Project: Smart three-sphere swimmer near a wall

2. Introduction to reinforcement learning and Q-learning

Luca Berti, L.G., C.P.

18 October 2021

Introduction in laymen terms

Learning:

- definition: acquiring knowledge that we didn't have
- purpose: doing something we weren't able to do before; improve our skills and be more efficient (optimize)
- how: reading, studying, experience (interaction)

Introduction in laymen terms

Learning:

- definition: acquiring knowledge that we didn't have
- purpose: doing something we weren't able to do before; improve our skills and be more efficient (optimize)
- how: reading, studying, experience (interaction)

Reinforcement learning:
An agent, by interacting with the environment, is able to learn a new task and, ideally, the optimal way to carry it out.

Introduction in laymen terms

Learning:

- definition: acquiring knowledge that we didn't have
- purpose: doing something we weren't able to do before; improve our skills and be more efficient (optimize)
- how: reading, studying, experience (interaction)

Reinforcement learning:
An agent, by interacting with the environment, is able to learn a new task and, ideally, the optimal way to carry it out. It is rewarded based on the actions it performs, reinforcing the behaviours that lead to better performances.

Necessary elements and questions

A reinforcement learning problem can be described by the following elements:

Necessary elements and questions

A reinforcement learning problem can be described by the following elements:

- **State**: s_t, state of the agent at time t

Necessary elements and questions

A reinforcement learning problem can be described by the following elements:

- **State**: s_t, state of the agent at time t
- **Action**: a_t, action taking the agent from state s_t to state s_{t+1}

Necessary elements and questions

A reinforcement learning problem can be described by the following elements:

- **State**: s_t, state of the agent at time t
- **Action**: a_t, action taking the agent from state s_t to state s_{t+1}
- **Reward**: r_{t+1} evaluates the action, i.e. how passing from s_t to s_{t+1} gives better performance

Necessary elements and questions

A reinforcement learning problem can be described by the following elements:

- **State**: s_t, state of the agent at time t
- **Action**: a_t, action taking the agent from state s_t to state s_{t+1}
- **Reward**: r_{t+1} evaluates the action, i.e. how passing from s_t to s_{t+1} gives better performance

Questions:

Necessary elements and questions

A reinforcement learning problem can be described by the following elements:

- **State**: s_t, state of the agent at time t
- **Action**: a_t, action taking the agent from state s_t to state s_{t+1}
- **Reward**: r_{t+1} evaluates the action, i.e. how passing from s_t to s_{t+1} gives better performance

Questions:

- can the state space be discrete, continuous?

Necessary elements and questions

A reinforcement learning problem can be described by the following elements:

- **State**: s_t, state of the agent at time t
- **Action**: a_t, action taking the agent from state s_t to state s_{t+1}
- **Reward**: r_{t+1} evaluates the action, i.e. how passing from s_t to s_{t+1} gives better performance

Questions:

- can the state space be discrete, continuous?
- can actions space be discrete, continuous?

Necessary elements and questions

A reinforcement learning problem can be described by the following elements:

- **State**: s_t, state of the agent at time t
- **Action**: a_t, action taking the agent from state s_t to state s_{t+1}
- **Reward**: r_{t+1} evaluates the action, i.e. how passing from s_t to s_{t+1} gives better performance

Questions:

- can the state space be discrete, continuous?
- can actions space be discrete, continuous?
- how does "learning" actually happens when we have these elements?

Necessary elements and questions

A reinforcement learning problem can be described by the following elements:

- **State**: s_t, state of the agent at time t
- **Action**: a_t, action taking the agent from state s_t to state s_{t+1}
- **Reward**: r_{t+1} evaluates the action, i.e. how passing from s_t to s_{t+1} gives better performance

Questions:

- can the state space be discrete, continuous?
- can actions space be discrete, continuous?
- how does "learning" actually happens when we have these elements?
- what is the mathematical theory hiding behind all of this?

Mathematical theory: Markov decision process

Definition (Decision process)

A decision process is defined by the four elements $(S, A, P(.), r(.,.))$:

- S is the state space of the process
- A is the action space, controlling the state dynamics
- $P(.|.,.) : S \times A \times S \to [0,1]$ are the transition probabilities among the states
- $r(.,.) : S \times A \to \mathbb{R}$ is the reward function

Mathematical theory: Markov decision process

Definition (Decision process)

A decision process is defined by the four elements $(S, A, P(.), r(.,.))$:

- S is the state space of the process
- A is the action space, controlling the state dynamics
- $P(.|.,.) : S \times A \times S \to [0,1]$ are the transition probabilities among the states
- $r(.,.) : S \times A \to \mathbb{R}$ is the reward function

Definition (Markov decision process)

A decision process is Markov if

$$\mathbb{P}(s_{t+1}|s_0, a_0, s_1, a_1, \dots, s_t, a_t) = \mathbb{P}(s_{t+1}|s_t, a_t),$$

i.e. the probability of reaching s_{t+1} depends only on s_t, a_t and not the whole history of the process. Ex: Swimming at low Reynolds number (no inertia, reversible flow)

Policy and value function

Definition (Markov policy)

Consider a probability law $\pi_t(a_t|s_t) : A \times S \to [0,1]$ such that $\sum_{a \in A} \pi_t(a_t|s_t) = 1$. Note that the probability law only depends on s_t.
A Markov policy is a sequence of these probability laws $\pi = (\pi_0, \pi_1, \ldots, \pi_t)$.

Given a policy π (which contains the information about action-state transitions), we want to "evaluate" its performance with respect to the reward function $r(.,.)$, that is

$$r^{\pi}(s) = \sum_{a \in A} \pi(a|s) r(s,a)$$

Definition (Value function)

It's a function $V^{\pi} : S \to \mathbb{R}$ that estimates the total expected reward following policy π ($\gamma < 1$):

$$V^{\pi}(s) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r^{\pi}(s_{t+k+1}) | s_t = s, \pi]$$

Q-function and optimal policy

The value function $V^\pi(s)$ tells us: what is the expected reward that I will get if I start from s and follow policy π? What about if the initial action is also fixed?

Definition (Q-function)

It's a function $Q^\pi : S \times A \to \mathbb{R}$ such that

$$Q^\pi(s,a) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r^\pi(s_{t+k+1}) | s_t = s, a_t = a, \pi].$$

One has $V(s) = \sum_a Q(s,a)\pi(a|s)$.
We want to obtain the *best result* we can from learning

Q-function and optimal policy

The value function $V^\pi(s)$ tells us: what is the expected reward that I will get if I start from s and follow policy π? What about if the initial action is also fixed?

Definition (Q-function)

It's a function $Q^\pi : S \times A \to \mathbb{R}$ such that

$$Q^\pi(s, a) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r^\pi(s_{t+k+1})|s_t = s, a_t = a, \pi].$$

One has $V(s) = \sum_a Q(s, a)\pi(a|s)$.
We want to obtain the *best result* we can from learning, so we want to get the *best expected reward*

Q-function and optimal policy

The value function $V^\pi(s)$ tells us: what is the expected reward that I will get if I start from s and follow policy π? What about if the initial action is also fixed?

Definition (Q-function)

It's a function $Q^\pi : S \times A \to \mathbb{R}$ such that

$$Q^\pi(s,a) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r^\pi(s_{t+k+1}) | s_t = s, a_t = a, \pi].$$

One has $V(s) = \sum_a Q(s,a)\pi(a|s)$.
We want to obtain the *best result* we can from learning, so we want to get the *best expected reward*, so we want to find the policy that maximise $V^\pi(s)$, i.e. the optimal policy.

Definition (Optimal policy)

A policy π^* is optimal if the associated V^π is optimal, that is if

$$V^*(s) = V^{\pi^*}(s) = \max_{\pi' \, Markov} V^{\pi'}(s) \quad \forall s \in S$$

Optimal value function

Theorem (Optimality: Bellman equation)

Suppose the policy is stationary, that is $\pi = (\pi_0, \pi_0, \ldots, \pi_0)$*.*
The optimal value function V^* *is the unique solution to the Bellman equation*

$$V^*(s) = \max_a \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') \right) \quad \forall s \in S.$$

The optimal Q function Q^* *is the unique solution to the Bellman equation*

$$Q^*(s,a) = \max_a \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a') \right) \quad \forall s \in S, \forall a \in A$$

Optimal value function

Theorem (Optimality: Bellman equation)

Suppose the policy is stationary, that is $\pi = (\pi_0, \pi_0, \ldots, \pi_0)$*.*
The optimal value function V^* *is the unique solution to the Bellman equation*

$$V^*(s) = \max_a \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') \right) \quad \forall s \in S.$$

The optimal Q function Q^* *is the unique solution to the Bellman equation*

$$Q^*(s,a) = \max_a \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a') \right) \quad \forall s \in S, \forall a \in A$$

NB1: here $P(.|.,.)$ are the transition probabilities that define the decision process.

Optimal value function

Theorem (Optimality: Bellman equation)

Suppose the policy is stationary, that is $\pi = (\pi_0, \pi_0, \dots, \pi_0)$.
The optimal value function V^* *is the unique solution to the Bellman equation*

$$V^*(s) = \max_a \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') \right) \quad \forall s \in S.$$

The optimal Q function Q^* *is the unique solution to the Bellman equation*

$$Q^*(s,a) = \max_a \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a') \right) \quad \forall s \in S, \forall a \in A$$

NB1: here $P(.|.,.)$ are the transition probabilities that define the decision process.
NB2: Bellman equation comes from maximising over the policies (remember $V^* = V^{\pi^*}$):

$$V^*(s) = \max_\pi V^\pi(s) = \max_\pi \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^\pi(s) \right)$$

Optimal policy

Theorem (Optimal policy)

A stationary policy is optimal if and only if its value function satisfies the Bellman equation, which means

$$\pi^* \in \arg\max_a \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') \right)$$

Proofs about this theorem and the previous one are contained in reference [1, French] or [2, English].

Optimal policy

Theorem (Optimal policy)

A stationary policy is optimal if and only if its value function satisfies the Bellman equation, which means

$$\pi^* \in \arg\max_a \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') \right)$$

Proofs about this theorem and the previous one are contained in reference [1, French] or [2, English].

We remark that the proof of the Bellman equations is based on a fixed point theorem, which will be the starting point from the algorithmic point of view.

Optimal policy

Theorem (Optimal policy)

A stationary policy is optimal if and only if its value function satisfies the Bellman equation, which means

$$\pi^* \in \arg\max_a \left(r(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') \right)$$

Proofs about this theorem and the previous one are contained in reference [1, French] or [2, English].

We remark that the proof of the Bellman equations is based on a fixed point theorem, which will be the starting point from the algorithmic point of view.

The two fundamental objects here are the value function and the optimal policy. Certain algorithms focus on finding one of the two, or both at the same time.

Algorithms: general overview

There are three families of reinforcement learning algorithms:

- Value based: they approximate the optimal value function
- Monte Carlo estimate of the expected return (can be used in the non-Markov case)
- Policy based: they approximate the optimal policy

Examples:

- Value based: SARSA, Q-learning
- Policy based: policy gradient, actor-critic

SARSA and Q-learning

In both algorithms, the focus is finding Q^*, as the value function can be computed as $V^* = \sum_a Q(s,a)\pi(a|s)$.

Definition (SARSA - *on policy*)

The update rule for SARSA is

$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha(r_t + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t)).$$

Definition (Q-learning - *off policy*)

The update rule for Q-learning is

$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha(r_t + \gamma \max_a Q^\pi(s_{t+1}, a) - Q^\pi(s_t, a_t)).$$

In the previous algorithms, α is the learning rate and γ the discount factor.
Difference: In SARSA you choose a_{t+1} following the same policy you use to approximate Q. In Q-learning, you choose a_{t+1} via the max function, which is different from the policy you use to approximate Q.

Q-learning algorithm

State and action spaces: finite (and possibly small)
The Q-function can be represented as a *matrix*!

Algorithm 1 Q-learning algorithm

Require: s_0 initial state; set $Q(s,a)$ to 0
for i=1 **to** N_{learn} **do**
Choose a_i using the policy given by Q
Take action a_i, observe the new displacement δ_i and state s_i
Update Q via
$Q^\pi(s_t,a_t) \leftarrow Q^\pi(s_t,a_t) + \alpha(r_t + \gamma \max_a Q^\pi(s_{t+1},a) - Q^\pi(s_t,a_t))$.
end for
Ensure: Q, r

Double Q-learning algorithm

Algorithm 2 Double Q-learning algorithm

Require: s_0 initial state; set $Q_A(s,a)$, $Q_B(s,a)$ to 0

for i=1 **to** N_{learn} **do**

  Choose a_i using the policy given by Q_A, Q_B

  Take action a_i, observe the new displacement δ_i and state s_i

  Choose (randomly) to update Q_A or Q_B

  if update Q_A **then**

    Let $a^* = \arg\max_a(Q_A)(s_i, a)$

    Update $Q_A(s,a) \leftarrow Q_A(s,a) + \alpha[r_n + \gamma Q_B(s_i, a^*) - Q_A(s,a)]$

  end if

  if update Q_B **then**

    Let $a^* = \arg\max_a(Q_B)(s_i, a)$

    Update $Q_B(s,a) \leftarrow Q_B(s,a) + \alpha[r_n + \gamma Q_A(s_i, a^*) - Q_B(s,a)]$

  end if

  $s \leftarrow s_i$

end for

Ensure: Q_A, Q_B, r

References

- [1] http://researchers.lille.inria.fr/~munos/papers/files/bouquinPDMIA.pdf
- [2] http://researchers.lille.inria.fr/~lazaric/Webpage/MVA-RL_Course14_files/notes-lecture-02.pdf
- Sutton, R. & Barto, A "Reinforcement Learning" (book)