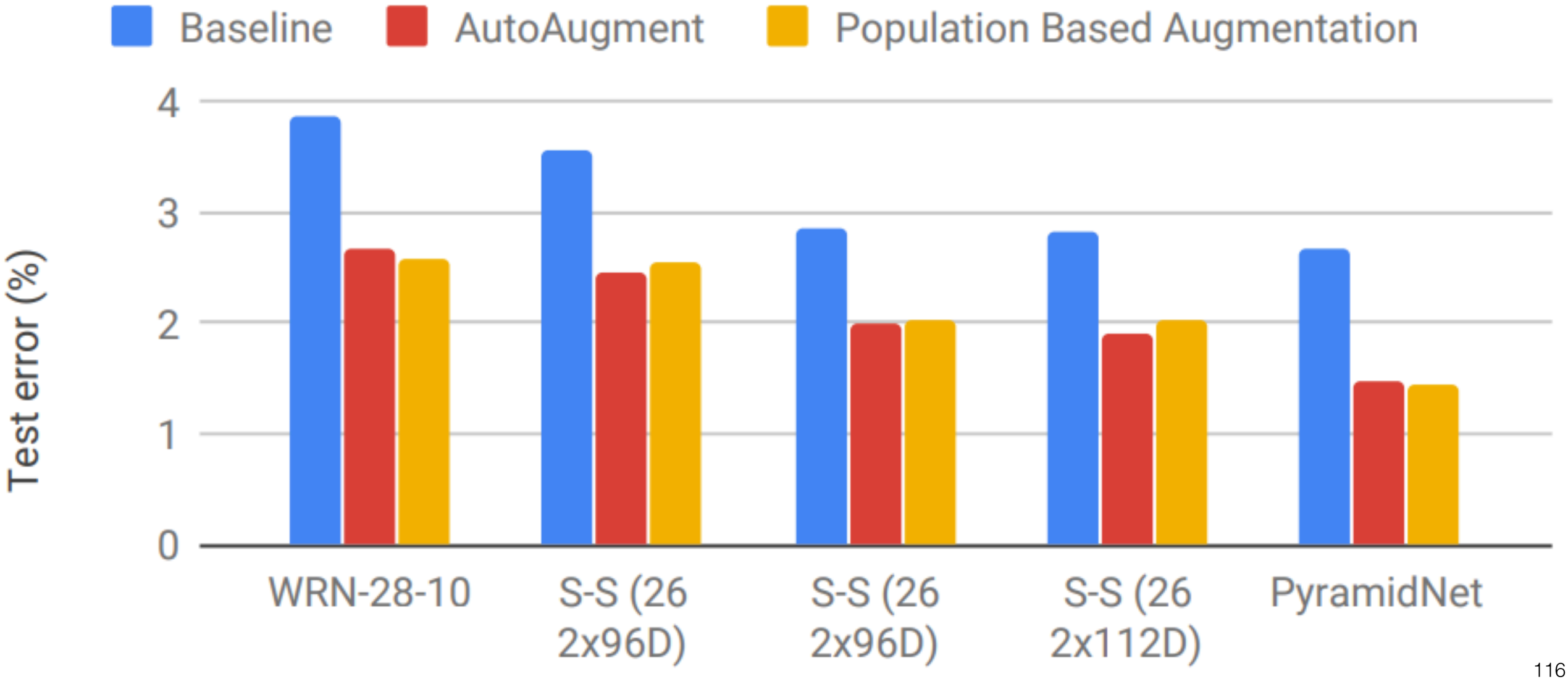
Known successes of RL
Learning to learn

- Training set augmentation
  (jittering, mirroring, occlusions, brightness/contrast/color variations)
- Learn augmentation policy (AutoAugment, PBA), which provides good generalization



https://arxiv.org/pdf/1905.05393.pdf

Known successes of RL - Learning to learn

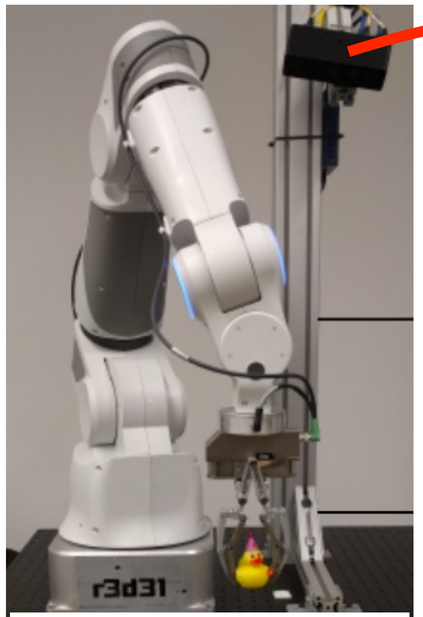


Known successes of RL - learning complex motions in simulation

[Heess 2017] https://arxiv.org/abs/1707.02286

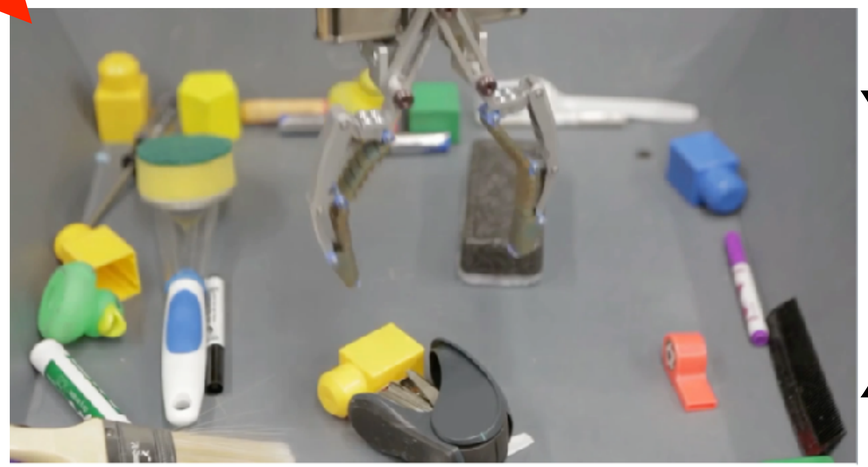
Known successes of RL
Learning complex motions in reality by paralelizing and automatizing rewards

manipulator+ RGB camera



image

$$= \pi_\theta($$



$$)$$

joint torques

Continues motion control from RGB(D)

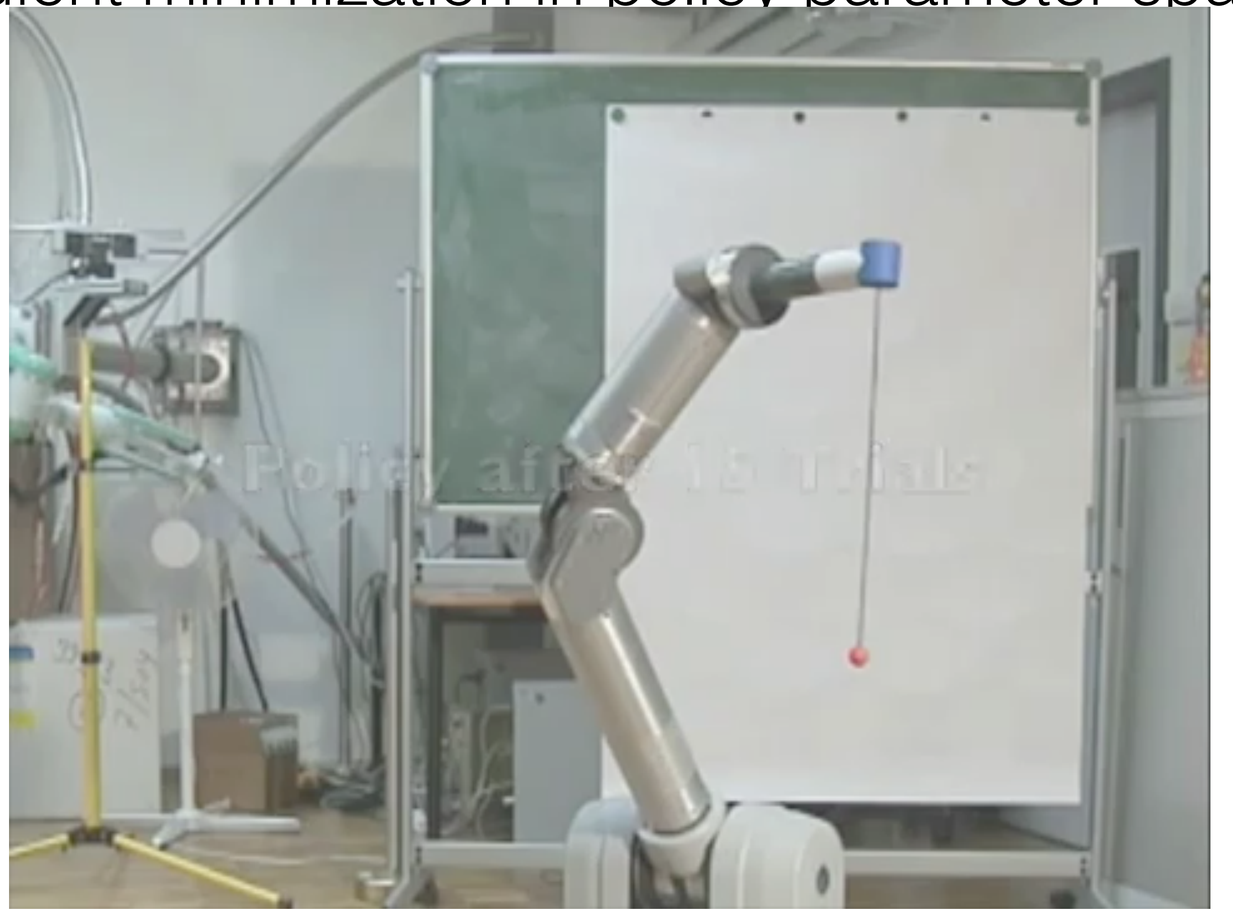
[Levine IJRR 2017] https://arxiv.org/abs/1603.02199


Source: Peter Pastor
r3d11
r3d24
r3d32

Known successes of RL

learning complex motions in reality by manually designing low-dim policy

- imitation learning from human demonstration
- **state space:** joint+ball positions, velocities, acceler.
- **action space:** motor torques
- gradient minimization in policy parameter space



Peters et al. NOW 2013

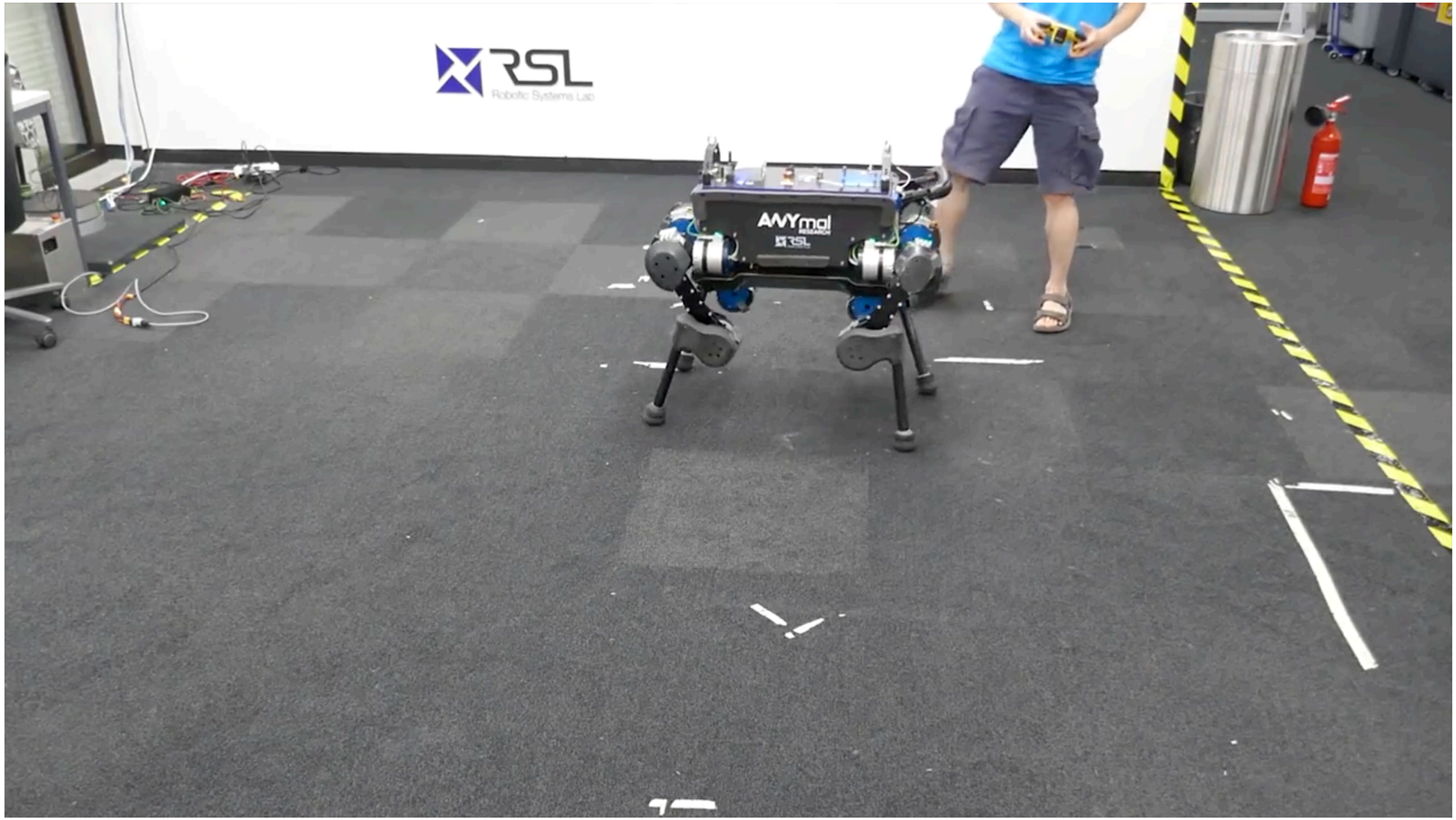
Motion and compliance control of flippers

[3] Pecka, Zimmermann, Svoboda, et al. **IROS/RAL/TIE(IF=6)**, 2015-2018

Known successes of RL

learning complex motions in reality by transfering policy from simulation

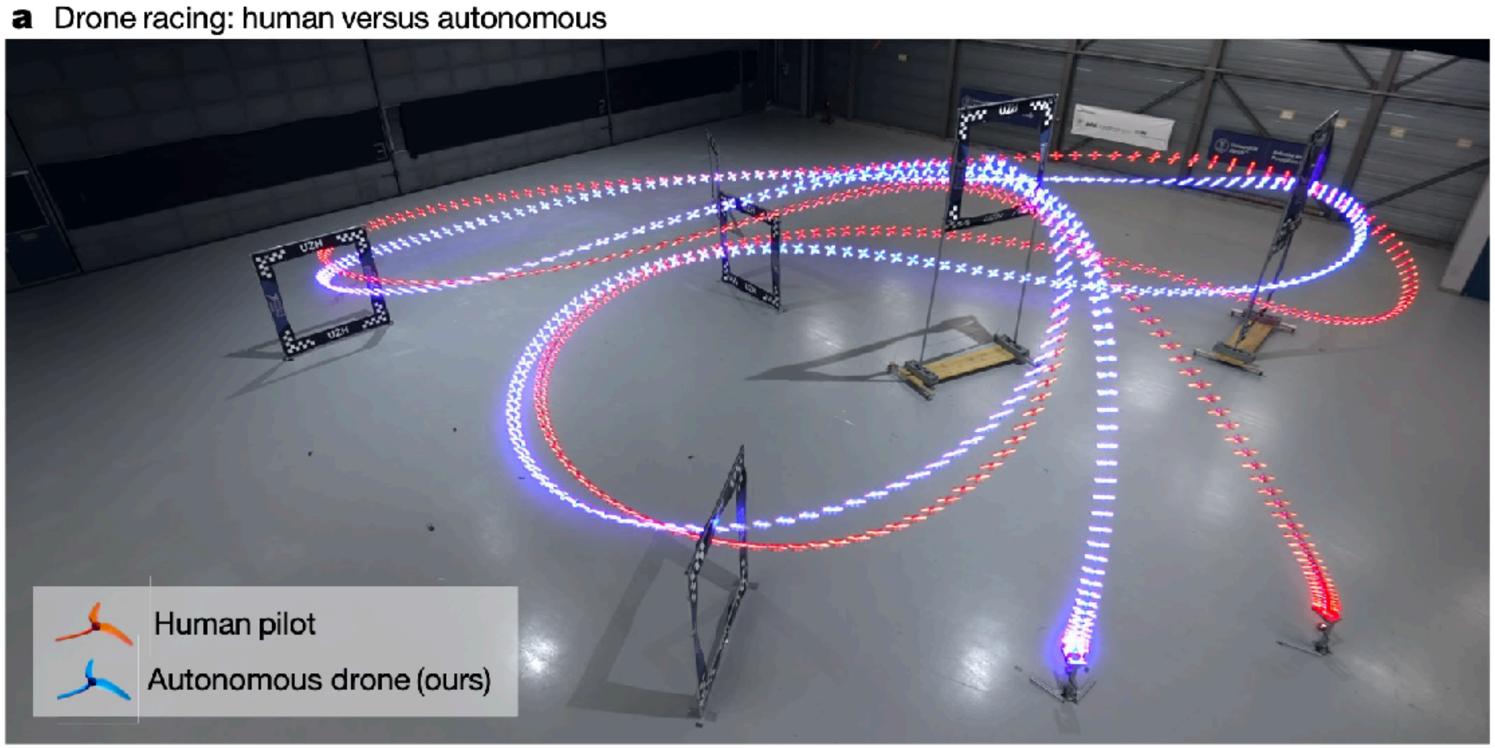
No visual inputs + flat terrain => simple domain transfer

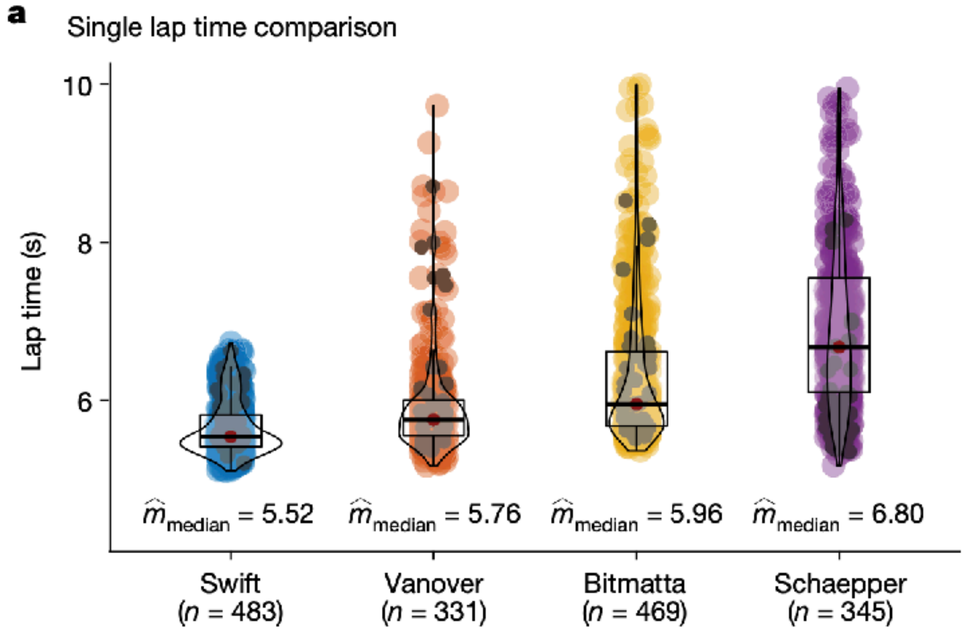


[Hwangbo, ETH Zurich, Science Robotics, 2018]

[Kumar 2020] Rapid Motor Adaptation for legged robot
https://ashish-kmr.github.io/rma-legged-robots/

[Kaufman Nature 2023] Champion-level drone racing using deep RL





- Simulation+residual model
- RL runs in simulation
- Method = PPO

[Kaufman Nature 2023] Champion-level drone racing using deep RL

Boston dynamics - Big dog - NO RL AT ALL

Typical problems

τ

Model identification:
- given some trajectories
  estimate model

$p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

$p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}')$

Model predictive control / Planning
- given the model and reward
  estimate optimal policy/plan

$\pi^* = \arg\max_{\pi} J_\pi$

τ

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}')$

Reinforcement learning:
- given rewards and trajectories,
  estimate optimal policy

$\pi^* = \arg\max_{\pi} J_\pi$

τ^*

Inverse reinforcement learning:
- given optimal trajectories
  estimate reward function

$r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$

Rewards engineering

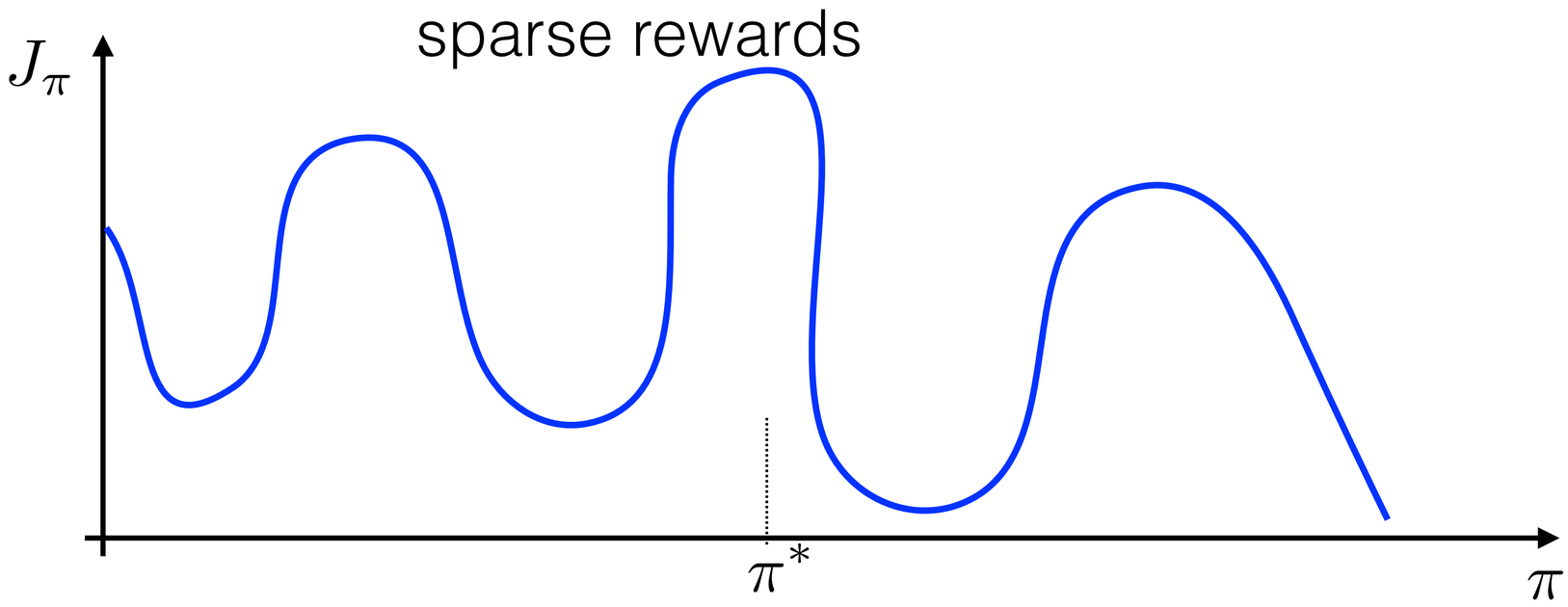
- Designing RL-friendly rewards is hard
- Sparse rewards are easier to design correctly (current speed)
- Dense rewards are easier to learn (winning the race)

Rewards engineering

- Sparse rewards are easier to design correctly
- Dense rewards are easier to learn

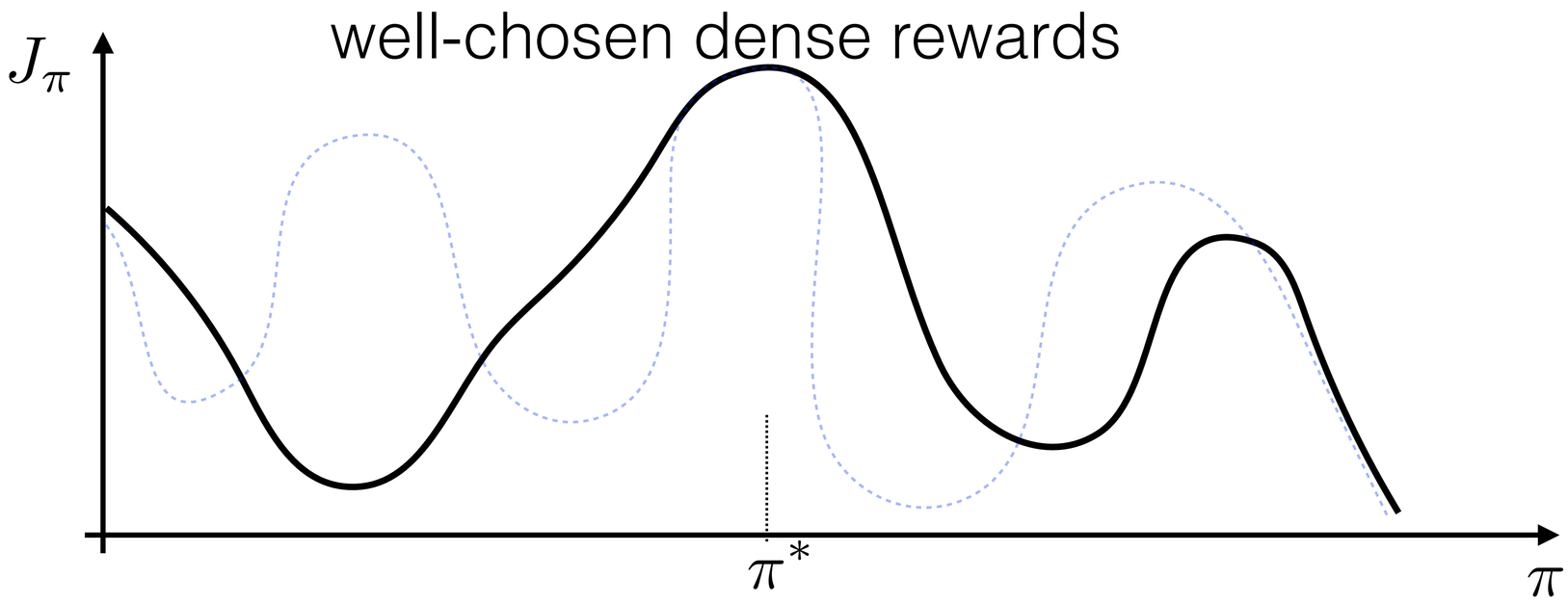
Rewards engineering

- Sparse rewards are easier to design correctly
- Dense rewards are easier to learn



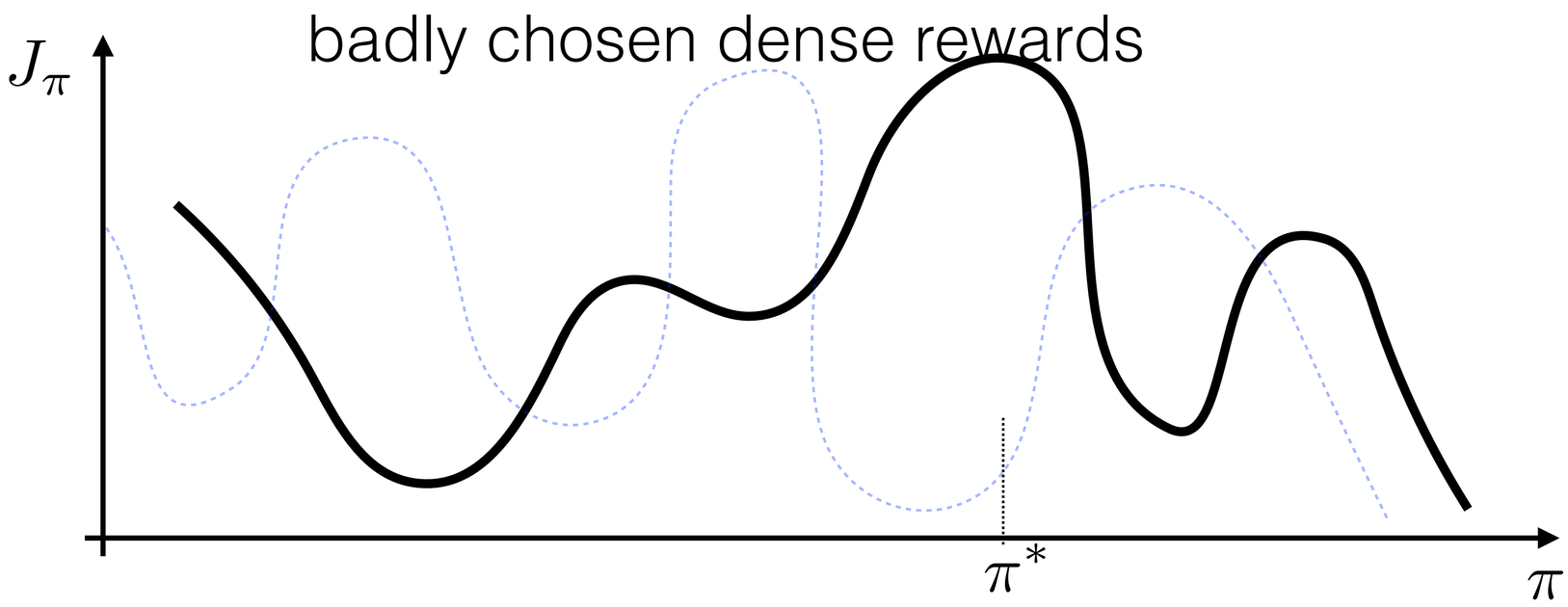
Rewards engineering

- Sparse rewards are easier to design correctly
- Dense rewards are easier to learn



Rewards engineering

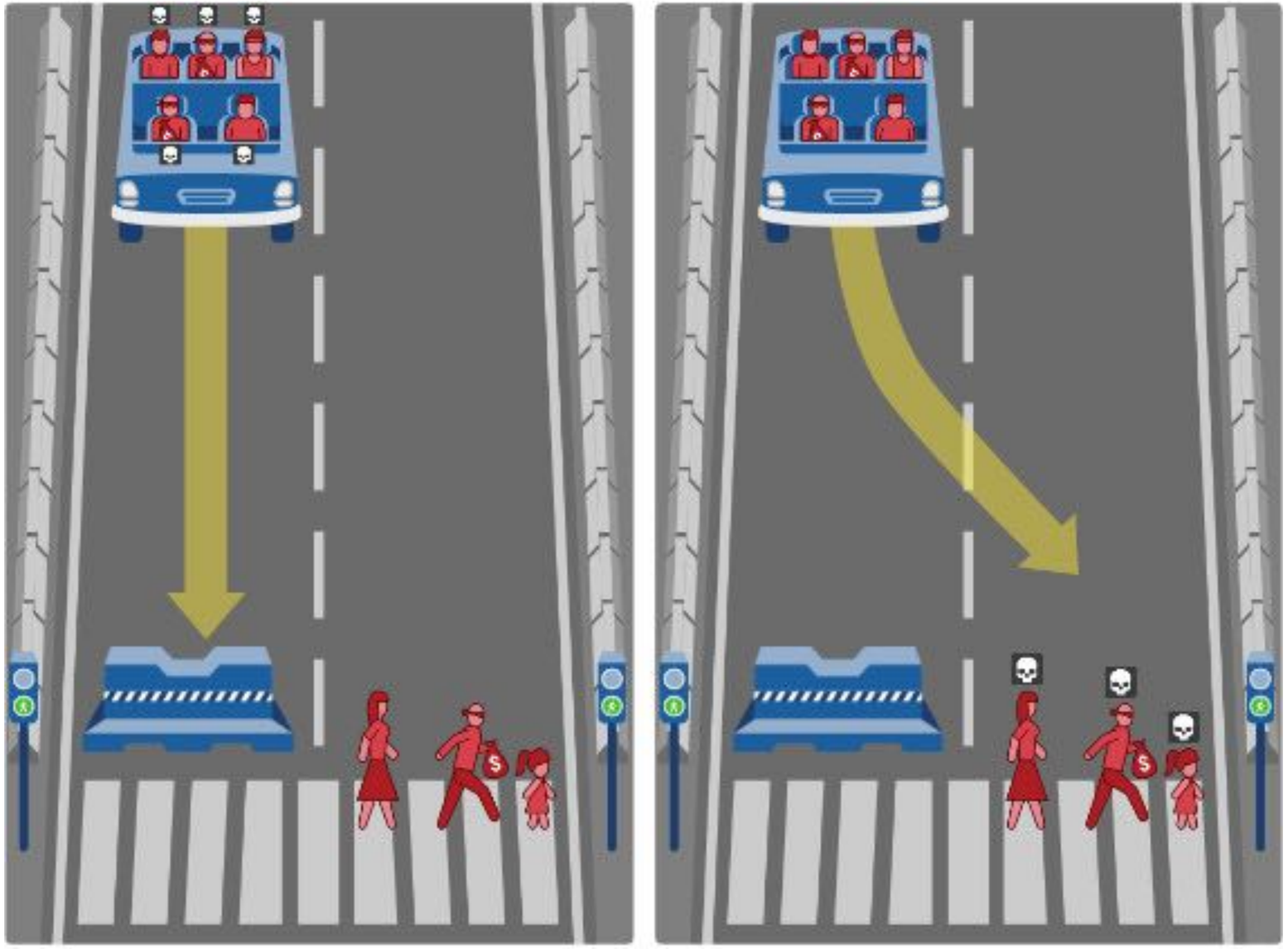
- Sparse rewards are easier to design correctly
- Dense rewards are easier to learn



Rewards engineering

- Boat racing (bad dense rewards):
  - sparse rewards (winning the race)
  - dense rewards (collecting powerups, checkpoints …)

Trolley problem



https://www.nature.com/articles/s41586-018-0637-6
[Moral Machine Experiment, Nature, 2018]

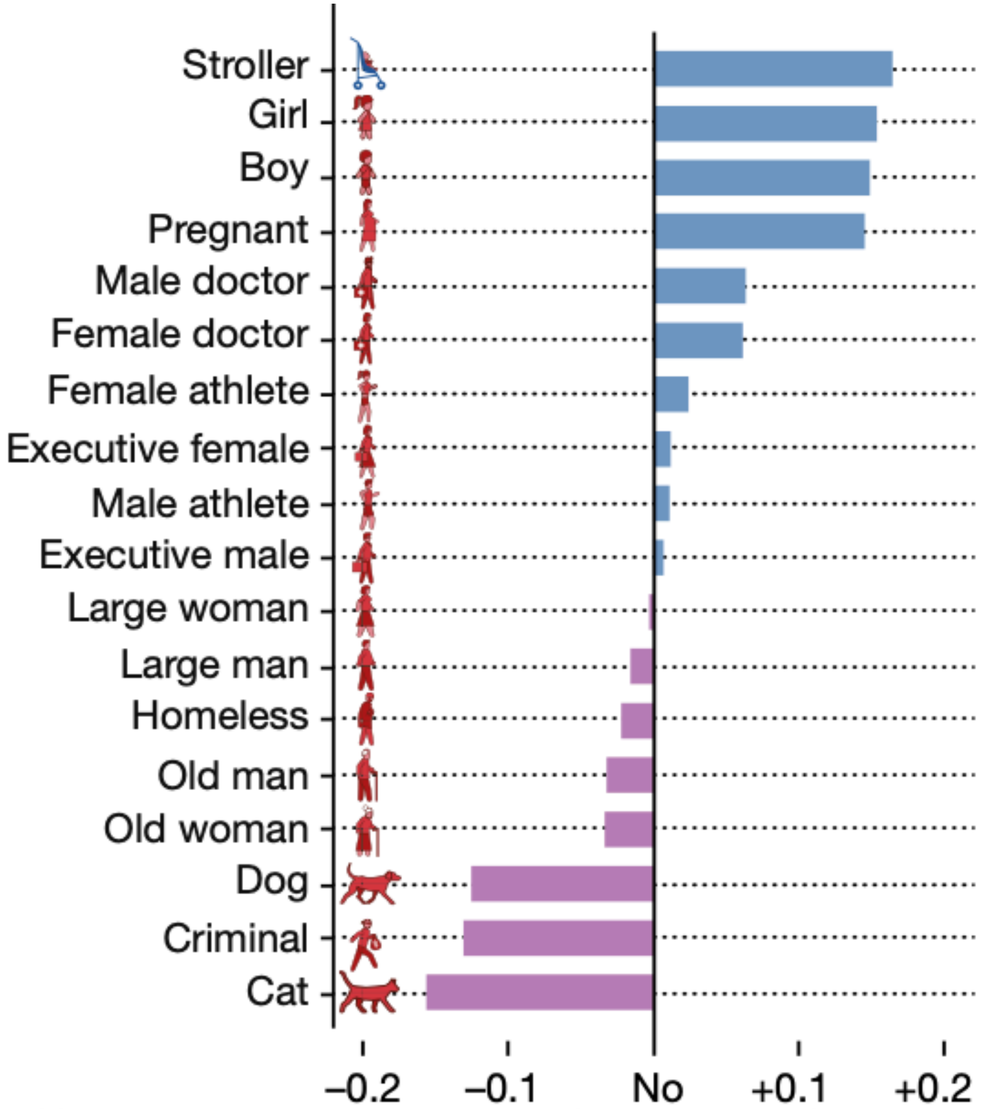
Trolley problem
estimated preference (normalized rewards) for life saving

Male, Female, Cat, Dog, Criminal, Boy, Girl, Stroller, Homeless, …

Guess your own preferences !

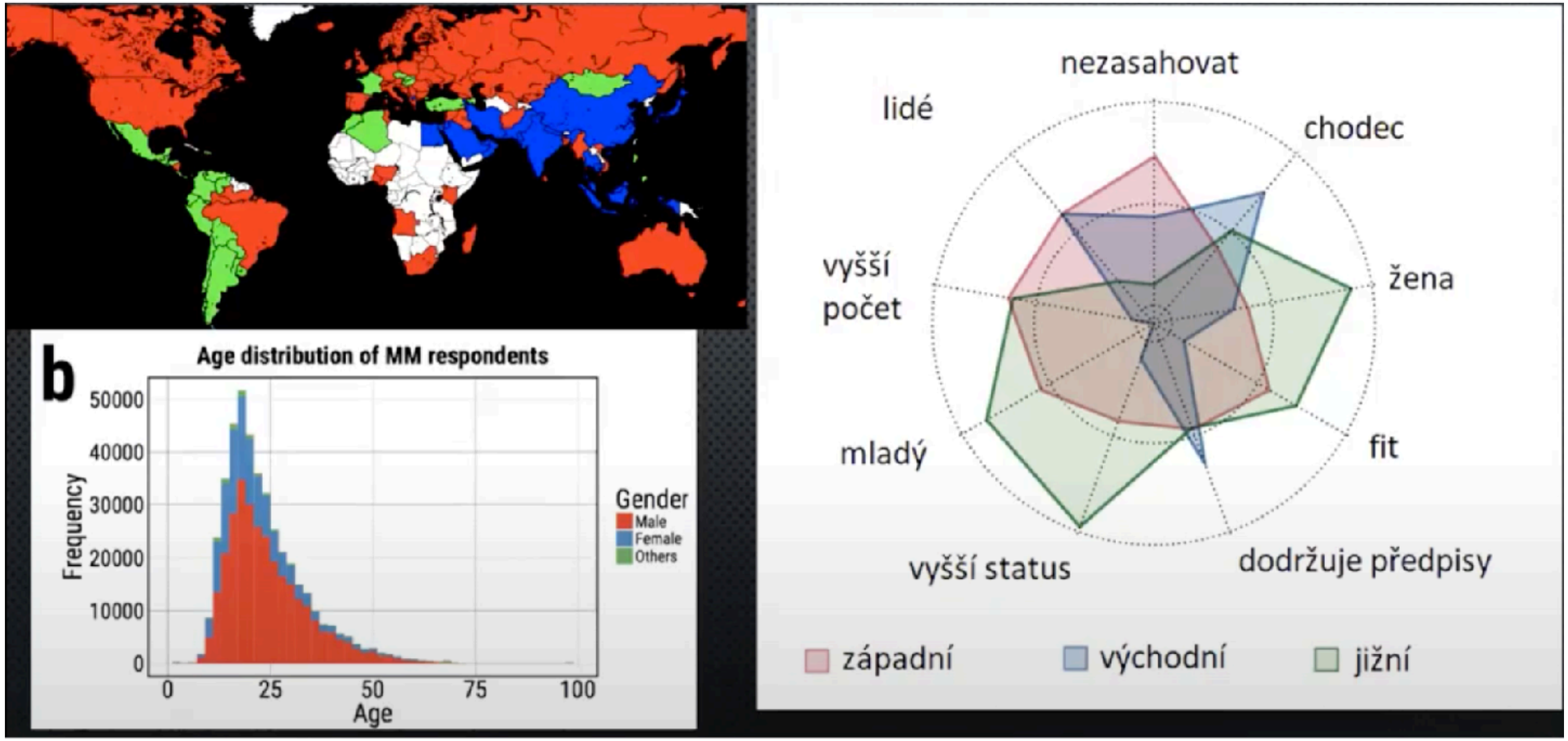
https://www.nature.com/articles/s41586-018-0637-6
[Moral Machine Experiment, Nature, 2018]

Trolley problem
estimated preference (normalized rewards) for life saving



https://www.nature.com/articles/s41586-018-0637-6
[Moral Machine Experiment, Nature, 2018]

Trolley problem
spatial distribution of life-saving preferences



https://www.moralmachine.net

https://www.nature.com/articles/s41586-018-0637-6
[Moral Machine Experiment, Nature, 2018]

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.
- Imitation learning setup

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.
- Imitation learning setup
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find policy $\arg\min_\theta \sum_{(\mathbf{x}_i, \mathbf{a}_i) \in \tau^*} \|\pi_\theta(\mathbf{x}_i) - \mathbf{a}_i\|_2^2$

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.
- Imitation learning setup (statistically inconsistent+ blackbox)
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find policy $\arg\min_\theta \sum_{(\mathbf{x}_i, \mathbf{a}_i) \in \tau^*} \|\pi_\theta(\mathbf{x}_i) - \mathbf{a}_i\|_2^2$
- Inverse reinforcement learning setup

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.
- Imitation learning setup (statistically inconsistent+ blackbox)
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find policy $\arg\min_\theta \sum_{(\mathbf{x}_i, \mathbf{a}_i) \in \tau^*} \|\pi_\theta(\mathbf{x}_i) - \mathbf{a}_i\|_2^2$
- Inverse reinforcement learning setup
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find reward function $r_{\mathbf{w}}$

$$\arg\min_{\mathbf{w}} \|\mathbf{w}\|_2^2$$

$$\text{subject to:} \sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \{\mathcal{T} \setminus \tau^*\}} r_{\mathbf{w}}(\mathbf{x}, \mathbf{u}, \mathbf{x}') \leq \sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \tau^*} r_{\mathbf{w}}(\mathbf{x}, \mathbf{u}, \mathbf{x}')$$

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.
- Imitation learning setup (statistically inconsistent+ blackbox)
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find policy $\arg\min_\theta \sum_{(\mathbf{x}_i, \mathbf{a}_i) \in \tau^*} \|\pi_\theta(\mathbf{x}_i) - \mathbf{a}_i\|_2^2$
- Inverse reinforcement learning setup
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find reward function $r_{\mathbf{w}}$

$$\arg\min_{\mathbf{w}} \|\mathbf{w}\|_2^2$$

$$\text{subject to: } \text{ReLU}\Big(\sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \{\mathcal{T} \setminus \tau^*\}} r_{\mathbf{w}}(\mathbf{x},\mathbf{u},\mathbf{x}') - \sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \tau^*} r_{\mathbf{w}}(\mathbf{x},\mathbf{u},\mathbf{x}') \Big) = 0$$

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.
- Imitation learning setup (statistically inconsistent+ blackbox)
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find policy $\arg\min_\theta \sum_{(\mathbf{x}_i, \mathbf{a}_i) \in \tau^*} \|\pi_\theta(\mathbf{x}_i) - \mathbf{a}_i\|_2^2$
- Inverse reinforcement learning setup
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find reward function $r_{\mathbf{w}}$

$$\arg\min_{\mathbf{w}} \|\mathbf{w}\|_2^2 + \mathrm{ReLU}\Big(\sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \{\mathcal{T} \setminus \tau^*\}} r_{\mathbf{w}}(\mathbf{x},\mathbf{u},\mathbf{x}') - \sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \tau^*} r_{\mathbf{w}}(\mathbf{x},\mathbf{u},\mathbf{x}') \Big)$$

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.
- Imitation learning setup (statistically inconsistent+ blackbox)
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find policy $\arg\min_\theta \sum_{(\mathbf{x}_i, \mathbf{a}_i) \in \tau^*} \|\pi_\theta(\mathbf{x}_i) - \mathbf{a}_i\|_2^2$
- Inverse reinforcement learning setup
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find reward function $r_{\mathbf{w}}$

$$\arg\min_{\mathbf{w}} \|\mathbf{w}\|_2^2 + \mathrm{ReLU}\Big(\sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \{\mathcal{T} \setminus \tau^*\}} r_{\mathbf{w}}(\mathbf{x},\mathbf{u},\mathbf{x}') - \sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \tau^*} r_{\mathbf{w}}(\mathbf{x},\mathbf{u},\mathbf{x}') \Big)$$

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.
- Imitation learning setup (statistically inconsistent+ blackbox)
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find policy $\arg\min_\theta \sum_{(\mathbf{x}_i, \mathbf{a}_i) \in \tau^*} \|\pi_\theta(\mathbf{x}_i) - \mathbf{a}_i\|_2^2$
- Inverse reinforcement learning setup
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find reward function $r_{\mathbf{w}}$

$$\arg\min_{\mathbf{w}} \|\mathbf{w}\|_2^2 + \mathrm{ReLU}\left(\sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \tau^{\mathrm{best}}} r_{\mathbf{w}}(\mathbf{x},\mathbf{u},\mathbf{x}') - \sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \tau^*} r_{\mathbf{w}}(\mathbf{x},\mathbf{u},\mathbf{x}')\right)$$

Learning from expert demonstrations

- Sometimes easier to provide good trajectories than good rewards.
- Imitation learning setup (statistically inconsistent+ blackbox)
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find policy $\arg\min_\theta \sum_{(\mathbf{x}_i, \mathbf{a}_i) \in \tau^*} \|\pi_\theta(\mathbf{x}_i) - \mathbf{a}_i\|_2^2$
- Inverse reinforcement learning setup
  1. Collect expert trajectories $\tau_1^*, \tau_2^*, \tau_3^*, \ldots$
  2. Find reward function $r_{\mathbf{w}}$

$$\arg\min_{\mathbf{w}} \|\mathbf{w}\|_2^2 + \mathrm{ReLU}\left(\sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \tau^{\mathrm{best}}} r_{\mathbf{w}}(\mathbf{x},\mathbf{u},\mathbf{x}') - \sum_{(\mathbf{x},\mathbf{u},\mathbf{x}') \in \tau^*} r_{\mathbf{w}}(\mathbf{x},\mathbf{u},\mathbf{x}')\right)$$

  3. Solve underlying RL/control task

Abbeel et al. IJRR 2010

- inverse reinforcement learning
- **state space:** angular and euclidean position, velocity, acceleration
- **action space:** motor torques
- learning reward function from expert pilot

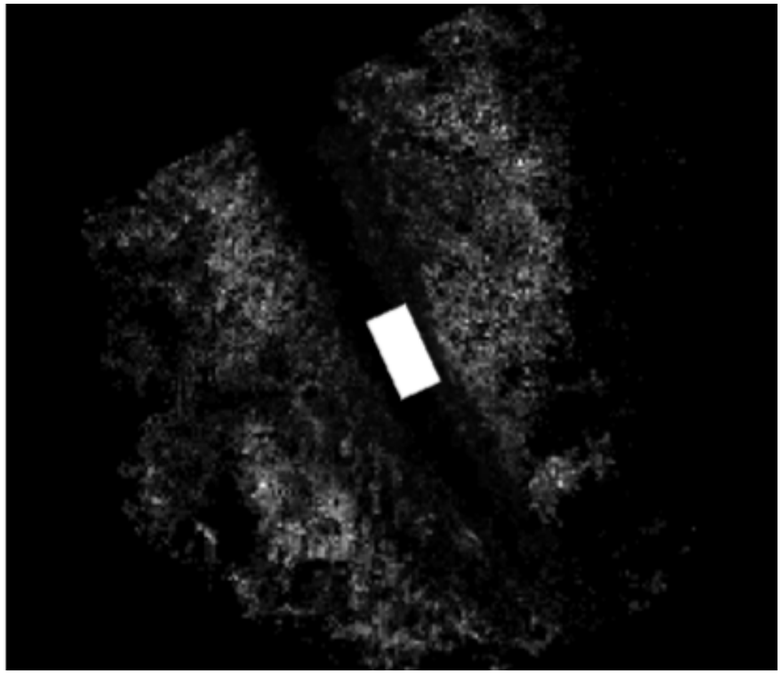
Abbeel et al. IJRR 2010

Silver et al. IJRR 2010

Similar to recent DARPA RACER
http://www.dtic.mil/dtic/tr/fulltext/u2/a525288.pdf

Silver et al. IJRR 2010

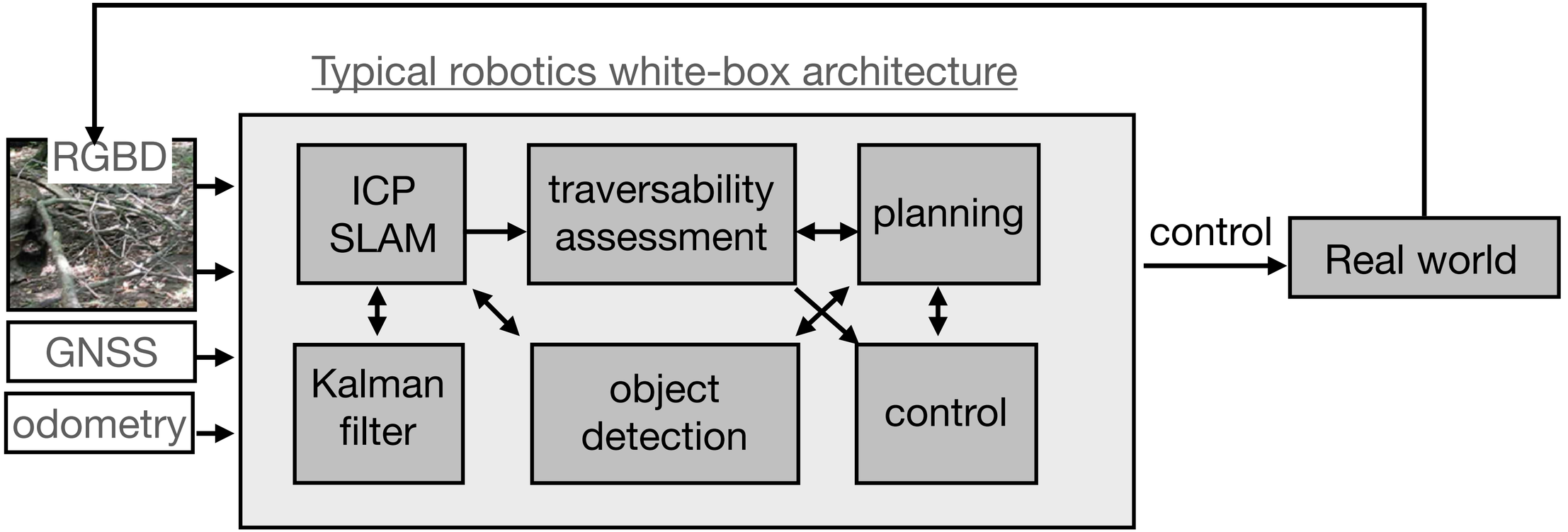
input image (state)



learned reward function
(traversability map)

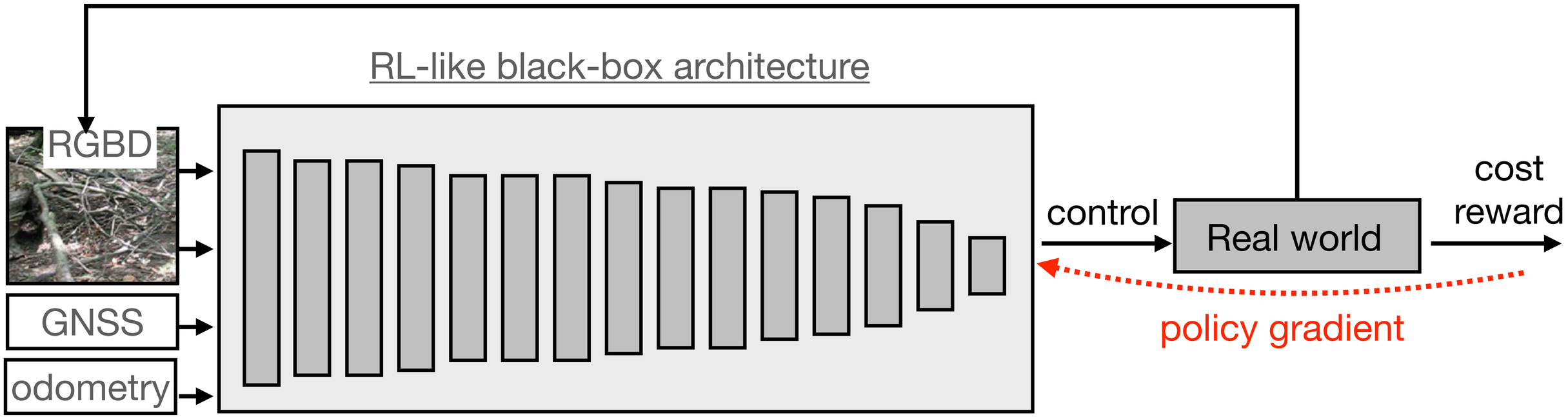
http://www.dtic.mil/dtic/tr/fulltext/u2/a525288.pdf

Typical robotics white-box architecture



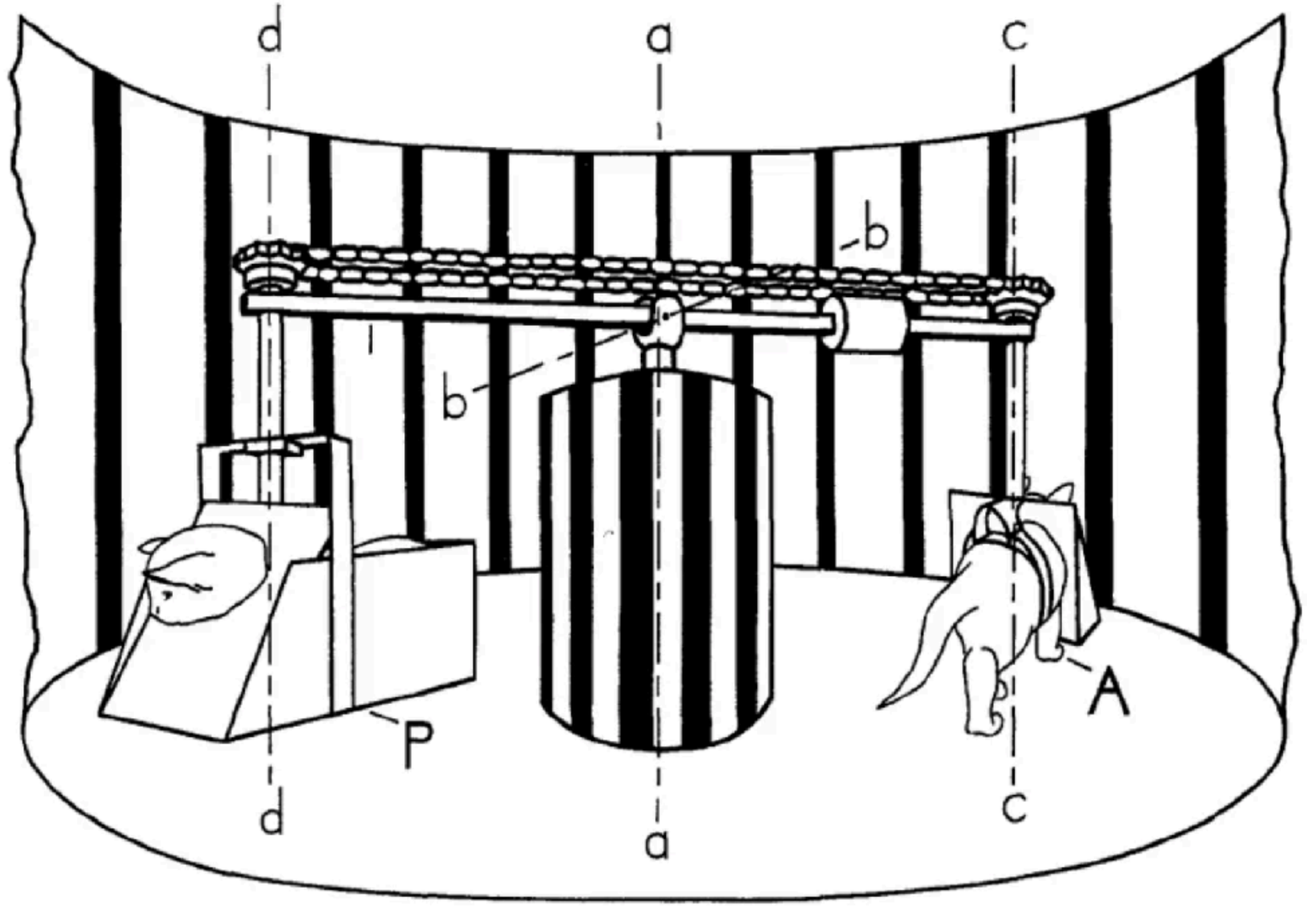
White-box architecture

- Explainable / interpretable
- Strong design bias
- Allows in-field tuning


RL-like black-box architecture
RGBD
GNSS
odometry
control
Real world
cost
reward
policy gradient

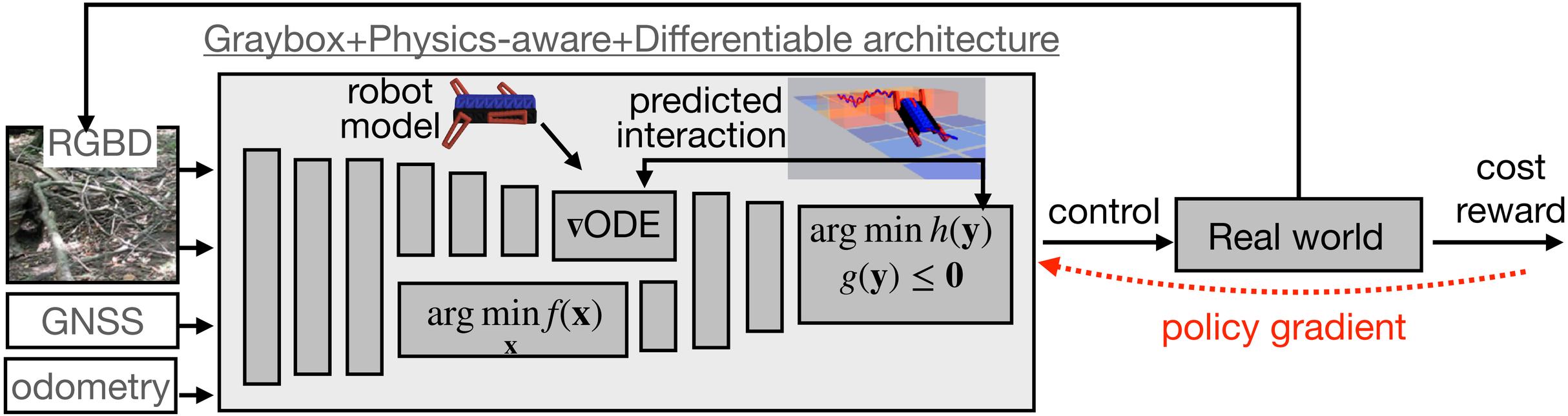
Fully differentiable black-box architecture

- Easy to design
- Poor generalization
- Smaller design bias
- Rewards tuning
- Self-actuated movement is necessary in order to develop normal perception.
- => independent training of components is bad idea



[Held and Hein, J. of Comparative Psychology, 1963]

Longterm goal: Explainable + self-learnable architecture



Next lecture topic: implicit layers