Reinforcement Learning

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

Università di Pisa

November 23, 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)?

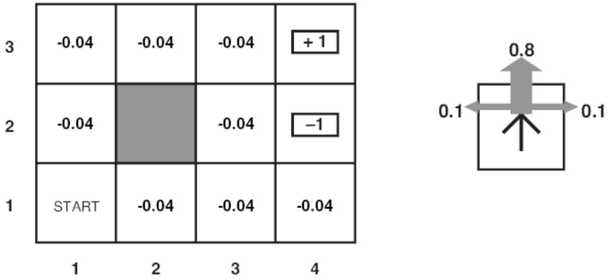
- We give the agent **rewards** based on its performance

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

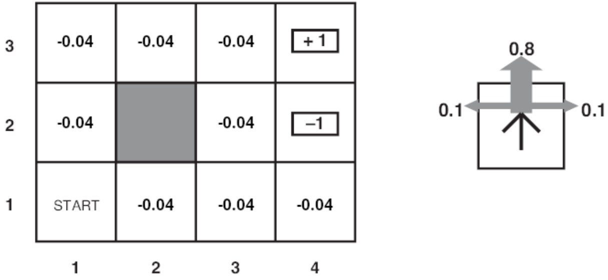
- Actions map states to states with a probability distribution
- Transition reward: $R(s, a, s')$
- Transition probability: $P(s' \mid s, a)$



(a) (b)

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**

Model-based reinforcement learning

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

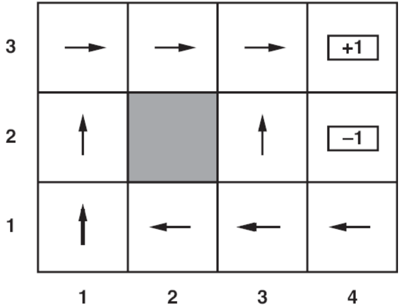
- Reward function: $R(s, a, s')$
- Probability function: $P(s' \mid s, a)$
- Utility function: U(s), maps states to **utility**

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

3	0.812	0.868	0.918	$\boxed{+1}$
2	0.762		0.660	$\boxed{-1}$
1	0.705	0.655	0.611	0.388
	1	2	3	4

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**:

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

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 (ADP)
- Temporal difference learning (TD)

Direct estimation

The goal is to approximate state utility under the given policy

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

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

We calculate utilities for $(1,2)$ given the trial:

$$(1{,}1) \xrightarrow[\text{Up}]{-0.4} (1{,}2) \xrightarrow[\text{Up}]{-0.4} (1{,}3) \xrightarrow[\text{Right}]{-0.4} (1{,}2) \xrightarrow[\text{Up}]{-0.4} (1{,}3) \xrightarrow[\text{Right}]{-0.4} (2{,}3) \xrightarrow[\text{Right}]{-0.4} (3{,}3) \xrightarrow[\text{Right}]{+1} (4{,}3)$$

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

We calculate utilities for $(1,2)$ given the trial:

$$(1{,}1) \xrightarrow[\text{Up}]{-0.4} (1{,}2) \xrightarrow[\text{Up}]{-0.4} (1{,}3) \xrightarrow[\text{Right}]{-0.4} (1{,}2) \xrightarrow[\text{Up}]{-0.4} (1{,}3) \xrightarrow[\text{Right}]{-0.4} (2{,}3) \xrightarrow[\text{Right}]{-0.4} (3{,}3) \xrightarrow[\text{Right}]{+1} (4{,}3)$$

$$U(1,2) = -0.04 - 0.04 - 0.04 - 0.04 - 0.04 + 1 = 0.8$$

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

We calculate utilities for $(1, 2)$ given the trial:

$$(1,1) \xrightarrow[\text{Up}]{-0.4} (1,2) \xrightarrow[\text{Up}]{-0.4} (1,3) \xrightarrow[\text{Right}]{-0.4} (1,2) \xrightarrow[\text{Up}]{-0.4} (1,3) \xrightarrow[\text{Right}]{-0.4} (2,3) \xrightarrow[\text{Right}]{-0.4} (3,3) \xrightarrow[\text{Right}]{+1} (4,3)$$

$$U(1,2) = -0.04 - 0.04 - 0.04 - 0.04 - 0.04 + 1 = 0.8$$

$$U(1,2)' = -0.04 - 0.04 - 0.04 + 1 = 0.88$$

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

We calculate utilities for $(1,2)$ given the trial:

$$(1,1) \xrightarrow[\text{Up}]{-0.4} (1,2) \xrightarrow[\text{Up}]{-0.4} (1,3) \xrightarrow[\text{Right}]{-0.4} (1,2) \xrightarrow[\text{Up}]{-0.4} (1,3) \xrightarrow[\text{Right}]{-0.4} (2,3) \xrightarrow[\text{Right}]{-0.4} (3,3) \xrightarrow[\text{Right}]{+1} (4,3)$$

$$U(1,2) = -0.04 - 0.04 - 0.04 - 0.04 - 0.04 + 1 = 0.8$$

$$U(1,2)' = -0.04 - 0.04 - 0.04 + 1 = 0.88$$

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

Adaptive dynamic programming

We can approximate the transition reward and probability functions (P and R) and apply **simplified Bellman's equation**:

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

When the policy is fixed, this gives a **linear equation**

Adaptive dynamic programming

We can approximate the transition reward and probability functions (P and R) and apply **simplified Bellman's equation**:

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

When the policy is fixed, this gives a **linear equation**

- Calculating P and R is easy when the environment is **fully observable**

Temporal difference learning

Still taking trials, but we **update** the utility to match frequently observed transitions

Temporal difference learning

Still taking trials, but we **update** the utility to match frequently observed transitions

Example

We calculate utilities for $(1, 3)$ given the trial:

$$(1,1) \xrightarrow[\text{Up}]{-0.4} (1,2) \xrightarrow[\text{Up}]{-0.4} (1,3) \xrightarrow[\text{Right}]{-0.4} (2,3) \xrightarrow[\text{Right}]{-0.4} (3,3) \xrightarrow[\text{Right}]{-0.04} (3,2) \xrightarrow[\text{Up}]{-0.04} (3,3) \xrightarrow[\text{Right}]{+1} (4,3)$$

Temporal difference learning

Still taking trials, but we **update** the utility to match frequently observed transitions

Example

We calculate utilities for $(1, 3)$ given the trial:

$$(1,1)\xrightarrow[\text{Up}]{-0.4}(1,2)\xrightarrow[\text{Up}]{-0.4}(1,3)\xrightarrow[\text{Right}]{-0.4}(2,3)\xrightarrow[\text{Right}]{-0.4}(3,3)\xrightarrow[\text{Right}]{-0.04}(3,2)\xrightarrow[\text{Up}]{-0.04}(3,3)\xrightarrow[\text{Right}]{+1}(4,3)$$

$$U^{\pi}(1, 3) = -0.04 + U^{\pi}(2, 3)$$

Temporal difference learning

Still taking trials, but we **update** the utility to match frequently observed transitions

Example

We calculate utilities for $(1,3)$ given the trial:

$$(1,1) \xrightarrow[\text{Up}]{-0.4} (1,2) \xrightarrow[\text{Up}]{-0.4} (1,3) \xrightarrow[\text{Right}]{-0.4} (2,3) \xrightarrow[\text{Right}]{-0.4} (3,3) \xrightarrow[\text{Right}]{-0.04} (3,2) \xrightarrow[\text{Up}]{-0.04} (3,3) \xrightarrow[\text{Right}]{+1} (4,3)$$

$$U^\pi(1,3) = -0.04 + U^\pi(2,3)$$

Then we can update with learning rate α:

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha \left[R(s, \pi(s), s') + \gamma U^\pi(s') - U^\pi(s) \right]$$

- From TD's point of view, ADP is TD with *simulated experiences*