Annual Report 2021
Paper 2021-02

GEOMET
@QUEEN'S
UNIVERSITY

Fundamentals of deep Q-Learning [1]

Sebastian Avalos (sebastian.avalos@queensu.ca)
Julian M Ortiz (julian.ortiz@queensu.ca)

Abstract

Reinforcement learning has achieved remarkable performances on oriented decision making problems. The agent-environment framework provides the principles for mapping states-and-actions to expected value rewards, maximizing the long-term total reward. The mapping function can be retrieved from look-up tables when the space of states and actions are small enough to maintain computational efficiency, or parametrized as an approximation of the underlying mapping. In real life problems, the environment is often incomplete, and the space of states and actions are non trackable or computationally unmanageable. Recent advances on Deep Learning have led to implement deep neural networks to approximate the mapping function, referred as deep Q-Learning. In this brief article, we review the building blocks of reinforcement learning with a final focus on the principles of deep Q-Learning.

1. Introduction

The field of reinforcement learning (RL) has its roots and draws insights from neurosciense, psychology, and computer science (Ludvig et al., 2011). Collaborative efforts have helped to strengthen the RL framework, providing methods and models on how agents (animals, humans or robots) learn to make decisions from past experiences of agent-environment interaction (Sutton and Barto, 2018). From a computational perspective, reinforcement learning is the framework of machine learning in which an agent is trained to maximize the reward over time as a result of chosen actions in a sequence of interactions within a particular environment. The learning process follows the principles of sequential decision making, where actions influence the immediate reward, the environment state, and all subsequent environment states, feasible actions and possible states. Therefore, the agent must learn the evaluate the quality of taking an action based on the current environment state and according to the immediate reward and delayed rewards.

To introduce concepts, we explore the following reduced-learning setting: every time, an agent must make a choice between k different actions, receiving an immediate reward drawn from a stationary distribution, without perceiving and/or altering the environment. The aim is to maximize the total reward in a finite number of choices (time steps). Let a_t be the action selected at time step t, and r_t the corresponding reward. The value of selecting the action a corresponds to the expected reward, and can be expressed as:

$$q_*(a) \doteq \mathbb{E}[r_t | a_t = a] \tag{1}$$

[1] Cite as: Avalos S, Ortiz JM (2021) Fundamentals of deep Q-Learning, Predictive Geometallurgy and Geostatistics Lab, Queen's University, Annual Report 2021, paper 2021-02, 14-21.

Maximizing the total reward is trivial when $q_*(a)$ is known for all possible actions a: select the action with highest value. Naturally, we do not have access to the real value $q_*(a)$ but it can be estimated based on previous experiences. Let $q_t(a)$ be the estimated value of selecting action a at time step t. Without considering when an action was taken but only the number of times it was taken and each of the corresponding rewards, a simple way to estimate the value of an action is by the sample-average method as:

$$q_t(a) = \frac{\sum_{i=1}^{t-1} r_i \cdot \mathbb{1}_{a_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{a_i=a}} \tag{2}$$

where $\mathbb{1}_{a_i=a}$ is 1 when $a_i = a$, and 0 otherwise. The action with highest value is drawn as:

$$a_t = \arg\max_a q_t(a) \tag{3}$$

The action(s) with highest estimated value is called a greedy action. We refer to the process of selecting a greedy action as *exploitation*, since the agent is exploiting the accumulated knowledge of previous experiences. We refer to the selection of non-greedy action as *exploration*. The latter allows the agent to update the estimate of non-greedy actions. The trade-off between exploration and exploitation is non trivial. Nevertheless, we can easily argue that exploration is fundamental in the early stages of a learning process, whereas exploitation is desired when certain stationarity is observed in the estimated values, in the latest stages of a learning process.

When looking at a single action that has been selected n times, we can compute the current estimated value q_n as:

$$\begin{aligned}
q_{n+1} &= \frac{r_1 + r_2 + \cdots r_{n-1} + r_n}{n} \\
q_{n+1} &= \frac{1}{n}\left(r_n + \frac{n-1}{n-1}\sum_{i=1}^{n-1} r_i\right) \\
q_{n+1} &= \frac{1}{n}\Big(r_n + (n-1)\cdot q_n\Big) \\
q_{n+1} &= q_n + \frac{1}{n}\Big(r_n - q_n\Big)
\end{aligned} \tag{4}$$

The previous update representation, from q_n to q_{n+1} knowing the last reward r_n and the number of times that the action has been taken n, has the structure:

$$\text{NewEstimation} \longleftarrow \text{OldEstimation} + \alpha \cdot [\text{Target} - \text{OldEstimation}] \tag{5}$$

The expression $[\text{Target} - \text{OldEstimation}]$ denotes the error between the desired value and the old estimation. The parameter α controls the rate in which the estimation value is updated, and is often expressed as a function of the time step and the corresponding action, $\alpha_t(a)$.

The distribution of reward probabilities has been assumed constant over time. In this scenario, an equal weight of previous experience is reasonable, such as $\alpha_t(a) = \frac{1}{n}$, which changes over time. For non-stationary situations, the intuition suggests to increase the weights to recent experiences and decrease the weights of old ones. Using a constant value $\alpha \in (0, 1]$ satisfies the desired property, transforming Eq. 4 into a weighted sum of the past reward and past estimation :

$$q_{n+1} = \alpha \cdot r_n + (1 - \alpha) \cdot q_n \tag{6}$$

By recursion, Equation 6 can be rewritten as a function of past rewards and the initial estimation as:

$$q_{n+1} = (1 - \alpha)^n \cdot q_1 + \sum_{i=1}^{n} \alpha \cdot (1 - \alpha)^{n-i} \cdot r_i \tag{7}$$

To guarantee convergence over time, we need α to satisfy both:

$$\sum_{t=1}^{\infty} \alpha_t(a) = \infty \qquad , \qquad \sum_{t=1}^{\infty} \alpha_t^2(a) < \infty \tag{8}$$

where the former expression implies enough steps to overcome initial conditions, while the latter expression implies a decrease in the step-size during learning to secure convergence. Note that the latter expression is not met when *alpha* is set constant, a desired property in non-stationary situations.

Until now, the learning process has focused on the estimation of the action values to maximize a total reward. Either stationary or non-stationary, the value of each action has been assumed unrelated to the context of learning. When the context is considered, the agent must learn how to evaluate actions conditioned to different situations. The dynamic of learning in a agent-environment framework is described in the following section.

2. The agent-environment framework

The interaction agent-environment is often discretized in time steps. Time steps are not required to represent the formal time dimension but rather sequential decisions steps. Formally, at each time step $t = 0, 1, 2, 3, ..., T$, the agent perceives the partial or complete state of the environment $s_t \in \mathcal{S}$ and must take an action $a_t \in \mathcal{A} = \{1, ..., |\mathcal{A}|\}$, receiving a single reward $r_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and modifying the environment into its next state s_{t+1}, as shown in Figure 1.

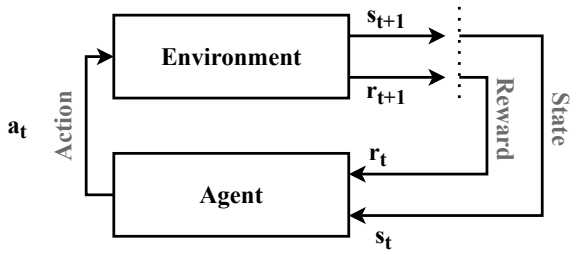

Figure 1: Reinforcement learning, agent-environment interaction scheme.

An agent-environment interaction results into a sequence of events (trajectory) in the form:

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, ... \tag{9}$$

Whenever T, $\mathcal{A}$, and $\mathcal{R}$ are finite subsets, the learning framework can be formally represented and described as a finite Markov Decisions Process (MDP). In MDP, the entire system is characterized by the mapping function from the pair [state, action] into [next state, reward]. Formally, let $s', s \in \mathcal{S}$, $r \in \mathcal{R}$, and $a \in \mathcal{A}$, the probability p of transition from state s into s' by taking action a and receiving the reward r is written as:

$$p(s', r|s, a) \doteq P(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a) \qquad , \qquad \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1 \tag{10}$$

where p represents the dynamic of the entire MDP system. Therefore, the probability of choosing an arbitrary action a depends only on the current state s and not on previous states.

Similar to the value estimation of an arbitrary action a in the non-associate task of Equation 7, we need to estimate the value of taking an arbitrary action a at any state s, denoted as $q(s, a)$. The cumulative reward G_t, at time step t, can be expressed from Equation 9 into Equation 11 as:

$$G_t = r_{t+1} + r_{t+2} + \cdots + r_T \tag{11}$$

with T as the final step. The previous formula works on finite oriented tasks in which the order of rewards is irrelevant and the agent-environment interaction sequence is finite. In order to make Equation 11 suitable for continuous oriented tasks or when the order of rewards matters, a discounted factor $\gamma \in [0, 1[$ is introduced, such that the cumulative discounted reward is computed as:

$$G_t \approx r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} + \gamma^3 \cdot r_{t+4} + ... + \gamma^{T-t-1} \cdot r_T = \sum_{k=0}^{T} \gamma^k \cdot r_{t+k+1} \tag{12}$$

From now on, we assume $T \to \infty$ without loss of generality. As the sequence of rewards depends on the sequence of actions taken over the sequence of states, we look for an estimator of the pair state-action, in terms of future rewards, to guide the agent. The agent acting behaviour on the environment is referred as the agent's *policy*.

Let $\pi(a|s)$ be the probability of chosen action a at the state s under the agent's policy π. The state-value function, $v_\pi(s)$, represents the expected total reward of the state s for policy π, and is formally expressed as:

$$v_\pi(s) = \mathbb{E}_\pi\big[G_t \big| s_t = s\big] \approx \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} \Big| s_t = s\right] \tag{13}$$

We define $q_\pi(s, a)$ as the action-value function, corresponding to the expected return of taking action a at state s under the policy π at time step t, and then following the same policy. It is computed as:

$$q_\pi(s, a) = \mathbb{E}_\pi\big[G_t \big| a_t = a, s_t = s\big] \approx \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} \Big| a_t = a, s_t = s\right] \tag{14}$$

Both, Equation 13 and Equation 14 can be estimated by previous experiences, similar to the simple average-method described earlier, when the number of states and actions are small enough to be stored and retrieved. When the space of action and/or state makes the store-and-retrieve

process inefficient, v_π and q_π can be parametrized, reducing the number of parameters describing the functions. When the space of actions and/or space become unmanageable or incomplete during the process of learning, the use of deep neural network architectures serves to map State-Actions with NextState-Rewards. The latter is referred as deep Q-Learning, and we elaborate on this concepts in the following section. Before that, we introduce the concept of Bellman equations by rewriting Equation 13 as:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi\big[r_{t+1} + \gamma \cdot G_{t+1} \big| s_t = s\big] \\ v_\pi(s) &= \sum_{a\in\mathcal{A}} \pi(a|s) \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s', r|s, a)\Big[r + \gamma \cdot \mathbb{E}_\pi\big[G_{t+1}\big|s_{t+1} = s'\big]\Big] \\ v_\pi(s) &= \sum_{a\in\mathcal{A}} \pi(a|s) \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s', r|s, a)\Big[r + \gamma \cdot v_\pi(s')\Big] \end{aligned} \tag{15}$$

The last expression translates into weighting each possible future response $[r + \gamma \cdot v_\pi(s')]$ by their probabilities of occurrence $\pi(a|s)p(s', r|s, a)$. In other words, it represents the value of a state as a function of the possible immediate rewards and value states.

Similarly, Equation 14 can be rewritten as:

$$q_\pi(s, a) = \mathbb{E}_\pi\big[r_{t+1} + \gamma \cdot G_{t+1} \big| a_t = a, s_t = s\big] \approx \mathbb{E}_\pi\left[r_{t+1} + \gamma \cdot \sum_{k=0}^{\infty} \gamma^k \cdot r_{(t+1)+k+1} \Big| a_t = a, s_t = s\right] \tag{16}$$

and by the principles of the Bellman equation, restated as:

$$q_\pi(s, a) = \mathbb{E}_\pi\big[r_{t+1} + \gamma \cdot q_\pi(s_{t+1}, a_{t+1}) \big| a_t = a, s_t = s\big] \tag{17}$$

We have from Equation 17 that the state-action value $q_\pi(s, a)$ can be decomposed into the immediate reward r_{t+1} of taking action a on the state s at time step t plus the discounted state-action value function $q_\pi(s_{t+1}, a_{t+1})$ at the next time step $t + 1$, recursively.

3. Deep Q-Learning

The Q-Learning technique was proposed by Watkins and Dayan (1992) as a simple approach for learning by successively improving the assessment of particular actions at particular states. The action-value function in Q-Learning is updated according to the expression:

$$q(s_t, a_t) \leftarrow (1 - \varepsilon) \cdot q(s_t, a_t) + \varepsilon \cdot \big[r_{t+1} + \gamma \cdot \max_{a_{t+1}} q(s_{t+1}, a_{t+1})\big] \tag{18}$$

where $\gamma \in [0, 1[$ and $\varepsilon \in [0, 1]$ are the discount factor and learning coefficient. When $\varepsilon : 1$ the action-value function is updated according to the received reward and discounted maximum action-value at the next state. When $\varepsilon : 0$ the action-value is not updated. This resembles the trade-off between exploration and exploitation. Indeed, we define an iteration as the moment when the agent has interacted with the environment through the entire time period or until the interaction has ended. Then, let ε_i be the epsilon value at the ith iteration, the ε_{decay} parameter controls the rate between exploration and exploitation as the training progresses, as $\varepsilon_{i+1} \leftarrow \varepsilon_i \cdot \varepsilon_{decay}$. Figure 2 illustrates the effects on ε_i by using $\varepsilon_{decay} : 0.99$

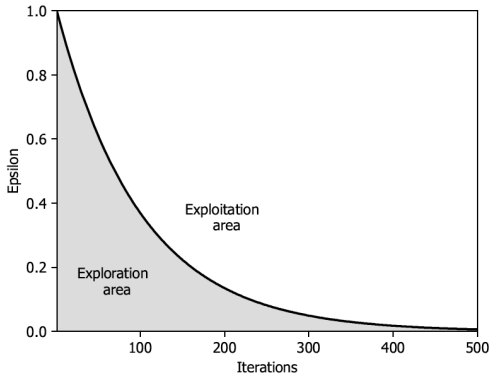

Figure 2: Epsilon greedy method. Exploration - exploitation dilemma.

The previous formulation requires to build a look-up table of size $\mathcal{S} \times \mathcal{A} \times T$ for all possible states (discrete), actions and time steps. Thus, the applicability of Q-Learning techniques has been constrained by computational power and limited to low-dimensional state and action spaces. In response, Mnih et al. (2015) proposed the Deep Q-Learning framework in which a deep neural network is trained to approximate the function action-value function $q(s_t, a_t)$. The latter directly extends the state space from a discrete to a continuous space. An extended theoretical and statistical analysis can be found at Fan et al. (2020). In the following, we focus on the main principles of Deep Q-Learning. We rewrite Equation 17 as function of states-and-actions as:

$$q_\pi(s_t, a_t) = \mathbb{E}_\pi\Big[r(s_t, a_t) + \gamma \cdot \sum_{s_{t+1} \in \mathcal{S}} p(s_{t+1}|s_t, a_t) \max_{a_{t+1}} q_\pi(s_{t+1}, a_{t+1})\Big] \tag{19}$$

Solving Equation 19, the optimal policy π corresponds to:

$$\pi(s) = \arg\max_{a \in \mathcal{A}} q(s, a) \tag{20}$$

The action-value function $q(s, a)$ is approximated by a deep neural network (DQN) that outputs a set of action-values of the form $q_\theta(s, \cdot)$ where θ corresponds to the set of neural network parameters. The use of a DQN allows the implementation of two tricks that accelerate the learning stage: replay memory and target network.

Let $\mathcal{M}$ be the set of experiences (memory set) stored in the form $(s_t, a_t, r_{t+1}, s_{t+1})$. Let $q_{\theta^*}(s, a)$ be a target network. The learning process starts with an empty memory set $\mathcal{M} = \emptyset$, random weights on the network parameter θ, and initial state s_0. The weights of the target network are initialized as $\theta^* = \theta$. At each iteration ith, we start from $t = 0$ until the interaction agent-environment ends. At each time t, the following steps are carried out:

1. With probability ε_i a random action is selected, and with probability $(1 - \varepsilon_i)$ the action is selected by $\arg\max_{a_t \in \mathcal{A}} q_\theta(s_t, a_t)$.
2. Once the action is executed, the immediate reward and the new state are stored in $\mathcal{M}$.
3. Randomly draw n transition samples from $\mathcal{M}$: $\{(s_j, a_j, r_j, s'_j)\}_{j \in [n]}$
4. For each sample, compute the target value $y_j = r_j + \gamma \cdot \arg\max_{a \in \mathcal{A}} q_{\theta^*}(s'_j, a)$

5. Update the network parameters using an optimization algorithm with a temporal learning rate α_t. For instance, using the gradient descent method:

$$\theta \leftarrow \theta - \alpha_t \cdot \frac{1}{n} \sum_{j=1}^{n} [y_j - q_\theta(s_j, a_j)] \cdot \nabla_\theta q_\theta \tag{21}$$

6. Every τ time steps, update the target network parameters as $\theta^* \leftarrow \theta$.

As a result, after training, the optimal policy $\pi_\theta(s)$ with respect to $q_\theta(s, a)$ is obtained as:

$$\pi_\theta(s) = \arg\max_{a \in \mathcal{A}} q_\theta(s, a) \tag{22}$$

4. Final remarks

We have covered the building blocks of reinforcement learning but many aspect have been left aside for simplicity, such as temporal difference learning, On-policy and Off-policy, SARSA, Monte Carlo tree search, exhaustive search, among others methods and principles. From the revised fundamentals, special attention is suggested on the following aspects when applying deep Q-Learning:

Environment representation To obtained valid, realistic and/or functional state-value and action-value functions, the environmental must be adequately represented in such a way that the agent is capable to interpret the differences between different states.

Reward The reward drives and guides the agent learning process. The reward must be in line with the long-term goal and must avoid pitfalls in which the agent would maximize the total reward without necessarily achieving the long-term goal.

Deep neural network The architecture of the network must be in accordance with the nature of the environment (spatial-temporal). Ensemble architectures would improve the capacity of approximating the action-value function in deep Q-Learning.

5. Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), funding reference number RGPIN-2017-04200 and RGPAS-2017-507956.

6. Bibliography

Fan, J., Wang, Z., Xie, Y., Yang, Z., 2020. A theoretical analysis of deep Q-learning, in: Learning for Dynamics and Control, PMLR. pp. 486–489.

Ludvig, E.A., Bellemare, M.G., Pearson, K.G., 2011. A primer on reinforcement learning in the brain: Psychological, computational, and neural perspectives. Computational neuroscience for advancing artificial intelligence: Models, methods and applications , 111–144.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. . nature 518, 529–533.

Sutton, R.S., Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Watkins, C.J., Dayan, P., 1992. Q-learning. Machine learning 8, 279–292.