Reinforcement learning and robot navigation using MDPs

Charles Dufour

March 28, 2018

Introduction

The problem

- Framework : the Disopt robot which can follow lines
- The problem : the robot should adapt its speed with respect to traffic lights
- How : using Markov Decision Process (MDP) and Reinforcement Learning (RL)

MDPs

Definition

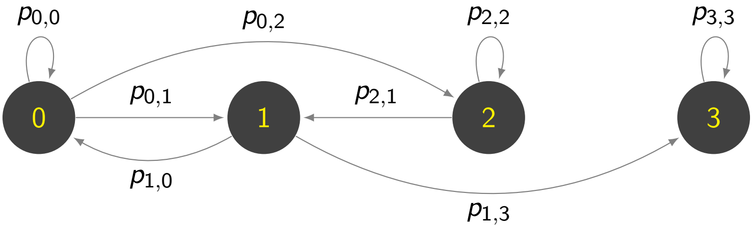
- A set of states $\mathcal{S} = \{s_0, s_1, s_2, \ldots, s_n\}$
- A set of actions $\mathcal{A} = \{a_1, a_2, a_3, \ldots, a_k\}$
- A transition function $T(a, s, s', r) = \mathbb{P}[s', r \mid a, s]$
- A reward function $R : \mathcal{S} \mapsto \mathbb{R}$
- A discount factor $0 \leq \gamma < 1$

Markov Property

The transitions only depends on the current state and the current action.

MDP example

MDPs can easily be represented by graphs :



The constraints are $\sum_j p_{i,j} = 1 \quad \forall i \in \mathcal{S}$

how to pick actions

Definition

A policy π is a probabilistic mapping from the set of states to the set of actions :

$$\pi : \mathcal{S} \mapsto \mathcal{A}$$

Issue

How to ?

How to asses the goodness of policies so we can find the best one ?
What is the best policy ?

how to asses the goodness of policies

Discounted return

$$G_t = \sum_{k=0}^{\infty} \gamma^k * R_{t+k+1}$$

how to asses the goodness of policies

Discounted return

$$G_t = \sum_{k=0}^{\infty} \gamma^k * R_{t+k+1}$$

action value while in a state s under π

$$q_\pi(s, a) = \mathbb{E}[G_t \mid S_t = s, A_t = a] \tag{1}$$

how to asses the goodness of policies

Discounted return

$$G_t = \sum_{k=0}^{\infty} \gamma^k * R_{t+k+1}$$

action value while in a state s under π

$$q_\pi(s, a) = \mathbb{E}[G_t \mid S_t = s, A_t = a] \tag{1}$$

state value under policy π

$$\begin{aligned} v_\pi(s) &= \mathbb{E}[G_t \mid S_t = s] \\ &= \sum_a \pi(a \mid s) \sum_{r,s'} p(s', r \mid s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{2}$$

how to asses the goodness of policies

how to compare two policies

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

how to asses the goodness of policies

how to compare two policies

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

Optimal policy

$$\pi_* \quad s.t. \quad \forall \pi : \pi_* \geq \pi$$

Bellman optimality equations

The optimal policy π_* has value functions : v_* and q_*

$$v_*(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')] \tag{3}$$

$$q_*(s, a) = \sum_{s',r} p(s', r \mid s, a)[r + \gamma \max_{a'} q_*(s', a')] \tag{4}$$

Another issue

computational issue

If we wanted to solve these equations directly, it would cost a lot of computational power to know exactly the value functions first and then to solve. So how do we do it ?

Another issue

computational issue

If we wanted to solve these equations directly, it would cost a lot of computational power to know exactly the value functions first and then to solve. So how do we do it ?

Approximation of value function

solving MDPs using dynamic programming

policy iteration

update rule :

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)(r + \gamma v_k(s'))$$

solving MDPs using dynamic programming

policy iteration

update rule :

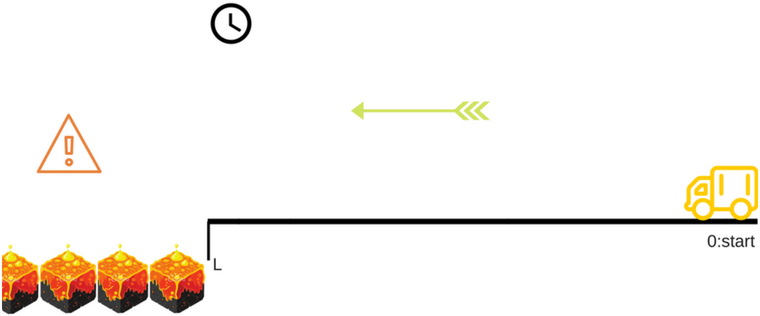
$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)(r + \gamma v_k(s'))$$

Policy Improvement

π / π' : old/new policy.

$$\pi'(s) = argmax_{a \in \mathcal{A}} q_{\pi}(s, a)$$

what have we done so far



modelization

States

modelization

States

- position $\{0,1,2,\ldots,L, \text{Lava}\}$

modelization

States

- position $\{0,1,2,\ldots,\mathrm{L}, \mathrm{Lava}\}$
- speed $\{\mathrm{low}, \mathrm{medium}, \mathrm{high}\}$

modelization

States

- position $\{0,1,2,\ldots,L, \text{Lava}\}$
- speed $\{\text{low, medium, high}\}$

Actions

modelization

States

- position $\{0,1,2,\ldots,L, \text{Lava}\}$
- speed $\{\text{low, medium, high}\}$

Actions

- decelerating

modelization

States

- position $\{0,1,2,\ldots,L, \text{Lava}\}$
- speed $\{\text{low, medium, high}\}$

Actions

- decelerating
- maintaining speed

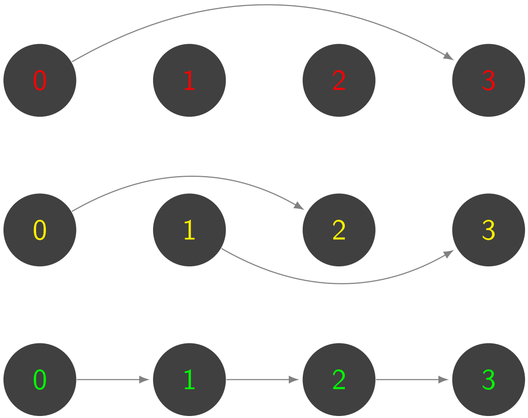
States

- position $\{0,1,2,\ldots,L, \text{Lava}\}$
- speed $\{\text{low, medium, high}\}$

Actions

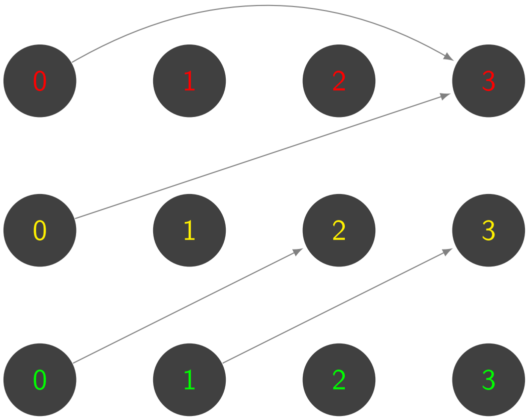
- decelerating
- maintaining speed
- accelerating

accelerating graph

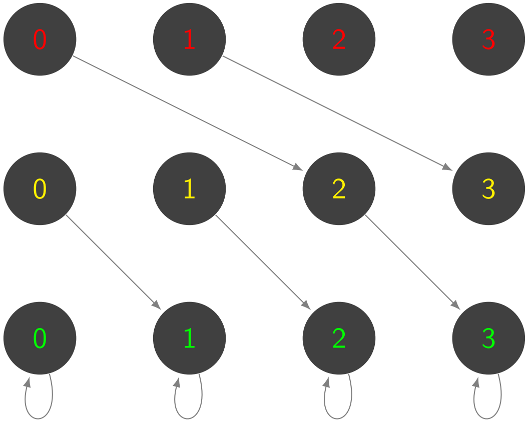


red : high speed
yellow : medium speed
green : low speed

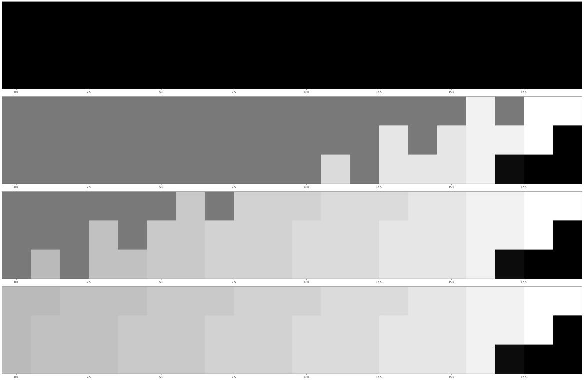
keeping the same speed graph

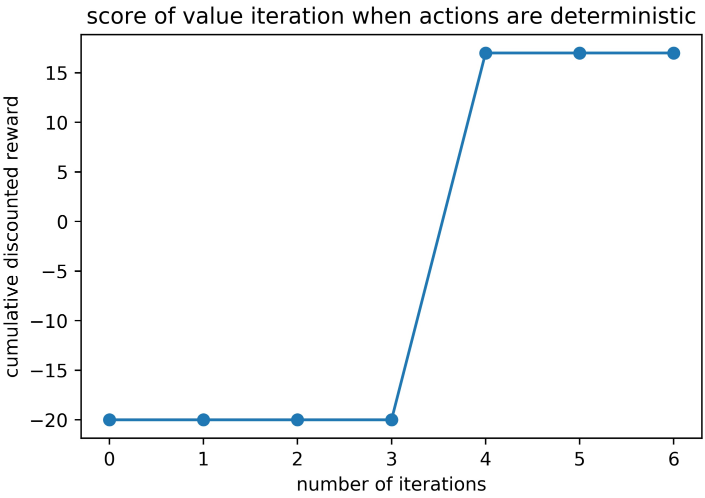


decelerating



States values at iterations $0, 2, 4$ and 6 (where stable policy is attained)





score of value iteration when actions are deterministic
cumulative discounted reward
15
10
5
0
−5
−10
−15
−20
0
1
2
3
4
5
6
number of iterations

What's next ?

already working on

- Add the traffic light into this setting

In a not so distant future

- Finding the distance from the robot's camera to the traffic light
- implement on the robot...

What's next ?

already working on

- Add the traffic light into this setting

In a not so distant future

- Finding the distance from the robot's camera to the traffic light
- implement on the robot... and pray that everything works well on the first try

What's next ?

already working on

- Add the traffic light into this setting

In a not so distant future

- Finding the distance from the robot's camera to the traffic light
- implement on the robot... and pray that everything works well on the first try

Ideas

- explore other algorithm and compare them : Monte-Carlo methods, time difference methods, Q learning, ...
- Neuro-dynamic programming ?