Reinforcement Learning

Niccolò Consigli n.consigli@studenti.unipi.it
Luca Seggiani l.seggiani@studenti.unipi.it

Università di Pisa

November 22, 2024

A new kind of learning

- Supervised learning: matching features to labels

A new kind of learning

- Supervised learning: matching features to labels
- Unsupervised learning: clustering data

A new kind of learning

- Supervised learning: matching features to labels
- Unsupervised learning: clustering data
- **Reinforcement learning**: learning from rewards

What is reinforcement learning (RL)?

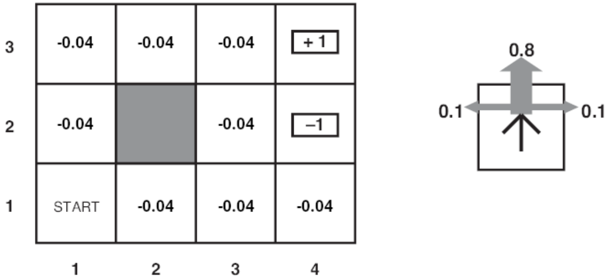
- We give the agent **rewards** based on its performance

What is reinforcement learning (RL)?

- We give the agent **rewards** based on its performance
- Markov property: we can view problems as **Markov decision processes** (MDPs)

A framework: Markov decision processes

- Actions map states to states with a probability distribution
- Transition reward: $R(s, a, s')$
- Transition probability: $P(s' \mid s, a)$



(a) (b)

Bellman's equation

Given a discount factor $\gamma \in [0, 1]$, the utility is given by:

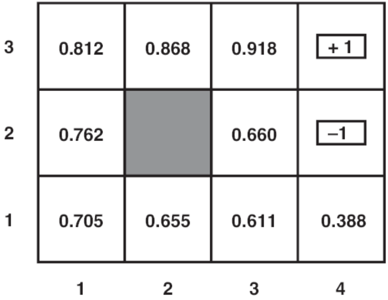
$$U(s) = \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) \left[R(s, a, s') + \gamma U(s') \right]$$

Model-based reinforcement learning

We maintain a transition model (an **MDP**):

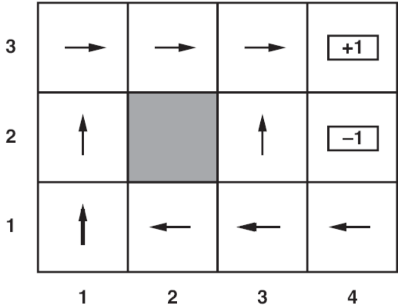
- Reward function: $R(s, a, s')$
- Probability function: $P(s' \mid s, a)$
- Utility function: U(s), maps states to **utility**

Once the model is learned, we can **maximize utility**



Model-free reinforcement learning

- Action-utility function: $Q(s, a)$ maps **actions** to **utility**
- Policy search: maps **states** to **actions**, essentialy a reflex agent



Passive reinforcement learning

We can't modify the policy, but we can figure out the utilities $U(s)$

- We use **policy iteration**:

$$U^{\pi}(s) = E\left[\sum_{t=0}^{+\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1})\right]$$

Three approaches for approximation:

- Direct estimation
- Adaptive dynamic programming
- Temporal difference learning

Direct estimation

The goal is to approximate state utility under the given policy

- Make **trials** and take the **reward-to-go** for each state

Example

$$(1, 1)$$

- This is inefficient! We can **exploit** the Markov property