



CTU

CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

Deep Learning Essentials

10. Reinforcement learning

Markov decision process, policy gradient methods, ...

Lukáš Neumann

Markov Decision Process (MDP)

- Describes behaviour of a decision-making agent (e.g. robot)

States:	$\mathbf{x} \in \mathcal{R}^n$	A situation the agent is in (e.g. position on a map, chess board configuration, ...)
Actions:	$\mathbf{u} \in \mathcal{R}^m$	An action the agent can do (e.g. move, place a tick, ...)
Model:	$p(\mathbf{x}' \mathbf{x}, \mathbf{u})$	When an action is taken, what happens next
Rewards:	$r(\mathbf{x}, \mathbf{u}, \mathbf{x}') \in \mathcal{R}$	A number given to the agent for certain states or actions
Policy:	$\pi(\mathbf{u} \mathbf{x})$	Agent's plan (strategy) that says what to do in given state

- The goal is typically finding the **Policy** with the highest **Reward**

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

Markov Decision Process (MDP)

- Let's consider a 3x4 grid world. The agent starts at cell (1,1) and aims to reach the Diamond at while avoiding Fire. The agent can move in four directions but cannot go through the wall.

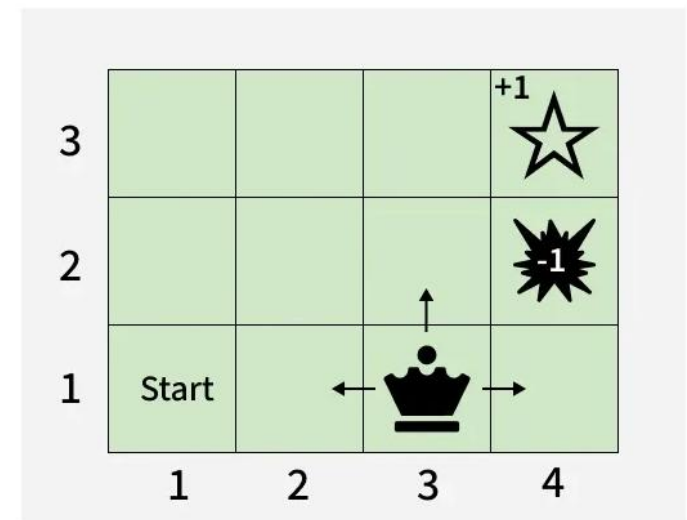
States: Agent position in the grid, position of the Fire and the Diamond

Actions: UP, DOWN, RIGHT, LEFT

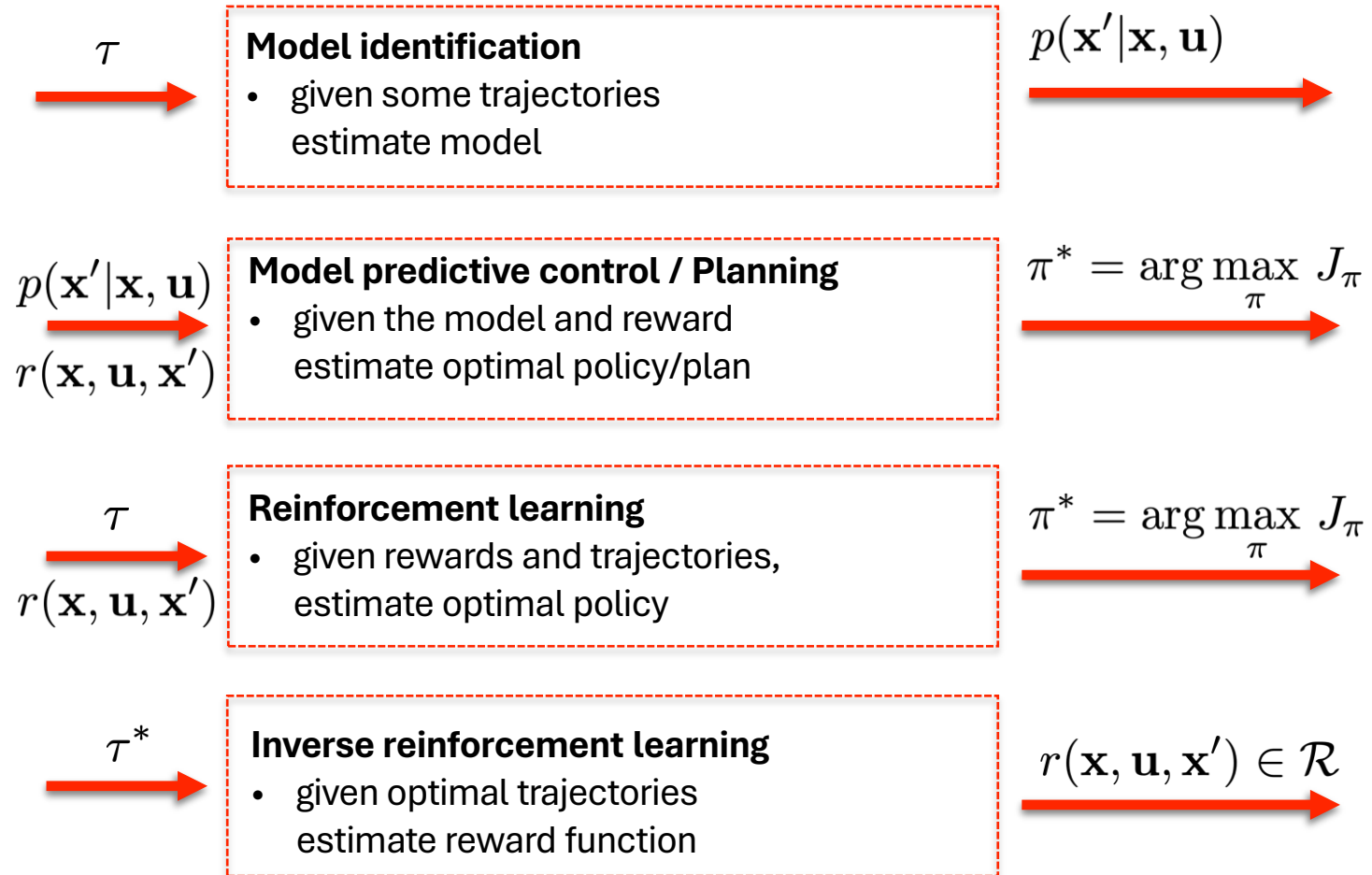
Model: With 80% chance the agent goes in intended direction, 20% random

Rewards: +1 for reaching Diamond,
-1 for Fire, -0.01 for each move

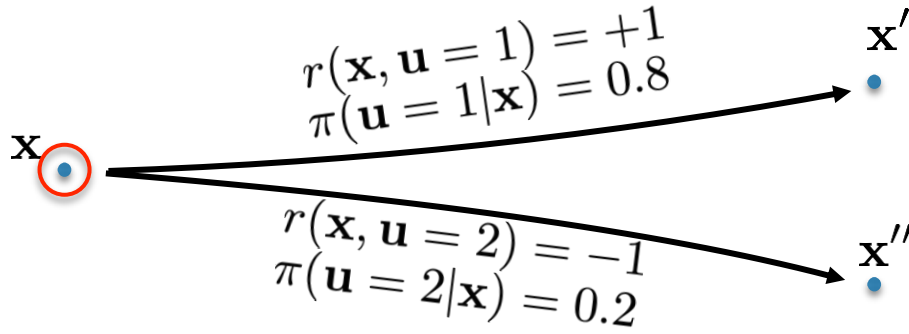
Policy: Go UP/DOWN until agent's Y axis the same as the Y axis of the Diamond.
Then go LEFT/RIGHT until agent's X axis matches the Diamond



MDP Tasks

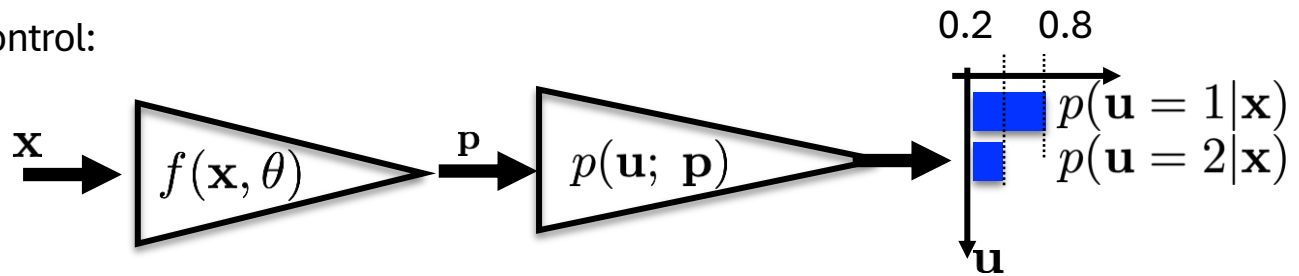


Markov Decision Process (MDP)

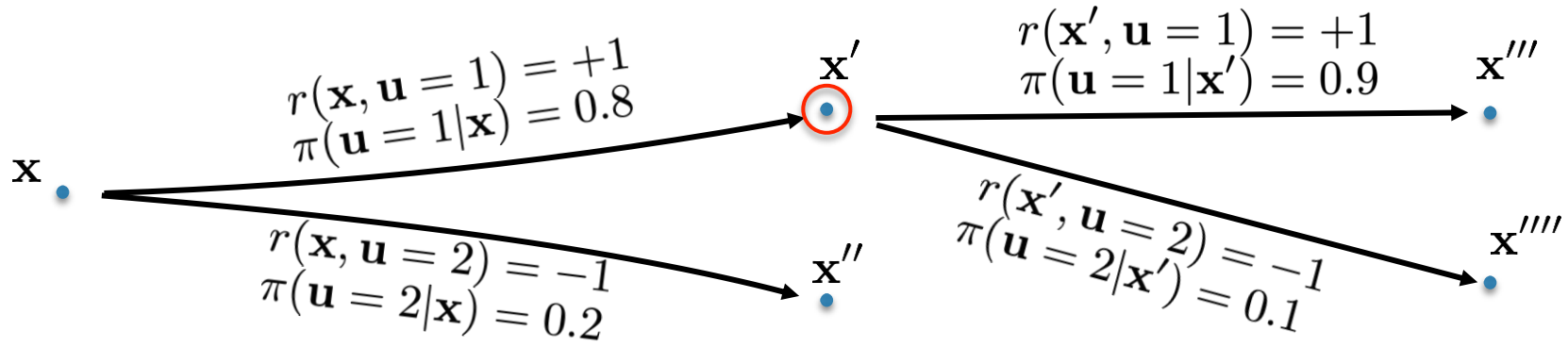


Stochastic policy for discrete control:

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

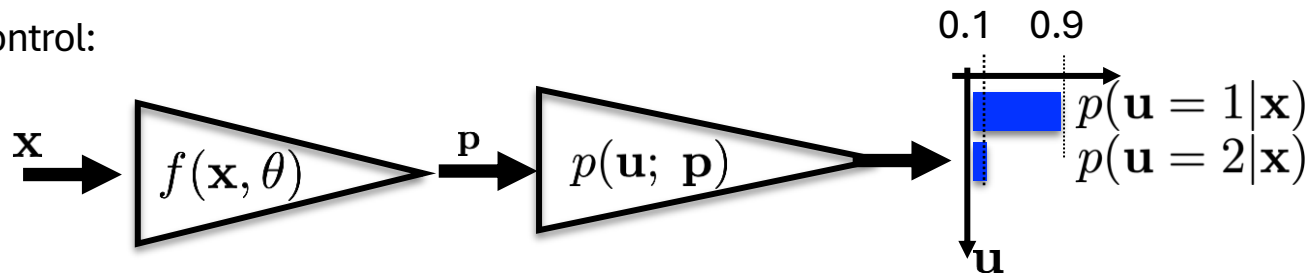


Markov Decision Process (MDP)

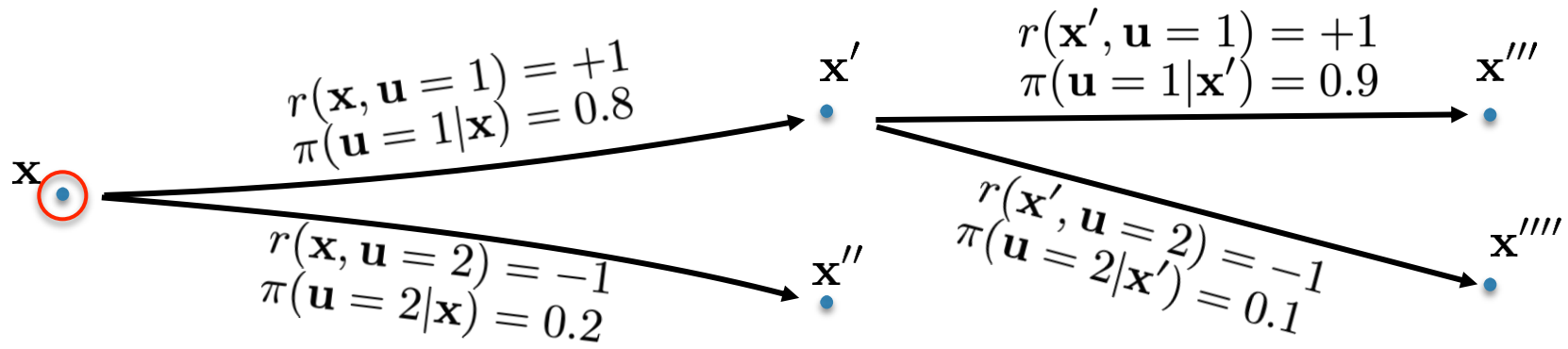


Stochastic policy for discrete control:

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



Markov Decision Process (MDP)

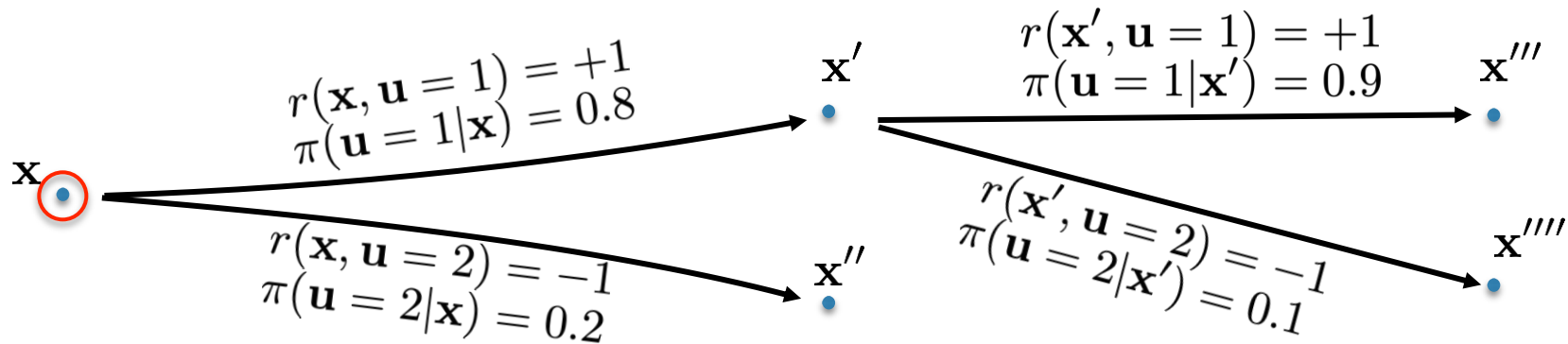


$V(\mathbf{x})$ value is the discounted sum of the rewards to be earned on average from state \mathbf{x}

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

$$V^\pi(\mathbf{x}) = ???$$

Markov Decision Process (MDP)

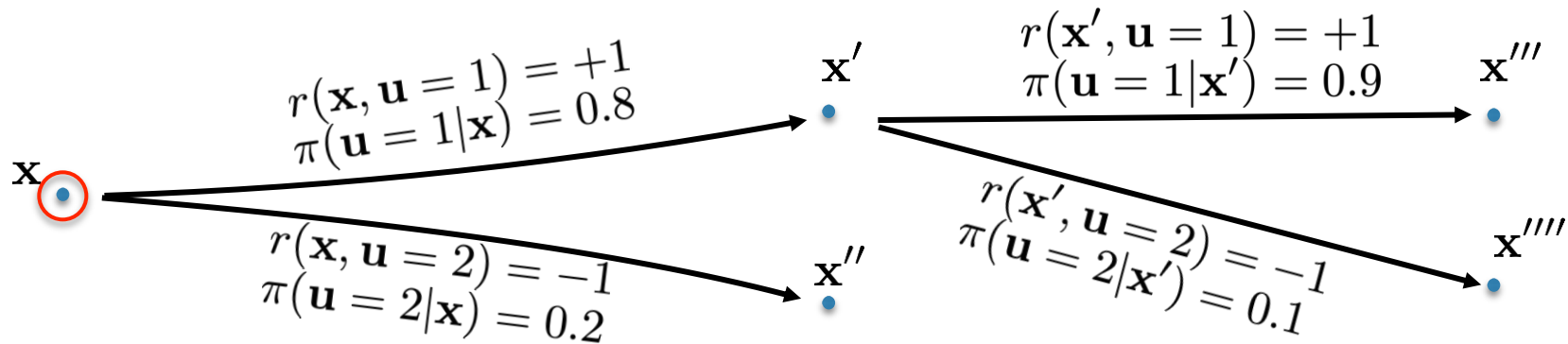


$V(\mathbf{x})$ value is the discounted sum of the rewards to be earned on average from state \mathbf{x}

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

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

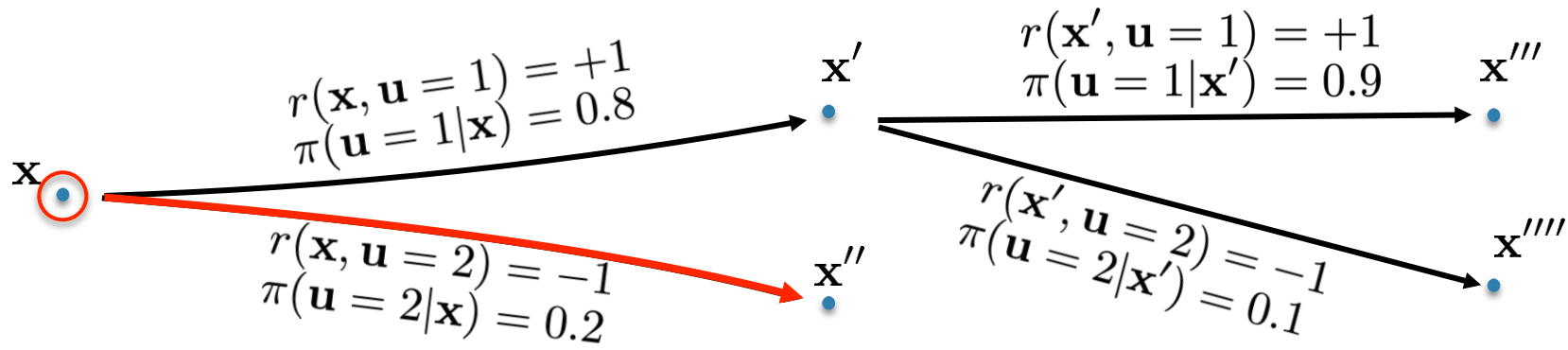
Markov Decision Process (MDP)



$Q(\mathbf{x}, \mathbf{u})$ value is the discounted sum of the rewards to be earned on average from state \mathbf{x} by taking the action \mathbf{u}

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

Markov Decision Process (MDP)

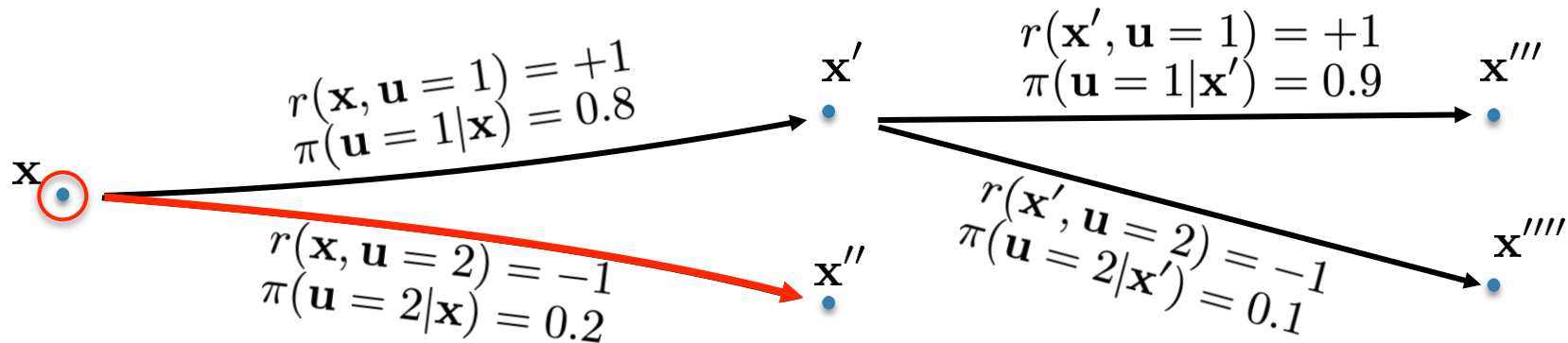


$Q(\mathbf{x}, \mathbf{u})$ value is the discounted sum of the rewards to be earned on average from state \mathbf{x} by taking the action \mathbf{u}

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

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

Markov Decision Process (MDP)

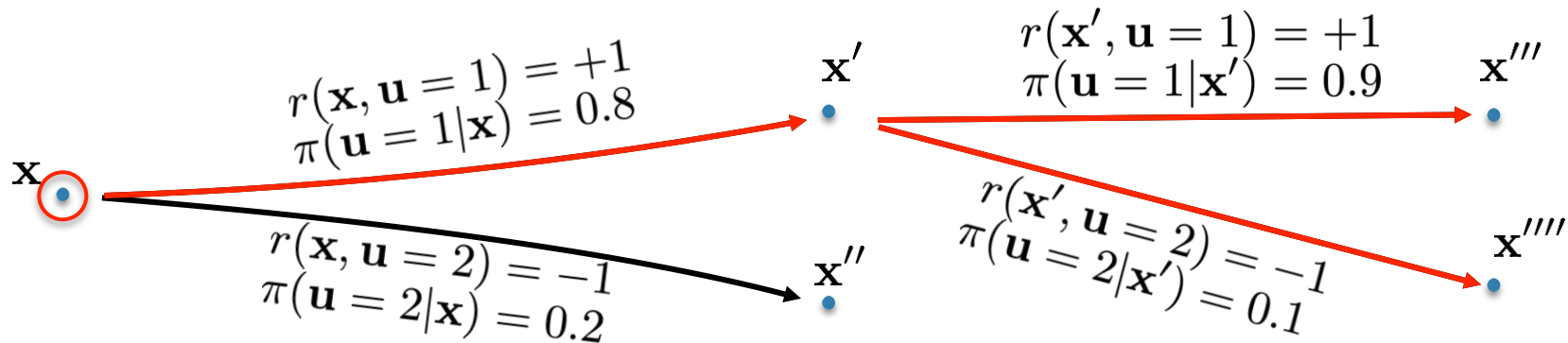


$Q(\mathbf{x}, \mathbf{u})$ value is the discounted sum of the rewards to be earned on average from state \mathbf{x} by taking the action \mathbf{u}

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

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

Markov Decision Process (MDP)



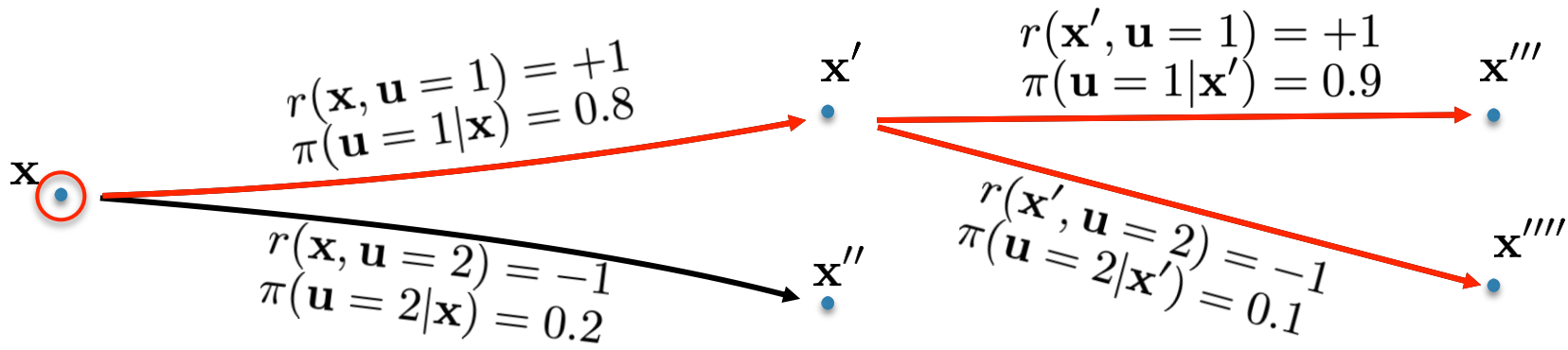
$Q(\mathbf{x}, \mathbf{u})$ value is the discounted sum of the rewards to be earned on average from state \mathbf{x} by taking the action \mathbf{u}

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

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

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

Markov Decision Process (MDP)



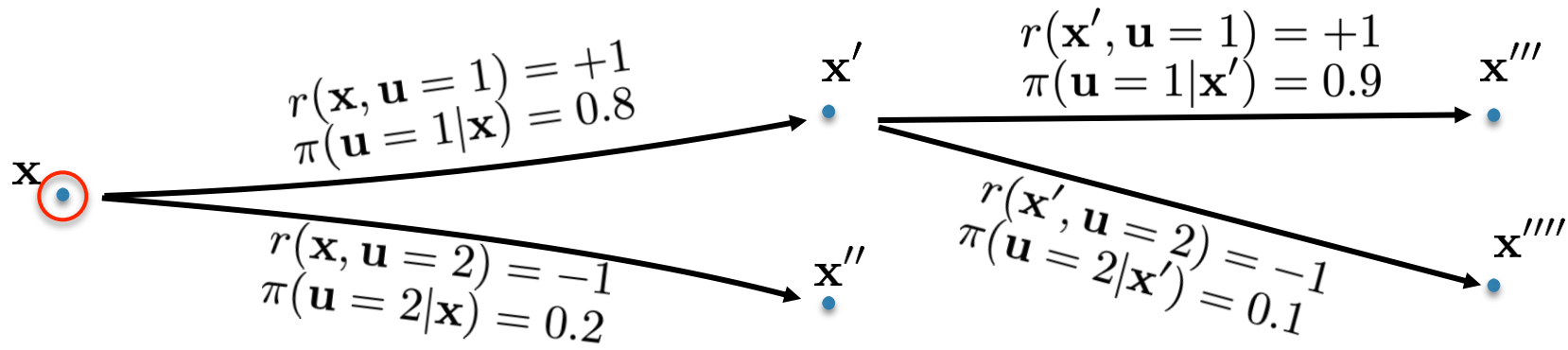
$Q(\mathbf{x}, \mathbf{u})$ value is the discounted sum of the rewards to be earned on average from state \mathbf{x} by taking the action \mathbf{u}

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

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

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

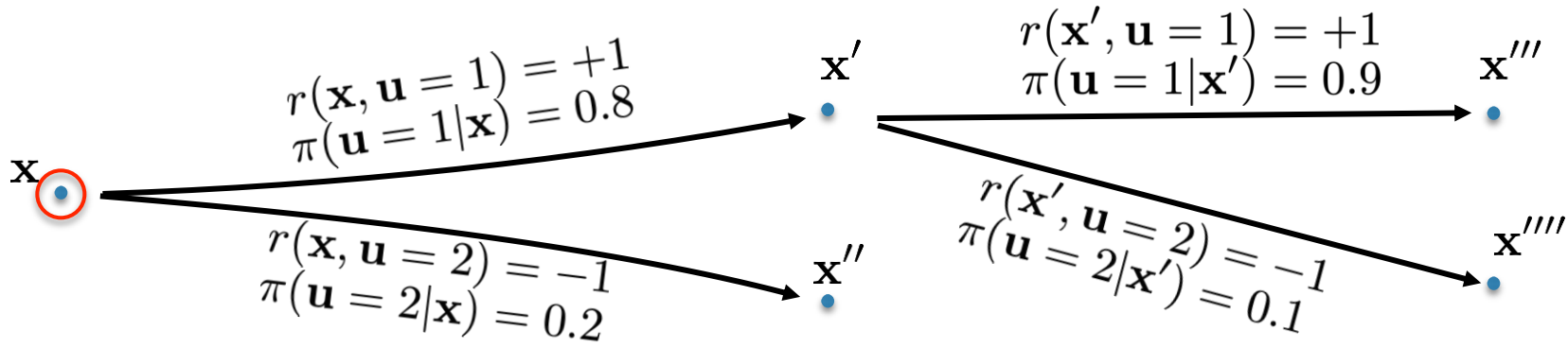
Markov Decision Process (MDP)



Advantage function $\mathbf{A}(\mathbf{x}, \mathbf{u})$ is the average gain/loss in reward by taking the action \mathbf{u} compared to the baseline policy

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

Markov Decision Process (MDP)



Advantage function $\mathbf{A}(\mathbf{x}, \mathbf{u})$ is the average gain/loss in reward by taking the action \mathbf{u} compared to the baseline policy

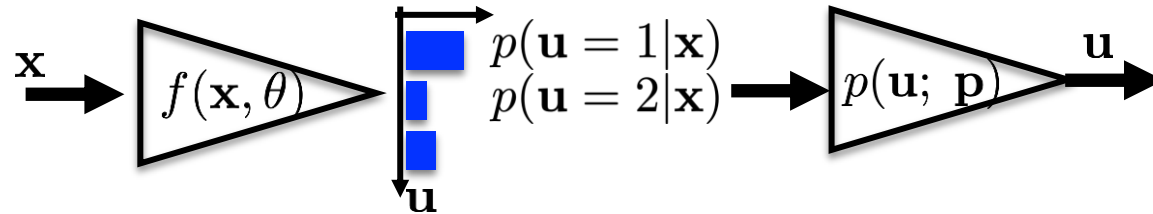
$$A^\pi(\mathbf{x}, \mathbf{u} = 1) = Q^\pi(\mathbf{x}, \mathbf{u} = 1) - V^\pi(\mathbf{x}) = 1.8 - 1.24 = 0.56$$

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

REINFORCE algorithm

Stochastic policy
discrete control:

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

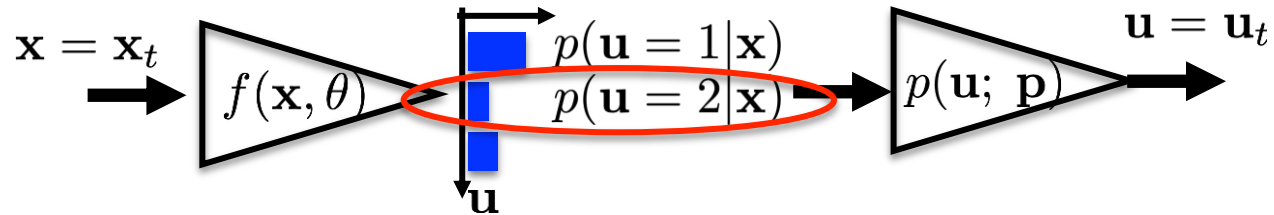


1. Initialize policy $\pi_{\theta}(\mathbf{u}|\mathbf{x}) = f(\mathbf{x}, \theta)$
2. Collect trajectories τ with policy π_{θ}
3. Optimize criterion: $\theta := \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}$ **What is the gradient ?**
4. Repeat from 2

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

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

REINFORCE algorithm



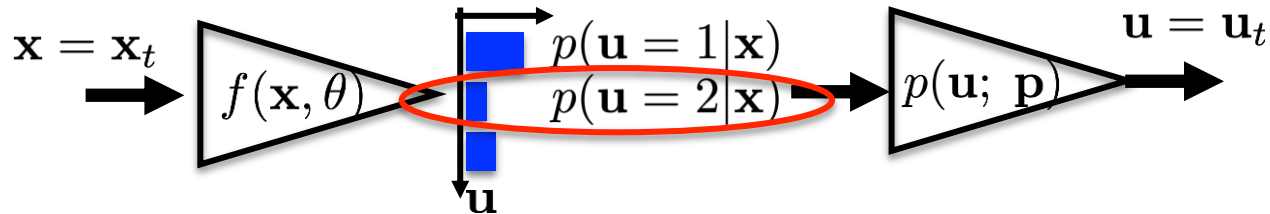
Assume that you took action \mathbf{x}_t in the state \mathbf{u}_t

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

Sum of rewards along the resulting trajectory.

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

REINFORCE algorithm



Assume that you took action \mathbf{x}_t in the state \mathbf{u}_t

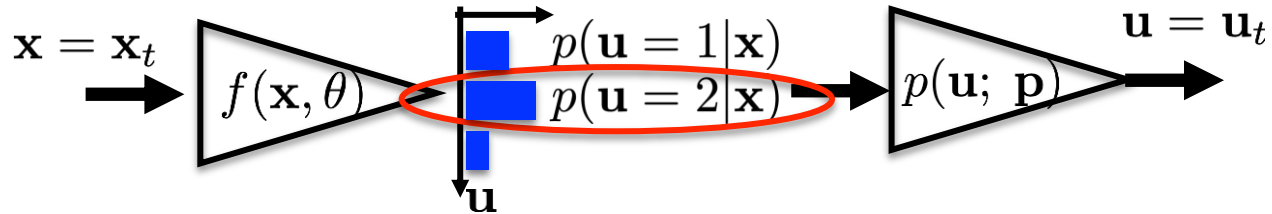
$$\frac{\partial J(\theta)}{\partial \theta} \approx \sum_{t=0}^T \frac{\partial \log(\pi_{\theta}(\mathbf{u}_t | \mathbf{x}_t))}{\partial \theta} \cdot r(\tau)$$

$r(\tau) = +100$

Sum of rewards along the resulting trajectory.

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

REINFORCE algorithm



Assume that you took action \mathbf{x}_t in the state \mathbf{u}_t

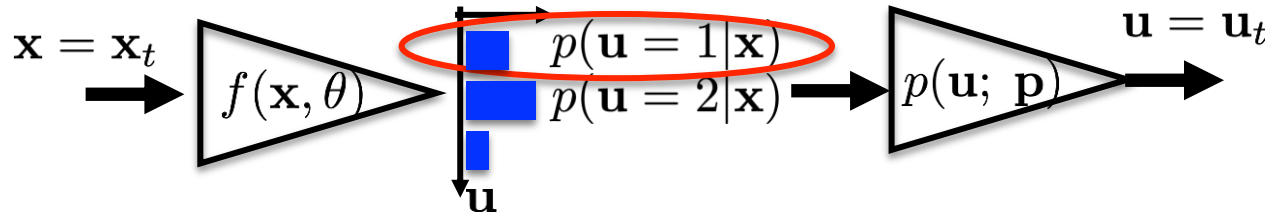
$$\frac{\partial J(\theta)}{\partial \theta} \approx \sum_{t=0}^T \frac{\partial \log(\pi_{\theta}(\mathbf{u}_t | \mathbf{x}_t))}{\partial \theta} \cdot r(\tau)$$

$r(\tau) = +100$

Sum of rewards along the resulting trajectory.

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

REINFORCE algorithm



Assume that you took action \mathbf{x}_t in the state \mathbf{u}_t

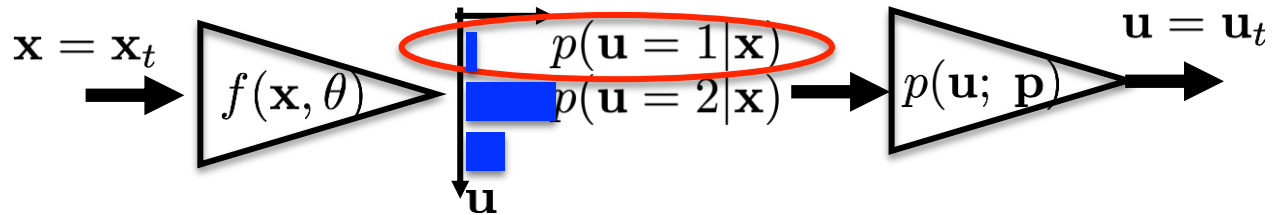
$$\frac{\partial J(\theta)}{\partial \theta} \approx \sum_{t=0}^T \frac{\partial \log(\pi_{\theta}(\mathbf{u}_t | \mathbf{x}_t))}{\partial \theta} \cdot r(\tau)$$

$$r(\tau) = -100$$

Sum of rewards along the resulting trajectory.

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

REINFORCE algorithm



Assume that you took action \mathbf{x}_t in the state \mathbf{u}_t

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

$$r(\tau) = -100$$

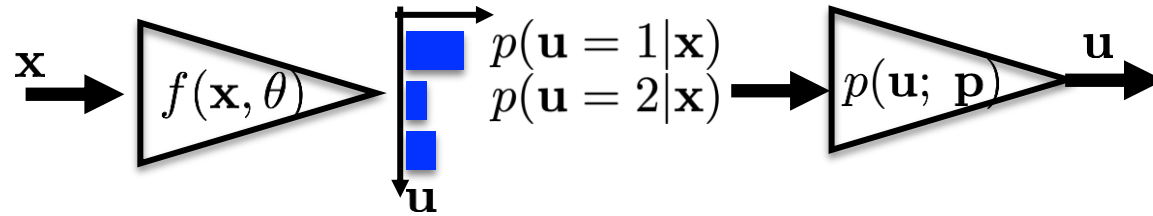
Sum of rewards along the resulting trajectory.

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

REINFORCE algorithm

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 $\pi_{\theta}(\mathbf{u}|\mathbf{x})$
3. Update policy:

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

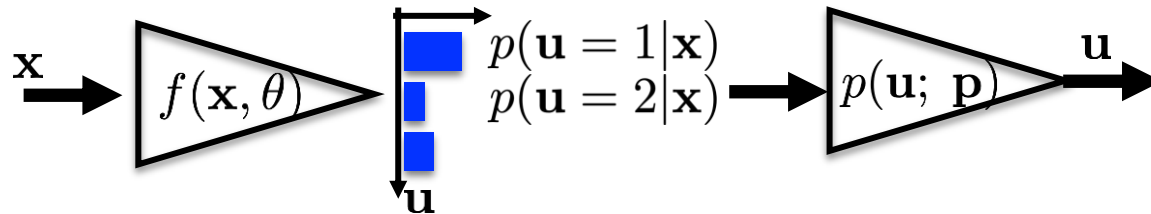
$$\theta := \theta + \alpha \frac{\partial J(\theta)}{\partial \theta}$$

4. Repeat from 2

Actor-Critic algorithm

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 $\pi_{\theta}(\mathbf{u}|\mathbf{x})$
3. Update policy (**Actor**)

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

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

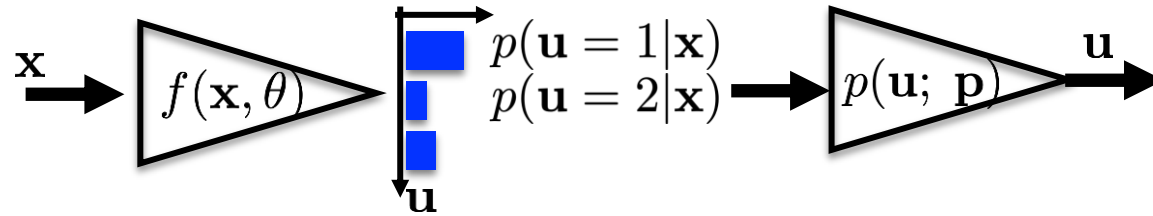
Advantage function

4. Repeat from 2

Actor-Critic algorithm

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 $\pi_{\theta}(\mathbf{u}|\mathbf{x})$
3. Update policy (**Actor**)

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

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

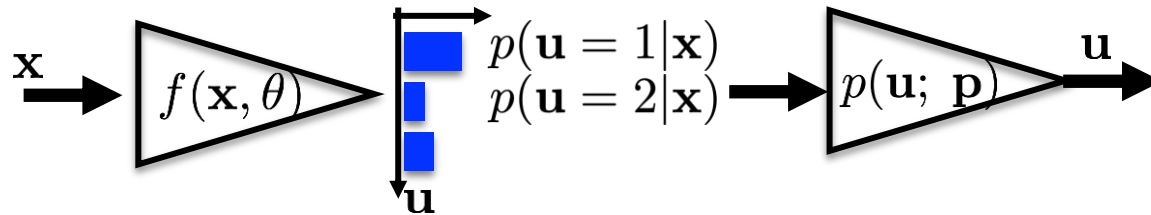
4. Repeat from 2

Advantage function
approximation

Actor-Critic algorithm

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 $\pi_{\theta}(\mathbf{u}|\mathbf{x})$
3. **Critic:** Update value function to predict observed values $V_{\omega}(\mathbf{x}) \leftarrow r + \gamma V_{\omega}(\mathbf{x}')$

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

4. **Actor:** Update policy

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

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

5. Repeat from 2

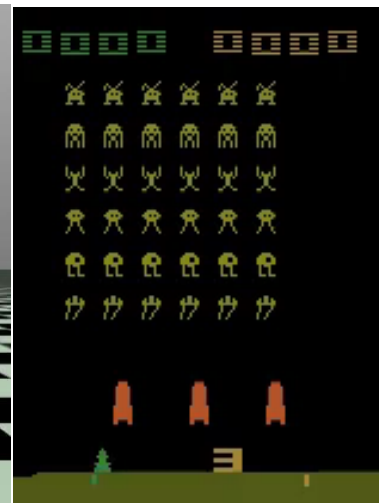
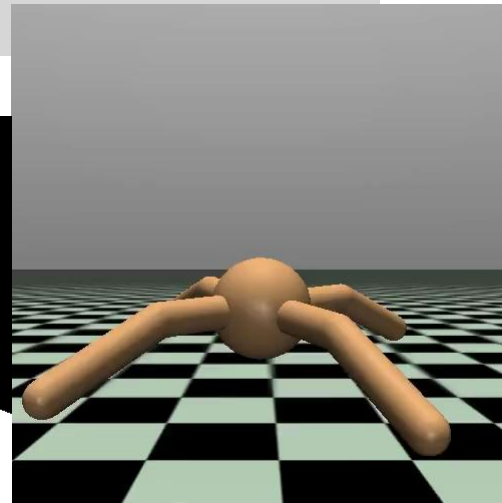
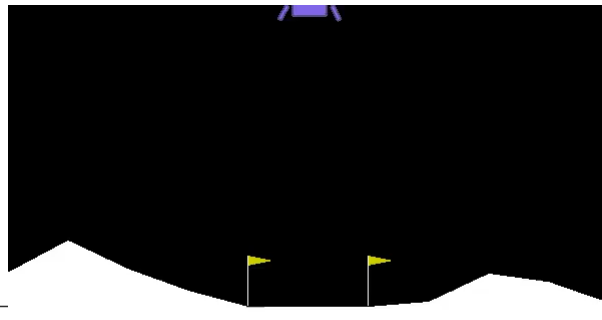
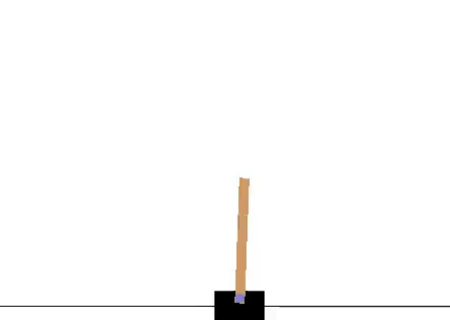
Reinforcement learning

- OpenAI Gym (Gymnasium)
- <https://github.com/openai/gym>

```
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

- Starcraft II
 - Deepmind AlphaStar beaten top-end professional human gamers 5:0
- No single best strategy
- Partially observable
- Longterm planning (delayed rewards from upgrades)
- Real-time
- Large action space



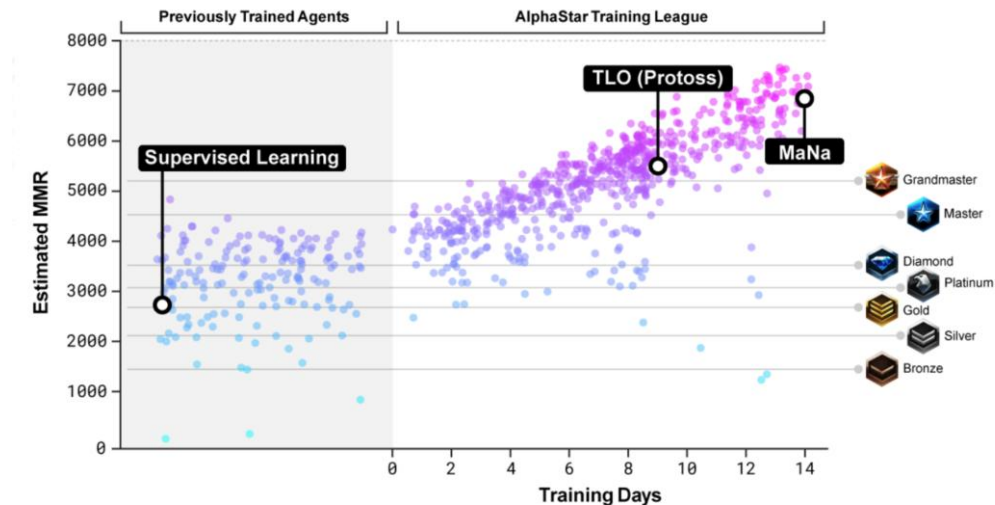
Reinforcement learning

- Starcraft II
 - Deepmind AlphaStar beaten top-end professional human gamers 5:0
- Supervised training to learn basic strategies



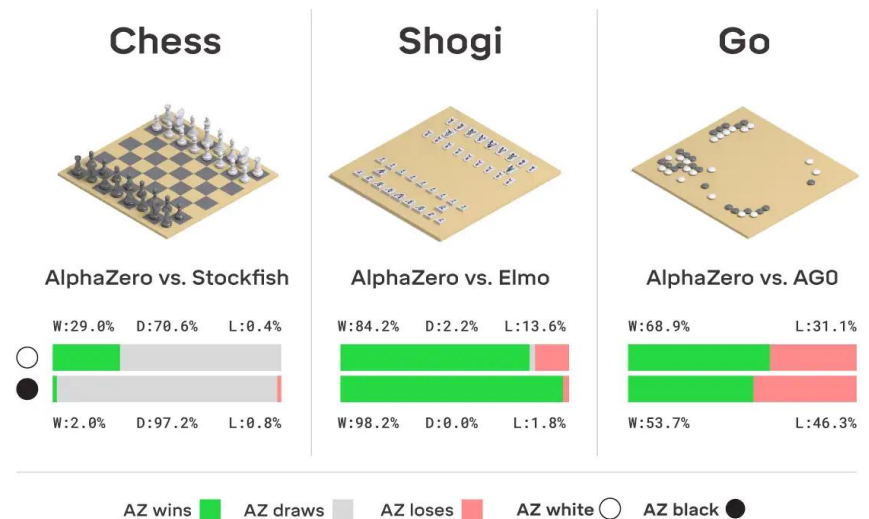
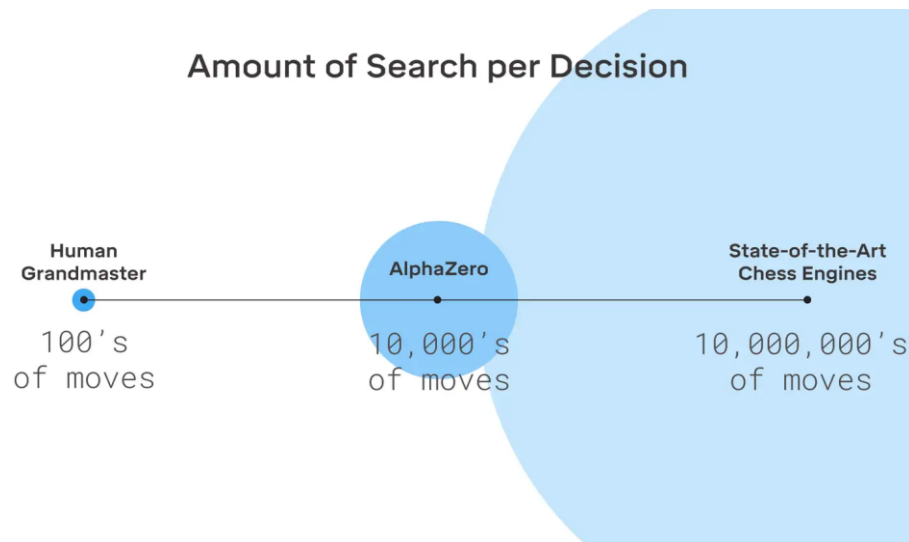
Reinforcement learning

- Starcraft II
 - Deepmind AlphaStar beaten top-end professional human gamers 5:0
- Supervised training to learn basic strategies
- Supervised training from played games
- 14 days of reinforcement learning by playing against 2 grandmasters



Reinforcement learning

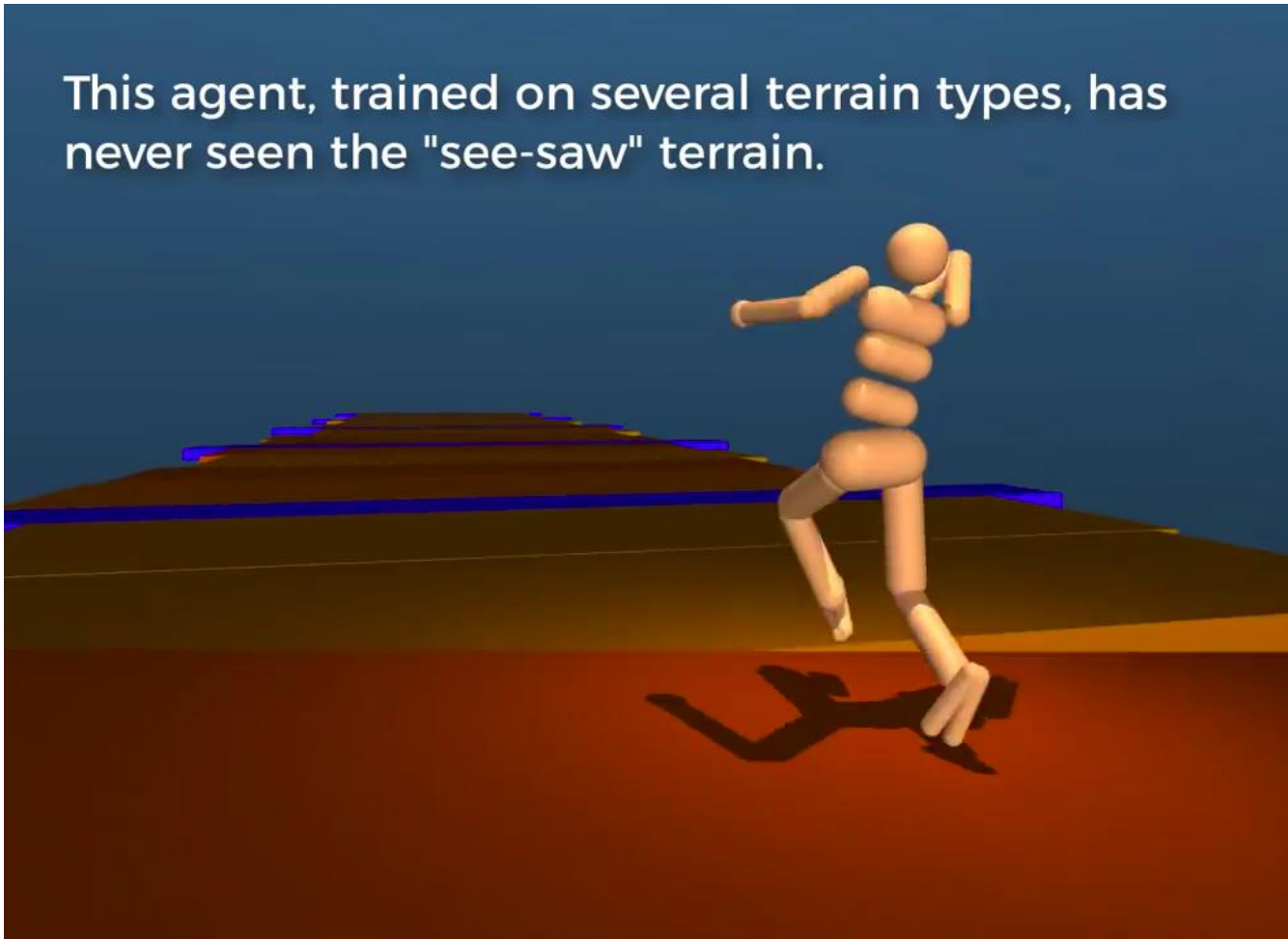
- Playing board games
- Brute-force algorithms have no chance in huge state-action spaces
→ RL guides the search efficiently



Reinforcement learning

- Learning complex movements in simulations

This agent, trained on several terrain types, has never seen the "see-saw" terrain.



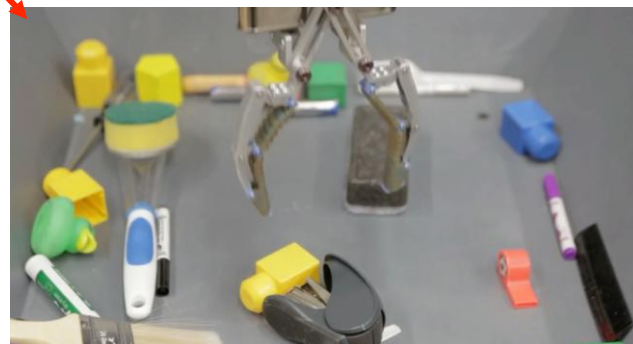
Reinforcement learning

- Learning complex motions in reality by parallelizing and automatizing rewards

manipulator+ RGB camera



image



$$\pi_{\theta} = (\quad)$$

Continuous motion control from RGB(D)

Reinforcement learning

- Learning complex motions in reality by parallelizing and automatizing rewards



Competencies gained for the test

- Markov Decision Process
- V-value, Q-value, Advantage function
- Policy gradient methods (REINFORCE)