École polytechnique fédérale de Lausanne

Disopt

Bachelor project

Reinforcement learning and robot navigation

Student:
Charles Dufour

Supervisors:
Jonas Racine
Prof. Friedrich Eisenbrand

Contents

1 Introduction

Explain that the agent has to maximize the expected reward after introducing RL and MDP (Xia, 2015)

2 MDP : Markov decision processes

Markov's decision processes are a tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, T, R, \gamma\}$:

- a set of states : $\mathcal{S} = \{s_0, s_1, \ldots, s_n\}$
- a set of actions : $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$
- a transition function : $T(s, a, s') \sim Pr(s' \mid a, s) \quad s, s' \in \mathcal{S}$ which gives the state transition probabilities
- a reward function : $R : \mathcal{S} \mapsto \mathbb{R}$
- a discount factor $\gamma \in [0, 1)$

The discount factor γ helps making our learning agent more or less far-sighted : γ can be interpreted as the relative importance it will give to future rewards compared to immediate.

$\mathcal{M}$ is a MDP if it respects the Markovian property which is that the transitions only depends on the current state and action :

$$\mathbb{P}(s_{n+1} \mid a_n, s_n, a_{n-1}, s_{n-1}, \ldots) = \mathbb{P}(s_{n+1} \mid a_n, s_n)$$

When an agent is learning in a MDP, what it observes is a sequence of states, actions and rewards: suppose the agent is in the state s_0 and chooses action a_1 and then end up in state s_1 with reward r_1; then the sequence observed is of the form : $s_0, a_1, s_1, r_1, a_2, s_2, r_2 \ldots$

Markov decisions processes are usually represented with graph : the nodes are the states and the directed edges from a node Q are the actions an agent can choose to make while being in state Q, which will bring the agent in the state represented by the node at the endpoint of the edge as we can see in figure 2.

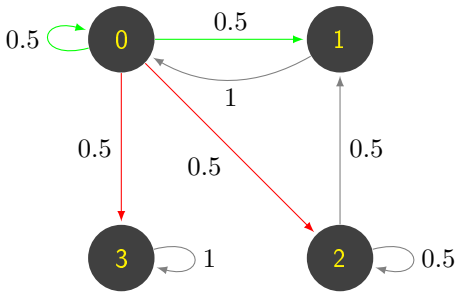

Figure 1: example of a graph representing a Markov Decision process, there are two actions possible from node 0 which takes to different states with some probability

3 Policies and Value functions

A policy : π is a stochastic mapping from states to action:

$$\pi : \mathcal{A} \times \mathcal{S} \mapsto [0, 1] \quad \text{s.t.} \quad \sum_a \pi(a \mid s) = 1 \quad \forall s \in \mathcal{S}$$

A policy is a formalisation of the decision making process : in each state, we follow the policy to decide which action to choose with some probability: $\pi(a \mid s)$ is the probability that $A_{t+1} = a$ if $S_t = s$.Hence maximizing the reward means finding a good policy.

We need to quantify the reward the agent has to maximize, so we define the return as :

$$G_t = r_{t+1} + \gamma r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = r_{t+1} + \gamma G_{t+1}$$

Now we need a way to compare policy between them : for that we use the *value functions* : the state value and the state-action value under a certain policy π. Intuitively they measure how good it is to be in a state (or respectively to take a specific action while being in a state) if we follow the policy afterwards.

Then we can define the value of taking action a in state s while following the policy π as the expected return we would receive if we follow π:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi\left[r_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\ &= \mathbb{E}_\pi\left[r_{t+1} \mid S_t = s\right] + \gamma \mathbb{E}_\pi\left[G_{t+1} \mid S_t = s\right] \\ &= \sum_a \pi(a \mid s) \sum_{s'} \mathbb{P}(s' \mid a, s) R(s') \\ &+ \gamma \sum_a \pi(a \mid s) \sum_{s'} \mathbb{P}(s' \mid a, s) \mathbb{E}_\pi\left[G_{t+1} \mid S_{t+1} = s', S_t = s\right] \end{aligned}$$

Using the Markovian property, $\mathbb{E}_\pi\left[G_{t+1} \mid S_{t+1}, S_t\right] = \mathbb{E}_\pi\left[G_{t+1} \mid S_{t+1}\right]$ so we get :

$$\begin{aligned} v_\pi(s) &= \sum_a \pi(a \mid s) \sum_{s'} \mathbb{P}(s' \mid a, s) \left[R(s') + \gamma \mathbb{E}_\pi\left[G_{t+1} \mid S_{t+1} = s'\right]\right] \\ &= \sum_a \pi(a \mid s) \sum_{s'} \mathbb{P}(s' \mid a, s) \left[R(s') + \gamma v_\pi(s')\right] \end{aligned} \tag{1}$$

And the value of taking action a while being in state s under a policy π following roughly the same method:

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}\left[G_t \mid S_t = s, A_t = a\right] \\ &= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid S_t = s, A_t = a\right] \\ &= \sum_{s'} \mathbb{P}(s' \mid a, s) \left[R(s') + \gamma v_\pi(s')\right] \end{aligned} \tag{2}$$

We can see from (1) and (2) that :

$$v_\pi(s) = \sum_a \pi(a \mid s) q_\pi(s, a) \tag{3}$$

These equations are called the Bellman equations and are used to find the value functions using dynamic programming.

4 Optimal policies and Optimal value function

For finite MDPs, the value function can define a partial order in the space of policies :

$$\pi \leq \pi' \Leftrightarrow v_\pi(s) \leq v_{\pi'}(s) \quad \forall s \in \mathcal{S}$$

An optimal policy is a policy which is greater or equal than any other policy. This is what we are interested to find.

We can notice there is no policy strictly better than every deterministic policy (Puterman, 1994) so there is always a deterministic optimal policy since there is only a finite number of deterministic policies.

We may have multiple optimal policies but they all have the same value functions otherwise they would not be optimal policies with respect to the partial order define earlier. We denote these functions q_* and v_*.

Bellman optimality equations The optimal policy π^* has optimal value functions : v_* and q_*, which satisfy the relations below :

$$\begin{aligned} v_*(s) &= \max_a q_*(s,a) \\ &= \max_a \sum_{s'} \mathbb{P}(s' \mid a, s) \left[R(s') + \gamma v_*(s')\right] \end{aligned} \tag{4}$$

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a\right] \\ &= \sum_{s'} \mathbb{P}(s' \mid s, a)[R(s') + \gamma \max_{a'} q_*(s', a')] \end{aligned} \tag{5}$$

These are called the *Bellman optimality equations*. What (4) tells intuitively is that the value of a state under π^* must equal the expected return for the best action we can take from at that state.

For finite MDPs these equations have a unique solution. We can note that if we know v_* or q_* a greedy approach to define a policy (best in the short term) becomes a long-term optimal solution : indeed defining a policy being greedy in function of v_* implies that you go to the best state possible, the one with the biggest expected reward.

5 Solving MDPs with dynamic programming

In general we don't have all the information we need to compute the exact value of v_* or even if we have them, we don't have the computational power needed. We often use approximation of value-function instead.

From now on we assume our MDPs are finite, even if it is possible to extend everything to infinite MDPs if we are careful enough to avoid the problematic ones. (Sutton & Barto, 1998)

5.1 Policy iteration

This method uses two processes : the first one is policy evaluation : we compute the value function of a policy for all the states $s \in \mathcal{S}$; then we use policy improvement to get a better policy by acting greedy.

Policy evaluation We begin by setting arbitrary $v(s) \quad \forall s \in \mathcal{S}$. Then we update using one of the following update rules :

- $v_{k+1}(s) = \sum_{a \in A} \pi(a \mid s) \sum_{s'} \mathbb{P}(s' \mid a, s)(R(s') + \gamma v_k(s'))$ this is called "iterative policy evaluation".
- we can use the same update rule as before, but use new information as soon as it is available : this kind of upgrade algorithm are called in places.

From this we derive the algorithm 1.

Policy improvement Then we try to find a better policy : we try to determine whether we should change $\pi(s)$ to $a \neq \pi(s)$. In order to do so, we try to first select action a while being in state s and then following the policy. If the expected reward we get by doing this choice is better than the one we get by simply following our policy, we should improve.

Mathematically speaking we will compare the value of taking action a while being in the state s : $q_\pi(s,a)$ to the value of s: $v_\pi(s)$. Then we would greedily improve our policy this way.

The greedy update rule we use to improve our policy is from which we derive algorithm 2 :

$$\pi'(s) = \arg\max_{a \in A} q_\pi(s,a)$$

Algorithm 1: Iterative policy evaluation (in place)

Input : policy to evaluate π
Output: $V \approx v_\pi$

Initialize $V = 0$
while $\Delta \geq \epsilon$ **do**
  $\Delta = 0$
  for $s \in \mathcal{S}$ **do**
    $v = V(s)$
    $V(s) = \sum_a \pi(a \mid s) \sum_{s'} \mathbb{P}(s' \mid a, s)[R(s') + \gamma V(s')]$
    $\Delta = \max(\Delta, \mid v - V(s) \mid)$
  end
end
return $V \approx v_\pi$

Algorithm 2: Policy improvement

Input : policy to improve π
Output: π' s.t : $\pi' \geq \pi$

for $s \in \mathcal{S}$ **do**
  $\pi'(s) = \underset{a \in A}{\text{argmax}}\ q_\pi(s, a)$
end
return π'

Policy Iteration By combining the two processes described before, we can derive an algorithm to sweep through all our states and upgrade our policy until the changes between each sweep is too small : it is controlled by a parameter ϵ and a parameter γ (which we talked about previously).This is the algorithm 3 which can be represented graphically by figure (2)

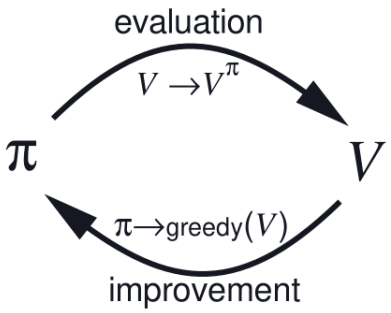

Figure 2: Graphical representation of the interactions between policy improvement and policy evaluation(from (Sutton & Barto, 1998))

Algorithm 3 terminates since there is only a finite number of deterministic policies. The reason why the algorithm 3 works is called the the *policy improvement theorem*:

Theorem 1. *Let π and π' be any pair of policies such that $\forall s \in \mathcal{S}$:*

$$q_\pi(s, \pi(s)) \geq v_{\pi'}(s) \tag{6}$$

then :

$$\pi \geq \pi' \tag{7}$$

Moreover if there is a strict inequality in all the states in 6 then there must be at least a strict inequality for one state in 7

Proof. The idea of the proof is to expand the q_π side until we get $v_{\pi'}(s)$ using Equation 2 in page 3.
Indeed we have :

$$\begin{aligned} v_\pi(s) &\leq q_\pi(s, \pi'(s)) \\ &= \mathbb{E}[r_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(a)] \\ &= \mathbb{E}_{\pi'}[r_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\ &\leq \mathbb{E}_{\pi'}[r_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1}) \mid S_t = s] \\ &= \mathbb{E}_{\pi'}[r_{t+1} + \gamma \mathbb{E}_{\pi'}[r_{t+2} + \gamma v_\pi(S_{t+2})] \mid S_t = s] \\ &= \mathbb{E}_{\pi'}[r_{t+1} + \gamma r_{t+2} + \gamma^2 v_\pi(S_{t+2}) \mid S_t = s] \\ &\leq \dots \\ &= v_{\pi'}(s) \end{aligned} \tag{8}$$

□

Algorithm 3: Policy Iteration

Input : arbitrary policy, stopping criterion ϵ
Output: estimation of the optimal policy and of its value function

Policy evaluation :
while $\Delta \geq \epsilon$ **do**
  $\Delta = 0$
  for $s \in \mathcal{S}$ **do**
    $v = V(s)$
    $V(s) = \sum_a \pi(a \mid s) \sum_{s'} \mathbb{P}(s' \mid a, s)[R(s') + \gamma V(s')]$
    $\Delta = \max(\Delta, \mid v - V(s) \mid)$
  end
end
Policy improvement :
policy-stable = true
for $s \in \mathcal{S}$ **do**
  old-action= $\pi(s)$
  $\pi(s) = \underset{a \in A}{\text{argmax}} q_\pi(s, a)$
  if *old-action* $\neq \pi(s)$ **then**
    policy-stable = false
  end
end
if *policy-stable* **then**
  return $V \approx v^* and \quad \pi \approx \pi^*$
else
  Go to Policy evaluation
end

5.2 Other types of DP method

Value Iteration Value iteration is another way of approximating an optimal policy : it combines in each of its sweep improvement and evaluation in algorithm 4.

Algorithm 4: Value iteration

Input : policy
Output: estimate of optimal policy
Initialize V arbitrarily **while** $\Delta \geq \epsilon$ **do**
  $v = V(s)$
  $V(s) = \max_a \sum_{s'} \mathbb{P}(s' \mid s, a)[R(s') + \gamma v_k(s')]$
  $\Delta = \max(\Delta, \mid v - V(s) \mid)$
end
return $\pi \approx \pi^* \ s.t : \pi(s) = \underset{a}{argmax} \sum_{s'} \mathbb{P}(s' \mid s, a)[R(s') + \gamma v_k(s')]$

Algorithm 4 finishes if the number of states is finite :

Proof. We define the Bellman operator : $\mathcal{T}_\pi : \mathbb{R}^{|\mathcal{S}|} \mapsto \mathbb{R}^{|\mathcal{S}|}$ in the following way :

$$(\mathcal{T}_\pi V)(s) = \max_{a \in \mathcal{A}} \sum_{s'} \mathbb{P}(s' \mid a, s) \left[R(s') + \gamma V(s')\right]$$

In particular this operator is a contraction in the infinity norm, and hence our convergence is equivalent to the well known fixed point problem. Indeed if we consider the inequality :

$$\mid \max_z f(z) - \max z h(z) \mid \leq \max_z \mid f(z) - h(z) \mid \tag{9}$$

Then having this inequality:

$$\begin{aligned}
\mid \mathcal{T}_\pi V(s) - \mathcal{T}_\pi V'(s) \mid \leq & \mid \max_{a \in \mathcal{A}} \sum_{s'} \mathbb{P}(s' \mid a, s) \left[R(s') + \gamma V(s')\right] - \max_{a \in \mathcal{A}} \sum_{s'} \mathbb{P}(s' \mid a, s) \left[R(s') + \gamma V'(s')\right] \mid \\
& \overset{(9)}{\leq} \max_{a \in \mathcal{A}} \mid \sum_{s'} \mathbb{P}(s' \mid a, s) \left[R(s') + \gamma V(s')\right] - \sum_{s'} \mathbb{P}(s' \mid a, s) \left[R(s') + \gamma V'(s')\right] \mid \\
& \leq \max_a \gamma \mid \sum_{s'} \mathbb{P}(s' \mid a, s) \left[V(s') - V'(s')\right] \mid \\
& \leq \gamma \max_s \mid V(s) - V'(s) \mid \\
& \leq \gamma \mid\mid V - V' \mid\mid_\infty
\end{aligned}$$

Now using the property that a contraction has a unique fixed point and that all sequences $V, \mathcal{T}_\pi V, \mathcal{T}_\pi^2 V, \ldots$ converges towards this fixed point we get the convergence.

□

There exists some others dynamic programming algorithm to solve these problems :

- Asynchronous Dynamic programming : it doe not sweep amongst all the states at each iteration. They are in place iterative DP algorithms.
- General Policy Iteration (GPI): they are mixing the two components of policy iteration (evaluation and improvement) a little bit more than the algorithm we already saw.

But the most important drawback to these methods is that they are model-based, meaning that we need to have a complete knowledge of the model to implement those.

5.3 Streetracer

6 Journal

28.02.2018 finished reading chapter 2 Sutton's book about the `k-bandits problem` : implementation of simple algorithms of the book on jupyter notebook in `Rl-sandbox`

01.03.2018 initiated the Latex journal

04.03.2108 read the notebook about `numpy` and tried to go on with the lecture of the literature–> have to read again the example about the golf

finished the chapter 3

12.03.2018 read chapter three again and made a summary of it

Then tried to attack the street racer problem

15.03.2018 tried to understand exactly what the problem of the street racing was about and tried to define a real reward function after coding the matrices for each action Then went back to studying chapter 4 in order to implement the policy evaluation/improvement functions

16.03.2018 finished reading chapter 4 of Sutton's book continued the summary

17.03.2018 finished typing the résumé so that I could concentrate on the actual code. I'm not sure about including a subsubsection about the efficiency of the DP method but for now I'm just putting the title to remember .

I have a little problem of references for my first graph but I'll see to that later

$$v_\pi(s) = R(s) + \gamma \sum_{s \in \mathcal{S}} P(s' \mid s, \pi(s)) v_\pi(s') \tag{10}$$

Et je pense que cette equation traduit bien le cas non stochastic, est-ce que j'explique en plus ?

Then I tried to finish the street racing program but too many bugs appeared so I stop for today, and I'll try again tomorrow

21.03.2018 Finished design the matrices and the model for the racing car, but issues seem to appear , the algorithm doesn't seem to work

22.03.2018 fixed issues in the code : still strange things happening : the score get optimum too quickly....

24.03.2018 fixed almost everything but still the convergence rate is abnormally high
Prepared first part of the presentation too

28.03.2018 worked on the presentation, read a lot of (Xia, 2015) and then on how to include the traffic lights into this framework

References

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1st ed.

Sutton, R. S., & Barto, A. G. (1998). *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed.

Xia, C. (2015). *Apprentissage Intelligent des Robots Mobiles dans la Navigation Autonome*. Ph.D. thesis. Thèse de doctorat dirigée par El Kamel, Abdelkader Automatique, génie informatique, traitement du signal et des images Ecole centrale de Lille 2015.
URL `http://www.theses.fr/2015ECLI0026`