RL for MDP
Chap. 2: Learning MDP

Idris Kharroubi

LPSM
Sorbonne Université

DUFE

Outline

The problem

The Reinforcement Learning problem can be summarized by the following question:

How to act on a system to optimize some numerical reward without a complete knowledge of the behavior of the system?

Supervised learning

A possible approach to solve this question is

1. estimate the behavior of the system by using examples of response of the system to a predetermined set of actions,
2. control the system based on the knowledge.

This approach is called **supervised learning** as we use an information given by the set of 'test actions'.

In interactive problems it is often impractical to obtain examples of desired behavior that are both correct and representative of all the situations in which the agent has to act.

Unsupervised learning

Unsupervised learning: typically consists in finding structure hidden in collections of unlabeled data.

Reinforcement learning is also **different** from unsupervised learning.

Even reinforcement learning does not rely on examples of correct behavior, it cannot be seen as a specific kind of unsupervised learning.

Reinforcement learning is indeed trying to **maximize** a reward signal instead of trying to **find hidden structure**.

New paradigm

Reinforcement learning is considered as a third machine learning paradigm, in addition to supervised learning, unsupervised learning (and perhaps other paradigms as well).

One of the challenges that arise in reinforcement learning, and not in other kinds of learning, is the trade-off between exploration and exploitation.

To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward.

But to discover such actions, one needs to try actions that have not been selected before.

The agent has to exploit what is already known in order to obtain a good reward, but an exploration is also needed to make better action selections in the future

Outline

Learning the model

In the case where the transition dynamics P and the reward function r for the MDP problem are unknown, the MDP becomes a reinforcement learning problem.

It consists in finding an optimal policy π (if it exists) while simultaneously learning the unknown P and r. The learning of P and r can be either explicit or implicit. This leads respectively to model-based and model-free RL.

We introduce in the sequel some standard RL terminology.

Agent-environment Interface

In RL, the learner or the decision-maker is called the **agent**.

The **environment** is the world where the agent operates/interacts. Interaction happens at times, $n = 0, 1, 2, 3, \ldots$ as follows.

- At the beginning of each time step n, the agent receives some representation of the environment's state, $s_n \in S$ and selects an action $a_n \in A$.
- At the end of this time step, in part as a consequence of its action, the agent receives a numerical reward r_t and a new state s_{t+1} from the environment.

The tuple (s_n, a_n, r_n, s_{n+1}) is a **sample** at time n and $h_n := \{(s_u, a_u, r_u, s_{u+1})\}_{u=0}^{n}$ is **the history** or **experience** up to n.

An RL algorithm is a finite sequence of well-defined policies for the agent to interact with the environment.

Exploration vs Exploitation

To maximize the accumulated reward over time, the agent learns to select the actions based on the past experiences (exploitation) and/or by trying new choices (exploration).

Exploration provides opportunities to improve performance from the current sub-optimal solution to the ultimate globally optimal one.

However, it is time consuming and computationally expensive.
On the other side, pure exploitation, though easy to implement, tends to yield sub-optimal global solutions.

Therefore, an appropriate trade-off between exploration and exploitation is crucial in designing RL algorithms to improve the learning and the optimization performance.

Simulator

In the procedure described above on how the agent interacts with the environment, the agent does so in an online fashion.

Namely, the initial state at time step $n+1$, s_{n+1} is the state that the agent moves to after taking action at from state s_n. This is a challenging setting since an efficient exploration scheme is needed.

Some results assume access to a simulator which allows the algorithm to query arbitrary state-action pairs and return the reward and the next state.

This allows the agent to "restart" the system at any time.
That is, the initial state at time step $n+1$ does not need to be the state that agent moves to after taking action from state s_n. This assumption significantly alleviates the difficulty of exploration.

Randomized Policy vs Deterministic Policy

We shall consider randomized policies instead of deterministic ones.

This means that at time n the strategy is not only a deterministic function $f_n(s_n)$ of the state but might be given by $\pi_n(s_n)$ where $\pi_n : S \to \mathcal{P}(A)$ is a random function.

This kind of strategy is also known in the control literature as a relaxed control and in game theory as a mixed strategy.

Despite possible existence of deterministic optimal policy, most of the RL algorithms adopt randomized policies to encourage exploration when RL agents are not certain about the environment.

Outline

Why evaluating performance?

As it is not possible to solve MDPs analytically in general we need to discuss numerical algorithms to determine the optimal policies.

To measure the efficiency of the different proposed algorithms we will need a measure of their performance.

There are classical types of performance measures for RL algorithms.

Criteria

- **Sample complexity.** Sample complexity is defined as the total number of samples required to find an approximately optimal policy.
- **Rate of convergence.** The rate of convergence calculates the number of iterations needed to reach a given accuracy while ignoring the number of samples used within each iteration.
- **Regret analysis.** The regret of a given policy π is defined as the difference between the cumulative reward of the optimal policy and that gathered by π.
- **Asymptotic convergence.** In addition to sample complexity and regret analysis discussed above, asymptotic convergence is another weaker convergence measure, which requires the error term to decay to zero as N goes to infinity (without specifying the order of N). This is often a first step in the theoretical analysis of RL algorithms.

Outline

RL components

An RL algorithm includes one or more of the following components:

1. a representation of a value function that provides a prediction of how good each state or each state/action pair is,
2. a direct representation of the policy $\pi(s)$ or $\pi(s, a)$,
3. a model of the environment (the estimated transition function and the estimated reward function) in conjunction with a planning algorithm (any computational process that uses a model to create or improve a policy).

Value and policy based methods

The first two components are related to what is called model-free RL. When the latter component is used, the algorithm is referred to as model-based RL.

Policy-based methods explicitly build a representation of a policy and keep it in memory during learning.

In our discussion of methodology, we focus on model-free RL algorithms for MDP.

In particular, we introduce some classical value-based and policy-based methods.