Method Exploration for Business Intelligence Using Reinforcement Learning in Big Data Scenarios

Wei Yang[1], Huaiwang Shi[2], Wei Zhang[1]

[1] School of Management Science and Engineering, Anhui University of Finance and Economics, Bengbu 233000, China;

[2] School of Accounting, Anhui University of Finance and Economics, Bengbu 233000, China

Abstract

The emergence of the big data era has generated a qualitative shift in the types of existing data structures. Traditional Business Intelligence technologies typically focus on cost control and plan execution reports for corporate operations, which fall short of modern businesses' need for precise decision-making. This study establishes the application strategy of reinforcement learning in business intelligence based on the environment of big data.

Keywords

Markov chain, big data, reinforcement learning.

1. Introduction

With the coming of the big data era, business intelligence is becoming more and more important in data information analysis and mining. In today's global market, where customer demands are diverse, firms need to quickly evaluate and utilize enterprise-generated information in order to stay ahead of the competition. Most traditional business intelligence solutions have an immature application foundation, and organizations have established business intelligence platforms intelligence technology application, as well as a lack of theoretical application analysis and application analysis of multiple technologies, will result in issues such as lagging information analysis, limited technology application by the enterprise environment, and difficult multi-state correlation analysis.

In the future, the reinforcement learning approach will be adopted to improve the technological methods and adaptability of business intelligence applications, thereby compensating for their deficiencies. Business intelligence plays a pivotal role in harnessing the power of enterprise information resources, seamlessly integrating with the features of the big data era. It serves as the foundation for creating and applying innovative data environments, empowering businesses to make well-informed operational decisions with unwavering support. As a result, the implementation of business intelligence research is particularly critical.

2. Related theories of reinforcement learning

Reinforcement learning is a type of learning that falls midway between supervised and unsupervised learning. It is a type of strategy-based learning. Reinforcement learning gathers information about the status of the environment by instantaneous interaction with it and evaluates actions made through feedback reinforcement signals. Through trial and error and selection, reinforcement learning learns the best strategy.

The environment and agent form the foundation of a reinforcement learning framework. Agent has a specific learning objective. All agents are conscious of their surroundings and direct their behavior in accordance with their objectives. If an Agent's behavior approach results in a favorable Reward from the environment for the Agent, the Agent is more likely to continue using that behavior strategy in the future. If, on the other hand, it results in a negative reward, the Agent's proclivity to perform this activity is diminished.

A strategy, a reward function, a value function, and a model of an optional environment are the four fundamental components of a reinforcement learning system [2].

(1)Policy

The Agent's action and overall performance are ultimately determined by the quality of the strategy, which is at the core of reinforcement learning. For each probable status, the policy outlines the series of activities the Agent should perform.

(2)Reward Function

The reward signal obtained from interacting with the environment is what powers the reward function. The reward signal R is typically a scalar signal, where the reward is represented by a positive number and the penalty by a negative number. The objective of reinforcement learning is to maximize the Agent's overall reward value.

(3)Value Function

The value function, also referred to as the evaluation function, takes a long-term view on the quality of a state (or state-action pair). The foundation of the reinforcement learning algorithm is the value function.

(4)Model of the environment

The external environment is simulated using the environment model. The model will forecast the following state and reward signal when the Agent takes an action in a specific state. Trial-and-error learning and planning are viewed as a method of acquiring experience in reinforcement learning. Reinforcement learning systems become closely related to dynamic programming techniques when models and planning are introduced.

Let’s start with Q-learning of reinforcement learning[3]. The learning issue under the MDP environment model of the Markov decision process is mostly resolved by Q-learning. Single-step Q-Learning is a method for delayed learning that was simply developed from the principles of dynamic programming. Policy and value functions in Q-Learning are represented by a two-dimensional query table with state-action pair indexed. For each state x and action a:

$$Q^*(x,a) = R(x,a) + \gamma \sum_y P_{xy}(a) V^*(y) \tag{2-1}$$

In this case, $R(x,\alpha) = E\{r_0 \mid x_0 = x, \alpha_0 = \alpha\}$, the probability that a state x will change to a different state, y, is expressed by the number $P_{xy}(a)$. The Q-Learning method retains an estimate Q^* of the function, indicated by $\hat{Q}^*$. Based on the action taken and the reward value received, it modifies the value of $\hat{Q}^*$. When the immediate reward is included, the $\hat{Q}^*$ value is changed in accordance with Sutton's prediction bias, also known as the TD error, which is the difference between the discounted value of the future state and the Q value of the present state-action pair:

$$r + \gamma \hat{V}^*(y) - \hat{Q}^*(x,a) \tag{2-2}$$

In this case, the next state to which action an is moved from state x is y, and r is the immediate reward value. $\hat{V}^{**}(x) = \max_a \hat{Q}^*(x,a)$.The following equation determines how the $\hat{Q}^*$values are updated:

$$\hat{Q}^*(x,a) = (1-\alpha)\hat{Q}^*(x,a) + \alpha\big(r + \gamma \hat{V}^*(y)\big) \tag{2-3}$$

In this case,$\alpha \in (0,1]$.The α parameter determines the learning rate and how much confidence should be placed in the corresponding update.

The Q-Learning algorithm uses TD(0) as an estimation factor for the expected return value. We may be aware that $\pi(x) = \arg\ \max_a \hat{Q}^*(x,a)$ defines the current estimate of the function Q^* as a greedy strategy, which chooses the action based on the biggest estimated Q-value. When each state changes its estimate, the first-order Q-Learning method does not specify in clear terms what the Agent should do[4]. The Agent has the capacity to carry out any activity. The Agent must repeatedly try each of the available actions in each state in order to determine the best Q-function.

3. Overview of Business Intelligence

A multidisciplinary analysis and decision-making system known as a business intelligence system can assist organizations in swiftly finding hidden information in a complex business environment and enhancing the precision and effectiveness of organizational decision-making and organizational prediction. The advantages of business intelligence are shown in Table 3-1.

Table 3-1: The advantages of business intelligence

Project	Existing Data	Business Intelligence
Data Structure	Unstructured or semi-structured	Highly structured data
Data Management	Chaos without order, can only meet the basic application of enterprise business	Integrated management of enormous data
Information Mining Level	The data contains a wealth of untapped information.	Exploiting the data's implied information

Due to the theoretical nature of most business intelligence research, there are significant challenges when it comes to actually implementing corporate management, technology, and culture. The following are the primary issues:

(1)Enterprises have a variation in how they think about business intelligence. Enterprise managers neglect their own management in favor of competing with rivals, and there is a deviation in the specific implementation procedure.

(2)The enterprise cannot guarantee that the data it obtains will be of high quality. Due to the shortcomings in the information systems of numerous businesses, the expenses associated with loading, transforming, and extracting data for business intelligence purposes will surge. Additionally, this will have a detrimental impact on the accuracy of analysis outputs.

(3)The data is more fragmented. It is difficult to share information between various departments of an enterprise. Business intelligence systems need to integrate resources to analyze more accurate solutions.

4. The combining method of Reinforcement Learning with Business Intelligence

Any business intelligence issue can be viewed as an agent. Transition relations exist as long as states may be separated, there are relationships between states, and these transition connections have rewards associated with them. It can swiftly create findings and continuously modify itself in response to changes in the external data environment by using reinforcement learning techniques.

4.1. The state transition problem

(1)Division of states in business intelligence

Having a distinct boundary, a finite number of states divided by its properties, and the transitions between the states are the first issues to be resolved in business intelligence. According to the real circumstance, polymorphic Markov-related theoretical knowledge may be used to split the state specifically. The states involved can be divided manually, and if there are more states, the Clementine approach in machine learning can be used to divide the states. Numerous techniques exist for dividing states, most commonly using established division criteria or subjective state classification, sample mean-standard deviation classification, clustering analysis, and state division based on simulation value. After analysis, we partition the pertinent data into N states, each of which represents state i, where i=1,2,3,...,n.

(2)The problem of dynamic transitions between states

The dynamic Process of Markov Decision Process (MDP) has the following steps: Allow an agent to begin in state S_0, before selecting action a_0 from A. With probability P_{sa}, the agent randomly moves to the following state S_1, then performs action a_1, then randomly moves to state S_2, then performs action a_2... We may depict the total reward functions obtained by the entire process $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} \cdots$ using the following diagram:

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots \tag{4.1}$$

Choose the best set of actions that maximize the overall reward weighted sum expectation.

$$E[R(s_0) + \gamma(s_1) + \gamma^2 R(s_2) + \cdots \tag{4.2}$$

When we are already in a state s, we will use some policy π to choose the next action a to perform and then transition to another state. This action selection process is called A policy, and each policy is actually a state-to-action mapping function π:S→A. Given π we also give a=π(s).

In order to tell the difference π is good or bad, we need to define the value function-the discounted cumulative return.

$$V^{\pi}(s) = E[R(s_0) + \gamma(s_1) + \gamma^2 R(s_2) + \cdots |s_0 = s, \pi] \qquad (4.3)[5]$$

In the current state s, after choosing a good strategy, the value function is the reward weighted sum expectation. Given π, we are also given a future action, which will go through a number of states, and each state will have a reward value. The states closer to the current state will have more influence on the action, and the weight will be higher. From a recursive point of view, the value function V of the current state s can be viewed as the sum of the reward R(s) of the current state and the value function of the next state. The above equation becomes:

$$V^{\pi}(s) = R(s_0) + \gamma(E[R(s_1) + \gamma(s_2) + \gamma^2 R(s_3) + \cdots]) = R(s_0) + \gamma V^{\pi}(s') \tag{4.4}$$

Although, given π, a is unique in A given state s, A→S may not be a many-to-one mapping. From the equation above, using Bellman's equality, we can get：

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s') \tag{4.5}$$

In Equation 3.5, s' denotes the next state, R(s) is the current state R, which is called immediate reward. You could also write $\gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s')$ as $E_{s' \sim p_{s\pi(s)}}[V^{\pi}(s')]$, which is the expected value of the next state value function. The next state conform to $P_{s\pi(s)}$ distribution.

The purpose of finding V is to find an optimal action policy π in the current state s. The optimal V^* is defined as follows:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \tag{4.6}$$

We can pick an optimal policy π from the alternativ:

$$V^*(s) = R(s) + \max_{a\in A} \gamma \sum_{s'} P_{sa}(s') V^*(s') \tag{4.7}$$

Having defined the optimal V^*, we define the optimal policy π^*:S→A as follows:

$$\pi^*(s) = \arg\max_{a\in A} \sum_{s'\in S} P_{sa}(s') V^*(s') \tag{4.8}$$

Choosing the optimal π^* also determines the next optimal action a for each state s.

$$V^*(s) = V^{\pi^*}(s) \geq V^{\pi}(s) \tag{4.9}$$

The optimal value function V^* of the current state is obtained under the optimal execution policy π^*. The mapping S→A can be generated, and this mapping is the optimal mapping.[3]

4.2. Algorithms for solving specific policies of the MDP

Here, we mainly discuss two efficient algorithms for solving specific policies of finite-state MDPS, only for the case where the MDP is finite-state and finite-action, $|S| < \infty, |A| < \infty$.

4.2.1. Value iteration method

First, we initialize V(s) to 0 for each s; Second, loop until convergence for each state s, update V(s)

$$V(s) := R(s) + \max_{a\in A} \textstyle\sum_{s'} P_{sa}(s')V(s') \tag{4.10}$$

The value iteration strategy makes use of Equation 3.10, and there are generally two strategies for implementing the inner loop:

Synchronous iterative method: All of V(s) are initially in the state of zero in the first iteration following startup. For each s, determine the new V(s)=R(s)+0=R(s).

The new V(s), which is obtained after computing each state, is initially stored. After calculating each new value V(s) for s, they are all updated uniformly.

Asynchronous iterative method: Corresponding to the synchronous iteration, for each state s, the new V(s) is obtained, which is not stored and directly updated.After the first iteration, most V(s)>R(s).

Regardless of which of the two you use, V(s) will eventually converge to $V^*(s)$. We apply the formula to determine the matching ideal policy π^* given V^*. As you solve for V^*, you can also solve for π^*.

4.2.2. policy iteration method

The Value iteration method makes the V converge to V^*. However, the policy iteration method focuses on π such that π converges to π^*. A mapping π from S to A will be determined at random, followed by a loop until convergence

{ First let $V:=V^{\pi}$, then update π(s) :let $\pi(s) := \arg\max_{a\in A} \sum_{s'} P_{sa}(s')V(s')$ for every state s}

$$\pi(s) := \arg\max_{a\in A} \textstyle\sum_{s'} P_{sa}(s')V(s') \tag{4.11}$$

The V of V:=V^{π} can be found by the Bellman equation.

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s')V^{\pi}(s') \tag{4.12}$$

We can find $V^{\pi}(s)$ for all states s.

According to the result of V:=Vπ, the optimal a in the current state s is selected. Then we can update π(s).

It's hard to say which is better and which is worse for value iteration and policy iteration. Policy iteration generally converges faster for smaller MDPS. But for very large MDPS (many states), value iteration is easier.

4.2.3. MDP parameter estimation

It is necessary to estimate the state transition probability and reward function R(s) parameters from the data in practice because they cannot be obtained explicitly (usually S, A, and 2 are known).

Suppose we have several state transition routes, see Figure 1.

$$s_0{}^{(1)} \xrightarrow{a_0{}^{(1)}} s_1{}^{(1)} \xrightarrow{a_1{}^{(1)}} s_2{}^{(1)} \xrightarrow{a_2{}^{(1)}} s_3{}^{(1)} \xrightarrow{a_3{}^{(1)}} \cdots$$

$$s_0{}^{(2)} \xrightarrow{a_0{}^{(2)}} s_1{}^{(2)} \xrightarrow{a_1{}^{(2)}} s_2{}^{(2)} \xrightarrow{a_2{}^{(2)}} s_3{}^{(2)} \xrightarrow{a_3{}^{(2)}} \cdots$$

$$\cdots$$

Figure 1: State transition routes

In this case, is the state corresponding to the j-th transition path at time i. is the action to be performed in state . Every transition path has a finite number of states, and every transition chain comes to an end when it either reaches a certain number of steps or reaches the terminal state.

The state transition probabilities can be estimated using maximum likelihood estimation if we have numerous transition chains like the one above (which is the same of having samples).

The denominator is the number of times an is carried out in state s, and the numerator is the number of times an is reached from state s after carrying out action.[6] The division represents the likelihood that, after executing an in state s, it will transition to . If the denominator is 0, then let .

This formulation also applies to online updates. This formulation also applies to online updates. When we have some new transition paths, we can modify the numerator and denominator (add the new count). The correction modifies the transition probability. More new transition pathways that are more accurate may appear in accordance with the altered likelihood.

After estimating the transition probabilities and reward functions, the MDP issue can be solved using value iteration or policy iteration. For instance, without knowing the probabilities of the state transitions, we can combine parameter estimation with value iteration as follows: First, random initialization . Second, in the loop until convergence { count the number of state transitions occurring in on the sample, which is used to update and R; update V with the estimated parameters (using value iteration); re-derive based on the updated V}.

5. Conclusion

For the best outcomes in big data settings, business intelligence and reinforcement learning should be organically blended. Algorithms for reinforcement learning should also be continually improved to match the needs of the dynamic business intelligence environment.

Acknowledgements

This paper is supported by the University Scientific Research Project of Anhui University of Finance and Economics (ACKYC19049)

About the author:

Wei Yang, Female, her research interests include big data analysis and applications.

Huaiwang SHI, Male, his research interests include international accounting.

Wei Zhang, Male, his research interests include information management and information system.

References

[1] Purohit, C. S., Manna, S., Mani, G., & Stonier, A. A. (2021). Development of buck power converter circuit with ANN RL algorithm intended for power industry. Circuit World, Vol. 47 No. 4, pp. 391-399.

[2] Baochang Zhang, Ce Li, Nana Lin. (2020). Machine Learning and Visual Perception. New York: Walter de Gruyter GmbH.

[3] Zhong Liu, Haihong Li, Quan Liu. (2008).Research on Reinforcement Learning Algorithms. Computer Engineering and Design, 29(22): 5805-5809.

[4] Jing Peng, Ronald J. Williams. (1994). Incremental multi-step Q-learning. Proceedings of the 11th International Conference on Machine Learning, ICML 1994, 226-232.

[5] Fang, D., Guan, X., Peng, Y., Chen, H., Ohtsuki, T., & Han, Z. (2021). Distributed Deep Reinforcement Learning for Renewable Energy Accommodation Assessment With Communication Uncertainty in Internet of Energy. IEEE Internet of Things Journal, 8(10), 8557–8569.

[6] Jaimes, L. G., Llofriu, M., & Raij, A. (2015). CALMA, an algorithm framework for mobile just in time interventions. SoutheastCon 2015, pp. 1-5.