Reinforcement learning and robot navigation

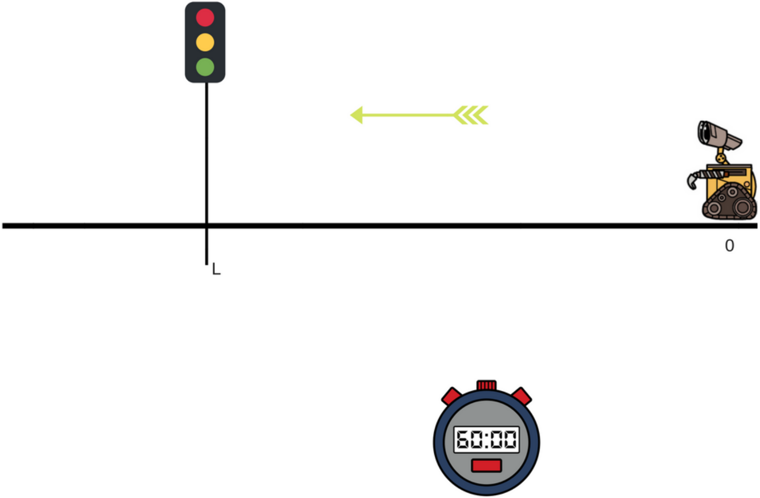
Charles Dufour

May 2, 2018

The problem

- Framework: a raspberry pie 3 robot which can follow lines
- The task: the robot should adapt its speed with respect to traffic lights
- How: using Reinforcement Learning (RL) and Markov Decision Process (MDP)

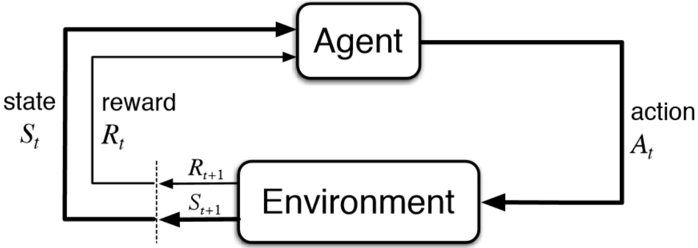
The task



Presentation

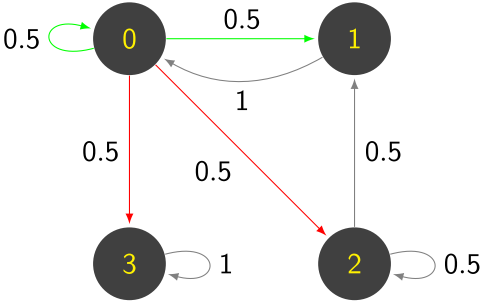
- Part I: theoretical insight
- Part II: results from first implementations

Reinforcement learning



The agent's job is to find a behavior that maximizes the long-run sum of values of the rewards.

Markov Decision Process intuition



Sequence of events : $s_0, a_1, s_1, r_1, a_2, \ldots$

Markov Decision Process

Definition: (MDP)

- 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 $T(a, s, s') = \mathbb{P}[s' \mid a, s]$
- 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.

How to pick actions

Definition: (policy)

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

$$\pi : \mathcal{A}x\mathcal{S} \mapsto [0,1]$$

s.t. $\sum_a \pi(a \mid s) = 1 \quad \forall s \in \mathcal{S}$

Issue

How to

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

How to asses the quality of policies

We are interested in maximizing the discounted return: G_t

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

How to asses the quality of policies

We are interested in maximizing the discounted return: G_t

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

Action value while in a state s under π

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

How to asses the quality of policies

We are interested in maximizing the discounted return: G_t

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

Action value while in a state s under π

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

State value under policy π

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

Recursive definition of value functions

action value

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

with $r = R(s')$

Recursive definition of value functions

state value

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

with $r = R(s')$

How to asses the quality of policies

How to compare two policies

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

How to asses the quality of policies

How to compare two policies

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

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_*

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

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

Finding optimal policy from value functions

$$\pi_*(s) = \underset{a}{\operatorname{argmax}}\ q_*(s, a)$$

Another issue

Computational issue

$\mid \mathcal{S} \mid$ linear equations to solve to evaluate policy
$\mid \mathcal{S} \mid$ non linear equations to solve the Bellman optimality equation

Another issue

Computational issue

$\mid \mathcal{S} \mid$ linear equations to solve to evaluate policy
$\mid \mathcal{S} \mid$ non linear equations to solve the Bellman optimality equation

Approximation of value function
and
Policy iteration

Solving MDP using dynamic programming

iterative policy evaluation

update rule :

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \sum_{s'} T(a, s, s') \left[R(s') + \gamma v_k(s') \right]$$

Solving MDP using dynamic programming

iterative policy evaluation

update rule :

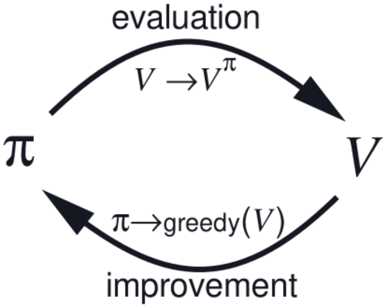
$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \sum_{s'} T(a, s, s') \left[R(s') + \gamma v_k(s') \right]$$

policy improvement

π / π' : old/new policy.

$$\pi'(s) = \operatorname*{argmax}_{a \in \mathcal{A}} q_\pi(s, a)$$

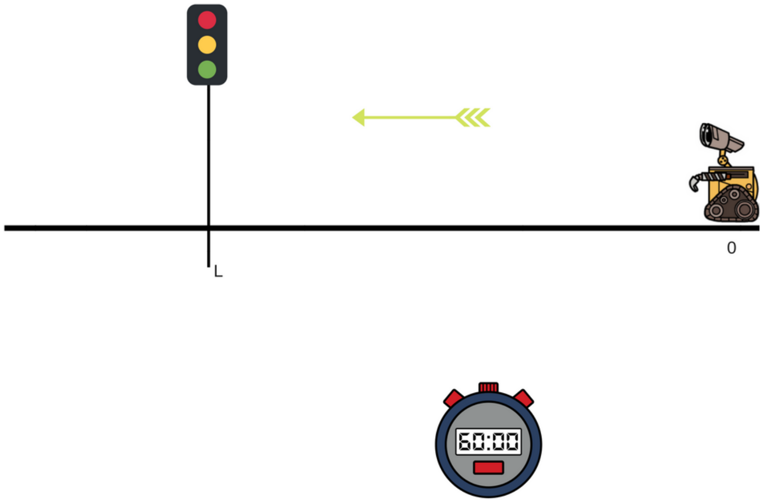
Policy iteration algorithm



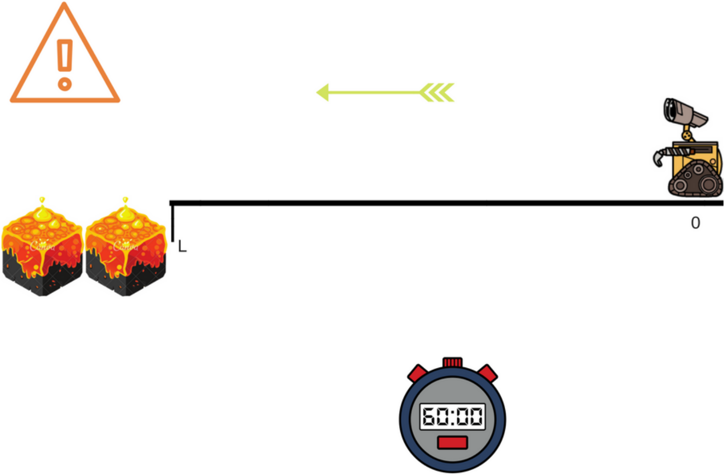
[1]

[1] From (Sutton & Barto, 1998)

Our problem



First a simpler problem



Modelization

states

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

Modelization

states

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

actions

- decelerating
- maintaining speed
- accelerating

Modelization

states

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

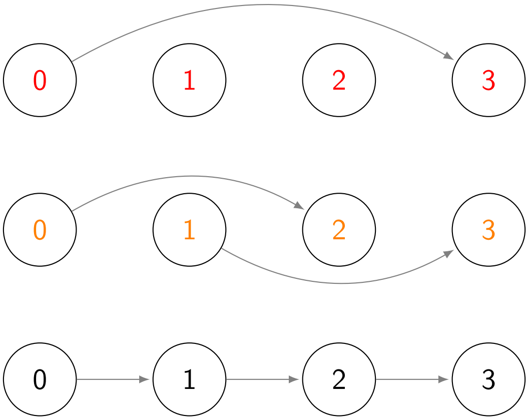
actions

- decelerating
- maintaining speed
- accelerating

reward function

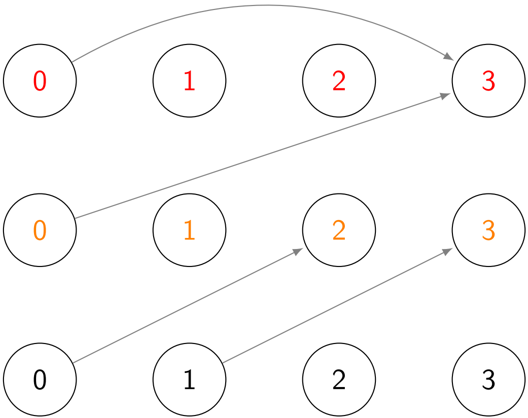
- Lava: reward of -L
- L in low speed: reward of $+$L
- any other state : reward of -1

Keeping the same speed graph

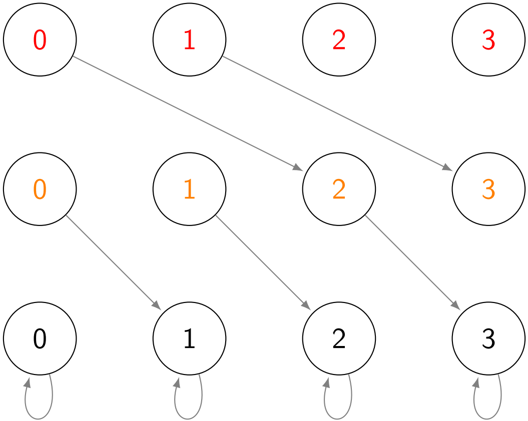


red : high speed
orange : medium speed
black : low speed

Accelerating graph



Decelerating



State value function at different iterations

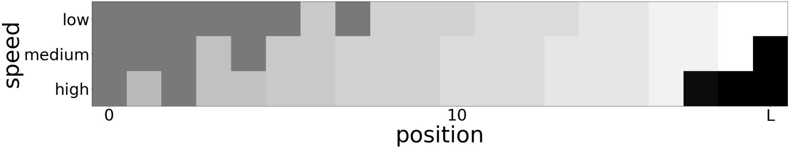
the brighter the better the value

States values at iteration 0

State value function at different iterations
the brighter the better the value

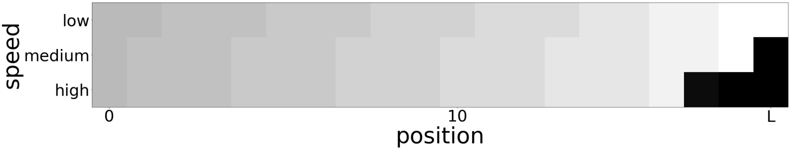
States values at iteration 2

State value function at different iterations
the brighter the better the value



States values at iteration 4

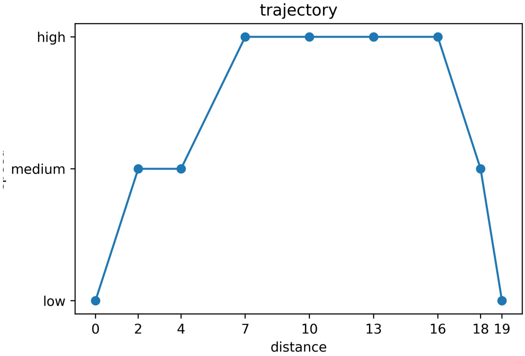
State value function at different iterations
the brighter the better the value



States values at iteration 6 (where stable policy is attained)

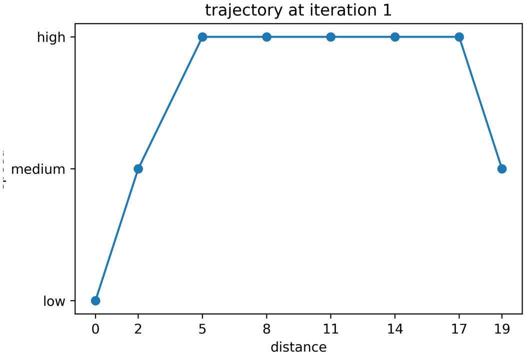
Results for deterministic actions

One of the optimal trajectories



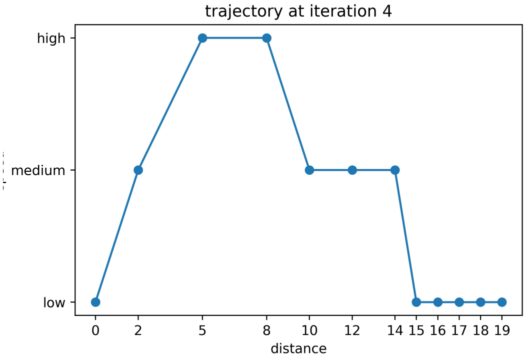
Results for stochastic actions

decelerating results in keeping the same speed with probability 1/4



Results for stochastic actions

decelerating results results in keeping the same speed with probability 1/4



where stable policy is attained

low
medium
high
0
10
L
position

Conclusion

- We introduced the theoretical framework: Reinforcement learning and Markov Decision Processes
- We solved a simple problem where we know the model (the transition function in particular)

What's next ?

already working on

- Add the traffic light into this setting
- Find the distance from the robot's camera to the traffic light

In a not so distant future

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

What's next ?

already working on

- Add the traffic light into this setting
- Find the distance from the robot's camera to the traffic light

In a not so distant future

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

Ultimately

Implement everything on the robot ...

What's next ?

already working on

- Add the traffic light into this setting
- Find the distance from the robot's camera to the traffic light

In a not so distant future

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

Ultimately

Implement everything on the robot ... and pray that everything works well