International Journal of Internet, Broadcasting and Communication Vol.14 No.2 192-198 (2022)
http://dx.doi.org/10.7236/IJIBC.2022.14.2.192

IJIBC 22-2-23

Reinforcement learning multi-agent using unsupervised learning in a distributed cloud environment

Seo-Yeon Gu*, Seok-Jae Moon**, Byung-Joon Park***

**Master, Department of Computer Science, Kwangwoon University, Korea.*
***Professor, Department of Artificial Intelligence Institute of Information Technology, KwangWoon University, Korea*
****Professor, Department of Computer Science, Kwangwoon University, Korea.*
E-mail: {gsy0207, msj80386, bjpark}@kw.ac.kr

Abstract

Companies are building and utilizing their own data analysis systems according to business characteristics in the distributed cloud. However, as businesses and data types become more complex and diverse, the demand for more efficient analytics has increased. In response to these demands, in this paper, we propose an unsupervised learning-based data analysis agent to which reinforcement learning is applied for effective data analysis. The proposal agent consists of reinforcement learning processing manager and unsupervised learning manager modules. These two modules configure an agent with k-means clustering on multiple nodes and then perform distributed training on multiple data sets. This enables data analysis in a relatively short time compared to conventional systems that perform analysis of large-scale data in one batch.

Keywords: *Cloud, Reinforcement Learning, Clustering, Unsupervised Learning, Data Anlaysis*

1. Introduction

Companies have built and utilized their own data analysis systems according to their business characteristics in a distributed cloud. However, as the number of tasks increases, more effective data analysis using a distributed data system that is interoperable between companies is required [1]. In addition, as the complexity of work increases, the areas where data are collected become more diverse and more numerous, so it is essential to consider the characteristics of specific work areas and agents [2]. The efficiency of data analysis for these multiple agents can be solved through a big data analysis platform using machine learning techniques such as reinforcement learning [3]. In this paper, we propose an unsupervised learning-based data analysis agent to which reinforcement learning is applied. The environment created by multiple agents is based on the environment of multi-agent reinforcement learning. The proposed agent consists of two modules: *Reinforcement Learning Processing Manager* and *Unsupervised Learning Manager*. This agent is a grouping of several nodes using k-means clustering, which is unsupervised learning, and enables distributed training on N data sets that are randomly collected. As each agent efficiently distributes a lot of data, data analysis is possible in a relatively short time. The structure of this paper is as follows. Chapter 2 examines related work, and Chapter 3 describes the agent composition and reinforcement learning techniques proposed in this paper. Chapter 4 describes the agent performance analysis, and finally, chapter 5 concludes.

Manuscript Received: April. 18, 2022 / Revised: April. 20, 2022 / Accepted: April. 22, 2022
Corresponding Author: msj8086@kw.ac.kr
Tel: Fax: +82-10-916-4751
Professor, Department of Artificial Intelligence Institute of Information Technology, KwangWoon University, Seoul, Korea

2. Related Work

2.1 Multiple Reinforcement Agent

Early approaches of multi-agent reinforcement learning were proposed in which individual agents learn their value functions and policies independently. In general, in the case of Independent Q-Learning (IQL) [4] based on Q learning, each agent independently performs Q learning [5], and further extended to DQN [6]-based deep reinforcement learning. A technique [6] has also been proposed. In other words, IQL simply performs distributed execution. Therefore, it is difficult to perform effectively because of the instability (non-stationarity) caused by simultaneous learning and search of agents. In order to solve this instability, a technique [7] focusing on stability based on past experience was introduced. However, in this technique, it is not easy to obtain a multi-agent behavior policy that can show coordinated cooperative behavior among team members because each agent learns the behavior policy independently of each other based on the limited observation ability. Conversely, approaches of centralized training techniques have been proposed by completely excluding the independence of agents. Central-centralized training is a method to improve instability by naturally regulating cooperation between agents. However, as the size and number of individual agents increases, the complexity of centralized training increases, making it difficult to perform effectively. Typically, the classic central-centralized training method is a method using coordination graphs [8].

2.2 K-means Clustering

Since K-Means Clustering is unsupervised learning among machine learning fields, only input data is given and no correct answer is given. When no correct answer is given, it is an algorithm that identifies the similarity between the given input data and divides similar data into K clusters [3]. As a result, K clusters and K centroids, which mean the average value of each cluster, are derived. This K-Means Clustering algorithm is being used in various fields, such as establishing customized marketing strategies through customer group analysis of companies, grouping news articles on portal sites by topic, and forecasting weather through clustering of climate data.

3. Proposed Agent

3.1 Cluster-based multi-agent environment in distributed environment

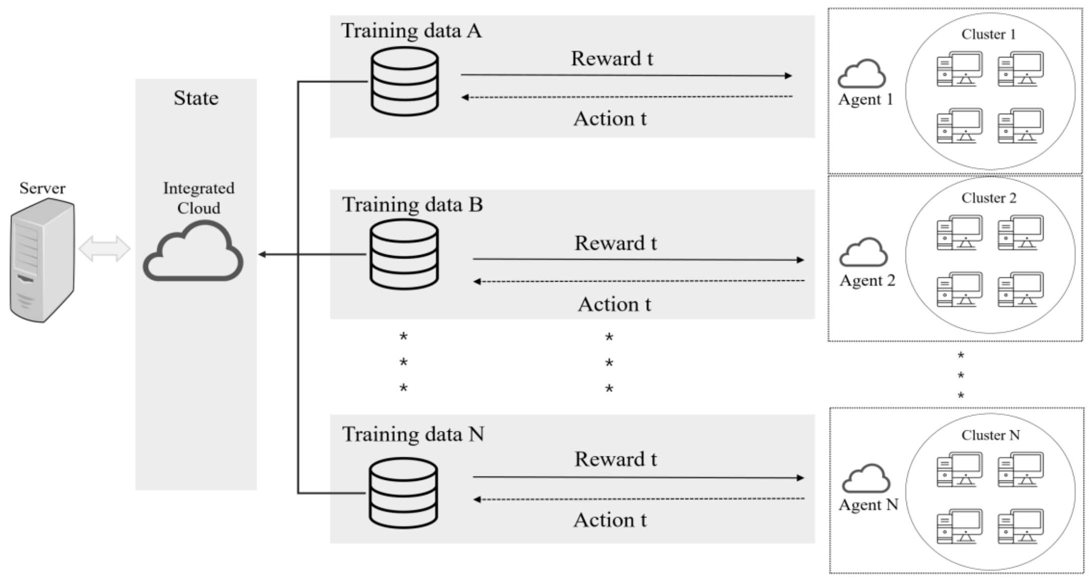

Figure 1. Cluster-based multi-agent environment in distributed environment

Figure 1 shows the cluster-based multi-agent environment in the distributed environment proposed in this paper. The server and the integrated cloud interact and collect data generated from multiple resources. Thereafter, big data is divided into N sets. Then, it is transmitted to several agents consisting of a cluster consisting of N nodes. After confirming the receipt of data, the agent trains the data using reinforcement learning. The trained results are sent back to the server in the form of 'actions'. This process is repeated until the point at which the reward is maximized, that is, the target query is reached.

3.2 Configuration of Multi-Agent

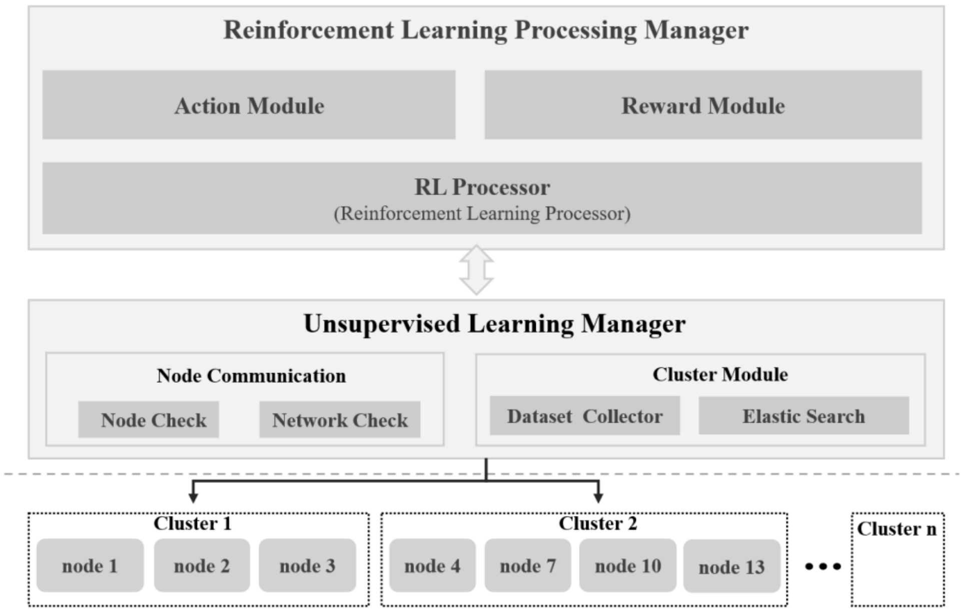

Figure 2. Multi-agent and cluster architecture

The agent consists of a reinforcement learning processing manager and an unsupervised learning manager. Reinforcement Learning Processing Manager consists of Action Module, Reward Module, and RL_Processor, and Unsupervised Learning Manager consists of Node Communication Module and Cluster Module. The following is a description of each module.

3.2.1 Unsupervised Learning Manager

- Node Communication Module: A module responsible for communication between nodes. Before clustering, the state of each node is checked, and the executable node is checked and the result is derived. It is possible to construct a flexible node network by checking whether the node can be executed or not. Receives information about nodes selected to configure a cluster from the Cluster Module, and transmits training data to the configured cluster for learning.
- Cluster Module: The Cluster Module receives node information from the Node Communication Module and actually performs clustering. It creates an optimal learning environment by continuously communicating with the Node Communication Module.
- Data Collector: The Data Collector module is a module that collects trained data information from each node. The final output, that is, the Action, is sent from the Action Module to the Main Server, and each result is collected in the Dataset Collector.

3.2.2 Reinforcement Learning Processing Manager

- Action Module: In this module, the action in reinforcement learning is taken by the agent to control the environment in the direction to maximize the reward, and the action in the proposed agent is the result of learning based on the reward (learned dataset) can be seen. Action Module is a module that receives the trained dataset.
- Reward Module: A module that manages the rewards the agent receives from the Main Server. When training is started for the first time, big data from the integrated cloud is used as training data. This training data is transmitted to the Reward Module. The transmitted data is divided into N sets and delivered to RL_Processor. Reinforcement learning interacts with the environment and learns to maximize rewards. When reward comes from the main server, the reward module compares whether this is the target query or training result.
- RL_Processor (Reinforcement Learning Processor): RL_Processor sends a node communication request to the Node Communication module of the unsupervised learning manager to deliver the N data sets delivered from the Reward Module to each cluster.

3.3 Execution Structure

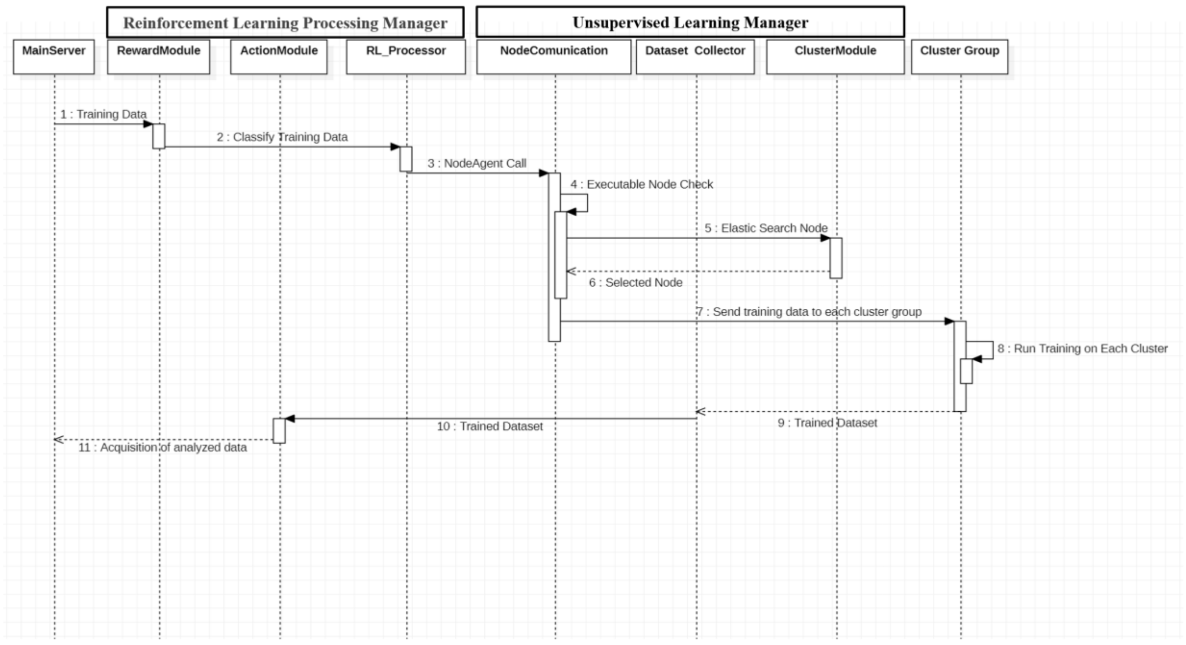
Reinforcement Learning Processing Manager
Unsupervised Learning Manager
MainServer
RewardModule
ActionModule
RL_Processor
NodeComunication
Dataset Collector
ClusterModule
Cluster Group
1 : Training Data
2 : Classify Training Data
3 : NodeAgent Call
4 : Executable Node Check
5 : Elastic Search Node
6 : Selected Node
7 : Send training data to each cluster group
8 : Run Training on Each Cluster
9 : Trained Dataset
10 : Trained Dataset
11 : Acquisition of analyzed data

Figure 3. Sequence Diagram

Figure 3 shows the data flow of the proposal agent as a sequence diagram. The above sequence represents the flow of the entire system and is repeated until the reward is maximized.

1. Training Data: Raw data is collected from the Main Server and delivered to the Reward Module in the form of large-scale data.
2. Classify Training Data: Divided into N data sets in the Reward Module.

3 ~ 6: Node information interacts to create a cluster, and packets are sent and received for smooth system communication.

7. Send training data to each cluster group: Map N data sets and N clusters.
8. Run Training on Each Group: Train the model in each cluster.
9. Trained Dataset: When N datasets are trained in each cluster, the results are sent to the Dataset Collector.
10. Trained Dataset: N outputs from the Dataset Collector are collected as one output, and it is sent to the

Action Module.
11. Acquisition of analyzed data: In the Action Module, the output is treated as an Action of reinforcement learning and sent to the Main Server.

3.4 Implementation Algorithm

```
Input D: Training Data Collected by Main Server
Output O: Trained Dataset
BEGIN:
    tA,tB,...,tN = ActionModule.ClassifyData(D) # Split training data into N
    i = 0
    for i in range(n): # Repeat until reward is maximum.
# Request for cluster creation and node communication
    RL_Processor.CallNode(tN)
# Save the non-executable node information.
non_executable_node = NodeCommunication.NodeCheck(tN)
# Information of node communication state
Info_N=ClusterModule.ElasticSearch(non_executable_node,tN)
# Create N clusters.
N_clusters = ClusterModule.MakeClusters(Info_N, tN)
    # Send a set of N datasets to each cluster.
    NodeCommunication.Send_tN_to_Nclusters(tN, N_clusters)
    # Proceed with training and return the output.
    Run training tN on each cluster: return N_trained_dataset

# Convert N outputs to one output.
    Output_trained_dataset = DatasetCollector.Collect(N_trained_dataset)
# Send the output to MainServer.
    ActionModule.Send_to_MainServer(Output_trained_dataset)
    if Output_trained_dataset == TargetQuery: # End iteration when reward is max.
        break
end for
END
```

Algorithm 1. Sequence Algorithm

The multi-agent system is implemented in Python. Algorithm 1 lists the execution of each module according to the sequence.

4. Experiments and Results

In order to measure the performance of the agent proposed in this paper, a comparative experiment was performed with a system that integrates the existing dataset in one server and proceeds with learning. In the experiment, the energy cost dataset provided by the public data portal was used, and the performance was compared by measuring the execution time until the completion of learning the given data. The experimental environment and method for performance measurement are as follows.

- Execution Environment: VMware, Windows Server 2020, MySQL, Linux Ubuntu 5 nodes, 4 agents
- Setting experimental goals and assumptions: It aims to build a cloud environment that can minimize energy

use costs by learning the energy cost dataset.
- Measure the response failure rate of the node according to the increase in the number of nodes in the agent.

In this experiment, the number of nodes in the agent is set to 5, and the result represents the average of the measured values obtained through a total of 50 experiments.

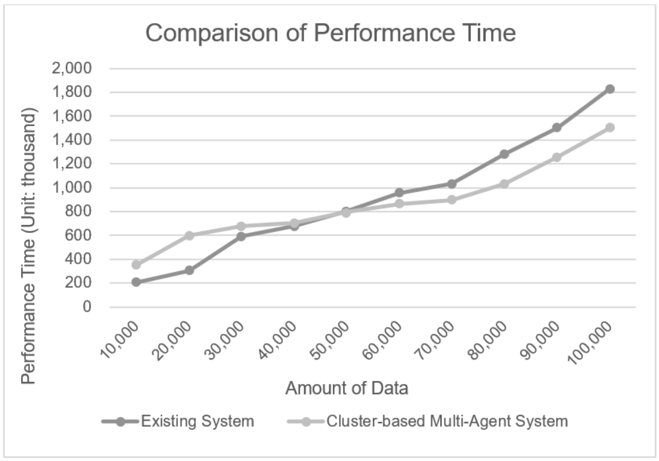

Figure 4. Comparison of execution time between the existing system and the proposed agent system

Figure 4 is a chart comparing the execution time of the existing system and the proposed agent system. When there is relatively little data, the execution time of the proposed agent system is faster, but it can be seen that the performance of the existing system is advanced as the amount of processed data increases. The reason is that, in the proposed agent system, it is expected that the response time is slower than in the existing system because it requires an agent adjustment time by distributing data into sets and determining the resource information of nodes.

Table 1. Cluster response failure rate according to the number of nodes in the agent

	cluster1	cluster2	cluster3	cluster4	cluster5	cluster6	cluster7	cluster8
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	1.35	1.82	0.00	1.46
7	3.57	4.62	2.85	3.23	4.11	3.82	2.90	3.04
8	7.49	8.32	7.25	8.12	8.86	9.72	10.03	11.01
9	11.24	10.98	10.23	13.32	14.48	15.21	16.87	17.54
10	20.13	21.43	22.54	22.01	23.54	24.03	26.65	25.64

The response coupling rate in this experiment refers to a case where learning is performed on each agent and the result learned by *MainServer* does not reach normally. When the number of nodes constituting the agent is 5, a normal response can be obtained, but as the number of nodes increases, the response defect rate increases, making it impossible to guarantee desired information. In addition, it can be seen that the response defect rate increases as the number of agents constituting the system and the number of nodes within the agent increases. The reason these results were derived is that, in the same vein as the experiment performed in Figure 4, it takes a lot of time for step-by-step adjustment of the agent system.

5. Conclusion

In this paper, we proposed a cluster-based multi-agent that applied reinforcement learning in a distributed cloud environment. In Figure 4, a real dataset was applied to the proposal agent, resulting in improved performance compared to the existing system. This demonstrates that clustering multiple nodes and performing distributed learning on data sets can perform data analysis more efficiently than traditional systems when data volumes are large. In Table 1, as the number of nodes in the agent increases, the response failure rate increases. This shows that as the number of clusters and the amount of data to be analyzed increases, the execution time for different nodes to return results varies depending on the operational state of the resource. Therefore, there is a need for a follow-up study on the efficient resource management method of the node in the agent.

Acknowledgement

This work is financially supported by Korea Ministry of Environment(MOE) Graduate School specialized in Integrated Pollution Prevention and Control Project.

References

[1] K. Al-Gumaei, A. Müller, J. N. Weskamp, C. S. Longo, F. Pethig and S. Windmann, "Scalable Analytics Platform for Machine Learning in Smart Production Systems",*2019 24th IEEE International Conference o n Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1155-1162, DOI: https://doi.org/10.1109/ETFA.2019.8869075

[2] Byung-Hyeon Yoo, Debrani Devi, Hyeon-Woo Kim, Hwa-Jeon Song, Kyung-Moon Park, and Seong-Won Lee, “A Survey on Recent Advances in Multi-Agent Reinforcement Learning,” Electronics and Telecomm unications Trends, vol. 35, no. 6, pp. 137–149, Dec. 2020. DOI: https://doi.org/10.22648/ETRI.2020.J.350614

[3] Kim, Bum-Kyu, Hong-Joo Yoon, and Jun Ho Lee. "A Study on the Distribution of Cold Water Occurrenc e using K-Means Clustering." The Journal of the Korea institute of electronic communication sciences 16. 2, pp.371-378, 2021. DOI: https://doi.org/10.13067/JKIECS.2021.16.2.371

[4] M. Tan, “Multi-agent reinforcement learning: Independent vs. cooperative agents.” in Proc. of the Tenth In ternational Conference on Machine Learning (ICML), pp.330-337, 1993. DOI: https://doi.org/10.1016/B978-1-55860-307-3.50049-6

[5] C. Watkins, “Learning from delayed rewards,” Ph.D. Thesis, University of Cambridge England, 1989

[6] V. Mnih, et al., “Human-level control through deep reinforcement learning,” Nature, pp.529–533, 201 DOI: https://doi.org/10.1038/nature14236

[7] J. N. Foerster, et al., “Stabilising experience replay for deep multi-agent reinforcement learning” in Procee dings of The 34th International Conference on Machine Learning (ICML), pp.1146-1155, 2017 DOI: https://doi.org/10.48550/arXiv.1702.08887

[8] Fei Chen and Wei Ren, "On the Control of Multi-Agent Systems: A Survey", Foundations and Tren ds® in Systems and Control: Vol. 6: No. 4, pp 339-499, 2019 DOI: http://dx.doi.org/10.1561/2600000019