Reinforcement Learning: A Friendly Introduction

Chapter · August 2021

DOI: 10.1007/978-3-030-84337-3_11

5 authors, including:

Dema Daoun

2 PUBLICATIONS **9** CITATIONS

SEE PROFILE

Zulfikar Alom

University of Insubria

29 PUBLICATIONS **660** CITATIONS

SEE PROFILE

Zeyar Aung

Khalifa University

121 PUBLICATIONS **2,328** CITATIONS

SEE PROFILE

Mohammad Azim

The University of Sydney

50 PUBLICATIONS **556** CITATIONS

SEE PROFILE

Reinforcement Learning: A Friendly Introduction

Dema Daoun[1], Fabiha Ibnat[1], Zulfikar Alom[1], Zeyar Aung[2], and Mohammad Abdul Azim[2(✉)]

[1] Department of Computer Science, Asian University for Women, Chittogram, Bangladesh
{dema.daoun,fabiha.ibnat,zulfikar.alom}@auw.edu.bd
[2] Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, United Arab Emirates
zeyar.aung@ku.ac.ae,mohammad.azim@auw.edu.bd

Abstract. Reinforcement Learning (RL) is a branch of machine learning (ML) that is used to train artificial intelligence (AI) systems and find the optimal solution for problems. This tutorial paper aims to present an introductory overview of the RL. Furthermore, we discuss the most popular algorithms used in RL and the Markov decision process (MDP) usage in the RL environment. Moreover, RL applications and achievements that shine in the world of AI are covered.

Keywords: Artificial intelligence · Reinforcement learning · Markov decision process · Bellman optimality

1 Introduction

Decades ago, science fiction books and movies introduced to the public a concept known as artificial intelligence (AI), where people are replaced by robots or machines that perform human-like work more effectively and efficiently. Eventually, these imaginations are becoming true as almost wherever we go, we see devices are working in different fields and doing various tasks. This phenomenon made people afraid of losing their jobs, being replaced, and being dominated by machines. The fear raises the question, "Will robots exceed human intelligence and control everything on this planet?" To answer this question, we need to understand the meaning of AI and machine learning (ML) by focusing more on reinforcement learning (RL).

ML is an automatic way of analyzing data without human involvement through learning from the data, identifying patterns, and then taking action. There are four popular types of learning approaches consisting of (i) supervised learning, (ii) unsupervised learning, (iii) semi-supervised learning, and (iv) reinforcement learning [32].

The Khalifa University, UAE partially support this research.

I. Awan et al. (Eds.): Deep-BDB 2021, LNNS 309, pp. 134–146, 2022.
https://doi.org/10.1007/978-3-030-84337-3_11

Supervised learning is a widely studied branch of machine learning approaches. In this approach, training is accomplished by utilizing labeled data with a supervisor who determines the correct action that had better be. The algorithm learns from in cooperation with the sets of inputs and their corresponding right outputs. And then, the algorithm finds the errors and modifies the model according to the error. Supervised learning can be used in predicting future events from historical data [32].

Unsupervised learning is another branch of machine learning technique where the algorithms have to figure out the correct action by themselves depending only on the inputs. Unsupervised learning is based on exploring the data and finding intrinsic patterns or groups instead of learning from the labeled data. This branch is used widely in transactional data [32].

Semi-supervised learning is a branch of machine learning that uses both labeled and unlabeled data for training. It uses a small amount of labeled data and a more considerable amount of unlabeled data and is suitable when the cost of labeling is high [32].

RL is a distinct branch of machine learning that differs from the ones mentioned earlier. In RL, an agent interacts with the environment to maximize its rewards through taking actions and learning from its consequences. To make the agent take action, it uses two methods: (i) exploitation (where the agent depends on its experience (trails and errors) to take action), and (ii) exploration (where the agent chooses to take new action not related to its previous experience) [40]. RL is similar to self-learning, where no one or nothing can help in knowing what the right action is. Therefore, the agent should keep trying and gathering information to reach the right action and reward.

RL emphasizes the core issue of AI, i.e., the representation of information [27]. Nowadays, the RL technique is quite famous for organizations dealing with broad and complex problems frequently. This paper aims to discuss some of the RL aspects and highlight a subset of remarkable RL applications. In the following sections, RL's main components, algorithms, and achievements are discussed. Also, the applications of the Markov Decision Process (MDP) and Bellman optimality equation in RL are mentioned. The RL challenges and future direction are mentioned after highlighting some of the advantages and disadvantages of RL.

Section 2 provides a background of RL. Section 3 describes RL taxonomy. Section 4 provides an RL overview consisting of components of RL agents, MDP, Bellman optimality equation. Section 5 presents a simple example of RL. The paper finally discusses the advantages and disadvantages of the RL systems in Sect. 6 and the conclusions in Sect. 7.

2 Background

RL techniques were improving year by year, which caused expanding their applications in our real life. RL could do remarkable achievements in the last three decades, and it has been used in different fields and domains.

2.1 RL Achievements

Recently, RL accomplished remarkable achievements in the AI world, where most of these achievements are in the gaming sector. In 1993, IBM developed a program called "TD-Gammon". This program uses RL to play backgammon at extreme levels and challenge the best players in the world [39].

In 2006, RL was used in autonomous helicopter flight. Two approaches have been used to attain it. First, a pilot is required to help in finding the helicopter dynamics model and a reward function. Second, RL algorithms' usage is needed to find an optimized controller for the model [1].

In 2013, DeepMind introduced the first deep learning model to play Atari 2600 at professional levels using deep learning and RL combination. They trained the model using a Q-learning algorithm where the inputs are raw pixels and the outputs are value-functions [25].

In 2016, DeepMind again developed a program called "AlphaGo" using RL. This program plays a challenging classical game called "Go," and it could defeat the best Go player by 5-0 games [34].

In 2017, DeepMind introduced a program called "Alpha Zero" which can play chess, shogi, and Go. This program could reach extreme levels in 24 h without only knowing the rules of each game. It does not have specific hypermeters for each game, and it uses the same hypermeters for all [35].

2.2 Real-Life Applications

RL is widely used in many fields like robotics, games, finance, transportation, etc. It is used in solving problems and building systems that can improve themselves with experiences.

Game playing is the most popular RL application in the world of AI. Many researchers used RL in the environment of a very general class of games. One of the RL game applications is Samuel's checkers playing system that defeated America's best checkers players. This artificial player is trained by playing games against and by given the rule of the games consisting of an incomplete list of parameters and a sense of direction [31]. The temporal difference algorithm is used in backgammon games by Tesauro, where he used three layers of neural networks for training [39].

Robotics and Control machines are other applications of RL. RL is used to build a two-armed robot that learns from experience. In this robot, complex non-linear control tasks and dynamic programming make the application known as linear-quadratic-regulator design. Also, RL is used in robots that push boxes by using the Q-learning algorithm. The application is considered difficult because of its vast number of uncertain results.

In the food industry, machines fill up food containers with a variable number of non-identical products. These machines operate according to many set-points, and the chosen set-points depend on the change in the product's characteristics and task constraints [19].

RL is used in intelligent transportation systems. Multi-agent reinforcement learning reduces traffic congestion by using adaptive traffic signal control, where each agent controls traffic lights around a single traffic junction. The reinforcement learning approach faced some challenges in this application because of the instability in the single-agent case. Van der Pol and Oliehoek devised a scalable system for the problem using Deep Q-Learning algorithms and a coordination algorithm. The success is due to removing the simplified assumptions and inclusion of a new reward function [29].

Resources management in computer clusters RL is used to build systems that learn from their experience. This application helps in minimizing the job slowdown process by learning the allocation and schedule of computer resources. Here, the optimal policy is found using the REINFORCE algorithm with the baseline value [24]. RL applications are also used in the online web system auto-configuration [7], optimize chemical reactions [43], online personalized news recommendation [42], real-time advertising [18], and in natural language processing [33].

2.3 Current Challenges and/or Opportunities

RL faces some challenges, and finding solutions to these challenges will expand the areas where RL can be applied.

1. System Delay: RL natural systems face issues regarding delays in state sensation, actuators, or reward feedback. RL systems take time to improve agents' learning, especially in recommender systems, which might need about a week to determine the reward [23]. Some researchers applied a learning algorithm that itself can know the effects of delays [15]. Also, in some RL problems, like in Atari, re-distributing rewards throughout time is applied to find return-equivalent MDP that gives almost the same optimal policy as in the original MDP [4].
2. Non-Stationary: In many RL systems, it is challenging to observe the state of physical items, like wear and tear or the amount of building up, and observations about users' mental state. This issue has been solved in two ways: first, combining history and the agent's observations using the DQN approach. The second is to use recurrent networks that help track and recover hidden states [12].

However, RL aims to figure out how to use multi-task learning to allow the agent to do multiple tasks instead of specialization [8]. Another challenge is that the model-free RL needs many trials to learn, and that is not acceptable in the real world because it causes danger and sometimes threatens people's lives.

3 RL Taxonomy

Reinforcement aims to find the best solution for a problem by using algorithms. Numerous algorithms have been applied to reach the goal of optimization. These algorithms can be divided into three distinct groups: (i) value-based, (ii) policy-based, and (iii) model-based.

3.1 Value-Based Methods

In the value-based methods, the agent aims to maximize the value function to get a long-run return under a specific policy [14]. The value-based group includes Q-Learning and SARSA (State-Action-Reward-State-Action). Q-Learning is an off-policy and model-free algorithm based on an action derived from another policy. This method follows some steps to update the value consisting of (i) initialize the Q-table, (ii) choose the action, (iii) perform the action and measure the reward and the new state, (iv) update the Q-table, and (v) repeat this process until it reaches the terminal state.

SARSA is an on-policy algorithm based on the agent's current action and policy. It follows one approach, which is the same as the behavior and target policy. The difference between Q-Learning and SARSA is that SARSA rewards are selected based on the same policy and original action counter to Q-learning. In general, the on-policy algorithms are better in performing than off-policy, but it is worse in finding the target policy [37].

3.2 Policy-Based Methods

In the policy-based methods, the agents aim to find the optimal policy that returns maximum reward without value function [14]. In policy-based methods, the neural networks are used to realize the optimal policy in two ways: gradient-based and gradient-free [38]. An example of policy-based is the REINFORCE Algorithm which is a policy gradient method. To understand the REINFORCE Algorithm, it is required to understand the meaning of the policy gradient methods, i.e., an on-policy depends on optimizing parameterized policies considering the long-term rewards. REINFORCE algorithm uses Monte Carlo methods (use to solve a large number of computer problems) [11] to estimate the returns because it is based on the entire trajectory.

Another example of a policy-based algorithm is the actor-critic method. This method is a temporal difference method that implements its policy indecently from the value function. It is called actor-critic as it selects actions and gives critiques to the actor responsible for the action [20].

3.3 Model-Based Methods

The agents aim to perform in the created environment in the model-based methods by relying on a model better. Some popular algorithms like Q-learning, SAC, and DDPG are model-free and are characterized by stability and optimality. However, model-based algorithms are more effective, and it uses artificial neural networks to predict the environment transactions and reward function [26]. For instance, Dyna-Q is an algorithm used to build a policy using both actual and simulated experiences. These experiences improve both the model and the approach through model learning, direct reinforcement learning, and planning.

4 RL: An Overview

To understand RL and know how it can be used, it is essential to get familiar with few concepts before, like RL components, MDP, and Bellman optimality equation.

4.1 Components of RL Agent

The RL system has four main components: (i) policy, (ii) reward function, (iii) value function, and (iv) the model of the environment.

Policy: Here, the policy (π) is the core goal of RL. The policy is the agent's behavior function that determines the way of behaving under certain circumstances [37]. In other words, the policy is the strategy used to achieve the agent's goal of conducting some tasks. The policy may include a straightforward function or a very complex one, and it can be stochastic. Mathematically, the policy is defined using the MDP, a tuple with four sets (s, a, P, R). MDP suggests actions to each possible state. Here, set s comprises the agent's internal status. Set a includes the agent's actions. The set P is a matrix of the transition probabilities from one state to another that modifies the agent's status. The set R is the agents' rewards.

Reward Function: The reward R of action is evaluated utilizing the reward function. Here, the process uses the agents' status as inputs and returns rewards as outputs of real numbers [10]. Note that R is the feedback from the environment and the main factor for updating the policy. It accurately describes the agent's state and defines the goal in the RL problem. The reward signal distinguishes between bad and good events for the agent to reach the maximum utility level for total received rewards in the long run. The reward depends on the action and the environment of the agent. Thus, to change the reward, there are two possible ways: (i) direct through action and (ii) indirect through the environment. When the action is changed, that will directly affect the reward. It will also affect the environment, which, in turn, affects the reward "indirectly" [37]. In general, the agent's reward differs when different actions are applied.

Value Function: The value functions $V(s)$ determine what is good in the long run by measuring how much good to be in a specific situation. The value function predicts the total amount of reward that an agent can get in the potential future [37]. The value function aims to find the optimal policy that maximizes the expected rewards by evaluating the states and selecting the best action. Three methods may achieve the optimality: (i) value iteration, (ii) policy evaluation, and (iii) policy iteration [16]. Moreover, the value function has four types of operations:

- The on-policy value function is where the agent starts in a state and acts according to the policy. After that, the function will give expected returns.
- The on-policy action-value function is where the agent starts in a state and suddenly an action, which is not from the policy, is taken and applied till the end, then the function will give expected return.
- The optimal value function is where the agent starts in a state and acts according to the optimal policy. After that, the function will give an expected return.
- The optimal value-action function is where the agent starts in a state. Suddenly, an action, which is not from the policy, is taken and acting according to the optimal policy till the end, then the function will give expected return [2].

Model of Environment: The model of the environment is the place where the agents operate. The model of the environment allows deducting the environmental behavior after looking at the state and action. It is used to build a view of future situations to decide which course of action is better to take [37]. The environment gives two signals to the agent, one on its next state and another on reward [41].

4.2 Markov Decision Process

Markov Decision Process (MDP) is a sequence of decisions that describes the environment for RL. It has four components: set of states (s), set of actions (a), transition probabilities (P), and real-valued reward function on states $R(s)$. The main property of the MDP is transition probabilities, where the probability of the next state depends only on the current state and current action without considering the previous states and actions. The probability of the next state is formulated in the following equation:

$$P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_0 \dots s_t, a_0 \dots a_t). \tag{1}$$

According to this equation, the probability of the following state only depends on the current state and current action, regardless of the previous states and actions:

$$s_0 \dots s_t, a_0 \dots a_t. \tag{2}$$

4.3 Bellman Optimality Equation

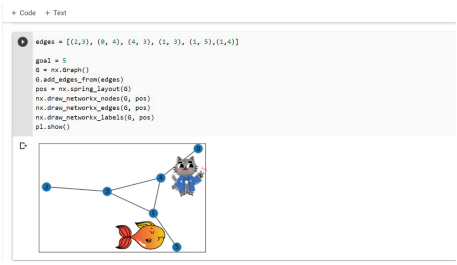
Richard Bellman found that using dynamic programming in solving MDP problems reduces the computational burdens [5]. MDP problem is, in fact, an optimality problem because the agent, as discussed before, aims to maximize its rewards [6]. It can be mathematically expressed as follows:

$$V^*(s) = \max_a q^*(s, a). \tag{3}$$

In this equation, the first part of the equation means maximizing the rewards. And the second part shows how the value function is related to itself [36]. This

equation is nonlinear and has no closed-form solutions. However, some methods introduced iterative solutions like Q-learning [30]. Note that this function raises a concept of q that stands for quality. Quality, in this case, represents how valuable a given action is in gaining some future reward.

(a) RL big picture



(b) Agents in the environments (Python code and output graphics)

```
[18] MATRIX_SIZE = 6
     M = np.matrix(np.ones(shape =(MATRIX_SIZE, MATRIX_SIZE)))
     M *= -1

     for point in edges:
         print(point)
         if point[1] == goal:
             M[point] = 100
         else:
             M[point] = 0

         if point[0] == goal:
             M[point[::-1]] = 100
         else:
             M[point[::-1]]= 0
             # reverse of point

     M[goal, goal]= 100
     print(M)
```

(c) Calculating rewards (Python code)

```
(2, 3)
(0, 4)
(4, 3)
(1, 3)
(1, 5)
(1, 4)
[[ -1.  -1.  -1.  -1.   0.  -1.]
 [ -1.  -1.  -1.   0.   0. 100.]
 [ -1.  -1.  -1.   0.  -1.  -1.]
 [ -1.   0.   0.  -1.   0.  -1.]
 [  0.   0.  -1.   0.  -1.  -1.]
 [ -1.   0.  -1.  -1.  -1. 100.]]
```

(d) Reward matrix

Fig. 1. RL example (cat catching fish).

5 Example of RL

Here, an example of reinforcement learning is presented that further clarifies how reinforcement learning precisely works. In this example, RL is used to create a cat and fish game where the cat has to find the fish [13]. The cat will keep looking for the fish in different possible paths. Upon successfully finding the fish, it will get 100 points. Hence, the cat will constantly learn to find fish and become a skilled fish hunting cat over time.

Figure 1 depicts various RL aspects in this example, where, Fig. 1a provides a big functional picture of RL, Fig. 1b presents the agents in the environment, Fig. 1c shows the reward policy implementation in the RL systems. Finally, Fig. 1d depicts the derived reward matrix in the system.

To make the cat learn from its experience, the first thing is to create an environment where the fish can be found without guiding the cat to the route of finding fish.

Here, five nodes are assigned to connect these nodes in different ways and create paths. The agent, i.e., the cat, will look for the courses to find out the fish. Also, the connection of the trails is defined in edges.

There are several paths in this environment in the figure, and the cat is located at node 0. As per the RL algorithm, the reward matrix determines how much the cat will get (as a reward) if it reaches the target fish and how much it will get if she does not. Now, there are many −1's in the reward matrix which indicates there is no direct connection between these nodes. Moreover, there are some (0) that notifies there is a direct connection between these nodes. For example, there is only one direct connection with node 0 which is node 4. That can be expressed as follows [−1. −1. −1. −1. 0. −1] where (0, 0) = −1, (0, 1) = −1, (0, 2) = −1, (0, 3) = −1, (0, 4) = 0, (0, 5) = −1.

After that, a function is created to determine the cat's next step from its current state. This function also finds the optimal policy that gives maximum rewards. Thus, the policy will be constantly updated, and these values will form the Q-matrix. In other words, the more times the cat goes through paths, the more information about the environment in the Q-matrix.

In the last stage, the agent will learn over time when the cat will search for the fish. Each time, the cat will choose a different path starting from node 0. The Q-matrix is constantly updated by the information received from a cat's movements. Thus the cat will be more efficient over time.

6 Recent Developments

This section presents some of the recent research done in RL and could make significant improvements.

6.1 Graph Convolutional RL

Graph convolutional is applied in RL to optimize the policy. This technique provides a multi-agents model in graphs because agents are not stables as they keep moving. Thus, agents face difficulties in learning the abstracts representations of interaction. Compared to the other cooperative methods, the graph convolutional RL technique improves agents' convolutional through regularizing the temporal relation [17].

6.2 Measuring the Reliability of RL Algorithms

Chan and other researchers realized the lack of reliability in RL and found a set of metrics to measure the reliability of RL algorithms quantitatively from different aspects. To analyze the performance of RL algorithms, evolution must be done during training and after learning. Also, these metrics measure reliability from various elements, such as reproducibility and stability [9].

6.3 Behavior Suite for RL

Researchers introduced a collection of designed experiments that look into the core of an agent's capabilities, called "Behavior Suite." This set of experiments capture the main issues in designing learning algorithms that can be developed. Also, it studies an agent's behavior through its performance [28].

6.4 The Ingredients of Real World Robotics RL

A group of researchers discussed the elements of a robotic learning system that does not require human intervention to ensure the process of learning. This automatic learning system is independently improved using real-world collected data, and learning here is feasible only using onboard perception without manually designed reset reward function [44].

6.5 Network Randomisation

A Simple Technique for Generalisation in Deep Reinforcement Learning improves deep RL agents' generalization ability by applying a randomized neural network that perturbs input observations. This approach allows agents to adapt to new domains because the agent learns robust features from the different environments [21].

7 RL: Pros and Cons

Like any techniques, RL has both advantages and disadvantages, and knowing them helps determine the best methods that can be applied to solve a specific problem.

7.1 Advantages

RL made a great revolution in the world of machine learning. Thus, it has many advantages compared to other machine learning types or compared to other AI methods. Here are some of the advantages:

- It gives successful rewards regardless of the environment's size because it depends on observing the environment to take the action that gives reward [22].
- It sustains change for a long time.
- In RL, errors have less chance to happen again because the agent learns from its experience and corrects its errors by training.
- It is better than humans in completing tasks, even in the games; RL applications could defeat the best players in the world.

7.2 Disadvantages

RL has some disadvantages that make it not suitable in some specific situations even with all these advantages.

- It takes a long time to find the optimal solution for big real-world problems [3].
- It is difficult to directly train some systems that need fixed logos of the system's behavior to be learned.
- Most real-world systems are fragile and expensive.
- The physical system destroys itself and its environment [12].
- If lots of reinforcement is applied, it leads to an overload of loads and diminishes the results.
- It creates a serious problem when it is used in some real-world applications like self-drive cars.

8 Conclusion

RL is a fascinating and vital topic regarding its wide usage, applications, and achievements. RL depends on self-experience and training to find the optimal policy and get maximum rewards. This tutorial paper described RL's main components and its popular algorithms and explained the MDP used widely in RL. This paper summarized most RL applications and achievements in the real world and included its advantages, disadvantages, current challenges, and opportunities.

References

1. Abbeel, P., Coates, A., Quigley, M., Ng, A.Y.: An application of reinforcement learning to aerobatic helicopter flight. In: Advances in Neural Information Processing Systems, pp. 1–8 (2007)
2. Achiam, J.: Introduction to RL (2018). BOpen AI. https://spinningup.openai.com/en/latest/spinningup/rl_intro.html
3. Arabnejad, H., Pahl, C., Jamshidi, P., Estrada, G.: A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling. In: Proceedings of the 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 64–73 (2017)
4. Arjona-Medina, J.A., Gillhofer, M., Widrich, M., Unterthiner, T., Brandstetter, J., Hochreiter, S.: RUDDER: return decomposition for delayed rewards. arXiv preprint arXiv:1806.07857 (2018)
5. Bellman, R.: On the theory of dynamic programming. Proc. Natl. Acad. Sci. U.S.A. **38**(8), 716 (1952)
6. Bellman, R.E., Dreyfus, S.E.: Applied Dynamic Programming. Princeton University Press, Princeton (2015)
7. Bu, X., Rao, J., Xu, C.Z.: A reinforcement learning approach to online web systems auto-configuration. In: Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems, pp. 2–11 (2009)
8. Caruana, R.: Multitask learning. Mach. Learn. **28**(1), 41–75 (1997)

9. Chan, S.C., Fishman, S., Canny, J., Korattikara, A., Guadarrama, S.: Measuring the reliability of reinforcement learning algorithms. arXiv preprint arXiv:1912.05663 (2019)
10. De Luca, G.: What is a policy in reinforcement learning? (2020). Baeldung. https://www.baeldung.com/cs/ml-policy-reinforcement-learning
11. Dimov, I.T., Tonev, O.I.: Monte Carlo algorithms: performance analysis for some computer architectures. J. Comput. Appl. Math. **48**(3), 253–277 (1993)
12. Dulac-Arnold, G., Mankowitz, D., Hester, T.: Challenges of real-world reinforcement learning. arXiv preprint arXiv:1904.12901 (2019)
13. Fazly, R.: Data science book (2020). GitHub. https://github.com/FazlyRabbiBD/Data-Science-Book/blob/master/8-ReinforcementLearning.ipynb
14. Guru99: Reinforcement learning: what is, algorithms, applications, example (2020). Guru99. https://www.guru99.com/reinforcement-learning-tutorial.html
15. Hester, T., Stone, P.: TEXPLORE: real-time sample-efficient reinforcement learning for robots. Mach. Learn. **90**(3), 385–429 (2013)
16. Hui, J.: RL – value learning (2018). Medium. https://jonathan-hui.medium.com/rl-value-learning-24f52b49c36d
17. Jiang, J., Dun, C., Huang, T., Lu, Z.: Graph convolutional reinforcement learning. arXiv preprint arXiv:1810.09202 (2018)
18. Jin, J., Song, C., Li, H., Gai, K., Wang, J., Zhang, W.: Real-time bidding with multi-agent reinforcement learning in display advertising. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 2193–2201 (2018)
19. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. J. Artif. Intell. Res. **4**, 237–285 (1996)
20. Konda, V.R., Tsitsiklis, J.N.: On actor-critic algorithms. SIAM J. Control Optim. **42**(4), 1143–1166 (2003)
21. Lee, K., Lee, K., Shin, J., Lee, H.: Network randomization: a simple technique for generalization in deep reinforcement learning. arXiv preprint arXiv:1910.05396 (2019)
22. Manju, S., Punithavalli, M.: An analysis of Q-learning algorithms with strategies of reward function. Int. J. Comput. Sci. Eng. **3**(2), 814–820 (2011)
23. Mann, T.A., et al.: Learning from delayed outcomes via proxies with applications to recommender systems. In: International Conference on Machine Learning, pp. 4324–4332. PMLR (2019)
24. Mao, H., Alizadeh, M., Menache, I., Kandula, S.: Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM Workshop on Hot Topics in Networks, pp. 50–56 (2016)
25. Mnih, V., et al.: Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
26. Moazami, S., Doerschuk, P.: Modeling survival in model-based reinforcement learning. arXiv preprint arXiv:2004.08648 (2020)
27. Mondal, A.K., Jamali, N.: A survey of reinforcement learning techniques: strategies, recent development, and future directions. arXiv preprint arXiv:2001.06921 (2020)
28. Osband, I., et al.: Behaviour suite for reinforcement learning. arXiv preprint arXiv:1908.03568 (2019)
29. Van der Pol, E., Oliehoek, F.A.: Coordinated deep reinforcement learners for traffic light control. In: Proceedings of the NIPS 2016 Workshop on Learning, Inference and Control of Multi-Agent Systems, pp. 1–8 (2016)

30. Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems. Technical report, Cambridge University Engineering Department, UK (1994)
31. Samuel, A.L.: Some studies in machine learning using the game of checkers. IBM J. Res. Dev. **3**(3), 210–229 (1959)
32. SAS: Machine learning: what it is and why it matters (2020), SAS. https://www.sas.com/en_us/insights/analytics/machine-learning.html
33. Sharma, A.R., Kaushik, P.: Literature survey of statistical, deep and reinforcement learning in natural language processing. In: Proceedings of the 2017 IEEE International Conference on Computing, Communication and Automation, pp. 350–354 (2017)
34. Silver, D., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
35. Silver, D., et al.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815 (2017)
36. Singh, A.: Reinforcement learning: Bellman equation and optimality (Part 2). Towards Data Sci. (2019). https://towardsdatascience.com/reinforcement-learning-markov-decision-process-part-2-96837c936ec3
37. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
38. Taylor, G., Burmeister, R., Xu, Z., Singh, B., Patel, A., Goldstein, T.: Training neural networks without gradients: a scalable ADMM approach. In: Proceedings of the 33rd International Conference on Machine Learning, pp. 2722–2731 (2016)
39. Tesauro, G.: TD-Gammon, a self-teaching backgammon program, achieves master-level play. Neural Comput. **6**(2), 215–219 (1994)
40. Woergoetter, F., Porr, B.: Reinforcement learning. ScholarPedia **3**, 1448 (2008)
41. Zhang, J.: Reinforcement learning - model based planning methods. Towards Data Science (2020). https://towardsdatascience.com/reinforcement-learning-model-based-planning-methods-5e99cae0abb8
42. Zheng, G., et al.: DRN: a deep reinforcement learning framework for news recommendation. In: Proceedings of the 2018 World Wide Web Conference, pp. 167–176 (2018)
43. Zhou, Z., Li, X., Zare, R.N.: Optimizing chemical reactions with deep reinforcement learning. ACS Cent. Sci. **3**(12), 1337–1344 (2017)
44. Zhu, H.: The ingredients of real-world robotic reinforcement learning. arXiv preprint arXiv:2004.12570 (2020)