Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/39/20230577

Design and analysis of adaptive control systems: Adaptive control algorithms using machine learning

Zikai Zhou

The University of Manchester, Manchester, UK, M13 9PR

zikai.zhou@student.manchester.ac.uk

Abstract. A growing field of research is the use of adaptive control algorithms with machine learning techniques like Q-learning and SARSA. With prospective applications in robotics, healthcare, and other fields, this interdisciplinary method strives to combine the stability and robustness of conventional control systems with the self-learning skills of reinforcement learning. Making sure that systems are stable and that learning occurs effectively is the main research problem. This paper demonstrates the stability and convergence of existing adaptive control algorithms when integrating with machine learning. There are mainly four primary methods of control algorithms: reinforcement learning, neutral network, support vector machine and deep learning. Reinforcement learning is the main focus of this paper. Data efficiency, robustness, and generalization are the main problems with reinforcement learning. Q-learning and SARSA (State, Action, Reward, State', Action') are two algorithms for reinforcement learning. The research will be done by analyzing these two algorithms based on existing material and the actual application of these two algorithms. SARSA is believed to be more safe as its on-policy methodology, and Q-learning is believed to be more proactive as its off-policy methodology.

Keywords: Q- learning, Sarsa, Machine Learning, Adaptive Control.

1. Introduction

To optimize the quality, productivity and efficiency of a system, adaptive control systems using machine learning are used. Adaptive control algorithms allow systems to interact with changing parameters. The algorithms have the ability to adjust controller parameters based on uncertainty and variation from external factors. So that the optimum output can be maintained by the system. Reinforcement learning is a typical algorithm used when adaptive control systems integrates with machine learning.

Reinforcement learning is used to find the optimum operation policy for the controller, which means the decision is made based on the current state of the environment. The ultimate goal is to maximize the reward after getting through a series of steps. This is a crucial branch of machine learning. Reinforcement learning was firstly introduced by psychologists studying animal behaviour learning. Start from 1980s, reinforcement learning began to take shape as a coherent field of study due to the pioneering work of Andrew Barto and Richard Sutton. One of the most successful work was done in 2013, DeepMind published a paper called 'Playing Atari with Deep Reinforcement Learning' and Deep-Q network was firstly introduced. This algorithm has shown extreme performance on Atari 2600

DOI: 10.54254/2755-2721/39/20230577

games. And this is a milestone for reinforcement learning, and this has shown the huge potential of this algorithm in other fields, for example, automatic vehicles and robot-assisted surgery.

Today, reinforcement learning is used in a wide variety of fields, including as robotics, self-driving cars, resource allocation, financial investments, and many more. Simultaneously, reinforcement learning is an important direction in artificial intelligence research, with substantial theoretical and practical implications for realising genuinely autonomous intelligent systems. Nonetheless, numerous issues remain for reinforcement learning, such as stability and sample efficiency, the trade-off between exploration and exploitation, over-optimization, and bias. Addressing these challenges will necessitate additional research into the theory and techniques of reinforcement learning.

This paper will focus on the existing model of reinforcement learning and analysis. The research method will be mostly based on theoretical analysis of existing reinforcement learning algorithms. Computer stimulation will be used to test the performance of a particular model. And finally, research will be conducted on real-world experiments and actual applications of reinforcement learning. This paper will also make improvement suggestions that will benefit future studies in this field.

2. Q-Learning

One of the core reinforcement learning algorithms is Q-learning. For a given finite Markov decision making process, it is a model-free, off-policy algorithm that is used to determine the best action-selection policy. In a similar vein, it aids agents (such as robots or software agents) in selecting the most profitable course of action based on their current situation.

The fundamental principle of Q-learning is to learn an action-value Q(s,a) that, when applied, ultimately yields the expected utility (future reward) of doing a certain action an in a specific state s. Through interaction with its environment, the agent interacts with this function to gain experience. The Q-values are updated repeatedly to represent the agent's learning about which actions generate the most reward in which states. The agent's ultimate objective is to maximise the cumulative reward over time.

The Q-values are updated using the rule:[1]

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)] \quad (1)$$

s is current state

a is action taken in state s

r is immediate reward after taking action a in state s

s' is new state after taking action a

α is learning rate (how much Q-value update we adopt in each update)

γ is discount factor (models the agents' consideration of future rewards)

Exploration vs. Exploitation

The trade-off between exploration and exploitation is one of the main problems with Q-learning. Exploration refers to when an agent tests out several actions to learn about their effects, whereas exploitation refers to when the agency chooses the action it thinks will produce the highest predicted payoff. The " ε-greedy policy" [2] is a popular tactic used to combat this. Here, the agent often chooses the action that satisfies the condition with the highest Q-value, but on rare occasions (with probability), it will select a random action, allowing exploration.

2.1. Application

The first deep learning model to successfully learn control policy directly from high sensory inputs using reinforcement learning was released by DeepMind Technologies. Based on a Q-learning-trained convolutional neural network, this model was created. Raw pixels are used as the input, and a value function that estimates the rewards is produced as the output. Without changing the learning methods, this strategy is applied to seven Atari 2600 games. And the outcome of these games revealed superhuman performance.

In this scenario, we assume that an agent interacts with the Atari emulator's environment through a series of actions, observations, and rewards. The agent chooses one action from the whole range of

DOI: 10.54254/2755-2721/39/20230577

lawful actions, A = {1,...,K} at each time step. The emulator receives the action and changes the internal state and game score. The agent cannot see the emulator's internal state; it can only see an image from the emulator, which is a vector of unprocessed pixels. It also receives a payout at the same time to signify game scores. The order of the actions and observations is taken into consideration, and it is expected that each sequence will end in a finite number of time-steps. As a result, there is now a sizable but finite Markov decision process (MDP) where each different state denotes a sequence.

As mentioned before, the aim of agent is to maximize the future rewards when interacting with emulator. This application has assumed that future reward has been discounted by ϒ per time step, and future discount returned at time-t can be defined as: (T is the time step where game terminates) [3].

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \tag{2}$$

We define optimal action value function as:(π is policy mapping sequences to actions) [3].

$$Q^*(s,a) = \max_\pi E[(R_t|s_t) = s, a_t = a, \pi] \tag{3}$$

The optimal value function obeys an very important rule called Bellman equation. The Bellman equation is founded on the following insight: if the optimal value Q*(s',a') of the sequence s' at the next time step is known, then the best selection will be the action that yields the maximum value of $r + \gamma Q^*(s', a')$[3].

$$Q^*(s,a) = E_{s' \sim \varepsilon}[r + \gamma \max_{a'} Q^*(s', a')|s, a] \tag{4}$$

The core concept of reinforcement learning algorithms is to iteratively update the Bellman equation to estimate the action-value function. $Q_{i+1}(s,a) = E[r + \gamma \max_{a'} Q_i(s', a')|s, a]$. These value algorithms eventually arrive at the ideal action value function,$Q_i \to Q^*$ as $i \to \infty$. IDue to the separation while calculating each sequence, this strategy is impracticable in real-world situations. An approximator for functions is frequently used. Although a linear function is frequently employed in reinforcement learning, non-linear functions can also be used on occasion. We called network approximator with width θ as Q-network. Loss function Li(θi) can be used to train Q-network: [3]

$$L_i(\theta_i) = E_{s,a \sim p(.)}[(y_i - Q(s,a;\theta_i))^2] \tag{5}$$

$y_i = E_{s' \sim \varepsilon}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})|s, a]$ is the target for iteration i and p(s,a) is a probability distribution over sequence s and function a that we call behaviour distribution.

Differentiate the loss function with respect to weight, we can get the gradient function:[3]

$$\nabla_{\theta_i} L_i(\theta_i) = E_{s,a \sim p(.);s' \sim \varepsilon}[(r + \gamma \max_{a'} Q^*(s', a'; \theta_{i-1}) - Q(s,a,\theta_i)) \nabla_{\theta_i} Q(s,a,\theta_i)] \tag{6}$$

Instead of computing all of the expectations in the aforementioned gradient, it is frequently more economical computationally to optimise the loss function using stochastic gradient descent. If the weights are updated after each time step and the expectations are changed to single samples from the behaviour distribution and the emulator ε, respectively, the well-known Q-learning process is reached.

The Q-learning method's ability to use emulator ε samples directly rather than having to expressly build an emulator ε is undoubtedly its greatest benefit. This algorithm therefore lacks a model. In addition, it is against policy. It learns about the greedy approach a = maxa Q(s, a;θ) as it adheres to a behaviour distribution that guarantees sufficient exploration of the state space. In practise, the behaviour distribution is frequently chosen using a greedy approach that selects a random action with probability $1 - \varepsilon$ and then continues the greedy strategy with probability.

2.2. Evaluation

The model-free character of Q-learning, a fundamental reinforcement learning algorithm, its guarantee of convergence in finite regions, and its off-policy learning nature, which permits flexibility in exploration techniques are all benefits. It is very flexible, working well with function approximators like neural networks. It is primarily designed for discrete action spaces, has intrinsic exploration-exploitation issues, tends to overestimate Q-values, and can display instability when combined with non-linear

DOI: 10.54254/2755-2721/39/20230577

function approximators. These drawbacks make it difficult to use. To address its shortcomings, a number of extensions have been developed, including Deep Q Networks (DQNs) and Double Q-learning. Despite being strong and theoretically sound, Q-learning's effectiveness is subtle and context-dependent, and depending on the particular application, its performance might vary greatly.

3. Sarsa: On-policy TD control

The agent can employ temporal-difference (TD) approaches to determine the best course of action if learning a model is not practical [4]. The agent's action-value function, which predicts the long-term discounted reward it will receive if it takes a particular action in a given state, is updated every time it takes an action using the feedback that results from that action. Under specific assumptions, the best action-value function, from which an ideal policy may be easily inferred, is guaranteed to be obtained using TD techniques.

The Sarsa method is the most popular TD technique. The update rule for Sarsa uses five components, which are represented by the names current state s, current action a, and immediate reward r, next sate st+1, next action at+1. The significance of state-action pairings is now shown when we take into consideration modifications from one state-action pair to the next. Because both of these scenarios feature Markov chains with reward procedures, they are formal counterparts. The theorems establishing the convergence of state values under TD(0) also apply to the analogous method for action values.[5]

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (7)$$

The update is complete upon each transition from nonterminal state St. We continually estimate q_π for the behaviour policy π using the on-policy technique. Change in the direction of greediness is also occurring. The algorithm is displayed below:[5]

Initialize $Q(s, a), \forall s \in A(s), \text{arbitarily, and } Q(\text{terminal} - \text{state}, \cdot) = 0$
Repeat(for each episode):
 Initialise S
 Choose A from S using policy derived from Q $(\text{e.g.}, \epsilon - \text{greedy})$
 Repeat(for each step of episode):
  Take action A, observe R, S'
  Choose A' from S' using policy derived from Q $(\text{e.g.}, \epsilon - \text{greedy})$
  $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$
  $S \leftarrow S'$; $A \leftarrow A'$;
Until S is terminal

How the policy relates to q has an impact on the stability and convergence of the Sarsa algorithm, such as through ε-greedy or ε-soft strategies. Satinder Singh has personally conveyed that Sarsa, under conditions like infinitely visiting all state-action pairs and the policy eventually becoming greedy, to an optimal policy and its accompanying action-value function is likely. Specifically, this can be achieved using ε-greedy strategies with ε set as 1/t. However, this assertion hasn't been formally documented in published works.

One of the application of SARSA is introduced in 'FlapAI Bird: Training an Agent to Play Flappy Bird Using Reinforcement Learning Techniques' [6]. The implementation is in the Link below: https://github.com/taivu1998/FlapAI-Bird

3.1. Evaluation

An on-policy reinforcement learning method that directly learns Q-values of state-action pairings through environmental interactions is the SARSA (State-Action-Reward-State-Action) algorithm. Under some circumstances, it is assured to converge, balancing exploitation and exploration through greedy means. Large state or action spaces require function approximators, which add complexity and the possibility of instability even though they are computationally efficient in their simplest form. As a strategy that adheres to policy, SARSA has a tendency to be conservative, putting safety first in settings where experimentation could lead to serious consequences. Its success depends on task characteristics

DOI: 10.54254/2755-2721/39/20230577

and hyperparameter optimisation, with applications ranging from robotics to gaming. Since SARSA takes into account the current, frequently exploratory policy, it can be more cautious than Q-learning, an off-policy method, particularly in the early learning stages.

4. Compare Q-learning and Sarsa

4.1. Update rule

The algorithms use unique equations to refresh their Q-values. For SARSA, the update formula is: $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$, with 'a'' representing the chosen action in the state 's'' based on the current policy. Alternatively, Q-learning employs the update: $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max Q(s', a') - Q(s, a)]$, where the highest value is derived from evaluating all possible actions 'a'' in the state 's''.[6]

4.2. Convergence

The two are also distinguished by convergent features. If all state-action pairings are infinitely visited and certain requirements for the policy and learning rate changes are satisfied, SARSA promises convergence to an optimal policy. On the other hand, Q-learning offers a guarantee of convergence to the optimal policy, assuming an infinite number of visits to each state-action combination and precise limitations on the learning rate, even if the exploratory policy, such as ε-greedy, isn't optimal.

4.3. Learning approach

The methods of learning used by SARSA and Q-learning are fundamentally different. Being an on-policy algorithm, SARSA learns the Q-value based on the course of action dictated by the current policy, which is frequently a -greedy strategy. The abbreviation "SARSA" stands for the order of State, Action, Reward, State, and Action, and it indicates this. Q-learning, in contrast, takes an off-policy attitude. Without respect to the policy being used, it learns the Q-value by assuming that the action that follows is optimal and updates its Q-value depending on the largest reward attainable from the succeeding state.

4.4. Exploration strategy

SARSA and Q-learning differ in their approaches to exploration. Throughout its learning trajectory, SARSA frequently employs -greedy exploration, potentially making it more conservative. This is due to the fact that it updates Q-values while taking the policy's exploration into consideration. While Q-learning can also engage in -greedy exploration, it based its updates on the assumption that an optimal policy will be in place moving forward, making it a more aggressive strategy.

4.5. Risk and safety

The algorithms differ in how they handle risk while learning. Because SARSA is on-policy, it tends to be safer in situations where taking improper, exploratory actions can result in severe fines. Q-learning can be riskier during the learning phase in situations where exploration might be harmful because it is designed to learn about the best course of action without necessarily following it.

4.6. Application and performance

There are differences in the real-world applications and performance indicators. Because of its potential conservatism, SARSA frequently finds favour in situations where serious penalties are attached to improper behaviour. Q-learning is better suited for situations when there aren't many substantial downsides introduced by exploration because of its effective push towards optimal policy learning.

In conclusion, the environment's characteristics and the precise objectives of the reinforcement learning assignment frequently influence the decision between SARSA and Q-learning. Q-learning aggressively pursues an optimal policy, possibly at the expense of more exploration-induced errors, whereas SARSA is more cautious and secure.

DOI: 10.54254/2755-2721/39/20230577

5. Conclusion

SARSA and Q-learning stand out as two potent algorithms in the broad field of reinforcement learning. Each is praised for its unique methods and significant contributions to the field. The unique difficulties of a given problem have a significant impact on the decision between them or the potential for incorporating their features into creative methods.

SARSA, which is renowned for its on-policy methodology, conducts itself like an attentive pupil. Its current policy directs its frequently greedy activities, and it learns from the results of those acts. By using a purposeful learning approach, SARSA makes careful to consider how actions will affect the real world. SARSA's careful approach makes it appropriate in circumstances where a slip-up can result in substantial setbacks, analogous to a student improving his methods after receiving feedback.

In contrast, Q-learning's off-policy approach is bolder and more proactive. It is intended to anticipate the best outcomes in the future, changing its Q-values in accordance with what it believes to be the best course of action, regardless of the actual decision taken. The exploratory character of Q-learning encourages further research and, under some circumstances, can hasten learning. However, in situations where mistakes have serious repercussions, this aggressiveness can also prove dangerous.

Their different Q-value update methods reflect their diametrically opposed ideologies. While Q-learning optimistically seeks the greatest possible future reward without regard to its actual viability, SARSA updates based on its next likely action. Their divergent strategies highlight how various problems demand various methods.

In the real world, SARSA could be compared to a careful financial planner weary of market fluctuations, whereas Q-learning is like an intrepid businessperson always looking for the next big thing and unafraid of potential dangers.

Picking a favourite is easier than choosing between the two. The decision is guided by the particular goals, the situation at hand, and the anticipated outcomes. SARSA is the recommended option when safety is of the utmost importance. But Q-learning takes the stage when speed and discovery are desired.

Future algorithms in reinforcement learning may include components from SARSA and Q-learning, balancing their differences and maximising their strengths. This is conceivable given the current state of the field. Such cooperative developments show potential for more effectively tackling complex real-world issues.

References

[1] Richard S. Sutton and Andrew G. Barto (2014) Reinforcement Learning: An Introduction, The MIT Press, London England Baeldung (2023) Epsilon-Greedy Q-Learning.

[2] https://www.baeldung.com/cs/epsilon-greedy-q-learning#:~:text=The%20epsilon%2Dgreedy%20approach%20selects,what%20we%20have%20already%20learned.

[3] Volodymyr M, Koray K, David S (2013) Playing Atari with Deep Reinforcement Learning, DeepMind Technologies, 1-8

[4] R. S. Sutton (1988), "Learning to predict by the methods of temporal differences," Machine Learning, vol. 3, pp. 9–44

[5] R. S. Sutton and A. G. Barto(1988), Reinforcement Learning: An Introduction. Cambridge, Massachussets: MIT Press

[6] Tai vu, Leon Tran (2020), Flap AI Bird: Training an Agent to Play Flappy Bird Using Reinforcement Learning Techniques, Stanford University, 1-12