

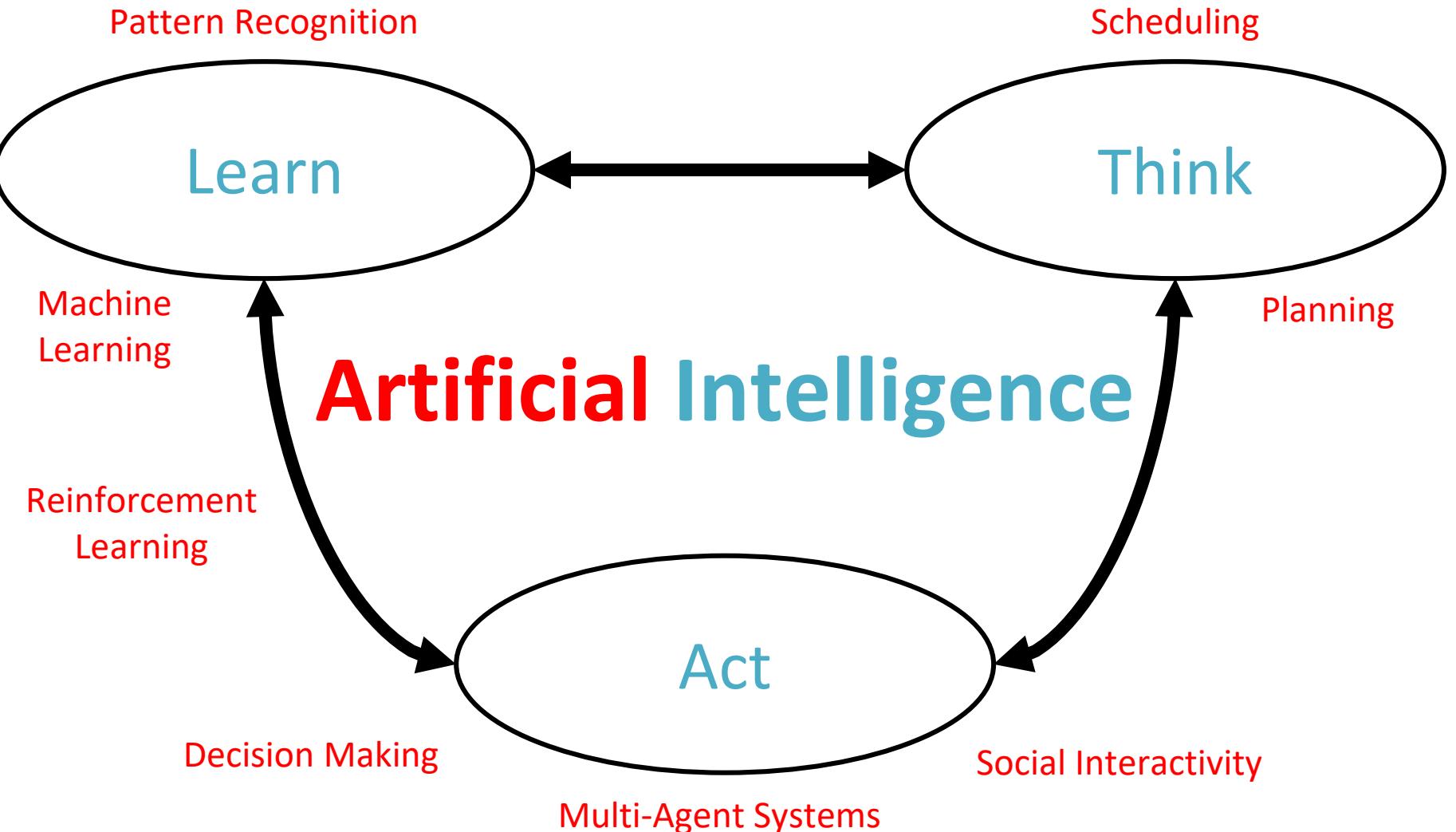
Praktikum Autonome Systeme

# Reinforcement Learning

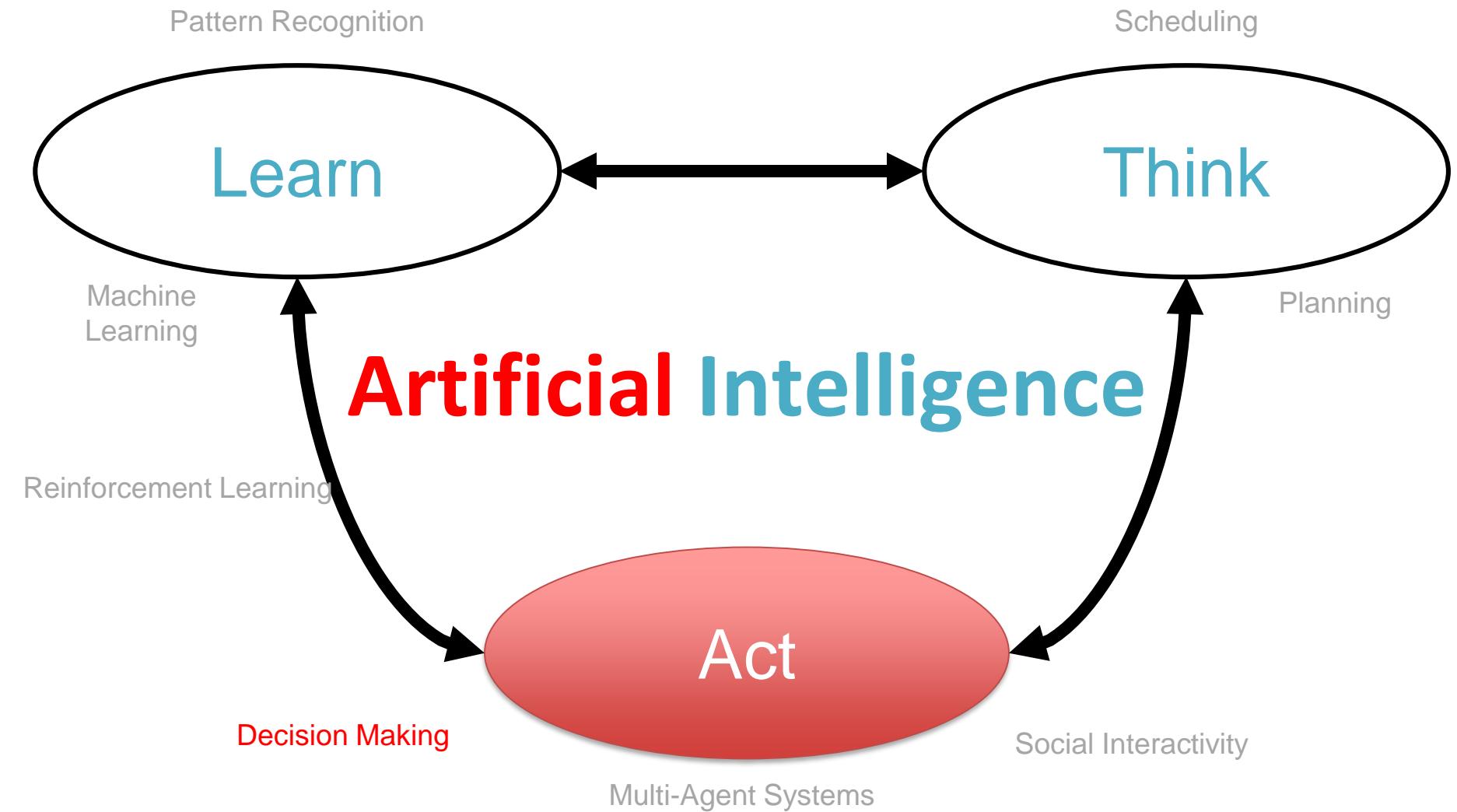
Prof. Dr. Claudia Linnhoff-Popien  
Thomy Phan, Andreas Sedlmeier, Fabian Ritz  
<http://www.mobile.ifi.lmu.de>

SoSe 2019





# **Recap: Automated Planning**

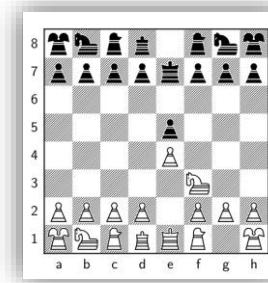


# Overview

## Multi-Armed Bandits



## Sequential Decision Making

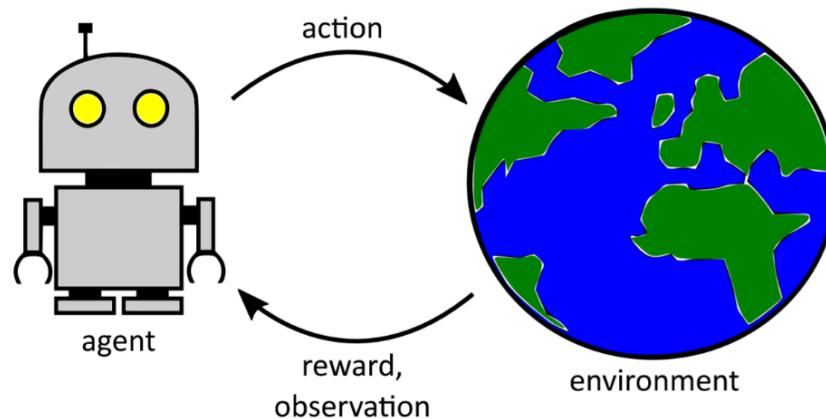


- Only a single state
- Multiple actions with possibly different reward

- Multiple states
- Multiple actions
  - with different rewards
  - and different transitions

# Sequential Decision Making

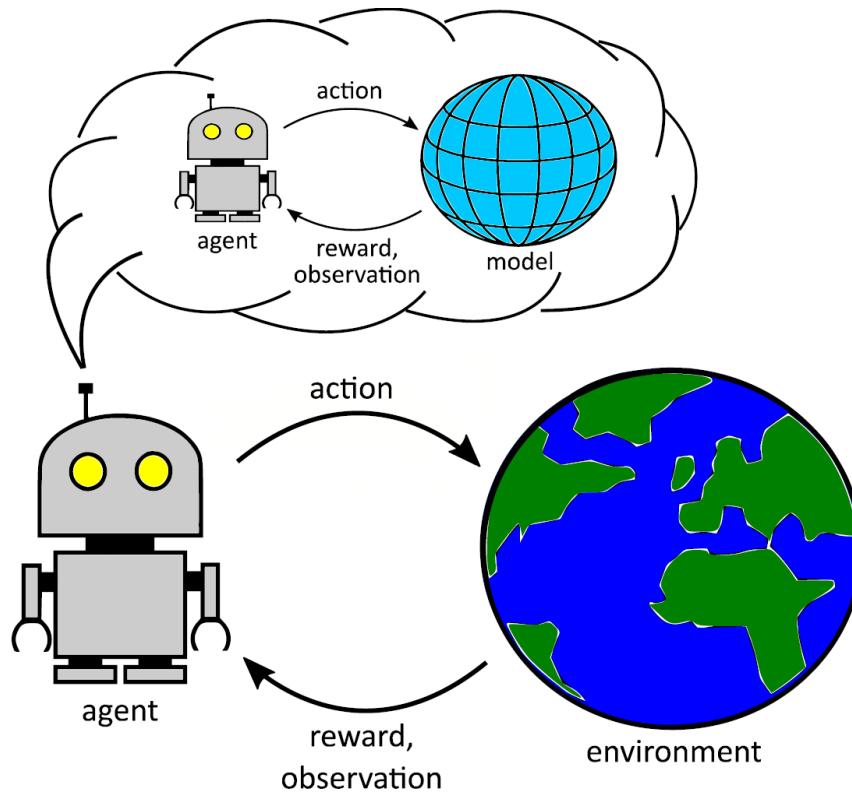
- **Goal:** Autonomously select actions to solve a (complex) task
  - time (i.e. state) is probably important
  - maximize the **expected reward** for each state



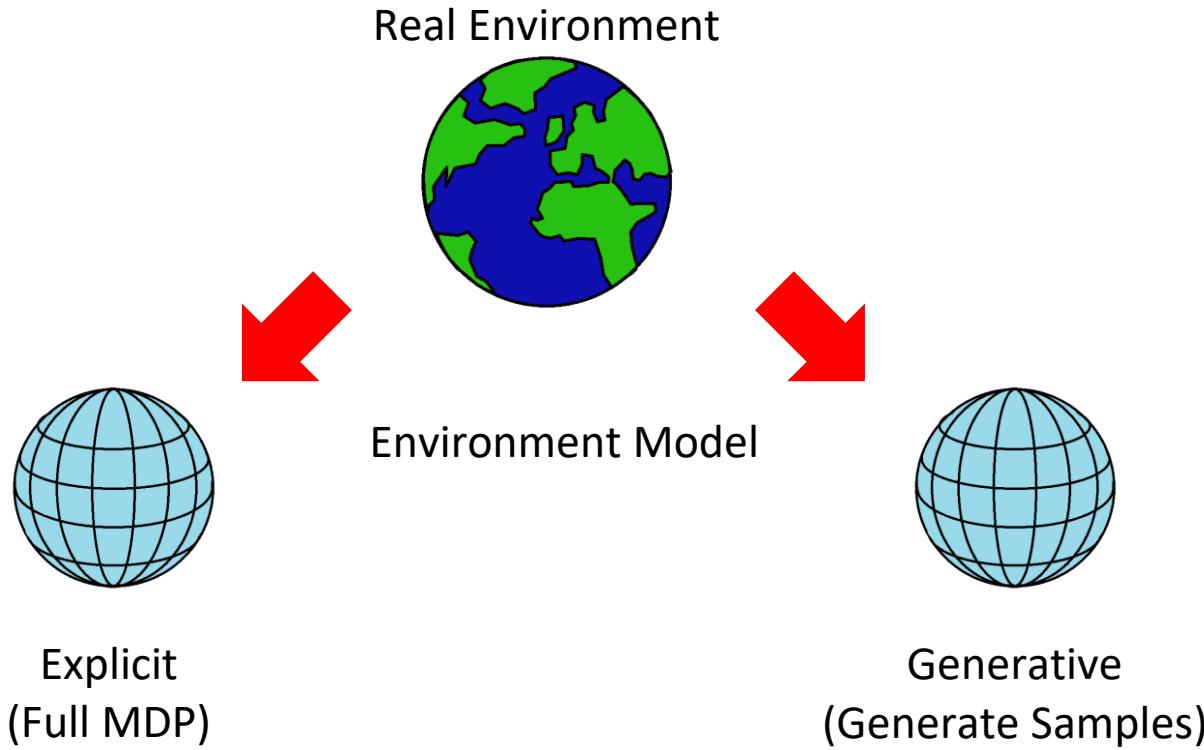
# Automated Planning

---

- **Goal:** Find (near-)optimal policies  $\pi^*$  to solve complex problems
- Use (heuristic) lookahead search on a **given model**  $\widehat{M} \approx M$  of the problem



# Explicit Model vs. Generative Model

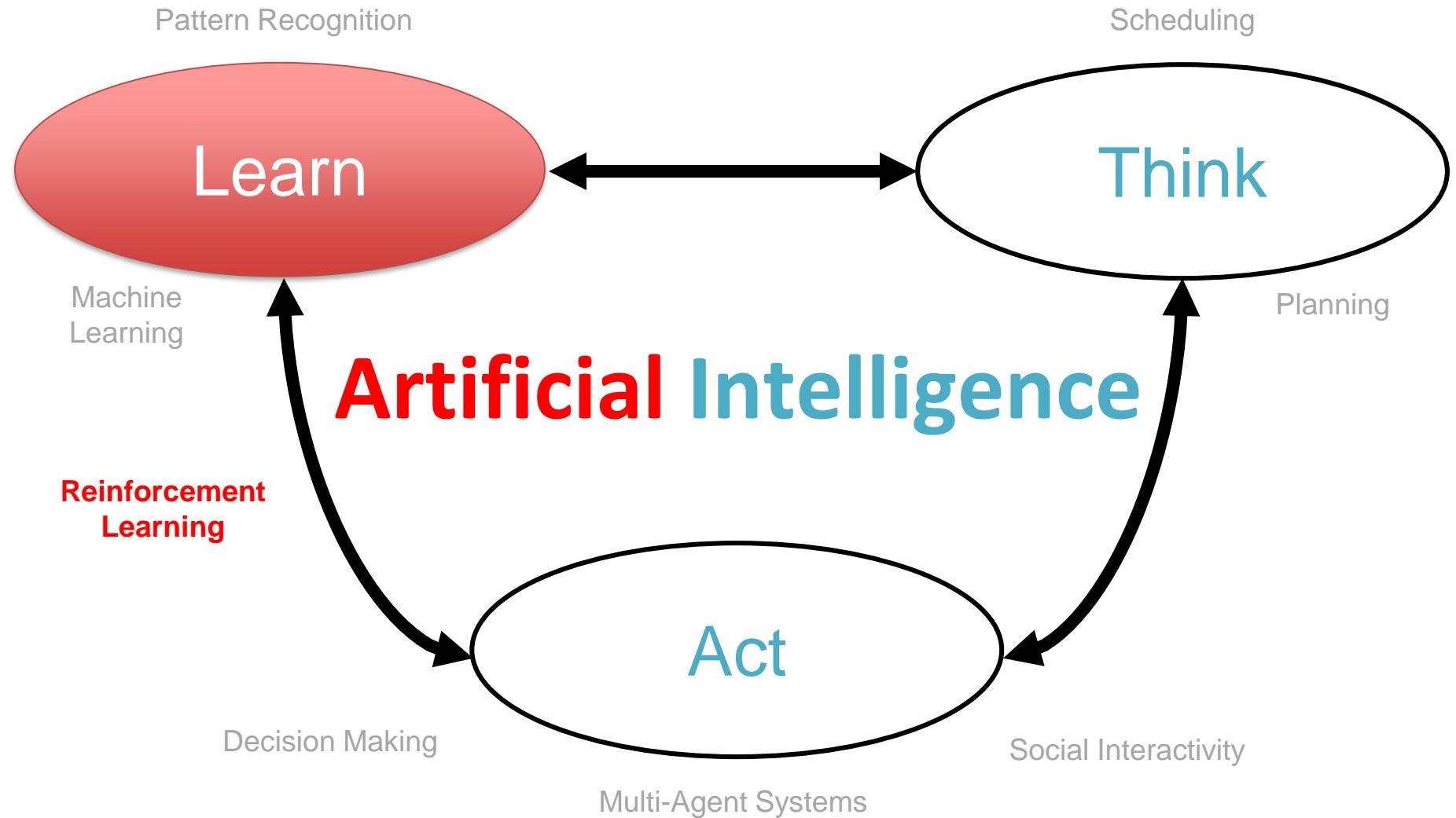


## Dynamic Programming

- Uses Transition Probabilities  
 $P(s_{t+1}|s_t, a_t)$
- Policy Iteration /  
Value Iteration

## Monte Carlo Planning

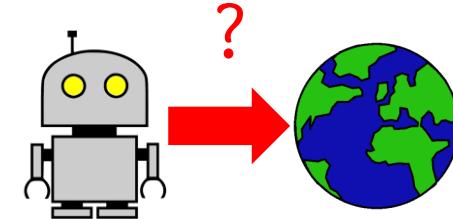
- Uses Samples from the Model to approximate  $V^*$  /  $Q^*$
- MC Rollout / MCTS



# Model Free Reinforcement Learning

---

**What if we don't have a model?**



Step1: Model Free Prediction

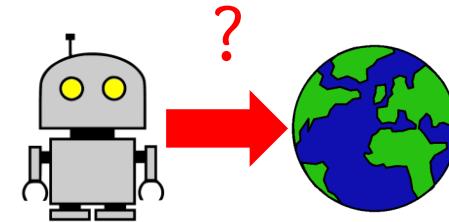
Step2: Model Free Control

# Model Free Reinforcement Learning

---

What if we don't have a model?

Step1: Model Free Prediction



“How good is the policy  $\pi$ ? ”

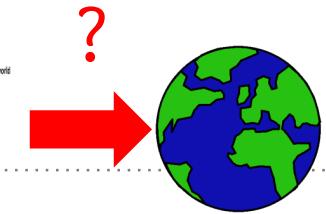
“What is the value of state  $s$  (when using  $\pi$ )? ”

- We have a policy  $\pi$  and want to evaluate it, i.e.:
- Estimate the Value Function  $V_\pi$  of an unknown MDP
- **Note: we don't yet try to improve the policy!**  
This will be done later in “model free control”

# Model Free Reinforcement Learning

## Model Free Prediction

### Evaluation Method 1: Monte Carlo (MC)



## Evaluation Method 1: Monte Carlo (MC)

- **Simply perform MC rollouts directly in the real world**
  - Complete some episodes (i.e. use policy  $\pi$ )
  - Calculate the value as the mean of the returns

### • First-Visit Monte-Carlo Policy Evaluation

Goal: evaluate a state  $s$

- The **first** time a state is visited in an episode
- Increment a counter  $N(s)++$
- Increment the total return  $S(s) \leftarrow S(s) + G_t$

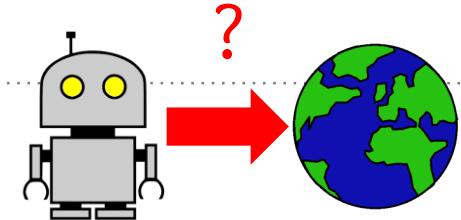
Recall:  $G_t$  is the total discounted reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{t-1} R_t$$

- Value is then estimated as  $V(s) = \frac{S(s)}{N(s)}$

# Model Free Reinforcement Learning

## Model Free Prediction



- **First-Visit Monte-Carlo Policy Evaluation**

- Value is then estimated as  $V(s) = \frac{S(s)}{N(s)}$

### Note: This is not iterative!

If we want to update  $V(s)$  after every episode, the mean can also be calculated incrementally:

- use the **incremental mean**

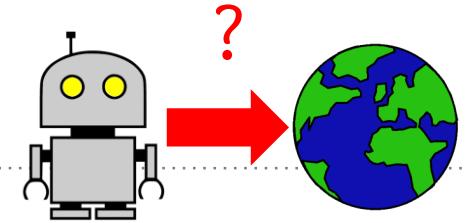
$$V(s_t) + \frac{1}{N(S_t)} (G_t - V(S_t))$$

- or the **running mean** (forgets old episodes)

$$V(s_t) + \alpha (G_t - V(S_t))$$

# Model Free Reinforcement Learning

## Model Free Prediction



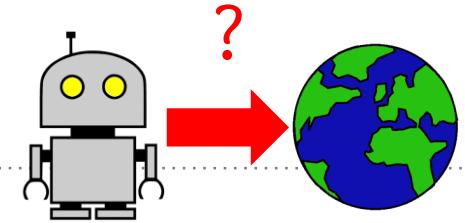
## Evaluation Method 2: Temporal difference learning (TD)

“Why learn only after the end of episodes?”

- TD learns from incomplete episodes (i.e. it *bootstraps*)
- Simplest TD: TD(0)
- Every step: Update the Value  $V(S_t)$  toward the estimated return:
  - $R_{t+1} + \gamma V(S_{t+1})$
  - $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$

# Model Free Reinforcement Learning

## Model Free Prediction

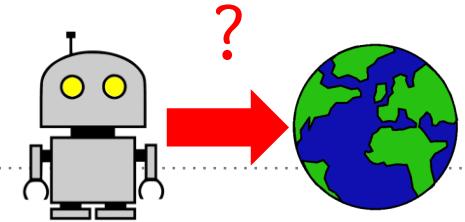


### What is better - MC or TD learning?

- No clear winner
- TD can learn before the end of episodes
  - ... or if there never are final ends!
- MC has high variance, but NO bias
  - good convergence
- TD has low variance, but is biased
  - converges, but not always when using function approximation

# Model Free Reinforcement Learning

Can we improve the policy without a model?

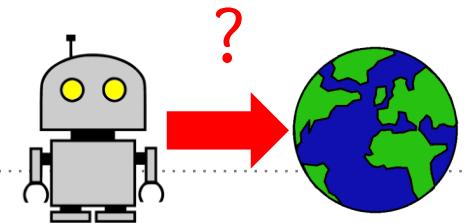


## Step2: Model Free Control

- **We want to improve a policy  $\pi$**
- Optimize the value function of an unknown MDP

# Model Free Reinforcement Learning

Why use Action-Value Functions?



## Recap: Model-Free VS Model-Based Policy Iteration

- Computing improvements using  $V(s)$  requires the transition probabilities of the MDP:

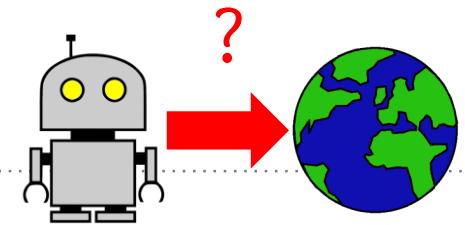
$$\pi'(s) = \operatorname{argmax}_{a \in A} R_s^a + P_{ss'}^a V(s')$$

- No model needed when using the Action-Value function  $Q$ :

$$\pi'(s) = \operatorname{argmax}_{a \in A} Q(s, a)$$

# Model Free Reinforcement Learning

Improve the policy after rollouts



## $\varepsilon$ -greedy Monte-Carlo Control

- Same idea as before, perform some rollouts
- Evaluate, i.e. calculate  $Q(S_t, A_t)$ :

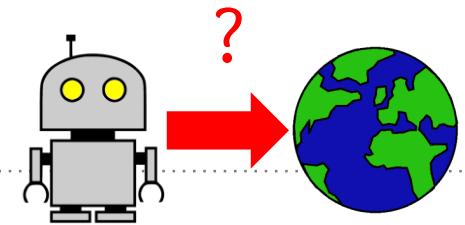
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} (G_t - Q(S_t, A_t))$$

- Improve using the new Q:

$$\pi \leftarrow \varepsilon\text{-greedy}(Q)$$

# Model Free Reinforcement Learning

Improve the policy every step



**Why not use TD instead and update every step?**

→ **Sarsa (State, Action, Reward, State', Action')**

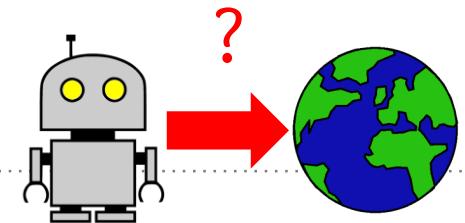
$$Q(S, A) \leftarrow Q(S, A) + \alpha \underbrace{(R + \gamma Q(S', A') - Q(S, A))}_{\text{TD target}}$$

**MC VS Sarsa:**

Empirical  $G_t$  was replaced by  
current reward R + discount \* estimated return  
from the next step (taking action a')

# Model Free Reinforcement Learning

Improve the policy every step



## N-Step Sarsa

- We can also consider more than the last reward:

$$n=1 \text{ (Sarsa)} \quad q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1})$$

$$n=2 \quad q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2})$$

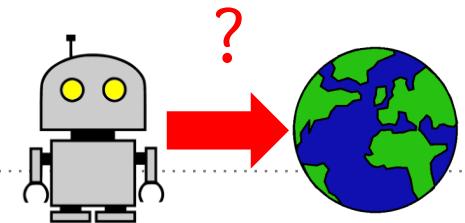
...

$$n=\infty \text{ (MC)} \quad q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n Q(S_{t+n})$$

$$Q(S, A) \leftarrow Q(S, A) + \alpha (q_t^{(n)} - Q(S, A))$$

# Model Free Reinforcement Learning

Improve the policy every step



## Sarsa: Pseudo Code

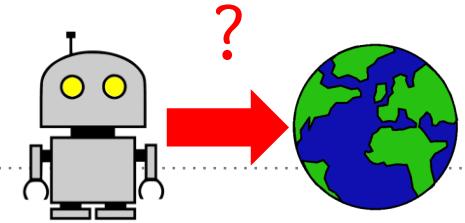
$$Q(S, A) \leftarrow Q(S, A) + \alpha (R + \gamma Q(S', A') - Q(S, A))$$

---

```
Repeat n_episode times:  
    s = reset environment  
    a = policy(s)  
    For t in max_step_times:  
        s', r, done = environment.step(a)  
        a' = policy(s')  
        td_target = r + discount * Q(s', a')  
        td_error = td_target - Q(s, a)  
        Q[s, a] = Q(s, a) + alpha * td_error  
        s = s';  
        a = a'  
        if done or t == max_step_times:  
            break
```

# Model Free Reinforcement Learning

Can we act A and learn B?

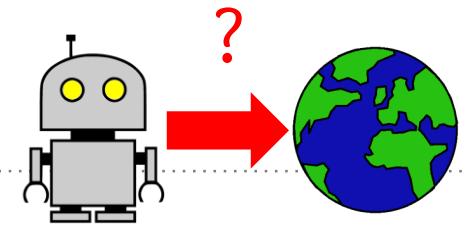


## On-Policy VS Off-Policy Learning

- Notice how both MC Control and Sarsa used the current policy to choose the next action
- We can also learn off-policy:
  - Choose the next action using one policy  $\mu(a|s)$
  - But evaluate using a different policy  $\pi$  and it's  $q_\pi(s, a)$
- This way, we can learn from observing others
- Or re-use experience collected using an old policy

# Model Free Reinforcement Learning

## Q Learning



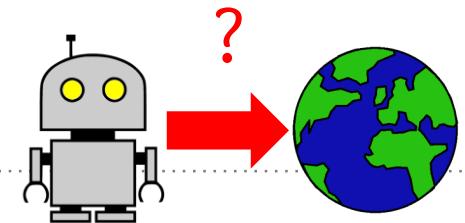
### Off-Policy learning of the action-value function: Q-Learning

$$Q(S, A) \leftarrow Q(S, A) + \alpha (R + \gamma \max_{a'} Q(S', a') - Q(S, A))$$

**Sarsa VS Q-Learning:** The successor action  $a'$  is not taken based on the behaviour policy but greedy ( $\rightarrow \max$ )

# Model Free Reinforcement Learning

## Q Learning



## Q-Learning: Pseudo Code

$$Q(S, A) \leftarrow Q(S, A) + \alpha (R + \gamma \max_{a'} Q(S', a') - Q(S, A))$$

---

```
Repeat n_episode times:  
    s = reset environment  
    For t in max_step_times:  
        a = policy(s)  
        s', r, done = environment.step(a)  
        td_target = r + discount * max(Q(s'))  
        td_error = td_target - Q(s, a)  
        Q[s, a] = Q(s, a) + alpha * td_error  
        s = s'  
        if done or t == max_step_times:  
            break
```

**Thank you!**