Deep Reinforcement Learning: From Q-Learning to Deep Q-Learning

Fuxiao Tan[1(✉)], Pengfei Yan[2], and Xinping Guan[3]

[1] School of Computer and Information Engineering, Fuyang Normal University, Fuyang 236037, Anhui, China
fuxiaotan@gmail.com
[2] School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, China
pengfei.yan.ia@foxmail.com
[3] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
xpguan@sjtu.edu.cn

Abstract. As the two hottest branches of machine learning, deep learning and reinforcement learning both play a vital role in the field of artificial intelligence. Combining deep learning with reinforcement learning, deep reinforcement learning is a method of artificial intelligence that is much closer to human learning. As one of the most basic algorithms for reinforcement learning, Q-learning is a discrete strategic learning algorithm that uses a reasonable strategy to generate an action. According to the rewards and the next state generated by the interaction of the action and the environment, optimal Q-function can be obtained. Furthermore, based on Q-learning and convolutional neural networks, the deep Q-learning with experience replay is developed in this paper. To ensure the convergence of value function, a discount factor is involved in the value function. The temporal difference method is introduced to training the Q-function or value function. At last, a detailed procedure is proposed to implement deep reinforcement learning.

Keywords: Deep reinforcement learning · Q-learning · Deep Q-learning · Convolutional neural networks

1 Introduction

In 2006, the concept of deep learning (DL) was proposed by Hinton et al., which was originated from the research of artificial neural networks [1]. Deep learning is a new field in the study of machine learning. Its motivation lies in the establishment and simulation of the human brain by artificial neural networks, which mimics the human brain's mechanism to interpret data, in the fields of computer vision, image processing, speech and audio [2].

The important factors for the success of deep learning include the strong fitting and generalization ability of the model, the high-density of computing

D. Liu et al. (Eds.): ICONIP 2017, Part IV, LNCS 10637, pp. 475–483, 2017.
https://doi.org/10.1007/978-3-319-70093-9_50

power, and the huge amount of training data. Based on the nonlinear operation of a large number of neurons, the deep neural network can acquire a strong fitting and generalization ability. Using a high-density computing device such as a GPU, the network model can be trained on a vast amount of training data (million level) in an acceptable amount of time (days) [3].

Machine learning is a multidisciplinary and interdisciplinary subject, which involves many subjects such as probability theory, statistics, approximation theory, convex analysis and computational complexity theory. Machine learning is a specialized discipline, which investigates how the computer simulates or realizes human learning behavior to acquire new knowledge or skills and reorganizes existing knowledge structures to continually improve their performance. Machine learning is the core of artificial intelligence, which is also the fundamental approach to make computers intelligent [6].

In the field of machine learning, based on the theories of Markov decision processes and temporal difference learning, reinforcement learning is still facing many difficulties and challenges [7], such as high dimensional approximate optimal strategy of continuous space, the stability of the learning algorithms, structured hierarchical optimization, reward function design, and so on [8].

1.1 Deep Learning and Neural Networks

The concept of deep learning is originated from the research of artificial neural networks (ANNs). The mathematical model of ANNs is made of layers of neurons [9]. ANNs are distributed parallel information processing algorithms, which are used to simulate the behavior of animal neurons. Depending on the complexity of the system, ANNs realize the purpose of processing information by adjusting the interconnection between large numbers of internal nodes.

The so-called deep learning is the neural network composed of multi-layer neurons to approximate the function of machine learning. The structure of deep learning is multilayer perceptron with multiple hidden layers. By combining low-level features to form more abstract high-level representations of attribute categories or features, deep learning can discover the distributed characteristics of data [10].

1.2 Deep Reinforcement Learning

The hottest branches of machine learning are deep learning and reinforcement learning. Reinforcing learning (RL) is a process of continuous decision-making [11]. The characteristics of RL is not giving any data annotations, but just providing a return function that determines the results of the current state (such as "good" or "bad"). In the essence of mathematics, it is a Markov decision process [12]. The ultimate goal of reinforcement learning is to obtain the expectation of optimal overall return function in the decision making process [13].

Reinforcement learning and deep learning are two very different algorithms of machine learning, but the deep learning algorithm can be used to implement reinforcement learning, which forms deep reinforcement learning (DRL).

Combining reinforcement learning with deep learning, it is possible to find an agent that can solve any human level task. Reinforcement learning defines the goal of optimization, while deep learning gives the mechanism of operation, which is the approach to characterize the problem and the way to solve the problem [14]. Therefore, deep reinforcement learning has a kind of ability to solve many complex problems. Thus, it will present more intelligent behaviors. So, deep reinforcement learning can be understood as part of the reinforcement learning achieved by deep learning. In essence, DRL can be considered as a learning and thinking algorithm [15].

2 Reinforcement Learning

Combining the perceptive ability of deep learning with the decision-making ability of reinforcement learning, deep reinforcement learning is a kind of artificial intelligence method which is closer to the human learning modes.

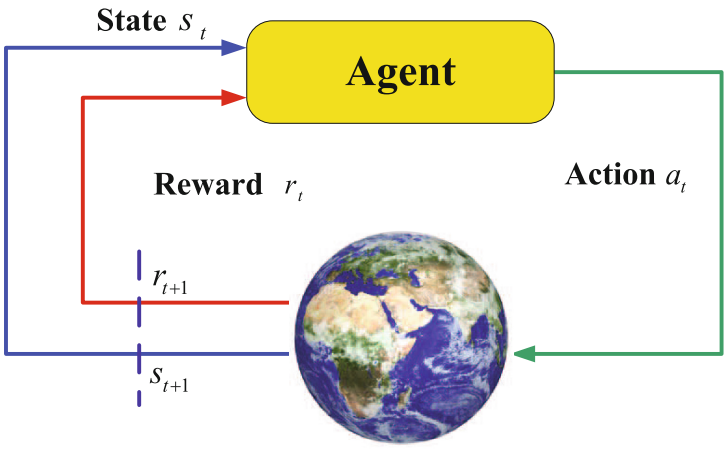

Fig. 1. The agent-environment interaction in reinforcement learning.

In Fig. 1, the agent and environment interact at each discrete time steps of a sequence, $t = 0, 1, 2, 3, \cdots$. At each time step t, the agent receives some representation of the environment's state $s_t \in S$, where S is the set of possible states and the action is $a_t \in A(s_t)$, where $A(s_t)$ is the set of actions available in state s_t. At time $t + 1$, as a consequence of its actions, the agent receives a numerical reward $r_{t+1} \in R$ and finds itself in a new state s_{t+1}.

At each time step, the agent implements a mapping from state to probabilities of each possible action. This mapping is called the agent's policy and is denote as π_t, where $\pi_t(s, a)$ is the probability that $a_t = a$ if $s_t = s$. Reinforcement learning methods specify how the agent changes its policy as a result of its experience. The agent's goal is to maximize the total amount of reward received over the long term.

2.1 Markov Decision Process

In general, Markov decision process (MDP) is denoted as a tuple $\{A, S, R, P\}$, where S is the state space of the environment, A is the action space of the agent, $R : S \times A \times S \to \Re$ is a reward function, and $P : S \times A \times S \to [0, 1]$ is a state transition function. In Fig. 1, for the RL structure, we define a function $\pi : S \times A \to \Pr(A)$ as a stochastic stationary policy of MDP, where $\Pr(A)$ is a probability distribution in the action space. So, the stationary policy π directly maps states to distributions over the action space, which is denoted as $a_t = \pi(s_t), t \geq 0$.

In sequential making, RL can be modeled as a MDP. At each time step t, the agent receives a state s_t and selects an action a_t from action space A, following a policy $\pi(a_t | s_t)$, i.e., mapping from state s_t to action a_t. Then, it receives a scalar reward r_t, and transitions to the next state s_{t+1}, according to the model, reward function $R(s, a)$ and state transition probability $P(s_{t+1} | s_t, a_t)$.

The set of states and actions, together with rules for transition from one state to another, make up a Markov decision process. One episode of this process (e.g. one game) forms a finite sequence of states, actions and rewards:

$$\{(s_0, a_0, r_1), (s_1, a_1, r_2), \cdots, (s_{t-1}, a_{t-1}, r_t)\} \tag{1}$$

where s_t represents the state, a_t is the action and r_{t+1} is the reward after performing the action. The episode ends with terminal state s_t (e.g. game over screen). A Markov decision process relies on the Markov assumption, that the probability of the next state s_{t+1} depends only on current state s_t and action a_t, but not on preceding states or actions.

2.2 Value Function

According to the theory of Markov decision process, the total reward for one episode can be calculated: $R = r_1 + r_2 + \cdots + r_n$. Thus, the total future reward from time t onward can be expressed as: $R_t = r_t + r_{t+1} + \cdots + r_{t+n}$.

If the same actions are performed and the same rewards next time are also got, the total future reward will be diverged. Therefore, the above rewards are substituted by discounted future reward:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-t} r_n \tag{2}$$

where γ is the discount factor between 0 and 1. It is easy to see, that discounted future reward at time step t can be expressed in terms of the same thing at time step $t + 1$:

$$R_t = r_t + \gamma(r_{t+1} + \gamma(r_{t+2} + \cdots)) = r_t + \gamma R_{t+1} = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{3}$$

A good strategy for an agent would be to always choose an action that maximizes the (discounted) future reward.

Along the state trajectories by policy π, let J_π be the expected total rewards. Without loss of generality, the goal of reinforcement learning is to estimate an optimal policy π^* which satisfies

$$J_{\pi^*} = \max_{\pi} J_\pi = \max_{\pi} E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \tag{4}$$

where r_t is the reward at time-step t, γ is a discount factor, and $E_\pi[\cdot]$ is the expectation with respect to the policy π and the state.

For a stationary policy π, in this paper, the state value function is defined as $V^\pi(s) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \,|s_0 = s \right]$. Thus, the optimal value function is defined as $V^{\pi^*}(s) = E_{\pi^*} \left[\sum_{t=0}^{\infty} \gamma^t r_t \,|s_0 = s \right]$.

To facilitate policy improvement, we defines the state-action value function $Q^\pi(s,a)$ as

$$Q^\pi(s,a) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \,|s_0 = s, a_0 = a \right] \tag{5}$$

and the optimal state-action value function

$$Q^{\pi^*}(s,a) = E_{\pi^*} \left[\sum_{t=0}^{\infty} \gamma^t r_t \,|s_0 = s, a_0 = a \right] \tag{6}$$

According to the Bellman optimality

$$V^\pi(s) = \sum \pi(a\,|s) E\left[R_{t+1} + \gamma V(s_{t+1}) \,|S_t = s\right] \tag{7}$$

the optimal value function satisfies

$$V^*(s) = E\left[R_{t+1} + \gamma \max^{\pi} V(s_{t+1}) \,|S_t = s\right] \tag{8}$$

and

$$Q^*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') \,|S_t = s, A_t = a\right] \tag{9}$$

where $R(s,a)$ is the expected reward received after taking action a in state s and s' is the successive state of s.

Thus, the optimal state-action value function is $Q^*(s,a) = \max_{\pi} Q^\pi(s,a)$. So, the optimal policy is computed by

$$\pi^*(s) = \arg\max_{a} Q^*(s,a) \tag{10}$$

In Fig. 1, after estimating the optimal state-action value function by $\tilde{Q}^*(s,a)$, a near-optimal policy can be obtained as

$$\tilde{\pi}^*(s) = \arg\max_{a} \tilde{Q}^*(s,a). \tag{11}$$

2.3 Q Learning

Temporal difference (TD) learning is a central idea in RL. It learns value function $V(s)$ directly from experience with the TD error, with bootstrapping, in a model-free, online, and fully incremental way. The update rule is

$$V(s_t) \leftarrow V(s_t) + \alpha [r_t + \gamma V(s_{t+1}) - V(s_t)] \tag{12}$$

where α is a learning rate. The $[r_t + \gamma V(s_{t+1}) - V(s_t)]$ is called TD error.

Similarly, Q-learning learns action value function with the update rule

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{13}$$

Q-learning is an off-policy control method [16].

3 Deep Reinforcement Learning

3.1 Deep Neural Network

We obtain deep reinforcement learning (deep RL) methods when we use deep neural networks to approximate any of the following component of reinforcement learning: value function, $V(s, \theta)$ or $Q(s, a; \theta)$, policy $\pi(a|\, s; \theta)$, and model (state transition and reward). Where the parameters θ are the weights in deep neural networks.

In the DeepMind paper [17], in order to obtain the four last screen images, there need resize them to 84×84 and convert to grayscale with 256 gray levels. Thus, there are $256^{84 \times 84 \times 4} \approx 10^{67970}$ possible game states. This means that there need 10^{67970} rows in the Q-table, and the number is a lot bigger than the number of atoms in the known universe.

Since many states (pixel combinations) do not occur, you can actually use a sparse table to include the states that are being accessed. Even so, many states are still rarely accessed. Maybe it will take lifetime of the universe for the Q-table to converge. Therefore, in algorithmic designing, the idealized scenario is to have a good guess for Q-values that has not yet been met.

To solve the above problems, it requires deep learning to take part in. The main role of neural networks is to derive features from highly structured data. Therefore, Q-function can be represented by neural network, which means the state and action as input and the corresponding Q-value as the output. In general, this is actually a classical convolutional neural network with three convolutional layers and followed by two fully connected layers.

3.2 Experience Replay

By now we have an idea how to estimate the future reward in each state using Q-learning and approximate the Q-function using a convolutional neural network. However, the main problem is that approximation of Q-values using

non-linear functions is not very stable. So, it requires some tricks to guarantee the convergence of the algorithm.

In deep learning, the widely-used of trick is experience replay. In algorithm execution, all experience $\{s, a, r, s'\}$ are stored in a replay memory. When conducting deep Q network training, the random minibatches from replay memory will replace the most recent transition. This will break the similarity of subsequent training samples, otherwise it is easy to direct the network into one of a local minimum. Furthermore, the experience replay makes the training task more similar to the usual supervised learning, which also simplifies the program's debug and algorithm testing.

3.3 Exploration-Exploitation

Reinforcement learning is a kind of a trial-and-error learning method. In the beginning, agent is not clear that the environment and the ways of working. The agent don't know what kind of action (behavior) is right, what kind is wrong. Therefore, agent needs to find a good policy from the experience of trial and trial to gain more reward in this process.

Therefore, in deep reinforcement learning, there is a tradeoff between Exploration and Exploitation. Exploration will give up some known information of reward and try a few new choice. Namely, under some kind of states, algorithm may have to learn how to choose the action for larger reward, but it doesn't make the same choice every time. Perhaps another choice that hasn't been tried will make the reward larger, that is the agent's goal to explore more information about environment. Moreover, exploitation refers to maximizing reward according to known information of agent.

3.4 Deep Q-Networks

To approximate the value function $Q(s, a; \theta)$ with parameters θ (that is, weights), a deep convolutional neural network can be applied in the designing of DRL. To carry out experience replay at each time t, the agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ is stored into a dat set $D_t = \{e_1, \cdots, e_t\}$.

The deep Q-learning updating at iteration i uses the following loss function

$$L_i(\theta_i) = E_{(s,a,r,s')}\left[\left(y_i - Q(s, a; \theta_i)\right)^2\right] \tag{14}$$

with

$$y_i = r + \gamma \max_{a'} Q\left(s', a'; \theta_i^-\right) \tag{15}$$

where θ_i^- are the network parameters used to compute the target at iteration i.

Without any generalization, it is common to use a function approximate to estimate the action-valued function

$$Q(s, a; \theta) \approx Q^*(s, a) \tag{16}$$

Differentiating the loss function with respect to the parameters, the gradient is

$$\nabla_{\theta_i} L(\theta_i) = E_{s,a,r,s'}\left[\left(r + \gamma \max_{a'} Q\left(s', a', \theta_i^-\right) - Q(s, a; \theta_i)\right) \nabla_{\theta_i} Q(s, a; \theta_i)\right] \tag{17}$$

From above, the deep Q-learning algorithm is as follows [5].

Algorithm 1. Deep Q-learning with experience replay.

Require:
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize the target action-value function $\hat{Q}$ with weights $\theta^- = \theta$

Ensure:
for episode = 1 to M **do**
  initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$;
  for $t = 1$ to T **do**
    With probability ε select a random action a_t;
    Otherwise select $a_t = \arg\max_a Q(\phi(s_t), a; \theta)$;
    Execute action a_t in emulator and observe reward r_t and image x_{t+1};
    Set $s_{t+1} = s_t$, a_t, x_{t+1} and preprocess $\phi_{t+1} = \phi(s_{t+1})$;
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D;
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D;
    Set $y_j = r_j$ if episode terminates at step $j + 1$, otherwise $y_j = r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right)$;
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameter θ;
    Every C steps reset $\hat{Q} = Q$, i.e., $\theta^- = \theta$;
  end for
end for

4 Conclusion

Combining deep learning with reinforcement learning, deep reinforcement learning is a method of artificial intelligence that approaches the human way of thinking. Furthermore, combining the deep convolution neural network with Q-learning, the algorithm of deep Q-learning is investigated in the paper.

There need to be pointed out that the essence of reinforcement learning is a markov decision process, reinforcement learning gives only a return function, which is the decision in the risk of a particular condition to perform an action (which may be a good result, or probably worst). However, RL performance depends on the extraction technology of artificial features, and deep learning just makes up for this shortcoming. The deep reinforcement learning algorithm is mainly aimed at the optimization problems of discrete-time systems. However, because there are continuous action space and continuous state space in the practical application, it also limits the application range of deep reinforcement learning.

Acknowledgments. This work was supported by the National Natural Science Foundation of China under Grant 61673117.

References

1. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
2. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
3. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
4. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
5. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
6. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013)
7. Werbos, P.J.: Approximate dynamic programming for realtime control and neural modeling. In: Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches. Van Nostrand Reinhold, New York (1992)
8. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
9. Liu, D.R., Wang, D., Wang, F.Y., Li, H.L., Yang, X.: Neural-network-based Online HJB solution for optimal robust guaranteed cost control of continuous-time uncertain nonlinear systems. IEEE Trans. Cybern. **44**(12), 2834–2847 (2014)
10. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
11. Liu, D.R., Wang, D., Li, H.: Decentralized stabilization for a class of continuous-time nonlinear interconnected systems using online learning optimal control approach. IEEE Trans. Neural Netw. Learn. Syst. **25**(2), 418–428 (2014)
12. Wei, Q.L., Liu, D.R., Lin, H.: Value iteration adaptive dynamic programming for optimal control of discrete-time nonlinear systems. IEEE Trans. Cybern. **46**(3), 840–853 (2015)
13. Liu, D.R., Wei, Q.L.: Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. IEEE Trans. Neural Netw. Learn. Syst. **25**(3), 621–634 (2014)
14. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. J. Artif. Intell. Res. **4**, 237–285 (1996)
15. Arel, I., Rose, D.C., Karnowski, T.P.: Deep machine learning - a new Frontier in artifical intelligence research. IEEE Comput. Intell. Mag. **5**(4), 13–18 (2010)
16. Watkins, C.J.H., Dayan, P.: Technical note: Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992)
17. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. In: NIPS Deep Learning Workshop, arxiv preprint arXiv:1312.5602 (2013)