Introduction to reinforcement learning

Milica Gašić

Dialogue Systems Group, Cambridge University Engineering Department

Course

- 8 lectures
- 1 practical session on 7th February
- Assessment: written report on coursework (upto 2000 words)
- Deadline 2nd March
- Sutton and Barto *Reinforcement learning, an introduction* second edition available at `http://incompleteideas.net/book/the-book-2nd.html`

In this course...

1. Introduction to reinforcement learning (RL)
2. Model-based and Model-free RL
3. Temporal difference (TD) methods
4. Function approximation for Value function
5. Actor-critic methods
6. Deep RL
7. Variance reduction
8. RL for continuous spaces and LQR

In this lecture...

Introduction to reinforcement learning

Learning from interaction

Markov assumption

Markov decision process

Value function

Bellman optimality equation

Different learning frameworks

Supervised

- learning from a training set of labelled examples provided by a knowledgeable teacher

Unsupervised

- Finds hidden structure in data, estimate density functions

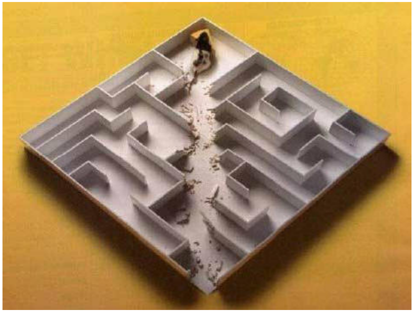
Reinforcement

- Learns from interaction and not from examples
- The goal is to maximise reward and not to find hidden structure

Learning from interaction

- Reinforcement learning involve learning what to do
- It maps solutions to actions as to maximize a numerical reward
- The agent is not told what to do but it must discover the best behaviour
- The actions that it takes affect future outcomes

Learning from interaction in practise



- Reinforcement learning in practise gives only an approximation to a true solution
- Real problem might be continuous and high dimensional

Exploration and exploitation dilemma

In reinforcement learning we have a goal-seeking agent that must simultaneously:

- **exploit** current knowledge
- **explore** new actions

The agent must try a variety of actions and progressively favour those that appear to be best.

Examples

Traditionally theoretical area, but nowadays example include:

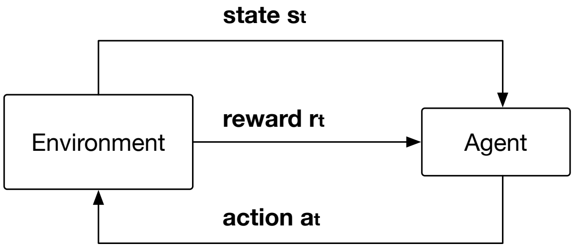
- Robot learning how to move
- Mastering the game of Go
- Teaching a dialogue system how to respond
- Online-advertisement
- Self-driving cars
- Medication dosage
- ...

Key properties of reinforcement learning

The agent in reinforcement learning

- Learns from **interaction** with the environment
- Needs to perform **planning**
- Has a **goal** or a number of sub-goals
- Must deal with **uncertain** environments
- Learns from **experience**

The agent-environment interface



- state $s_t \in \mathcal{S}$
- action $a_t \in \mathcal{A}$
- reward $r_t \in \mathbb{R}$
- policy $\pi_t(a \mid s) = p(a_t = a \mid s_t = s)$

Main elements in reinforcement learning

policy	defines behaviour of the agent
reward	defines the goal of the agent
value function	defines what is the goal in the long run, a prediction of the future reward
model	explains how the environment behaves: what are the transition probabilities for different states? It used for planning in *model-based* learning, otherwise the agent learns from trial and error in *model-free* learning

Example

A robot picks boxes and puts them on the shelf. What are states, actions and rewards?

Abstraction

Reinforcement learning offers an abstraction to the problem of goal-directed learning from interaction.
It proposes that the sensory, memory and control apparatus and the objective can be reduced to states, actions and rewards passing back and forth between the agent and the environment.

Goals and rewards

Reward hypothesis: The goal and the purpose of the agent can be thought of as the maximisation of the expected value of cumulative reward.
Question: Who computes the reward the environment or the agent?

Tasks

Episodic tasks: interaction terminates after a finite number of steps

Continuing tasks: interaction has no limit

Return

Episodic tasks: $R_t = r_{t+1} + r_{t+2} + \cdots + r_T$, where T is the final time step

Continuing tasks: $R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots$ where γ is the discount factor, $0 \leq \gamma < 1$

- for continuing tasks γ must be smaller than 1
- when $\gamma = 0$ the agent is *miopic*
- when $\gamma \to 1$ the agent is *farsighted*

Unified notation: $R_t = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}$

The Markov property

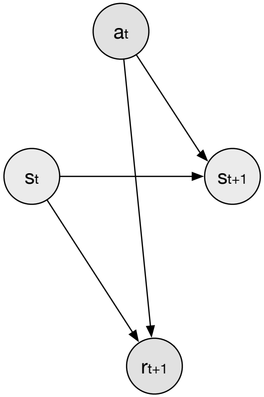
The state contains all information about the environment that is available to the agent. State representation deals with what information goes into the state.
The state satisfies the **Markov property** if it summarises past sensations compactly in such a way that all relevant information is retained. If the state satisfies the Markov property than the state at time $t + 1$ only depends on the state and the action at time t:

$$p(s_{t+1} = s', r_{t+1} = r \mid s_0, a_0, r_1, \ldots, s_{t-1}, a_{t-1}, r_t, s_t, a_t) = \\ p(s_{t+1} = s', r_{t+1} = r \mid s_t = s, a_t = a)$$

The Markov property

The best policy for choosing actions as a function of a Markov state is just as good as the best policy for choosing actions as a function of complete history.
The Markov property is important because it simplifies the policy optimisation process as the policy is simply a function of states.

Markov decision process



- $p(s', r \mid s, a) = p(s_{t+1} = s', r_{t+1} = r \mid s_t = s, a_t = a)$
- $r(s, a) = E[r_{t+1} \mid s_t = s, a_t = a] = \sum_{r \in \mathbb{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a)$

Markov decision process

Markov decision process is characterised by

state-transition probability $p(s' \mid s, a) = \sum_{r \in \mathbb{R}} p(s', r \mid s, a)$

expected reward $r(s, a, s') = \sum_{r \in \mathbb{R}} r p(r \mid s, a, s') = \frac{\sum_{r \in \mathbb{R}} r p(s', r | s, a)}{p(s' | s, a)}$

Value functions

How good is it for the agent to be in a particular state?

state-value function for policy π $V_\pi(s) = E_\pi[R_t \mid s_t = s]$

action-value function for policy π

$$Q_\pi(s, a) = E_\pi[R_t \mid s_t = s, a_t = a]$$

Bellman equation

Value functions satisfy recursive relations.

$$
\begin{aligned}
V_\pi(s) &= E_\pi[R_t \mid s_t = s] \\
&= E_\pi[\sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} \mid s_t = s] \\
&= E_\pi[r_{t+1} + \gamma \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+2} \mid s_t = s] \\
&= \sum_a \pi(a, s) \sum_{s'} \sum_r p(s', r \mid s, a) \\
&\quad (r + \gamma E_\pi[\sum_{k=0}^{T-1} \gamma^k r_{t+k+2} \mid s_{t+1} = s']) \\
&= \sum_a \pi(a, s) \sum_{s'} \sum_r p(s', r \mid s, a)(r + \gamma V_\pi(s'))
\end{aligned}
$$

Bellman equation

Bellman equation expresses the relationship between the value of a state and the value of its successor states. It averages over all possibilities weighted by their probability of occurrence.

Optimal value functions

Policy π is better or equal to policy π' iff $V_\pi(s) \geq V_{\pi'}(s)$ for every $s \in \mathcal{S}$.

$$\begin{aligned} V_*(s) &= \max_\pi V_\pi(s) \\ Q_*(s, a) &= \max_\pi Q_\pi(s, a) \\ &= E[r_{t+1} + \gamma V_*(s_{t+1}) \mid s_t = s, a_t = a] \end{aligned}$$

Bellman optimality equation

Bellman optimality equation for state-value function:

$$
\begin{aligned}
V_*(s) &= \max_a Q_*(s, a) \\
&= \max_a E_{\pi^*}[R_t \mid s_t = s, a_t = a] \\
&= \max_a E_{\pi^*}[\sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a] \\
&= \max_a E_{\pi^*}[r_{t+1} + \gamma \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+2} \mid s_t = s, a_t = a] \\
&= \max_a E_{\pi^*}[r_{t+1} + \gamma V_*(s_{t+1}) \mid s_t = s, a_t = a] \\
&= \max_a \sum_{s', r} p(s', r \mid s, a)[r_{t+1} + \gamma V_*(s')]
\end{aligned}
$$

Bellman optimality equation

Bellman optimality equation for action-value function:

$$\begin{aligned} Q_*(s,a) &= \max_a E_{\pi^*}[r_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q_*(s_{t+1}, a') \mid s_t = s, a_t = a] \\ &= \sum_{s',r} p(s', r \mid s, a)[r_{t+1} + \gamma \max_{a'} Q_*(s', a')] \end{aligned}$$

For a finite-state MDP this is a system of N non-linear equations with N unknowns. Once we get the optimal state-value function or the optimal action-value function we can retrieve the optimal policy.

Optimality and approximation

Critical aspect of the problem is the computational power available in a single time step.

tabular case
: Value functions can be expressed as table values

non-tabular case
: Value functions must be approximated using function approximations

The online nature of reinforcement learning makes it possible to approximate optimal policies in ways that put more emphasis into making close-to-optimal decisions for frequently used states than occasionally used states.

Summary

- Unlike supervised and unsupervised learning, reinforcement learning allows learning from interaction
- Elements of reinforcement learning are the policy, the reward, the value function and the model.
- The state comprises of all information that is relevant to policy optimisation. A state with Markov property only depends on the previous state and the action taken.
- Value functions satisfy Bellman equations.

Next lecture

- Dynamic programming and Monte Carlo methods