Theoretical aspects of Q-learning

Theoretical aspects of Q-learning

Masters thesis defense

Jacob Harder

Department of Mathematical Sciences

University of Copenhagen

26 June, 2020

Overview

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.
- $\rightarrow$ **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.

Greediness

In order to show existence of optimal policies and talk about algorithms which can determine such policies, we define the concept of *greediness*.

Greedy actions

The purpose of (most) value functions $V : \mathcal{S} \to \mathbb{R}$ is to give an estimate on how *good* a certain state is, in terms of the rewards one may expect after visiting it.

This give rise to the idea of *greedy actions*, that is, actions leading to states high *values* (according to V).

Definition greedy actions

Let $V : \mathcal{S} \to \mathbb{R}$ be a measurable value-function. We define

$$G_V(s) = \operatorname*{argmax}_{a \in \mathfrak{A}(s)} T_a V(s) \subseteq \mathfrak{A}(s)$$

as the set of **greedy** actions w.r.t. V.

Greedy policies

Greedy actions leads to *greedy policies*:

Definition: Greedy policy

Let $V : \mathcal{S} \to \mathbb{R}$ be a measurable value-function and let $\tau : \mathcal{S} \rightsquigarrow \mathcal{A} \in S\Pi$ be a stationary policy. If there exists a measurable $G_V^\tau(s) \subseteq G_V(s)$ such that

$$\tau(G_V^\tau(s) \mid s) = 1$$

for every $s \in \mathcal{S}$, then τ is called greedy w.r.t. V. We will often denote a V-greedy policy by τ_V.

Existence of greedy policies

Theorem (Existence of greedy policies)

Suppose $V : \mathcal{S} \to \mathbb{R}$ is *upper semicontinuous* and that

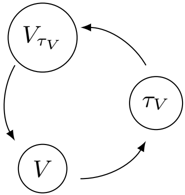
1. $\mathcal{S}$ and $\mathcal{A}$ are standard Borel.
2. The set of admissable actions $\mathfrak{A}(s) \subseteq \mathcal{A}$ is compact for all $s \in \mathcal{S}$ and $\Gamma = \{(s,a) \in \mathcal{S} \times \mathcal{A} \mid a \in \mathfrak{A}(s)\}$ is a closed subset of $\mathcal{S} \times \mathcal{A}$.
3. The transition kernel P is continuous.
4. The expected reward function $r = \int r' \, \mathrm{d}R(r' \mid \cdot)$ is upper semicontinuous and bounded from above.

Then there exists a deterministic policy π_V which is greedy for V.

If assumptions 1.-4. hold we will say that the MDP is *greedy*.

Policy iteration

Using the concepts we have defined one can get the following idea: Iteratively generate value functions by picking greedy policies and then evaluating these policies. This is called *policy iteration*.



Policy iteration is a well studied algorithm and can be shown to converges to optimum for a variety of environments. We will however move on to talk about a related concept called *value iteration*.

Operators on value functions

Before defining value iteration we introduce some operators

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:

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

Properties of the T-operators

Proposition (Properties of the T-operators)

$$V_{k,\pi} = T_{\tau_1} V_{k-1,(\tau_2,\dots)} = T_{\tau_1} \dots T_{\tau_k} V_0.$$

$$V_\pi = \lim_{k\to\infty} T_{\tau_1} \dots T_{\tau_k} V_0$$

For the stationary policy τ we have $T_\tau V_\tau = V_\tau$.

T and T_τ are γ-contractive on $\mathcal{L}_\infty(\mathcal{S})$.

V_τ is the unique bounded fixed point of T_τ in $\mathcal{L}_\infty(\mathcal{S})$.

This way T_τ can be interpreted as a 'one-step policy evaluation'. On the other hand T can be interpreted as a 'one-step evaluation, when always choosing greedy actions'.

Value iteration

Value iteration is the iterative application of the T-operator. The following theorem show why value iteration is a central idea Reinforcement Learning[3].

Theorem (Existence optimal policies & convergence of value iteration)

Given a greedy MDP we have that

$$V_k^* = T^k V_0 = T_{\tau_{k-1}^*} \dots T_{\tau_0^*} V_0 = V_{k,(\tau_{k-1}^*,\dots,\tau_0^*)}$$

The policy $(\tau_{k-1}^*, \dots, \tau_0^*)$ is a deterministic k-optimal policy where $\tau_k^* = \tau_{T^k V_0}$ is any deterministic greedy policy for $T^k V_0$ for any $k \in \mathbb{N}$. Furthermore $V^* = \lim_{k\to\infty} T^k V_0^*$, the greedy policy $\tau^* = \tau_{V^*}$ exists and an optimal policy.

[3] Actually value iteration is inherited from dynamic programming.

Convergence rates

We can also show that the optimal value function V^* is a fixed point of the Bellman optimality operator T.

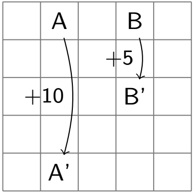
$$TV^* = V^*$$

This is often called *Bellman's optimality equation*.
Recalling that T is γ-contractive, by Banach's fixed point theorem we get exponential convergence rates for value iteration:

$$\left\| T^k V - V^* \right\| \leqslant \gamma^k \| V - V^* \|_\infty = \mathcal{O}(\gamma^k)$$

Example: Gridworld

The *gridworld*
MDP consist of 25 states $\mathcal{S} = [5]^2$
and 4 actions $\mathcal{A} = \{U, D, L, R\}$
for *up*, *down*, *left* and *right*
and moves the agent 1 square up,
down, left or right. A reward of
0 is given by default, except when



- *hitting the boundary* a reward of -1 is given
- when in $A = (2, 1)$ any action moves to $A' = (2, 5)$ and is rewarded 10.
- when in $B = (4, 1)$ any action moves to $B' = (4, 3)$ and is rewarded 5.

Finally $\gamma = 0.9$ is the standard value of the discount factor in this example.

Example: Gridworld

-0.50	10.00	-0.25	5.00	-0.50
-0.25	0.00	0.00	0.00	-0.25
-0.25	0.00	0.00	0.00	-0.25
-0.25	0.00	0.00	0.00	-0.25
-0.50	-0.25	-0.25	-0.25	-0.50

V_{1,τ_r}

3.31	8.79	4.43	5.32	1.49
1.52	2.99	2.25	1.91	0.55
0.05	0.74	0.67	0.36	-0.40
-0.97	-0.44	-0.35	-0.59	-1.18
-1.86	-1.35	-1.23	-1.42	-1.98

V_{400,τ_r}

Figure: Policy evaluations of the gridworld environment. Note that $V_{\max} \cdot \gamma^{400} = 100 \cdot (0.9)^{400} \approx 4.97 \cdot 10^{-17}$ so $V_{\tau_r,400}$ are very close to the true infinite horizon value functions V_{τ_r} (providing numerical errors are insignificant).

Example: Gridworld

9.00	10.00	9.00	5.00	4.50
8.10	9.00	8.10	4.50	4.05
0.00	8.10	0.00	4.05	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00

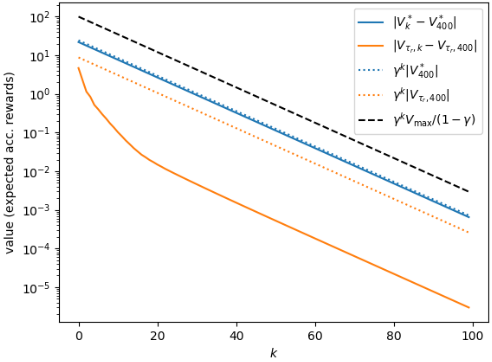
V_3^*

21.98	24.42	21.98	19.42	17.48
19.78	21.98	19.78	17.80	16.02
17.80	19.78	17.80	16.02	14.42
16.02	17.80	16.02	14.42	12.98
14.42	16.02	14.42	12.98	11.68

V_{400}^*

Figure: Optimal value functions of the gridworld environment. By the same upper bound as before we have $\left\|V^* - V_{400}^*\right\|_\infty < 4.97 \cdot 10^{-17}$.

Example: Gridworld



Convergence of gridworld value functions compared with the theoretical bounds. The black dashed line is the general theoretical bound for both T and T_τ by Banachs fixed point theorem and the maximum value $V_{\max} = R_{\max}/(1-\gamma)$. The dotted blue and orange uses $\left|V_k^*\right|$ and $|V_{\tau,k}|$ respectively, which might not be available. $(\gamma = 0.9)$.

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[4].

[4]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*.

Motivation for Q-functions

A clear advantage of working with Q-function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ rather than a value function $V : \mathcal{S} \to \mathbb{R}$, is that finding the optimal action $a^* \in \mathfrak{A}(s)$ at state s requires only a maximization over the Q-function itself: $a^* = \text{argmax}_{a \in \mathfrak{A}(s)} Q(s, a)$. This should be compared to finding an optimal action according to a value function V: $a^* = \text{argmax}_{a \in \mathfrak{A}(s)} r(s, a) + \gamma \mathbb{E}_{P(\cdot|s,a)} V$.

Greed with Q-functions

Formally we define greedy actions and policies w.r.t. a Q-function as 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

Moreover we define T-operators similar to ones for value functions

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 $T_a = T_{\delta_a}$.

Relation between value- and Q-functions

Theorem (Relations between Q- and value functions)

Let $\pi = (\tau_1, \tau_2, \dots) \in M\Pi$ be a Markov policy and $\tau \in S\Pi$ stationary. Then

- Policy evaluations are related by $\mathbb{E}_{\tau(\cdot|s)} Q_{k,\pi} = V_{k+1,(\tau,\pi)}(s)$.
- T_τ-operators are related by $T_\tau Q_{k,\pi}(s,a) = r + \gamma \mathbb{E}_{P(\cdot|s,a)} T_\tau V_{k,\pi}$.
- τ is greedy for $Q_{k,\pi}$ if and only if τ is greedy for $V_{k,\pi}$ and τ is greedy for Q_π if and only if τ is greedy for V_π.
- Optimal policies are related by $\max_{a \in \mathfrak{A}(s)} Q^*(s,a) = V^*(s)$ and

$$Q_k^*(s,a) = r(s,a) + \gamma \mathbb{E}_{P(\cdot|s,a)} V_k^*, \quad Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{P(\cdot|s,a)} V^*$$

Properties of Q-functions

Because of the close relations many properties are inherited from value function to Q-functions:

Proposition (Properties of Q-functions)

Let $\pi = (\tau_1, \tau_2, \dots) \in M\Pi$ be a Markov policy and $\tau \in S\Pi$ stationary. Then

$Q_{k,\pi} = T_{\tau_1} \dots T_{\tau_k} Q_0$ and $Q_k^* = T^k Q_0^*$.

$Q_\pi = \lim_{k\to\infty} Q_{k,\pi}$ and $Q^* = \lim_{k\to\infty} Q_k^*$.

T, T_τ are γ-contractive on $\mathcal{L}_\infty(\mathcal{S} \times \mathcal{A})$ and Q^*, Q_τ are their unique fixed points.

$Q^* = Q_{\tau^*}$

Q-iteration

Q-iteration is the analogue of value-iteration for Q-function. It can be stated in the form of an algorithm as follows:

Algorithm (Q-iteration)

Data: MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, number of iterations K
Initialize expected reward function $r \leftarrow \int x \,\mathrm{d}R(x \mid \cdot)$ and $\widetilde{Q}_0 \leftarrow r$.
for $k = 0, 1, \ldots K - 1$ **do**
  $\widetilde{Q}_{k+1} \leftarrow T\widetilde{Q}_k$
end
Output: $\widetilde{Q}_K$

In the context of a greedy MDP we immedially have that the output of the Q-iteration algorithm $\widetilde{Q}_K = Q_K^*$ is K-optimal.

Value iteration with Q-functions

Similar to value-iteration we can use Banach fixed point theorem with the contractive properties of the T-operator for Q-functions to obtain exponential convergence of Q-iteration:

Proposition (Convergence of Q-iteration)

Suppose the Q-iteration algorithm is run with a greedy MDP. Then the output $\widetilde{Q}_K = Q_K^*$ satisfy

$$\left\| Q^* - Q_K^* \right\|_\infty \leqslant \gamma^K V_{\max}$$

Why are we not done?

We have exponential convergence for the broad class of problems expressible as a greedy MDP. This class includes highly difficult environments such as control problems in time-descretized simulation environments such as computer games, including the game of *chess*. Are we then done?

Problems of Q-iteration

1. It is assumed that we know how to integrate over P and R.
   - The distributions of P and R might be impractical to work with in a computer.
   - It is common in RL to assume that P and R are unknown, thus including a variety of environments, which we have not yet considered.
2. It is assumed that we know how to represent Q functions in a feasible way in a computer.

Example: Chess

The state space of chess is very large (roughly $|\mathcal{S}_{\text{chess}}| \geqslant 10^{43}$). This means that if we were to use Q-iteration naively (with finite implementation as in the gridworld example) then we would have to store a vector of roughly $N \cdot 10^{43}$ real numbers for each Q-function we define, where N is the average number of admissable actions at each state $\mathfrak{A}(s), s \in \mathcal{S}$ which has been estimated to around $N \approx 35$ for chess. This requires roughly $1.4 \cdot 10^{45}$ bytes, if each number is stored as a single precision floating point number (4 bytes). For comparison the entire digital data capacity in the world is estimated less than 10^{23} bytes as of 2020. Needless to say this is beyond any practical relevance.

What have we done so far?

Model-dependent algorithms

Another frame