École polytechnique fédérale de Lausanne

Disopt

Semester project

Reinforcement learning and robot navigation

Student:
Charles Dufour

Supervisors:
Jonas Racine
Prof. Friedrich Eisenbrand

Contents

1 Theory

1.1 Introduction

Reinforcement learning
Reinforcement learning is learning what to do,how to map situations to actions so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them. [1]

Some technical terms :

- *policy* : it is a mapping from the states to the actions
- *value function* : what our learning agent is trying to optimize
- *model* of the environment : the laws governing the environment

"Reinforcement learning methods specify how the agent's policy is changed as a result of its experience"[1]

The usual way to formulate the reinforcement learning problem from a mathematical point of view is by using what we call Markov's decision processes.

1.2 MDP : Markov decision processes

Markov's decision processes are composed by :

- a set of states : $\mathcal{S}$
- a set of actions : A
- 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 : $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$
- The Markov property : the transitions only depends on the current state and action

The Bellman equation

$$v_\pi(s) = R(s) + \gamma \sum_{s \in \mathcal{S}} P(s' \mid s, \pi(s)) v_\pi(s') \tag{1}$$

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_\pi(s')] \tag{2}$$

These are two formulations of the bellman equation used to compute optimal policies by iteration : (1) is from the MOOC and (2) is from Sutton's book [1]

1.2.1 Policies and Value functions

A policy : $\pi : \mathcal{S} \mapsto A$ is a mapping from states to action. If we follow this policy (way of behaving) we can define the value function for a policy in order to compare them, which links a state and its expected reward if we follow this policy :

The discount factor helps making our learning agent more or less far-sighted : the greater γ the more "impact" will have a late reward on our reward sequence, hence making the agent more conscious about these actions.

We define the return as : $G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
And $\pi(a \mid s)$ is the probability that $A_t = a$ if $S_t = s$
Then we can define the value of taking action a in state s while following the policy π

$$q_\pi(s,a) = \mathbb{E}[G_t \mid S_t = s, A_t = a] = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a] \tag{3}$$

And the value of a state s under a policy:

$$v_\pi(s) = \mathbb{E}[G_t \mid S_t = s] = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^t R_{t+k+1} \mid S_t = s] \quad \forall s \in \mathcal{S} \tag{4}$$

1.2.2 Optimal policies and Optimal value function

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

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

An optimal policy is a policy which is greater or equal than any other policy.

Bellman optimality equations

$$v_*(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')] \tag{5}$$

$$q_*(s, a) = \sum_{s',r} p(s', r \mid s, a)[r + \gamma \max_{a'} q_*(s', a')] \tag{6}$$

For finite MDP's 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.

1.3 Solving MDP's

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.

1.3.1 Dynamic programming

2 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

References

[1] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.