RESEARCH ARTICLE

Reinforcement Learning Techniques to Game Environments

Ruben Chevez* and Kratika Naskulwar

Abstract

Games have been one of the methods of choice for machine learning researchers to test their new algorithms. It is characterized as being a controlled simulated environment with discrete or stochastic observations. Games are a set of simple to complex tasks where different iterations of the algorithms can be benchmarked against each other to compare their learning abilities and flexibility towards unexpected observations from the environment. In the following sections, we will discuss the state-of-the-art algorithms that were able to master many ATARI and other platform games during their development.

Keywords: machine learning; atari; artificial intelligence; reinforcement learning; simulation

Background

A Markov Decision Process (MDP) is often used to describe a reinforcement learning problem when the environment is fully observable. An MDP is made up of a tuple **(S, A, R, T)**, where **S** is the state or observation. **A** is the set of actions that can be chosen, **R(s, a)** is the reward function that returns the reward for an action taken in the given state, s. **T(s' — s, a)** is the probability of transition to a state, given an action taken in state. [1, 2]

Exploration is the ability for an agent to transition to new or unknown states of the environment without relying on previous experience. Exploitation is the ability for the agent to base its decision completely on experiences. [3] A balance between the two is necessary to start exploring the environment and after some time, start relying on experience to choose the most rewarding actions. This is often called the exploration vs exploitation trade-off. [4]

There are two main approaches to solve reinforcement learning problems: Policy-Based, and Value-Based agents. [5]

Methods

Christopher Watkins said that "Learning to act in ways that are rewarded is a sign of intelligence." [6] Biological systems have inspired many revolutionary methods of cognitive science in the search of automatic problem-solving systems. The following reinforcement learning techniques' categories are inspired by the OpenAI Docs. [7]

Model-Free RL

Q-Learning

Q-Learning is an alternative to model-based machine learning algorithms, because it does not generate a model to represent the environment. Representing the environment can be computationally costly depending on the complexity of the environment. Instead, the

*Correspondence: rubencg@mun.ca, ksnaskulwar@mun.ca

Computer Science, Memorial University, 230 Elizabeth Ave, St. John's, Newfoundland and Labrador, Canada. Tel.: 17097694877

agent computes Q-values, which mean the "quality"of an action taken on a given state, **s**. A policy can be derived by choosing the actions with the highest Q-values for the current state. [6]

The Q-values initialized randomly and then updated based on rewards gained, executing different actions using the Bellman Equation. Q-Learning uses techniques to let the agent explore, such as the Exploration vs Exploitation Trade-off and the Discount Factor, to give significance to the most recent actions. [8–10]

Deep-Q Learning

Deep-Q Learning was introduced and patented by the team at DeepMind as an improvement to the previous Q-Learning algorithm. Using Convolutionary Neural Networks as the environment's model aproximator, the algorithm is able to learn the best actions to take in complex environment more efficiently than the Q-tables in Q-learning. [11, 12]

Dueling DQN

The Dueling Network is a model-free neural system design. The system comprises two streams that shows the state value and advantage functions, sharing a single learning module. The two streams are joined to yield a single output as a Q function. The output consists of a set of Q values per action. [13]

The Dueling network separately estimates state value (scalar V) and advantages (A) for each action. The state and advantage values are combined, thereby yielding the output as Q values for each action. [14]

Double DQN

The Double DQN algorithm was introduced to mitigate the overestimation of Q-values in the basic DQN algorithm to achieve better performance on several games. Overestimation can be an issue in training performance and can sometimes cause less than optimal policies.

In the standard DQN algorithm, the target network selects the action and simultaneously evaluates the action quality which is likely to cause the selection of overestimated values. [15]

Prioritized Experience Replay

Prioritized Experience Replay is a strategy that allows past experiences to be recollected and utilized based on their significance. Samples are uniformly transitioned from replay memory and in the same order as they are experienced in previous methods. Prioritizing experience makes the learning more efficient when used with DQN.

The issue of over-fitting caused when experiences are prioritized with transition error is overcome by prioritizing experiences with stochastic prioritization. Bias introduced towards high priority samples causes more chances of high priority samples to be chosen. Bias is corrected by using Importance sampling weights (IS). This algorithm performs better on 41 games out of 49 than DQN, accomplishing a new state-of-the-art. [16]

Policy Gradients

A3C

A3C (Asynchronous advantage actor-critic) is a simple deep reinforcement learning algorithm. The Asynchronous gradient descent is used for the purpose of optimization of deep neural network controllers. A3C controls the policy and value estimation function. The value function and the policy are updated whenever a final state has occurred. Of the different optimization algorithms, RMSProp, was found to be considerably optimal than the other methods. A3C performance exceeds the state-of-the-art algorithms on the Atari domain methods. It uses a single multi-core CPU other than GPU for training half of the time. [17]

Ryan Partridge explains the three As in A3C. The first A, stands for actor. It is a set of actions and the state value of the current agent is called the critic. The second A, is asynchronous. A3C deals with multiple

independent agents with their individual weights and lets the multiple agents interact in parallel. The last A, stands for advantage. It contains the value and policy loss. The advantage equation is used to calculate the policy loss that improves the behavior of agent. [18]

TRPO

Trust Region Policy Optimization (TRPO) is an iterative policy gradient algorithm. This algorithm is viable for optimizing huge nonlinear policies. TRPO addresses the following issues of Policy Gradient Methods; training may be ruined in case of a large policy, convergence issue may arise due to the insensitive learning rate, and mapping changes between parameter space, and policy is hard and will weaken the sample efficiency. TRPO has two variants of sampling schemes, the first is the single path, which is based on individual trajectories and is used for policy gradient estimation. The second is the Vine method which is mostly used in policy iteration methods. This method constructs a set to perform multiple actions on each state from the set. The Vine method performs better than single path and efficiently estimates advantage values. [19, 20]

PPO

The Proximal Policy Optimization (PPO) algorithm is a policy gradient approach for reinforcement learning. Unlike standard policy gradient methods, PPO allows multiple gradient updates. PPO has the confidence and reliability merits of trust region policy optimization (TRPO), yet, is a lot easier to implement. Sampling the policy data and performing various epochs of optimization on the data sampled is altered to optimize the policy. To evaluate PPO, the experiments looked at the performance of different versions of the surrogate objective, and as a result, clipped probability ratios perform the best. [21]

Nicolas Heess et al. developed a dispersed version of Proximal Policy Optimization called DPPO to achieve better execution in rich, simulated environments. Experiments show that it is effectively salable and allow strong policy optimization with less parameter tuning. [22]

ACKTR

The Actor Critic using the Kronecker-Factored Trust Region (ACKTR) is an actor-critic method that uses Kronecker-factored approximation curvature (K-FAC). Kronecker-factored approximation is used to optimize both actor and critic. Gradient's co-variance matrix can be inverted effectively in the ACKTR algorithm. This algorithm optimizes the value function by using Gauss-Newton approximation. It generously increases the agent's performance and efficiency in Atari environments. ACKTR was evaluated on both discrete control tasks and continuous control tasks. This algorithm shows an improvement in sample efficiency by 2-3 times when compared with the first-order gradient method(A2C). [23, 24]

Deterministic Policy Gradients

DPG

The Deterministic policy gradient (DPG) is an off-policy actor-critic algorithm designed to be able to learn deterministic target policies from a policy of exploratory behavior. The policy gradient combines both state and action in stochastic policy gradients, while in deterministic policy gradients it combines with the state. The DPG is first used to derive the off-policy algorithm by using a differentiable function approximator. Then the action-value function is estimated. After that, the parameters of the policy are updated based on the approximation of action-value gradient. Results of DPG show significant improvement over stochastic policy gradients, especially in high dimensional space. In some cases when the functionality to introduce noise

in a controller is unavailable, DPG will still be applicable unlike stochastic policy gradients. [25]

DDPG

DDPG (Deep Deterministic policy gradient) is an off-policy, actor-critic, model-free algorithm that can be operated over continuous action spaces. This algorithm is based on DPG and addresses the issue of dealing with high dimensional and continuous action spaces. An important point of this algorithm is that it is simple and requires a simple actor-critic structure and a learning algorithm with quite a few moving parts. This makes it easy to execute and scale to numerous complex problems. Batch normalization is a deep learning technique which was used to address the issue of learning the model ineffectively when low dimensional feature vector observations are used for learning. [26] The experiment's problems were solved within 2.5 million steps, which is 20 steps less than those required for DQN. However, one constraint of DDPG is that it requires a larger set of training data to learn the algorithm. [27]

Combining Policy-Learning and Q-Learning

PGQL

PGQL stands for "Policy Gradient and Q-Learning" and is a data-efficient and stable algorithm that aims to combine the best of both methods. This connection allows estimating Q-values from the action preferences of the policy to which the Q-Learning updates were applied. The Bellman residual is said to be small when the penalty is small at a fixed point of a regularized policy gradient algorithm. PGQL introduces a hybrid between policy gradients and Q-Learning by adding an auxiliary update where it explicitly attempts to reduce the Bellman residual as estimated by the policy. The authors observed data efficiency and stability compared to both A3C and Q-Learning during testing in the OpenAI Gym environments. [28]

Evolutionary Algorithms

Evolutionary Strategies are algorithms inspired by natural evolution in which at every generation, a population is mutated and its fitness is evaluated. The highest scoring candidates are selected and recombined to form the next generation. This procedure is iterated until the objective is achieved or the maximum number of iterations is reached.

Evolutionary Strategies are characterized for being easy to parallelize, are more resilient to long delayed rewards, and avoiding gradients calculations since the only information needed is the scalar episode return and the random seed used to generate the perturbations in the mutation. Since back-propagation is not required, the computational power needed and memory is reduced by two-thirds. It is considered by the authors to have an advantage over policy gradient methods when the environment contains long episodes with many time-steps. Policy gradient methods use discounting rewards to overcome this problem. [29]

Intrinsic Motivation

Curiosity Driven Exploration

Curiosity-driven exploration is a method created to solve the sparse reward problem in reinforcement learning. Curiosity serves as an intrinsic reward that lets the agent be motivated to explore and learn new patterns. Deepak Pathak et al. states that "We formulate curiosity as the error in an agent's ability to predict the consequence of its own actions in a visual feature space learned by a self-supervised inverse dynamics model". [30, 31] This method significantly outperformed the state-of-the-art baseline algorithms such as A3C and PPO, in the absence or sparse reward environments. [32, 33]

Model-Based RL — Model is Learned

I2A

The team DeepMind developed the Imagination-Augmented Agents (I2As) and the authors claim that agents that have a model of the world usually perform better than agents who do not. The model is not always available so the model is built, even though it is imperfect because is learned, but ultimately, it is used to plan the next action. The final policy can be achieved by two different paths in the architecture of the algorithm. The first path is through a model-free approach and the second a model-based approach which tries to predict or imagine the next state. Both results are taken by the final component of the I2A which is the policy module and outputs the imagination-augmented policy. Sébastien Racanière et al. stated that "I2As can thus be thought of as augmenting model-free agents by providing additional information from model-based planning, and as having strictly more expensive power than the underlying model-free agent." [34]

Model-Based RL — Model is Given

AlphaGo

AlphaGo is the first computer program to defeat a professional human Go player and the first program to defeat a Go world champion. AlphaGo competed against the European Champion, Mr. Fan Hui, on October 2015, and Mr. Lee Sedol, who is the winner of 18 world titles March 2016, and won both matches. On January 2017, AlphaGo's upgraded version was disguised as the online player "Master", achieving 60 consecutive wins against professional players. On December 2017, DeepMind introduced AlphaZero, the improved version of its predecessor. [35–39]

AlphaZero

AlphaZero is an improved version of the previous AlphaGo. This algorithm is characterized for its ability to master not only Go but Chess and Shogi, all of which are complex in nature & require strategic reasoning. AlphaZero defeated all the state-of-the-art algorithms meant for each specific game by learning to play from scratch. It taught itself how to play by using Convolutional Networks, Monte Carlo Tree search for next state simulations, and was tuned using Bayesian optimization to find the best hyper-parameters. [40, 41]

Conclusion

The purpose of this review was to view the trends in reinforcement learning techniques targeted to solve game environments. Within the past few years, we have seen how many of these techniques have evolved, improved over its predecessors, and achieved higher performance in its specific task to solve. These techniques can be mainly categorized into two categories, model based and model free. Another categorization could be policy based, value based, and policy-value based. Among the most popular algorithms are PPO, A3C, and TRPO considered as the state-of-the-art and Deep-Q Learning as a route for beginners starting to learn the concepts of reinforcement learning. But as the No Free Lunch theorem states, there will be no algorithm that will master all of the rest in the different conditions or types of problems to solve. [42] Each technique has its own domain, their own downsides, and advantages.

References

1. Richard B. A Markovian Decision Process. Indiana Univ Math J. 1957;6:679–684.
2. Szepesvari C. Algorithms for Reinforcement Learning. In: Publishers MC, editor. Algorithms for Reinforcement Learning; 2009. p. 1–98. Draft of the lecture published in the Synthesis Lectures on Artificial Intelligence and Machine Learning. Available from: https://sites.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf.
3. Wang H, Zariphopoulou T, Zhou XY. Exploration versus exploitation in reinforcement learning: a stochastic control approach; 2019. Available from: https://arxiv.org/pdf/1812.01552.pdf.
4. Sutton RS, Barto AG. Reinforcement Learning: An Introduction. In: Reinforcement Learning: An Introduction. Cambridge, Massachusetts London, England: The MIT Press; 2017. p. 19.

5. Cheung D. A brief introduction to reinforcement learning; 2018. Available from: https://medium.freecodecamp.org/a-brief-introduction-to-reinforcement-learning-7799af5840db.
6. Watkins C. Learning from Delayed Rewards; 1989. Q-Learning. Available from: http://www.academia.edu/download/50360235/Learning_from_delayed_rewards_20161116-28282-v2pwvq.pdf.
7. OpenAI. Key Papers in Deep RL; 2018. Available from: https://spinningup.openai.com/en/latest/spinningup/keypapers.html.
8. BELLMAN R. THE THEORY OF DYNAMIC PROGRAMMING; 1954. Available from: https://projecteuclid.org/download/pdf_1/euclid.bams/1183519147.
9. Greaves J. Understanding RL: The Bellman Equations; 2019. Step-by-step derivation, explanation, and demystification of the most important equations in reinforcement learning. Available from: https://joshgreaves.com/reinforcement-learning/understanding-rl-the-bellman-equations/.
10. Martin M, Universitat politècnica de Catalunya. Reinforcement Learning Searching for optimal policies I: Bellman equations and optimal policies; 2011. Available from: http://www.cs.upc.edu/~mmartin/Ag4-4x.pdf.
11. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al.. Playing Atari with Deep Reinforcement Learning; 2013. Available from: https://deepmind.com/research/publications/playing-atari-deep-reinforcement-learning/.
12. Mnih V, Kavukcuoglu K, Silver D. Human-level control through deep reinforcement learning. Nature. 2015;5018:1–13.
13. Wang Z, Schaul T, Hessel M, van Hasselt H, Lanctot M, de Freitas N. Dueling Network Architectures for Deep Reinforcement Learning; 2016.
14. Dueling DQN to play Cartpole;. Available from: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788621755/10/ch10lvl1sec57/dueling-dqn-to-play-cartpole.
15. van Hasselt H, Guez A, Silver D. Deep Reinforcement Learning with Double Q-learning; 2015.
16. Schaul T, Quan J, Antonoglou I, Silver D. PRIORITIZED EXPERIENCE REPLAY; 2016.
17. Mnih V, Badia AP, Mirza M, Graves A, Harley T, Lillicrap TP, et al.. Asynchronous Methods for Deep Reinforcement Learning; 2016.
18. Asynchronous Advantage Actor Critic (A3C);. Available from: https://github.com/Achronus/Machine-Learning-101/wiki/Asynchronous-Advantage-Actor-Critic-(A3C)#actor-critic.
19. RL— Trust Region Policy Optimization (TRPO) Explained;. Available from: https://medium.com/@jonathan_hui/rl-trust-region-policy-optimization-trpo-explained-a6ee04eeeee9.
20. Schulman J, Levine S, Moritz P, Jordan M, Abbeel P, University of California, Berkeley, Department of Electrical Engineering and Computer Sciences. Trust Region Policy Optimization; 2017.
21. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal Policy Optimization Algorithms; 2017.
22. Heess N, TB D, Sriram S, Lemmon J, Merel J, Wayne G, et al.. Emergence of Locomotion Behaviours in Rich Environments; 2017.
23. Wu Y, Mansimov E, Liao S, Grosse R, Ba J. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation; 2017.
24. Martens J, Grosse R. Optimizing Neural Networks with Kronecker-factored Approximate Curvature;.
25. Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M. Deterministic Policy Gradient Algorithms; 2014.
26. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift.;.
27. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, et al.. CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING; 2016.
28. O'Donoghue B, Munos R, Kavukcuoglu K, Mnih V. COMBINING POLICY GRADIENT AND Q-LEARNING; 2017.
29. Salimans T, Ho J, Chen X, Sidor S, Sutskever I. Evolution Strategies as a Scalable Alternative to Reinforcement Learning; 2017.
30. Pathak D, Agrawal P, Efros AA, Darrell T. Curiosity-driven Exploration by Self-supervised Prediction. In: ICML; 2017. .
31. Blajer JAGW, Mariusz K. The Inverse Simulation Study of Aircraft Flight Path Reconstruction. Transport. 2002;XVII(3):103–107.
32. Hill A, Raffin A, Ernestus M, Traore R, Dhariwal P, Hesse C, et al.. Stable Baselines; 2018.
33. Dhariwal P, Hesse C, Klimov O, Nichol A, Plappert M, Radford A, et al.. OpenAI Baselines; 2017.
34. Racanière S, Weber T, Reicher DP, et al.. Imagination-Augmented Agents for Deep Reinforcement Learning; 2018. Available from: https://arxiv.org/pdf/1707.06203.pdf.
35. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, et al. Mastering the game of Go without human knowledge. Nature. 2017 October;550:354.
36. DeepMind. AlphaGo vs AlphaGo: self play games;. Available from: https://deepmind.com/research/alphago/alphago-vs-alphago-self-play-games/.
37. DeepMind. Innovations of AlphaGo;. Available from: https://deepmind.com/blog/innovations-alphago/.
38. DeepMind. Exploring the mysteries of Go with AlphaGo and China's top players;. Available from: https://deepmind.com/blog/exploring-mysteries-alphago/.
39. AlphaGo;. Available from: https://deepmind.com/research/alphago/.
40. David S, Thomas H, Julian S, Ioannis A, Matthew L, Arthur G, et al. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. CoRR. 2017;abs/1712.01815.
41. DeepMind. AlphaZero: Shedding new light on the grand games of chess, shogi and Go;. Available from: https://deepmind.com/blog/alphazero-shedding-new-light-grand-games-chess-shogi-and-go/.
42. Wolpert DH, Macready WG. No Free Lunch Theorems for Optimization; 1997. Available from: https://ti.arc.nasa.gov/m/proftle/dhw/papers/78.pdf.