Theoretical aspects of Q-learning

Masters thesis defense

Jacob Harder
Department of Mathematical Sciences
University of Copenhagen

26 June, 2020

Overview

The goal of RL

Q-functions

Value iteration

Q-learning as AI

Artificial Intelligence (AI)

↓

Machine Learning (ML)

↙ ↘

Reinforcement Learning (RL) | Supervised Learning, etc.

↓

Q-learning

Machine learning

Machine Learning is "the study of computer algorithms that improve automatically through *experience*".

- **Supervised learning**: Tasks are learned from data based on feedback from a *supervisor*. E.g. image classification.
- **Unsupervised learning**: Data is given without evaluatory feedback, general trends about the data are analysed. E.g. principal component analysis, and cluster analysis.
- $\rightarrow$[1] **Reinforcement learning**: Algorithms which learns through interactions with an *environment*.

[1] "$\rightarrow$": Our main area of focus in this thesis.

Challenges in RL

Challenges in Reinforcement Learning include:

- **Exploration-exploitation trade-off**. Training and performing occurs simultaneously so one optimizes the total reward on some time horizon. This is studied in e.g. the multi-armed bandit problem.
- → **Deriving optimal policies**. Training and performing is distinguished and emphasis is put on the expected performance of the final derived policy rather than rewards occuring during training.

The environment

The **environment** in RL is often formalized as a **Markov decision process** (MDP), which consists of

- $\mathcal{S}$ a measurable space of states.
- $\mathcal{A}$ a measurable space of actions.
- $P : \mathcal{S} \times \mathcal{A} \rightsquigarrow \mathcal{S}$ a transition kernel[2].
- $R : \mathcal{S} \times \mathcal{A} \rightsquigarrow \mathbb{R}$ a reward kernel discounted by
- a discount factor $\gamma \in [0, 1)$.
- $\mathfrak{A}(s) \subseteq \mathcal{A}$ a set of admissable actions for each $s \in \mathcal{S}$.

[2] Here $\rightsquigarrow$ denotes a *stochastic mapping* (to be defined soon).

Examples of MDPs

Examples of Markov decision processes include

- Board games where one plays against a fixed opponent, e.g. *chess* where the set of states $\mathcal{S}$ is the set of all obtainable chess-positions.
- Time-descretized physics simulations with action inputs and reward outputs, including most single player video games and the classic *cartpole* example (balancing a stick).

The probability kernels

Probability kernel

A **probability kernel** (also called a *stochastic mapping*, *stochastic kernel* or *Markov kernel*) $\kappa : \mathcal{X} \rightsquigarrow \mathcal{Y}$ is a collection of probability measures $\kappa(\cdot \mid x)$, one for each $x \in \mathcal{X}$ such that for any measurable set $B \subseteq \mathcal{Y}$ the function $x \mapsto \kappa(B \mid x)$ is measurable.

The transition probability measure $P(\cdot \mid s, a)$ of the pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ determines what states are likely to follow after *being* in state s and *choosing* action a. Similarly from the reward kernel R one obtains the measure $R(\cdot \mid s, a)$ determining the reward distribution following the timestep (s, a).

Policies

Given a Markov decision process one can define a **policy** π by sequence of probability kernels $\pi = (\pi_1, \pi_2, \dots)$ where $\pi_i : \mathcal{H}_i \rightsquigarrow \mathcal{A}$ and $\mathcal{H}_i = \mathcal{S} \times \mathcal{A} \times \cdots \times \mathcal{S}$ is the *history space* at the ith timestep.

Stochastic processes

An MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ together with a policy $\pi = (\pi_1, \pi_2, \dots)$ and a distribution μ on $\mathcal{S}$ give rise to a stochastic process $(S_1, A_1, S_2, A_2, \dots) \sim \kappa_\pi \mu$ such that for any $i \in \mathbb{N}$ we have $(S_1, A_1, \dots, S_i) \sim P\pi_{i-1} \dots P\pi_1 \mu$ where $P\pi_{i-1} \dots P\pi_1$ denotes the *kernel-composition* of the probability kernels $P, \pi_1, \dots, \pi_{i-1}$. We denote by $\mathbb{E}_s^\pi$ expectation over $\kappa_\pi \mu$ where $\mu = \delta_s$, that is, $S_1 = s$ a.s.

Policy evaluation

For a policy π we can define the policy evaluation function:

Policy evaluation

Denote by $r(s,a) = \int x \, \mathrm{d}R(x \mid s,a)$ the *expected reward function*.

We define the **policy evaluation function** by

$$V_\pi(s) = \mathbb{E}_s^\pi \sum_{i=1}^{\infty} \gamma^{i-1} r \circ \rho_i$$

where ρ_i is projection onto $(\mathcal{S}_i, \mathcal{A}_i)$.

This an example of a (state-) *value function*, as it assigns a real number to every state $s \in \mathcal{S}$.

Finite policy evaluation

Similar to the infinite horizon policy evaluation we can also consider a finite horizon version:

Definition: Finite policy evaluation

We define the function $V_{n,\pi} : \mathcal{S} \to \mathbb{R}$ by

$$V_{n,\pi}(s) = \mathbb{E}_s^\pi \sum_{i=1}^{n} \gamma^{i-1} r \circ \rho_i$$

called the kth **finite policy evaluation**[a].

[a] When $n = 0$ we say $V_{0,\pi} = V_0 := 0$ for any π.

Optimal value function

Definition: Optimal value functions

$$V_n^*(s) := \sup_{\pi \in R\Pi} V_{n,\pi}(s) = \sup_{\pi \in R\Pi} \mathbb{E}_s^\pi \sum_{i=1}^{n} r_i$$

$$V^*(s) := \sup_{\pi \in R\Pi} V_\pi(s) = \sup_{\pi \in R\Pi} \mathbb{E}_s^\pi \sum_{i=1}^{\infty} r_i$$

This is called the **optimal value function** (and the nth optimal value function). A policy $\pi^* \in R\Pi$ for which $V_{\pi*} = V^*$ is called an **optimal policy**. If $V_{n,\pi*} = V_n^*$ then π^* is called n-optimal.

Provided such an optimal policy π^* exists, obtaining such a policy is the ultimate goal of Reinforcement Learning.

Operators on value functions

The T-operators

For a stationary policy $\tau \in S\Pi$ and a value function $V : \mathcal{S} \to \mathbb{R} \in \mathcal{L}_\infty(\mathcal{S})$ we define the operators

The policy evaluation operator:

$$T_\tau V := s \mapsto \int r(s,a) + \gamma V(s') \, \mathrm{d}(P\tau)(a, s' \mid s)$$

The Bellman optimality operator:

$$TV := s \mapsto \sup_{a \in \mathfrak{A}(s)} \left(r(s,a) + \gamma \int V(s') \, \mathrm{d}P(s' \mid s,a) \right)$$

Q-functions

A **Q-function** is simply any function assigning a real number to every state-action pair. They are also called (state-) *action value functions*.

A **Q-learning** algorithm is any algorithm which uses Q-functions to derive a policy for an environment[3].

How to derive a policy from a Q-function? One way to do this is by picking *greedy actions*.

[3] Some authors refer to Q-learning as a specific variation of temporal difference learning, but this fails to capture many algorithms which are also referred to as *Q-learning algorithms*.

Greedy policies

Let $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ be a measurable Q-function and $\tau : \mathcal{S} \rightsquigarrow \mathcal{A}$ be a (stationary) policy.

Greedy policy

Define the set of *greedy actions* by $G_Q(s) := \operatorname{argmax}_{a \in \mathfrak{A}(s)} Q(s, a)$. If there exist a measurable set $G_Q^\tau(s) \subseteq G_Q(s)$ for every $s \in \mathcal{S}$ such that

$$\tau\left(G_Q^\tau(s) \,\middle|\, s\right) = 1$$

then τ is said to be **greedy** with respect to Q and is denoted τ_Q.

Q-function operators

Operators for Q-functions

For any stationary policy $\tau \in S\Pi$ and integrable Q-function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R} \in \mathcal{L}_\infty(\mathcal{S} \times \mathcal{A})$ we define

Next-step operator:

$$P_\tau Q(s,a) = \int Q(s',a') \, \mathrm{d}\tau P(s',a' \mid s,a)$$

Policy evaluation operator:

$$T_\tau Q(s,a) = r(s,a) + \gamma \int Q(s',a') \, \mathrm{d}\tau P(s',a' \mid s,a)$$

Bellman optimality operator:

$$TQ(s,a) = r(s,a) + \gamma \int \max_{a' \in \mathcal{A}} Q(s',a') \, \mathrm{d}P(s' \mid s,a)$$

where $T_a = T_{\delta_a}$.