Markov Decision Processes

Elena Zanini

1 Introduction

Uncertainty is a pervasive feature of many models in a variety of fields, from computer science to engineering, from operational research to economics, and many more. It is often necessary to solve problems or make decisions without a comprehensive knowledge of all the relevant factors and their possible future behaviour. In many situations, outcomes depend partly on randomness and partly on an agent decisions, with some sort of time dependence involved. It is then useful to build a framework to model how to make decisions in a stochastic environment, focusing in particular on Markov processes. The latter are characterised by the fact that the way they evolve in the future depends only on their present state, such that each process is independent of any events from the past. A variety of important random systems can be modelled as Markov processes, including, but not limited to, biological systems and epidemiology, queuing systems, financial and physical systems.

Due to the pervasive presence of Markov processes, the framework to analyse and treat such models is particularly important and has given rise to a rich mathematical theory.
This report aims to introduce the reader to Markov Decision Processes (MDPs), which specifically model the decision making aspect of problems of Markovian nature. The structure of its body is as follows. Section 2 provides a common formal definition for MDPs, together with the key concepts and terminology. Section 3 describes possible approaches to handle these models. First,we give a brief outline of the main two exact methods that have been used in the past, and the different perspectives they arise from. Then, in Section 3.2 we presents a concise overview of the most commons extensions of these models to cases of non-deterministic nature, accompanied by references for further readings. Finally, Section 4 introduces a prevalent issue in the field, namely optimal learning.

As previously mentioned, we only aim to equip the reader with the necessary notions to understand the general framework from MDPs and some of the work that has been developed in the field. Note that Putermans book on Markov Decision Processes [11], as well as the relevant chapter in his previous book [12] are standard references for researchers in the field. For readers to familiarise with the topic, *Introduction to Operational Research* by Hillier and Lieberman [8] is a well known starting text book in O.R. and may provide a more concise overview of the topic. In particular, they also offer a very good introduction to Markov Processes in general, with some specific applications and relevant methodology. A more advanced audience may wish to explore the original work done on the matter. Bellman's [3] work on Dynamic Programming and recurrence sets the initial framework for the field, while Howards [9] had a fundamental role in developing the mathematical theory. References on specific aspects are provided later in the relevant sections.

Finally, due to space restriction, and to preserve the flow and cohesion of the report, applications will not be considered in details. A renowned overview of applications can be found in White's paper, which provides a valuable survey of papers on the application of Markov decision processes, "classified according to the use of real life data, structural results and special computational schemes"[15]. Although the paper dates back to 1993 and much research has been developed since then, most of the ideas and applications are still valid, and the reasoning and classifications presented supports a general understanding of the field. Puterman's more recent book [13] also provides various examples and directs to relevant research areas and publications.

2 Definition

Although there are different formulations for MDPs, they all show the same key aspects. In this section, we follow the notation used by Puterman [12], which provides a fairly concise but rigorous overview of MDPs.

To start with, we have "a system under control of a decision maker [which] is observable as it evolves through time". A few characteristics distinguish the system at hand:

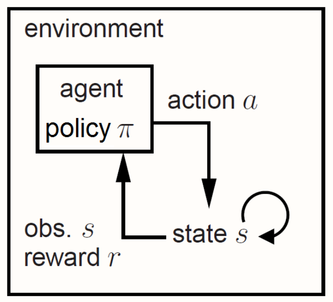

Figure 1: MDP structure

- a set **T** of *decision epochs* or *stages* t at which the agent observes the state of the system and may make decisions. Different characteristics of the set T will lead to different classifications of processes (e.g. finite/infinite, discrete/continuous).
- The *state space* **S**, where $\mathbf{S_t}$ refers to the states for a specific time t.
- The *action set* **A**, where in particular $\mathbf{A_{s,t}}$ is the set of possible actions that can be taken after observing state s at time t.
- The *transition probabilities*, determining how the system will move to the next state. Indeed, MDPs owe their name to the transition probability function, as this exhibits the markov property[1]. In particular, $p_t(j|s,a)$ defines the transition to state $j \in S_{t+1}$ at time $t+1$, and only depends on the state s and chosen action a at time t.
- The *reward function*[2], which determines the immediate consequence for the agent's choice of action a while in state s. In some cases, the value of the reward depends on the next state of the system, effectively becoming an "expected reward". Following simple probability rules, this can be expressed as
$$r_t(s,a) = \sum_{j \in S_{t+1}} r_t(s,a,j) p_t(j|s,a), \tag{1}$$
where $r_t(s,a,j)$ is the relevant reward in case the system will next be in state j.

A Markov Decision Process can then be defined by the quintuple $(T, S_t, A_{s,t}, p_t(j|s,a), r_t(s,a))$, with distinctions between types of MDPs relying on different assumptions. Note that the above serves as a general definition for the reader to grasp the key aspects of any MDP, and understand the reasoning behind the main approaches used, as presented in Section 3. The reader should refer to the suggested references for a more detailed review of different classes of MDPs, and specific developments of solution methods.

3 Algorithms

The objective of MDPs is to provide the decision maker with an optimal *policy* $\pi : S \rightarrow A$. Policies are essentially functions that rules, for each state, which action to perform. An optimal policy will optimise

[1]Some generalisation are available, where the transition and the reward functions may not only depend on the current state. Often, these can be seen as Stochastic Decision Processes.

[2]This "reward" takes both positive and negative values, indicating an income and a cost respectively.

(either maximise or minimise) a predefined objective function, which will aim to achieve different targets for different formulations. Then, the optimisation technique to use depends on the characteristics of the process and on the "optimality criterion" of choice, that is the preferred formulation for the objective function. MDPs with a specified optimality criterion (hence forming a sextuple) can be called *Markov decision problems*. Although some literature uses the terms *process* and *problem* interchangeably, in this report we follow the distinction above, which is consistent with the work of Puterman referenced earlier. For simplicity, we present the algorithms assuming the state and action spaces, S and A, are finite. Note that most concepts are applicable, given relevant adaptations, to general cases too, more details can be found in the given references.

In most situations, a policy π is needed that maximises some cumulative function of the rewards. A common formulation is the *expected discounted sum* over the given time horizon, which may be *finite* or *infinite*. The following formulation can be used:

$$\mathbb{E}\left[\sum_{t=0}^{h} \gamma^t R_t\right],$$

where $0 \leq \gamma < 1$ is the discount rate. Note that h will in fact be infinity in the infinite horizon case. The formula can also be adapted to situations where the reward depends not only on the time, but also on the current or future state of the system, the action chosen, the policy, or all of the above.

An important hypothesis, although still unproven, unifies all goals and purposes to the form given above, stating they may all be formulated as a maximisation of a cumulative sum of rewards [14].

A variety of methods have been developed during the years. Among these, exact methods work within the linear and the dynamic programming framework. We focus on the latter, and present in the following section two most influential and common exact methods available, namely value iteration and policy iteration. Section 3.2 will then consider an approximate method to approach non-deterministic cases. In both cases, we only provide a brief conceptual overview of the approaches considered.

3.1 Exact Methods: Dynamic Programming

As mentioned before, MDPs first developed from Bellman's work on dynamic programming [3], so it's not surprising that they can be solved using techniques from this field.

First, a few assumptions need to be made. Let the state transition function P and the reward function R be known, so that the aim is to obtain a policy that maximizes the expected discounted reward. Then let us define the *value* [3] V_π for policy π starting from state s as

$$V_\pi(s) := \mathbb{E}_\pi\left[\sum_{t=1}^{h-t} r_{t+i} \;\middle|\; s_t = s\right],$$

which gives the overall expected value of the chosen policy from the current to the final state (note that in this case we assume a finite-horizon of length h).

The standard algorithms proceed iteratively (although versions using systems of linear equations exist) to construct the following two vectors $V(s) \in \mathbb{R}$ and $\pi(s) \in A$, defined as follows:

- the optimal actions

$$\pi := \arg\max_a \left\{\sum_{s'} p_t(s'|s,a)\left[r_t(s,a,s') + \gamma V(s')\right]\right\}; \tag{2}$$

- the discounted sum of the rewards

$$V(s) := \sum_{s'} p_t\left(s'|s,\pi(s)\right)\left[r_t(s,\pi(s),s') + \gamma V(s')\right]. \tag{3}$$

[3]Some references sometimes refer to this as *utility*.

Note that $V(s)$ is the iterative version of the so-called *Bellman equation*, which determines a necessary condition for optimality to be obtained.

The main DP algorithms to solve MDP differ in the order they repeat such steps, as we briefly see in Sections 3.1.1 and 3.1.2. In both cases, what matters is that, given certain conditions on the environment, the algorithms are guaranteed to converge to optimality [3, 9, 13].

3.1.1 Value iteration

First proposed in by Bellman in 1957 [3], the *value iteration* approach, also called backward induction, does not compute the policy π function separately. In its place, the value of $\pi(s)$ is calculated within $V(s)$ whenever it is needed.
This iterative algorithm calculates the expected value of each state using the value of the adjacent states until convergence (that is, the improvement in value between two consecutive states is smaller than a given tolerance τ). As per usual in iterative methods, smaller tolerance values insure higher precision in results.
The algorithm follows the following logic shown on the right, and terminates when the optimal value is obtained.

```
Algorithm 1: Value Iteration (VI)

Initialise $V$ (e.g. $V = 0$) - only
needed for the first $\delta$ computation
to be performed

Repeat
$\delta = 0$
for each state $s$
    $V(s+1) = \max_{a \in A} \sum_{s'} p_t(s'|s,\pi(s)) [r_t(s,\pi(s),s') + \gamma V(s')]$
    $\delta =$ max$(\delta, |V(s+1) - V(s)|$
Until convergence ($\delta < \tau$)
```

3.1.2 Policy iteration

The body of research developed by Howard [9] first sparked from the observation that a policy often becomes exactly optimal long before value estimates have converged to their correct values. The *policy iteration* algorithm focuses more on the policy in order to reduce the number of computations needed whenever possible. First, an initial policy is chosen, often by simply maximising the overall policy value using rewards on states as their value.

Two steps follow:

1. *policy evaluation*, when we calculate the value of each state given the current policy until convergence;

2. *policy improvement*, to update the policy using eq.(3) until an improvement is possible.

The resulting iterative procedure is shown on the right.

The algorithm terminates when the policy stabilizes.

```
Algorithm 2: Policy Iteration (PI)

1. Initialise $V$ (e.g. $V(s_0)$)
   and compute $\pi(s) \in A$

2. Policy evaluation
   Repeat
   $\delta = 0$
   for each state $s$
       $V(s+1) := \sum_{s'} p_t(s'|s,\pi(s)) [r_t(s,\pi(s),s') + \gamma V(s')]$
       $\delta =$ max$(\delta, |V(s+1) - V(s)|$
   Until convergence ($\delta < \tau$)

3. Policy Improvement
   for each state $s$
       $\pi := \arg\max_a \{\sum_{s'} p_t(s'|s,a) [r_t(s,a,s') + \gamma V(s')]\}$
       if $\pi(s) = \pi(s-1)$
         stable policy found
       else
         go back to 2.
```

3.2 Handling uncertainty: POMDP and approximate methods

We assumed before that the state s is known, as well as the action distribution function, in order for $\pi(s)$ and $V(s)$ to be calculated. This is not often the case in real life applications.

A special class of MDPs called partially observable Markov decision process (POMDP) deals with cases where the current state is not always known. Although these are outside the scope of this project, we refer the reader to a noteworthy online tutorial [6], providing both a simplified overview of the subject and references to the key publications in the area.

Another kind of uncertainty arises when the probabilities or rewards are **unknown**. In these situations, the ideas presented can be used to develop approximate methods. A popular field concerned with such framework, especially in artificial intelligence, is that of *reinforcement learning*. This section is based on the publication of Sutton and Barto [14], whose work is an essential piece of research in the field and introduces RL algorithms. For a more extensive treatment of RL algorithms, the work of Bertsekas and Tsitsiklis [4] is a standard reference.

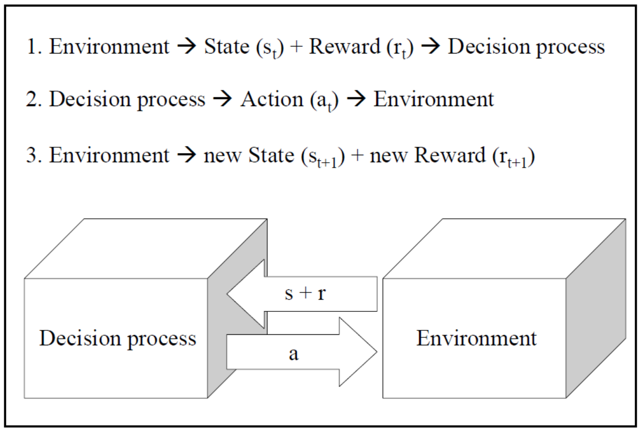
1. Environment → State (s_t) + Reward (r_t) → Decision process
2. Decision process → Action (a_t) → Environment
3. Environment → new State (s_{t+1}) + new Reward (r_{t+1})
s + r
a
Decision process
Environment

Figure 2: The figure shows model of a reinforcement learning system. First the decision process observes the current state and reward then the decision process performs an action that effects the environment. Finally the environment returns the new state and the obtained reward.[2]

Reinforcement learning methods often rely on representing the policy by a state-action value function $Q : S \times A \to \mathbb{R}$, where

$$Q(s,a) = \sum_{s'} P_a(s,s')(R_a(s,s') + \gamma V(s')).$$

The policy π from before is then just $\pi(s) := \arg\max_a Q(s,a)$.
Function Q essentially describes the scenario of where we choose action a, to then either continue optimally or according to the current policy. Although this function is also unknown, the key to reinforcement learning techniques is their ability to learn from experience. In practice, that corresponds to exploiting the information from the past and upcoming state of the system, that is from the triplets (s, a, s').

Similar algorithms to value iteration can then be performed. Define a *Q-learning update* to be

$$Q(s,a) := (1-\beta)\, Q(s,a) + \beta \left[R_t(s,a) + \gamma \max_{a'} Q(s',a') \right],$$

where $0 < \beta < 1$, the update is a given *learning rate*. We can then follow a similar approach to that in Section 3.1.1, known as *Q-learning*. Essentially, use the *Q-learning update* in place of the value function step, so to take into account the probabilistic framework.
As far as transition probabilities are concerned, a simulation approach may be used to obtain them, so that explicit specification are no longer necessary. This is a key distinguishing feature from the value and policy iteration algorithms, where transition probabilities are needed.
One of the advantages of reinforcement learning algorithms is that they can handle large MDPs where exact methods become infeasible, while also coping with a less comprehensive knowledge of the process.
A crucial issue agents are confronted with in RL is the trade-off between exploration and exploitation. Section 4 aims to summarise the main concept involved, and highlight areas where further research has been and still is undertaken.

4 Further research: exploration and exploitation

The trade-off between exploration and exploitation is a key issue in reinforcement learning problems. Say we want to maximise the rewards, then we could

- always choose the action a with highest expected Q-value for the current state s, or
- explore a non-optimal action with lower Q-value of higher uncertainty.

The latter approach may help to converge faster to the optimal solution, as an action of lower expected Q-value may ultimately bring a bigger reward than the current best-known one. Such a choice may potentially be disadvantageous, as this sub-optimal action may diminish the overall value. Even when that is the case, such an approach still allows us to learn more about the environment. Ultimately, by increasing our understanding of the environment, we may actually take better actions with more certainty.

Note that the research of an optimal balance between exploration and exploitation is an active field of research, due to the consequences of such choices in larger real world applications. Finding the right approach is especially important when some of the steps above are highly computationally expensive. This tutorial [7] provides an excellent overview of such issues, with a specific focus on dynamic programming.

5 Conclusions

Markov decision processes are essentially Markov chains with an immediate-cost function, and can be used to model a variety of situations where a decision maker has partial control over the system. They have a large number applications, both practical and theoretical, and various algorithms have been developed to solve them. In Section 2 we presented a formal definition of MDP, while Section 3 aims to introduce the main concepts that are at core of most approaches, the policy and the value function used in optimality equations. We then focused specifically the main exact methods that have been developed within a dynamic programming framework - value and policy iteration. We present a brief overview of the iterative procedure they involve in Section 3.1.1 and 3.1.2 respectively. We then consider cases where some probabilistic and uncertainty factors come into play. In particular, a popular subject with some non-deterministic characteristics is that of reinforcement learning, where systems can be formulated as a Markov decision processes. After a brief overview of this field, we draw attention to a common issue that arises from such conditions, namely the trade-off between exploration and exploitation. To conclude, we notice how such a issue is particularly important in highly computationally demanding environments, so that further efforts have and should been devoted to progress the research in the area.

References

[1] http://people.orie.cornell.edu/pfrazier.

[2] E. Andreasson, F. Hoffmann, and O. Lindholm. To collect or not to collect? machine learning for memory management. In *Java Virtual Machine Research and Technology Symposium*, pages 27–39, 2002.

[3] R. Bellman. A markovian decision process. Technical report, DTIC Document, 1957.

[4] D. P. Bertsekas and J. N. Tsitsiklis. Neuro-dynamic programming. 1996.

[5] A. N. Burnetas and M. N. Katehakis. Optimal adaptive policies for markov decision processes. *Mathematics of Operations Research*, 22(1):222–255, 1997.

[6] A. R. Cassandra. Partially observable markov decision processes. www.pomdp.org.

[7] P. I. Frazier. Learning with dynamic programming. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

[8] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research.* McGraw-Hill, 7th edition, 2005.

[9] R. Howard. Dynamic programming and markov decision processesmit press. *Cambridge, MA*, 1960.

[10] W. B. Powell and P. Frazier. Optimal learning. *Tutorials in Operations Research: State-of-the-art Decision-making Tools in the Information-intensive Age*, 2, 2008.

[11] M. L. Puterman. Markov decision processes. 1994. *Jhon Wiley & Sons, New Jersey.*

[12] M. L. Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[13] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*, volume 414. John Wiley & Sons, 2009.

[14] R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning.* MIT Press, 1998.

[15] D. J. White. *Markov decision processes.* John Wiley & Sons New York, NY, 1993.