LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Praktikum Mobile und Verteilte Systeme

# Applications and Further Challenges of Autonomous Systems

Prof. Dr. Claudia Linnhoff-Popien
André Ebert, Thomy Phan, Robert Müller, Steffen Illium
http://www.mobile.ifi.lmu.de

WiSe 2018/19

# Outline

- **An Introduction to Autonomous Systems**
  - Motivation, Definition and Challenges
  - Artificial Intelligence
- **Decision Making in Autonomous Systems**
  - Markov Decision Processes
  - Planning
  - Reinforcement Learning
- **Applications and Further Challenges**
  - Deep Reinforcement Learning
  - Combining Planning and Reinforcement Learning
  - Further Challenges

# Outline

- **An Introduction to Autonomous Systems**
  - Motivation, Definition and Challenges
  - Artificial Intelligence
- **Decision Making in Autonomous Systems**
  - Markov Decision Processes
  - Planning
  - Reinforcement Learning
- **Applications and Further Challenges**
  - Deep Reinforcement Learning
  - Combining Planning and Reinforcement Learning
  - Further Challenges

# → Deep Reinforcement Learning

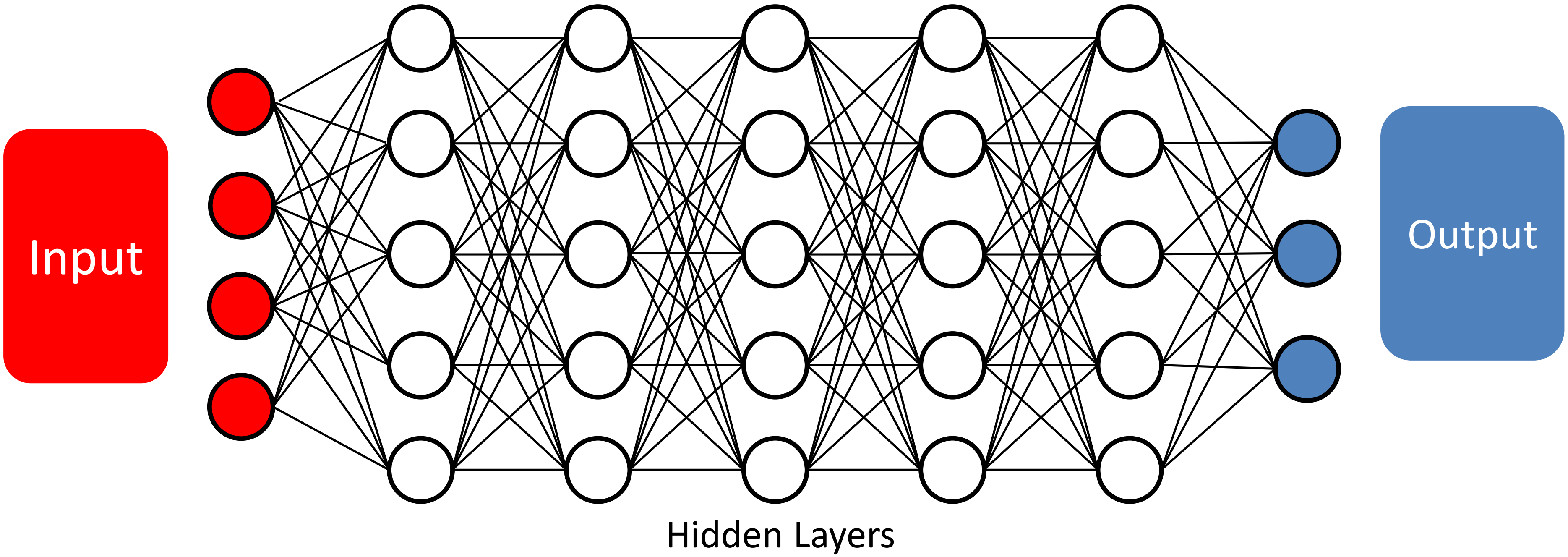
# Reinforcement Learning in Practice

- Focus on Model-Free Reinforcement Learning
- **Problem:** How to approximate $\pi^*$, $V^*$ and/or $Q^*$ for *each state*?

- Tabular Methods do not scale well with large state and action spaces
- Use Function Approximation instead:
  - Linear Combination of Features
  - Decision Tree / Forest
  - Nearest Neighbor
  - Neural Network
  - …

# Deep Learning

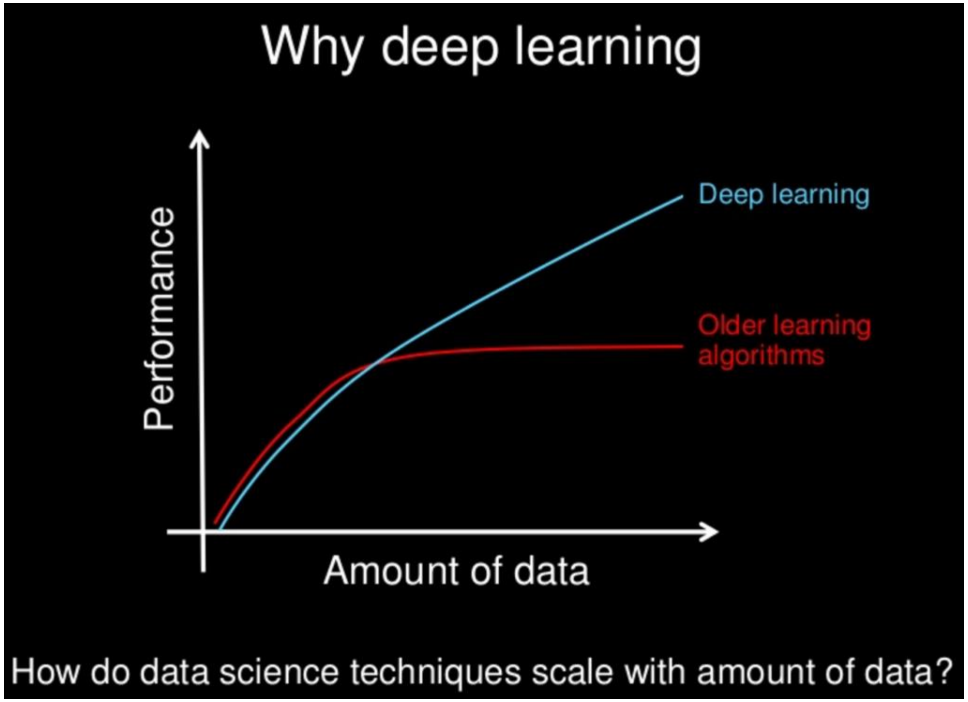
- **Deep Learning:** Neural Network with multiple hidden layers




- Enables **end-to-end learning** (feature learning + mapping) of tasks
- Typically trained with Stochastic Gradient Descent
- Works well for many complex tasks – but hard to interpret

# Why Deep Learning for Reinforcement Learning?

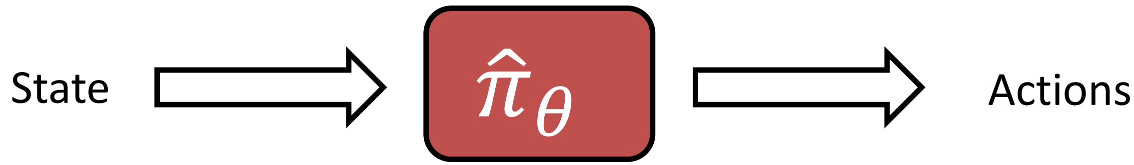
- Reinforcement Learning typically requires large amount of experience / data to solve complex problems (with large state and action spaces)
- Deep Learning scales well with large amount of data




Andrew Ng, What data scientistis should know about deep learning, 2015

# Model Free Deep Reinforcement Learning

- Approximate $\pi^*$, $V^*$ and/or $Q^*$ with Neural Networks (and weights $\theta$)
- Approximate $\pi^*: \mathcal{S} \rightarrow \mathcal{A}$ with $\hat{\pi}_\theta$:

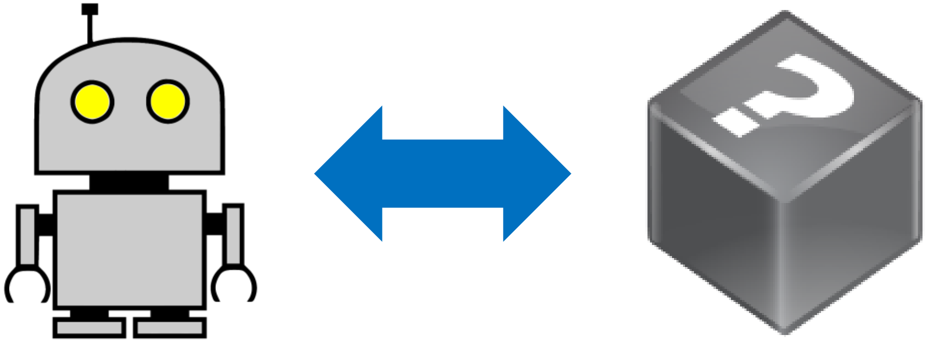



- Approximate $V^*: \mathcal{S} \rightarrow \mathbb{R}$ with $\hat{V}_\theta$:


- Approximate $Q^*: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ with $\hat{Q}_\theta$:

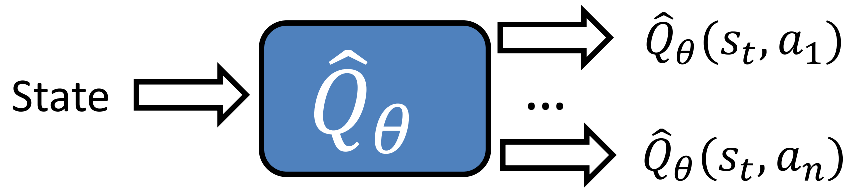
# Challenges of Deep Reinforcement Learning

- Large amount of (real-world) data required
- Correlation in Data (Overfitting/Oscillation)
- Non-Stationary Data Distribution
- Computational and Memory Resources
- Consideration of safety, risk, and uncertainty



# Example: Deep Q-Learning

- Approximate $Q^*: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ with Deep Q-Network $\hat{Q}_\theta$




- Use experience buffer $D$ to store the last $N$ experience samples
- After each time step:
  - Store $e_t = \langle s_t, a_t, r_t, s_{t+1} \rangle$ in $D$
  - Sample random minibatch of $n_{batch}$ experience samples to generate TD-targets:

$$y_t = r_t + \gamma \max_{a_{t+1} \in \mathcal{A}} \hat{Q}_\theta(s_{t+1}, a_{t+1})$$

  - Perform stochastic gradient descent on $\langle s_t, y_t \rangle$-pairs

# Example: Improvements of Deep Q-Learning

- Use second network $\hat{Q}_{\theta^-}$ to temporarily „freeze“ TD targets:

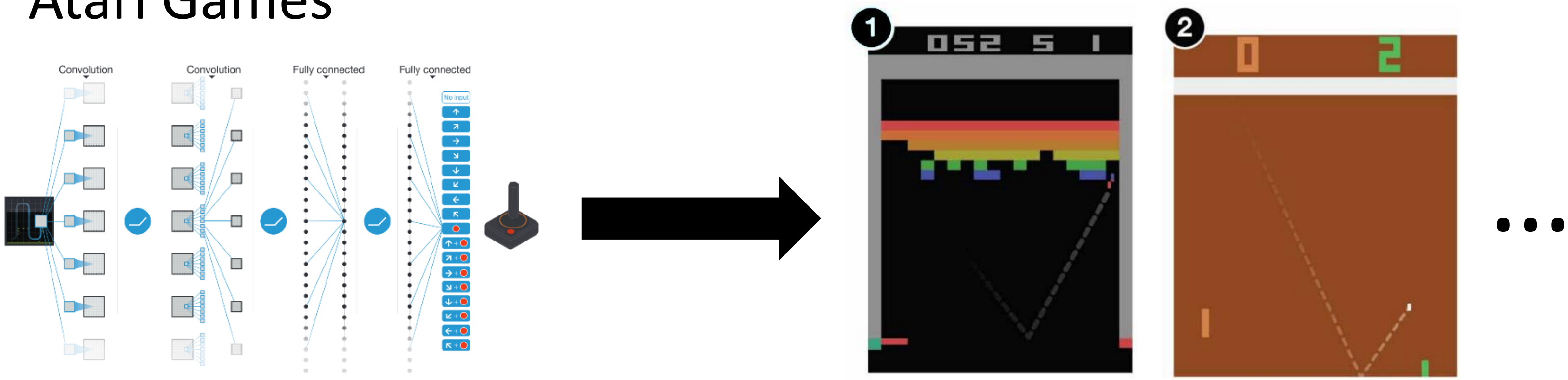
$$y_t = r_t + \gamma \max_{a_{t+1} \in \mathcal{A}} \hat{Q}_{\theta^-}(s_{t+1}, a_{t+1})$$

**target network prediction**

  - Periodically set $\theta^- = \theta$, after $C$ time steps (or soft update with $\alpha \in [0,1]$: $\theta^- = (1-\alpha)\theta^- + \alpha\theta$)
- Prioritize experience sampling according to TD error
- More variants:
  - Deep Recurrent Q-Network
  - Dueling Deep Q-Network
  - Distributional Deep Q-Network

# Success of Model-Free Deep Reinforcement Learning

- Atari Games




V. Mnih et al., Human-level control through deep reinforcement learning, Nature, 2015

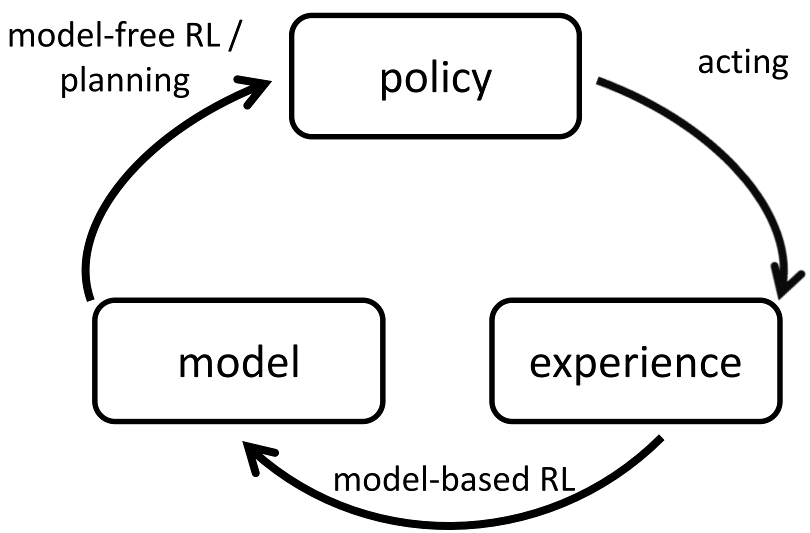
- OpenAI Five


https://blog.openai.com/openai-five/

# → Combining Planning and Reinforcement Learning

# Dyna Architecture

- Learn value function / policy from simulated and actual experience
- Learn model from actual experience
- Sample experience from model

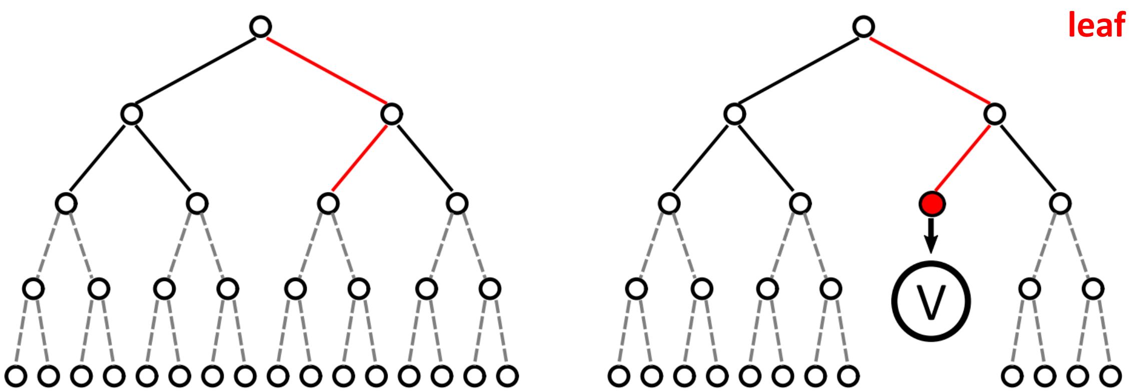


# Online Planning with Reduced Search Depth

- Online Planning usually has strict real-time requirements
  - limited computation budget
  - limited horizon
- **Problem:** how to evaluate actions and states with limited horizon $h$?
- Use learned value function $\hat{V}$ to evaluate leaf states, e.g.

$$G_t = \sum_{k=0}^{h-1} \gamma^k \mathcal{R}\big(s_{t+k}, \pi(s_{t+k})\big) + \gamma^h \hat{V}(s_h)$$

**leaf state evaluation**




T. Phan, L. Belzner, T. Gabor, K. Schmid, Leveraging Statistical Multi-Agent Online Planning with Emergent Value Function Approximation, 17th Conference on Autonomous Agents and Multiagent Systems, 2018

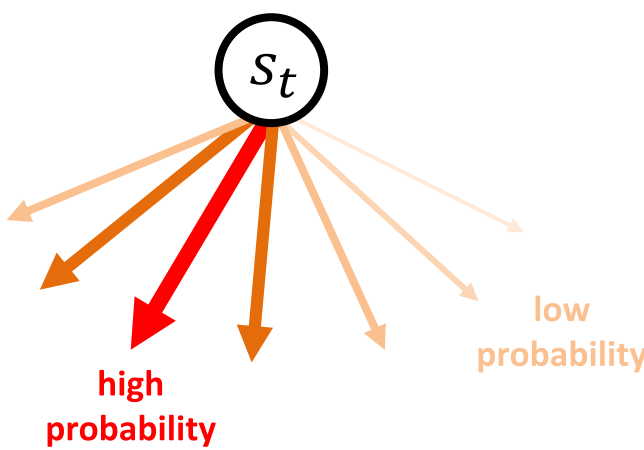
# Online Planning with Reduced Search Breadth

- Online Planning usually has strict real-time requirements
  - limited computation budget
  - limited horizon
- **Problem:** how to search with limited computation budget?
- Use learned policy $\hat{\pi}$ to recommend actions for exploration, e.g.

$$\pi_{search}(s_t) = argmax_{a_t \in \mathcal{A}}\{Q(s_t, a_t) + \hat{\pi}(s_t)\, U(s_t, a_t)\}$$

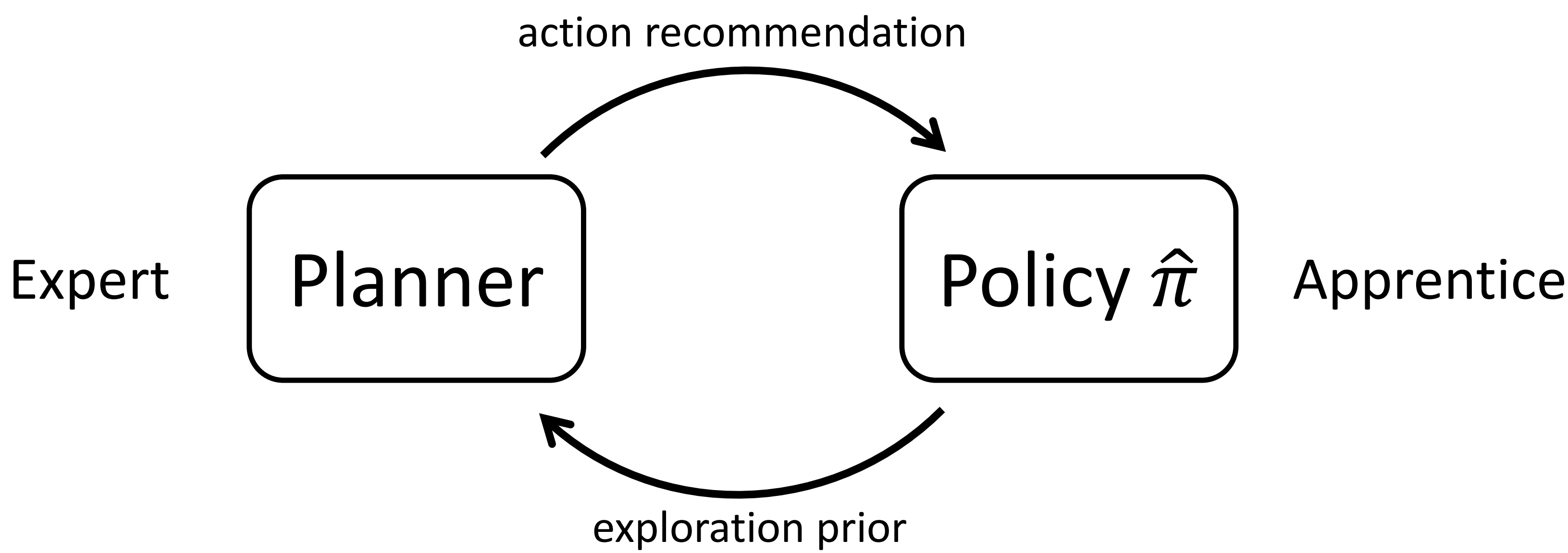
exploration prior

exploration term



# Expert Iteration

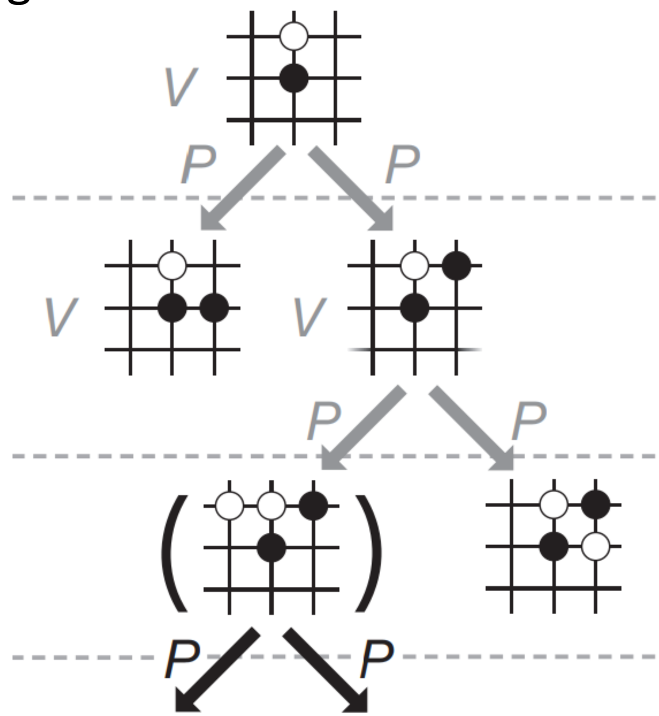
- Learn policy $\hat{\pi}$ from action recommendations of a model-based planning algorithm
- Improve model-based planner by using $\hat{\pi}$ as exploration prior




- Supervised learning without human knowledge (beyond model)

# Success Story

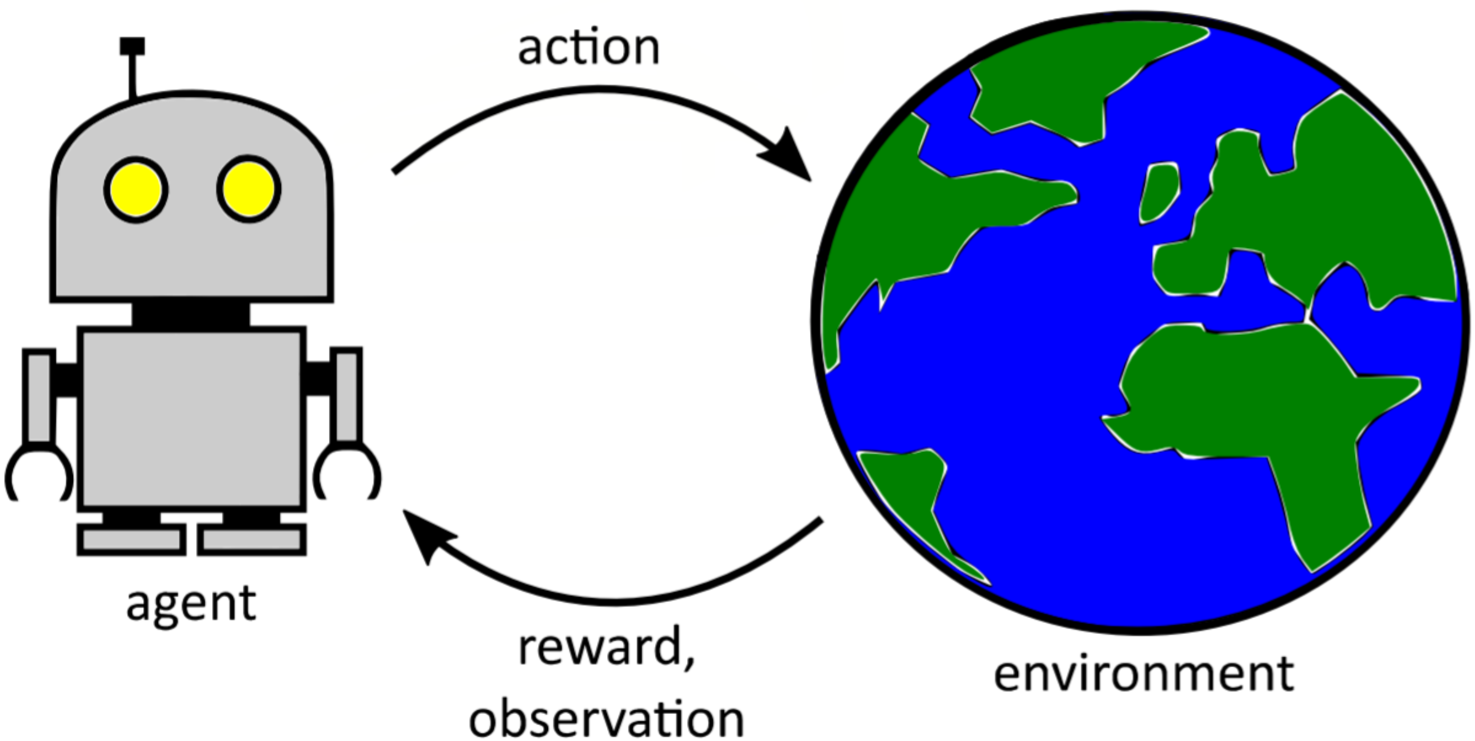
- AlphaGo Zero and AlphaZero
  - Monte Carlo Tree Search enhanced with neural networks to play Go, Chess and Shogi
  - Uses policy approximation $\hat{\pi}$ to bias node selection
  - Uses value function approximation $\hat{V}$ to evaluate nodes
- Alpha(Go)Zero learns $\hat{\pi}$ and $\hat{V}$ entirely from self-play
  - Only requires a generative model (rules, opponent model)
  - No human labeled data required




D. Silver et al., Mastering the game of go without human knowledge, Nature, 2017

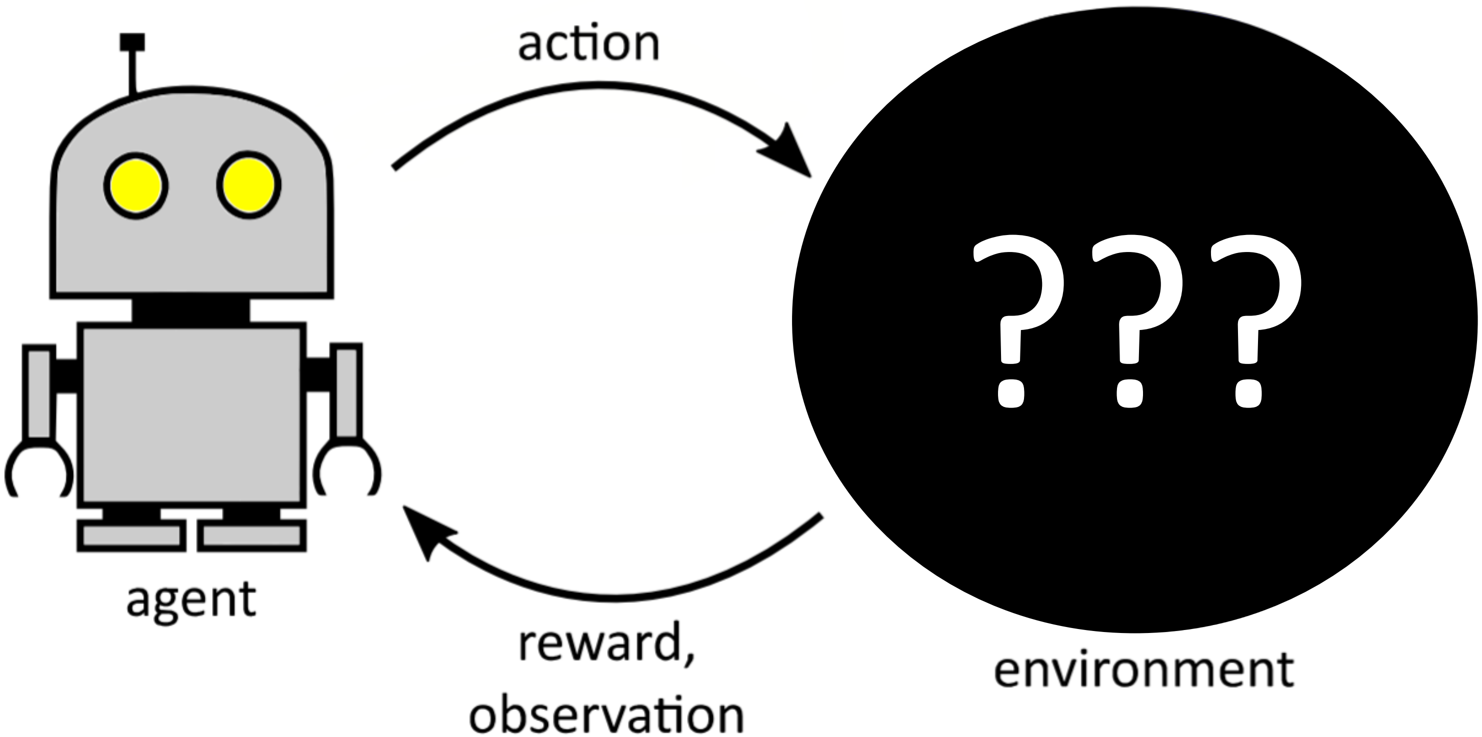
# → Further Challenges

# Partially Observable Environments




$$state = observation$$

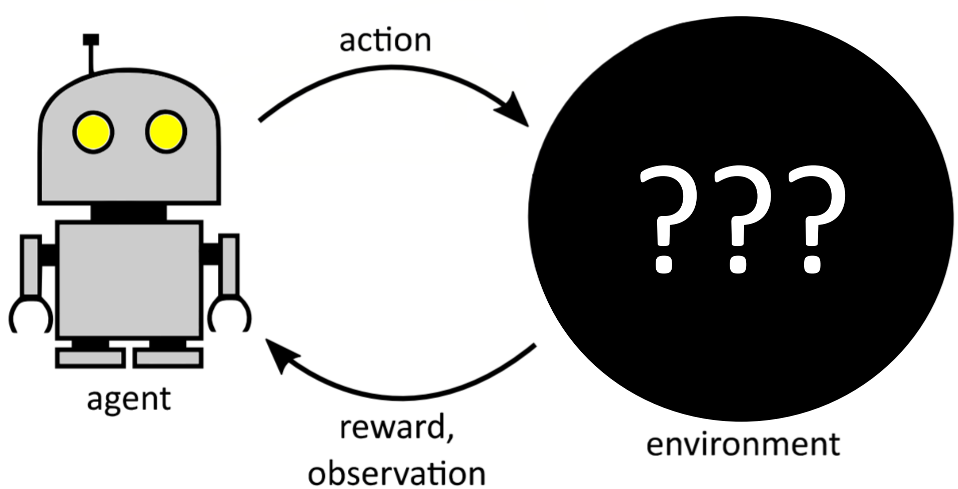
# Partially Observable Environments




$$state \neq observation$$

# Definition

- A Partially Observable MDP (POMDP) is defined as $\mathrm{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \Omega \rangle$:
  - $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}$ same as in MDPs
  - $\mathcal{O}$ is a (finite) set of observations
  - $\Omega(o_{t+1}|s_{t+1}, a_t) \in [0, 1]$ is the probability for observating $o_{t+1} \in \mathcal{O}$ when executing $a_t \in \mathcal{A}$ and transitioning to $s_{t+1} \in \mathcal{S}$



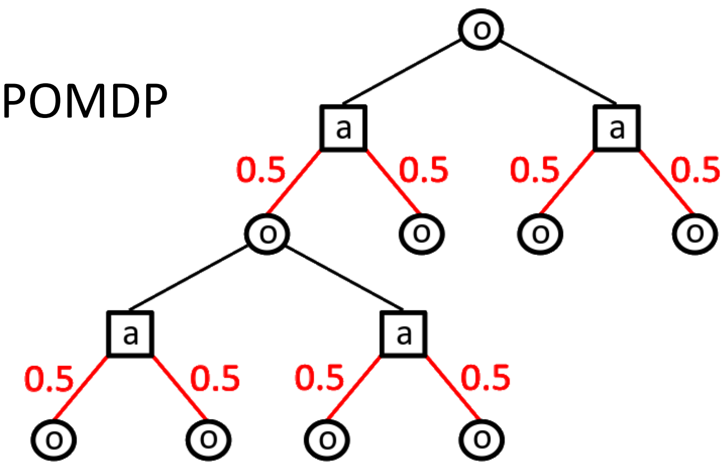
# POMDP Challenges

- States $s_t \in \mathcal{S}$ cannot be identified with observations $o_t \in \mathcal{O}$ :
  - Condition policy on **history** $h_t = [a_0, o_1, \dots, a_{t-1}, o_t]$ of actions and observations instead
  - Maintain **belief state** $b_{h_t}(s_t)$ (probability distribution over $s_t \in \mathcal{S}$)
  - Environment can be modeled as *history* or *belief state MDP*

- **Goal:** Find an *optimal policy* $\pi^*$ which maximizes the expected return for each history or belief state
  - *Curse of dimensionality*: belief state space $\mathcal{B}$ scales exponentially with the state space $\mathcal{S}$.
  - *Curse of history*: number of possible histories scales exponenetially with the horizon length of the problem.

# Scalable POMDP Solutions

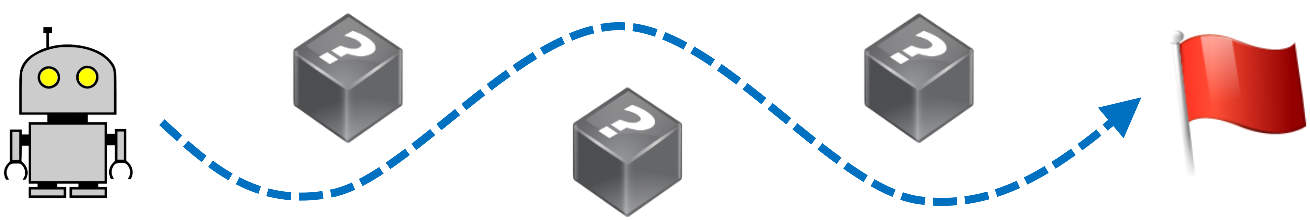
- Monte Carlo Planning
  - Requires generative model $\hat{M} \approx M$ of POMDP
  - Use $\hat{M}$ for belief state approximation (e.g., particle filter)
  - Use $\hat{M}$ to construct tree of histories




- Deep Recurrent Reinforcement Learning
  - Use Recurrent Neural Networks to train DQN on observation-sequences (histories!)
  - Basically the same mechanisms as DQN for MDPs (experience replay, target network, etc.)

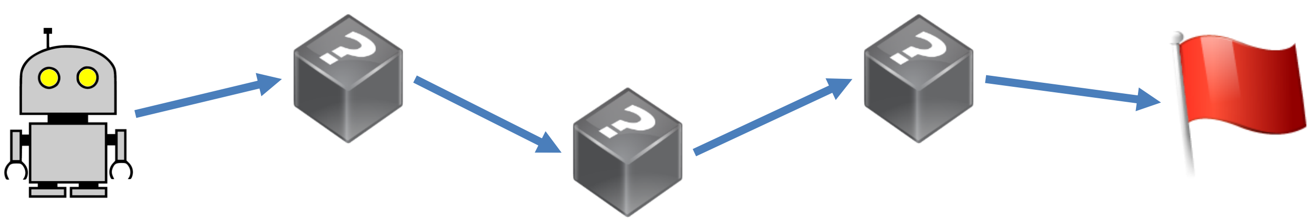
# Temporal Abstraction

- Real-world problems are often complex at different levels
- Humans often divide larger tasks into **subtasks** or **subgoals** which can be solved individually
  - Problem can be solved **top-down** by selecting subtasks
  - Problem can be solved **bottom-up** by solving subtasks
- **Example:** Navigation Task
  - Subtasks: reach object X
  - Select order of subtasks to be solved (high-level policy)
  - Solve subtasks, deal with low-level uncertainty, etc. (low-level policy)

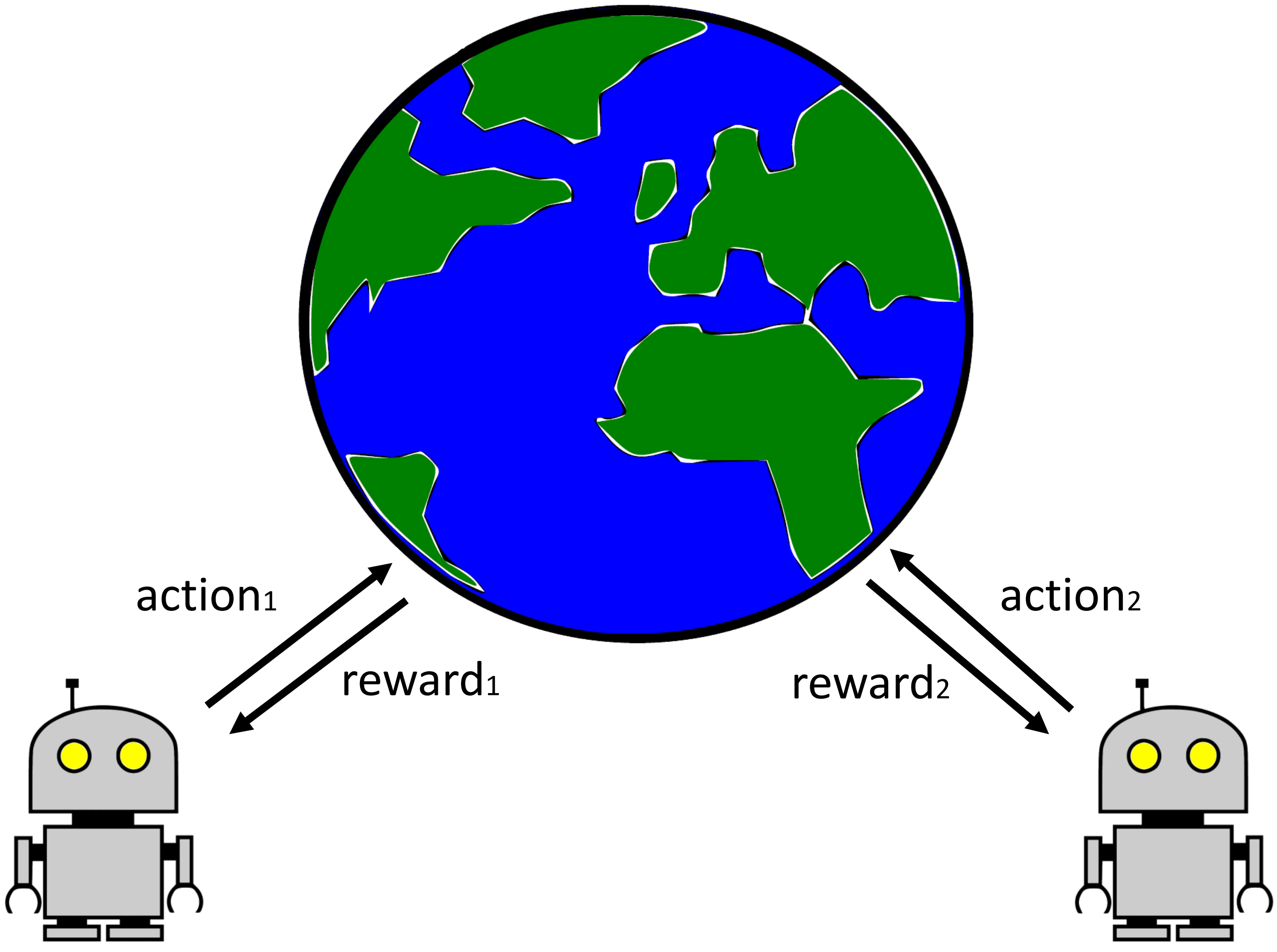


# Temporal Abstraction Challenges

- Specification of Subtasks
- Specification of (Sub)Policies
- Granularity
- Model Specification (for Planning)

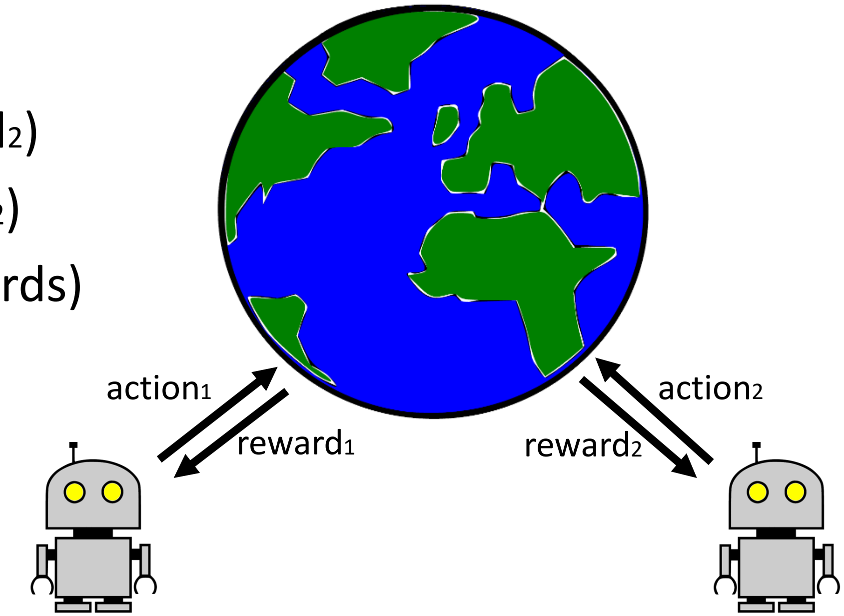


# Multi-Agent Systems (MAS)



# MAS Definitions

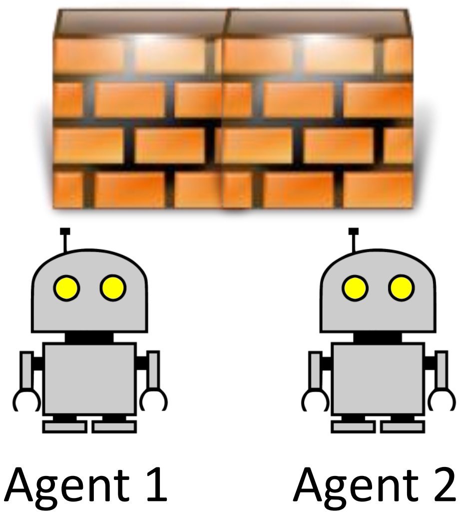
- Many ways to define a MAS
  - Competitive ($reward_1 = -reward_2$)
  - Cooperative ($reward_1 = reward_2$)
  - Self-interested (individual rewards)




- In MAS, actions of other agents have influence on the own reward
  - Non-stationarity (agents adapt according to own behavior)
  - Decentralized adaptation requires coordination!

# MAS Example

- Cooperative MAS with two mobile agents
- Both agents need to avoid an obstacle while maintaining formation



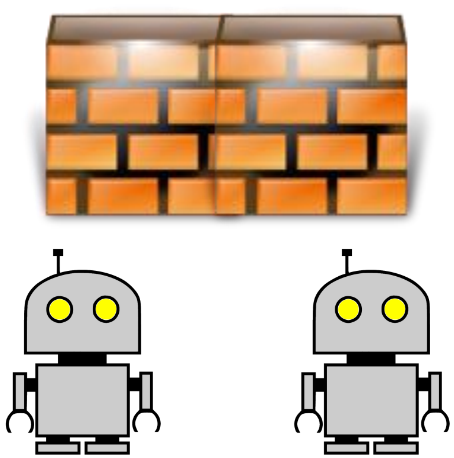

L. Busoniu et al., A comprehensive survey of multiagent reinforcement learning, IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 2008.

- Reward matrix:

| **Reward** | $left_2$ | $straight_2$ | $right_2$ |
|---|---|---|---|
| $left_1$ | **10** | -5 | 0 |
| $straight_1$ | -5 | -10 | -5 |
| $right_1$ | -10 | -5 | **10** |

# MAS Challenges

- Extremely High Complexity
- Credit Assignment (in cooperative MAS)
- Non-Stationarity
- Partial Observability



# Outline

- **An Introduction to Autonomous Systems**
  - Motivation, Definition and Challenges
  - Artificial Intelligence
- **Decision Making in Autonomous Systems**
  - Markov Decision Processes
  - Planning
  - Reinforcement Learning
- **Applications and Further Challenges**
  - Deep Reinforcement Learning
  - Combining Planning and Reinforcement Learning
  - Further Challenges

# Coming up next …

We are going to offer **new courses** in the summer semester!

- **Lecture:** Intelligent Systems
- **Practical Course:** Autonomous Systems

# Thank you!