Development of Human-level Decision Making Algorithm for NPPs through Deep Neural Networks : Conceptual Approach

Seung Geun Kim[a], Poong Hyun Seong[a*]
[a]*Department of Nuclear and Quantum Engineering, KAIST, 291, Daehak-ro, Yuseong-gu, Daejeon, 34141, Republic of Korea*
[*]*Corresponding author: phseong1@kaist.ac.kr*

1. Introduction

As human operators perform significant roles in nuclear power plants (NPPs), their contribution to the NPP safety is also considerable. Accordingly, many kinds of studies have been conducted for the reduction of human errors, and NPP safety in advance.

One of the strategies for reducing human errors is aiding human operators to work more efficiently and effectively. Studies on human-system interfaces (HSIs) and procedures are affiliated to this strategy. The other approach is shifting task subjects from human operators to machines. Development of operation support systems and automation systems are the representative examples.

Development of operation support systems and automation systems are closely related to machine learning field. However, since it is hard to achieve human-level delicacy and flexibility for complex tasks with conventional machine learning technologies, only operation support systems with simple purposes were developed and high-level automation related studies were not actively conducted.

With the dramatic growth on computing power and overflow of data, machine learning (especially deep neural network) is one of the most spotlighted research fields on recent years. Especially, it is well-known that deep neural networks perform overwhelmingly better than conventional machine learning algorithms in pattern (e.g. images) recognition [1], natural language processing [2], and many other fields.

Not only in data processing, recent technologies also facilitated the development of human-level decision making algorithms. In 2015, by the combination of convolutional neural network (CNN) with the concept of reinforcement learning (RL), artificial intelligence (AI) that self-learns to play various Atari 2800 games (e.g. breakthrough, pong, etc.) was developed, and surpassed the human-level playing in many games [3]. More surprisingly, the game of Go (i.e. Baduk) playing AI 'AlphaGo' was developed by 'Google Deepmind' in the next year, which defeated Fan-Hui (European Go champion) as 5:0, and Se-dol Lee (World Go champion) as 4:1. [4].

In the cases of other board games such as checker and chess, 'CHINOOK' defeated Marion Tinsley (World checker champion) in 1994 [5], and 'Deep Blue' defeated Garry Kasparov (World chess champion) as 2:1 (with 3 draw games) in 1997 [6] by using conventional methodologies with the investment of enormous amount of computing resources. However, unlike many other board games, the game of Go was considered as one of the most challenging area in machine learning field since its search space was too large (i.e. too many number of cases) for conventional methodologies although the computing power has been advanced dramatically. In this point of view, AlphaGo's victory against human champion has proven that recent paradigm of machine learning can perform well, even better than human in highly complex tasks.

For the further reduction of human errors in NPPs and automated NPP design, it is inevitable to develop high-level operation support systems and automation systems. Hence, continuous studies on the NPP application of new machine learning paradigm are essential.

The ultimate goal of this research is to develop the algorithm which conducts automated high-level decision making during the emergency situation of NPPs. In this paper, the concepts applied to the development of desired algorithm were introduced with the expected difficulties and suggested solutions.

2. General Concepts and Related Works

In this section, concepts for decision making algorithm are briefly described, and detailed steps for development are also explained.

2.1 Concept Outline

Since the ultimate goal of the agent (i.e. decision making algorithm) is to decide proper actions for transient mitigation, it is necessary to understand the general processes of decision making.

From the 1980s, many kinds of decision making models were suggested [7-9]. Although there are some minor differences, processes can be briefly summarized as followings; definition of goal, gathering information about current situation and possible options, evaluation of the expected consequences, option selection and application, feedback.

According to the steps of decision making, the agent should acquire current state's information first (goal is fixed as transient mitigation), and then search for proper action among many kinds of possible actions through the evaluation of various scenarios.

The most naïve and intuitive method for the implementation of searching process to AI is brute-force

search (i.e. exhaustive search) method, which simulates for every possible action until simulation ends. However, this approach is not feasible when the search space is huge just like as the game of Go case or NPP case. Owing to limited computing resources, breadth (i.e. number of action candidates) and depth (state evaluation ahead of time) of simulation should be reduced for practical implementation in such cases.

To reduce the breadth of simulation, policy network which deduces the conditional probabilities of choosing specific actions under the specific states is introduced. With the well-trained policy network, unrelated or inappropriate action candidates are excluded, so that the breadth of simulation can be reduced.

To reduce the depth of simulation, value network which deduces the expected outcomes under the specific states and policy is introduced. Well-trained value network can rationally evaluate the states without simulation till the end, so that the depth of simulation can be reduced.

Based on these two kinds of networks, the agent actually searches for optimal actions through Monte-Carlo tree search (MCTS) method, which is a heuristic search algorithm for decision processes.

Most of concepts that described in this paper were already suggested by other studies. However, it is not easy to apply these concepts at the same manner since nuclear field has many discriminative features. Accordingly, it is necessary to consider the distinct characteristics of nuclear field and adopt different methods, although the similar concepts are shared.

2.2 Preparing Simulation Environment

Unlike many other fields of machine learning applications, it is nearly impossible to acquire actual data of NPP emergency situations. Also, due to the extremely complex environment of NPP systems, trade-off between accuracy, simulation time and real-time intervention such as operator control action always exists for the simulation tools. Accordingly, the usage of proper simulation tools for acquisition of data is essential.

In this study, it is planned to use MARS (Multi-dimensional analysis of reactor safety) code as simulation tool. MARS is widely used simulation code developed by KAERI (Korea atomic energy research institute), and it is able to consider sufficiently detailed features of NPPs according to the input model.

One of the biggest drawback of MARS (and most of the simulation codes) is that real-time interventions such as operator control actions are hard to simulate. In order to consider interventions, following alternative steps are needed; stopping simulation before the intervention, revising the simulation input file, and restarting simulation with revised input file.

Automated real-time intervention is essential for RL in this study. Therefore, it is required to prepare self-simulation environment and automated input file generation (with considering intervention) program which currently does not exist.

2.3 Supervised Learning of Policy Network

As briefly described in section 2.1, policy network is for the modeling of posterior actions under specific states. In mathematical words, the policy is the set of conditional probabilities of specific action a_i ($i \leq n$, $\boldsymbol{n}$ is the number of total possible actions) under specific state s_j ($j \leq m$, $\boldsymbol{m}$ is the number of total possible states) which can be represented as following equation.

$$P(i, j) = p(a_i \mid s_j) \quad (1)$$

Where $\boldsymbol{P}$ is policy ($\boldsymbol{n}$ by $\boldsymbol{m}$ matrix), and $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{n}$, $\boldsymbol{m}$ are natural numbers.

For the first step of the algorithm development, training of supervised learning (SL) policy network is conducted based on the procedures in order to mimic the policies of operators (or procedures). Various procedures are analyzed to identify state-action relations, and situations represented in the procedures are imitated by simulation environment.

However, different to most other implementations, only limited number of paths is included in procedures which lead to extreme values of conditional probabilities (i.e. $p(a \mid s)$ close to zero or one) in most cases. Therefore, SL policy network alone is expected to be meaningless according to the goal since it would be almost same with conventional rule-based expert systems (if $\boldsymbol{A}$ then $\boldsymbol{B}$ systems). Still, it is desirable to conduct SL before RL since SL can provide brief direction of learning and significantly reduces the training time.

2.4 Reinforcement Learning of Policy Network

In order to strengthen the policy network, RL is conducted by utilizing self-simulation environment. As large amount of randomized data can be obtained from self-simulations, parameters of policy network also can be fine-tuned by using these data.

However, since SL policy network is expected to have extreme values of conditional probabilities, it is not appropriate to start training RL policy network directly from SL policy network (i.e. SL policy network is not flexible enough for training RL policy network). Instead, probability of random action selection is assigned in order to provide flexibility in making decisions (i.e. to avoid excessive dependency on procedures). The random action selection probability declines as training proceeds to avoid divergence and reduce training time.

If the initial random action selection probability is high, the output of RL policy network is more likely to less dependent to procedure based policies. On the contrary, if the initial random action selection probability is low, the output is likely to more dependent to procedures. This value is needed to be optimized by trial-and-error method, to balance the dependency on procedures.

2.5 Reinforcement Learning of Value Network

As briefly described in section 2.1, value network is for the prediction of outcomes under specific states and policy. In mathematical words, the value function represents the expected value of final outcome z_t under the state at time step t (s_t) and predicted future actions ($a_{t...T}$) according to the policy ($t \leq T$, T is the time step when the simulation ends) which can be represented as following equation.

$$V(s) = E[z_t \mid s_t = s, a_{t...T} \sim P] \quad (2)$$

Where V is value function and t, T are natural numbers.

After the training of RL policy network, value network is also trained by using self-simulation data. From the outcomes of self-simulation data, value function can be obtained and optimized.

However, there is possibility of over-fitting since successive actions are usually strongly correlated, which can be differed by single action but the target is to predict only final outcome. To avoid this problem, it is needed to generate simulation data which includes not only strongly correlated actions but also uncorrelated (or less correlated) actions.

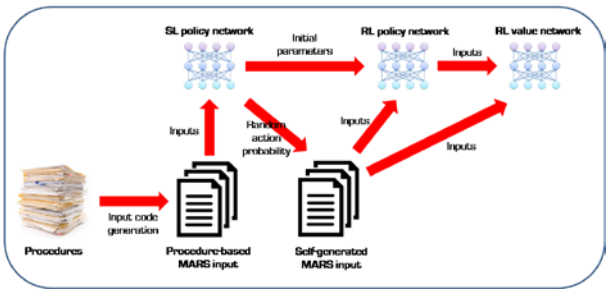

Fig. 1. Schematic of the training processes of policy and value networks.

2.6 Monte Carlo Tree Search

The actual selection of action is conducted by the methodology of MCTS, which combines the trained policy and value networks. The MCTS module searches and evaluates the most promising actions among vast search tree, and expands the search space when it is necessary. MCTS includes following four steps; selection, expansion, evaluation (simulation), and backup (back-propagation).

a. Selection : Starting from the root node (current state), edge (future state) with high priority is selected.

b. Expansion : If the simulation does not ends at the edge, new child nodes can be created according to the policy network. One of them is selected as like selection step.

c. Evaluation (simulation) : From the created and selected child node, value network predicts the final outcome. Independently, simple simulation according to the policy network can be conducted to deduce the final outcome.

d. Backup (back-propagation) : From the evaluation result, parameters of all parent nodes are updated.

With sufficient numbers of updates, MCTS module can reduce the number of cases by eliminating undesirable nodes, and be more accurate.

2.7 Constraints and Related Works

Overall structure of suggested algorithm is quite similar to the structure of AlphaGo. However, because of the huge gap between the nature of the game of Go and NPP, there are many constraints during the application of suggested concepts. Thus, additional concepts to resolve these constraints are needed to be applied.

This section describes the constraints and related works although some of them were already described in previous sections, since there are many important features.

a. Dynamic nature of NPP systems : State of NPP changes continuously, and implemented actions do not give immediate feedbacks. This problem can be solved by setting up proper time-gap between actions.

b. Restricted simulation environment : Real-time intervention (i.e. implementation of actions) is not easy for most of simulation codes, and self-simulation environment is currently unavailable. With the sequences of MARS code simulation stopping and restarting, real-time intervention problem can be solved. Development of self-simulation environment should be progressed by automated input file generation.

c. Unavailability of human-operated data : during the SL, it is impossible to mimic human operator's decisions directly from real data since human-operated data is almost unavailable. This problem can be solved by generation of procedure-based simulation data, with the assumption that most of human operators follow the instructions in procedures accurately.

d. Extreme conditional probability values from SL policy network : SL policy network trained with procedure-based simulation data would have extreme values of conditional probabilities, since instructions in procedures are well-defined in most cases. This problem can be solved by assignment of random action selection probability to assign flexibility in making decisions.

e. Importance of mitigation processes : Desirability of mitigation processes cannot be represented, although the mitigation paths are also important. By setting up proper reward function which considers not only final mitigation results but also mitigation processes, this problem can be solved.

3. Conclusions

As one of the efforts for reducing human error in NPPs and technical advance toward automation, the ultimate goal of this research is to develop human-level decision making algorithm for NPPs during emergency situations. The concepts of SL, RL, policy network, value network, and MCTS, which were applied to decision making algorithm for other fields are introduced and combined with nuclear field specifications. Since the research is currently at the conceptual stage, more research is warranted.

There are several expected limitations of this study. Firsty, similar to most of other machine learning related studies in nuclear field, heavy reliance on simulation tool is unavoidable. As a result, the performance of the developed algorithm would be heavily affected by the accuracy of simulation tool. Moreover, since it is planned to conduct research about one kind of DBA (design basis accident), additional studies to cover the whole situations in NPPs might be needed.

The results of this study can be applied for the improvement of procedures or safety related systems by comparing with developed algorithm's strategies and conventional procedure-based accident mitigation strategies. Also, with the concept of reinforcement learning, more precise algorithm can be expected since it can continuously tuned by based on additional data.

If the study progresses to the practical application stage, it is expected that human induced errors can be reduced by deploying actively-aiding support system based on developed algorithm. Furthermore, this study can be applied as original technology for NPP automation.

REFERENCES

[1] D. Ciresan, U. Meier, J. Schmidhuber, "Multi-column Deep Neural Networks for Image Classification", Technical Report, No. IDSIA-04-12, 2012

[2] F.A. Gers, J. Schmidhuber, "LTSM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages", IEEE Transactions on Neural Networks, Vol. 12, No. 6, p. 1333-1340, 2001

[3] V. Mnih et al., "Human-level control through deep reinforcement learning", Nature, Vol. 518, p. 529-533, 2015.2

[4] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search", Nature, Vol. 529, p.484-489, 2016.1

[5] J. Schaeffer et al, "CHINOOK the world man-machine checkers champion", AI Magazine, Vol. 17, No. 1, p. 21-29, 1996.

[6] M. Campbell et al, "Deep Blue", Artificial Intelligence, Vol. 134, p. 57-83, 2002.

[7] B. Jonathan, Rex V. Brown, "Teaching decision making to adolescents", Hillsdale, NJ: Lawrence Erlbaum Associates, p. 61-78, ISBN 0805804978, 1991.

[8] K. L. Guo, "DECIDE: a decision-making model for more effective decision making by health care managers", The Health Care Manager, Vol. 27, No. 2, p.118-127, 2008.6

[9] J. Pijanowski, "The role of learning theory in building effective college ethics curricula", Journal of College and Character, Vol. 10, No. 3, p.1-13, 2009.2