Machine learning approach for approximation of thermal-hydraulic code using k-NN

Seunghyoung Ryu [a*], Hyeonmin Kim [b], Seung Geun Kim [a], Jaehyun Cho [b*]
[a]*Artificial Intelligence Application & Strategy team/* [b]*Risk Assessment Research Team,*
Korea Atomic Energy Research Institute, 111, Daedeok-daero 989beon-gil, Daejeon, 34057, South Korea
[*]*Corresponding author: chojh@kaeri.re.kr*

1. Introduction

Digital twin is one of recent emerging technology that can be utilized for various nuclear engineering applications. A digital twin model is a representation of real system in virtual space which could interacts with the original entity in real time. Furthermore, one of distinguishing characteristic of digital twin is the ability of planning based on rapid simulation of future events. Based on the forecasting of future scenario, we could respond to events and problems more effectively; thus, it helps to aid in improving nuclear safety.

In nuclear engineering, thermal hydraulic (TH) code simulates TH dynamics of nuclear reactor of given scenario. Based on the simulation, we could anticipate the outcome of an accident or abnormal events in advance. However, due to the high complexity in computation of TH dynamics, simulation through TH code run requires long computation time ranging from minutes to hours. Therefore, a fast running model of TH code is necessary to develop a digital twin model for nuclear power plants. There are two types of digital twin models: a physics-based model utilizing reduced order modeling and a data-driven model based on machine learning technology.

In this study, we focus on the latter approach. In the previous literature, a data-driven fast-running model was developed based on deep learning [1, 2]. In [1], a conditional autoencoder network was utilized by training massive TH code run datasets. In [2], recurrent neural network based on long short-term memory was investigated considering the time-series characteristic of TH code. Following the problem formulation described in [2], we utilize *k*-nearest neighbor (*k*-NN) methods [3] which is a simple and robust machine learning method. Based on experimental results with *k*-NN, we check the feasibility of using data-driven methods in the development of the fast-running model.

2. Problem Formulation

Based on the work of [2], we formulate the operation of TH code as follows. Let $\mathrm{x} = [x_1, x_2, \dots, x_d] \in \mathbb{R}^d$ be the vector that defines the accident scenario which is also used as an input of TH code run. Then TH code calculates TH dynamics and outputs the value of target variables in time-series. This can be denoted as $\mathrm{Y} \in \mathbb{R}^{T \times D}$ where T is the length of output and D is the number of target variables. Simulation terminates when it meets ending conditions; thus, the length of the output can be differ according to the input data. For the training data $D_{train} = \{\mathrm{x}_i, Y_i\}$, and test data $D_{test} = \{\mathrm{x}_j, Y_j\}$, data-driven fast running model estimates Y_j from x_j by learning internal relations in D_{train}; the original code computes the output according to mathematically defined multiple equations.

3. *k*-NN-based approximation

k-NN is a well-known machine learning methodology that can be used in both regression and classification problems. Considering fast running model is regression model, and *k*-NN-based model can be described as follows. For a distance metric between two samples $d(x, y)$ and the number of nearest neighbors *k*, *k*-NN finds *k* neighbors of input x_j which has lowest distance $\mathrm{d}(\mathrm{x}_j, \mathrm{x}_i)$ among $\mathrm{x}_i \in D_{train}$. If we denote N_j as a set of indices for nearest neighbors of x_j where $|N_j| = k$, the estimation of Y_j is the average of $\{Y_i\}$ for $i \in N_j$. As a result, *k*-NN model maintains reference database of TH code run results and find the best matching accident scenarios to output the result of unknown accident scenario or new sample. To check the feasibility of using *k*-NN as a surrogate model, we compared the prediction error of *k*-NN according to various *k* and error metrics. In doing this, Euclidean distance is used for distance metric, and details of experimental setup are given in following subsections.

3.1 Dataset configuration

First, we obtain TH code result data to develop data-driven model. The used TH code is modular accident analysis program (MAAP) which simulates severe accident sequence for a risk analysis. We accumulate TH code result data from four accident scenarios as follows: total loss of component cooling water (TLOCCW); TLOCC, small loss of coolant accident (SLOCA) with diverse high-pressure pump injection and auxiliary pump injection; SLOCA-2, SLOCA with changing severe accident guidance (SAG) 2 and 3; SLOCA-29, medium loss of coolant accident (MLOCA); MLOCA. For each scenario, we obtain results of 2,000 code run with different safety system parameters that composes input vector of MAAP. For example, input vector of scenario TLOCCW consists of SAG2 (depressurization delay time and rated mass flow rate for one pressurizer of relief valve) and SAG3 (in-vessel injection volume flow rate and in-vessel injection dealy time), etc. Based on the truncated normal distribution, we randomly sampled parameter values and create a set of input vectors. Among the accumulated results, unexpectedly

terminated scenarios are omitted, thus final data set contains 7,631 TH code results. In addition, we select four output variables: PPS, PPSTRB, FREL(1), and FREL(2).

Next, we split training and test data based on stratified K-fold cross validation. The number of splits, K, is set to 5; four subsets are used as D_{train} and the remaining subset is used as D_{test}. The proportion of each scenario constituting a subset follows that of the entire dataset. Since each subset can be used as D_{test}, we report the average and median of prediction errors on different test sets.

3.2 Experimental setup

We compare the result on different $k \in \{1,3,5,7,9\}$ in terms of difference between actual simulation output and estimated result by the *k* neighbors. In doing this, we calculate mean squared error (MSE), mean absolute percentage error (MAPE), normalized absolute error (NAE), and symmetric mean absolute percentage error (SMAPE), respectively. Instead of using true value at denominator in MAPE, NAE is normalized by the mean of actual results, and SMAPE is normalized by sum of actual and predicted values.

3.3 Experimental results

We report the average error results of FREL(1) and PPS in Tables I and II, respectively. For corresponding k, the upper row describes the average value and lower row indicates median. Except for MSE, other error metric can be interpreted as percentage values. According to the result, approximation by *k*-NN shows different aspect from the errors of typical regression problems. We expect that the excessive value of MAPE is due to the scale issue of the target variable. True values near zero induce excessive percentage errors thus SMAPE exhibits more interpretable results compare to other metrics.

Table I: Error table for FREL(1)

k	MSE	NAE	MAPE	SMAPE
1	0.04	735k	17k	16.2
	1.7E-08	5.7	8.5	4.7
3	0.04	737k	17k	16.2
	1.7E-08	5.7	8.7	4.7
5	0.03	513k	12k	17.2
	2.5E-08	6.8	12.0	5.5
7	0.03	415k	12k	18.7
	3.5E-08	7.9	15.3	6.3
9	0.03	359k	12k	18.6
	4.1E-08	8.4	17.5	6.3

Table II: Error table for PPS

k	MSE	NAE	MAPE	SMAPE
1	3.5E+12	47.4	80.6	15.2
	8.5E+11	28.8	27.3	10.3
3	3.4E+12	45.0	78.3	15.2
	7.4E+11	25.7	25.7	10.3
5	2.5E+12	44.9	76.5	15.1
	8.1E+11	26.1	30.2	10.1
7	2.2E+12	44.7	75.8	15.3
	8.0E+11	26.6	33.8	10.5
9	2.1E+12	44.8	75.8	15.4
	7.7E+11	27.0	34.2	10.6

In addition, we observed that the median of error is a lot lower than the mean of error. For example, median of MAPE for PPS ranges 25~34 and median of MAPE fo FREL(1) ranges 8~17. This results indicates that there are outliers inducing high approximation error.

4. Conclusions

In this study, we analyze the result of using *k*-NN for approximation of TH code results. *k*-NN is simple and easy to utilize compared to recent machine learning methods. However, the error results shows that there exists outliers inducing excessive approximation errors. In other words, the results can be quite different , even if the inputs are similar. Therefore more robust approach for fast approximation is required. To resolve this, deep learning or *k*-NN-deep learning hybrid models can be developed for further research, and the result of *k*-NN can be used for a bench mark results.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Science, ICT, and Future Planning of the Republic of Korea and the National Research Foundation of Korea (NRF-2020M2C9A1061638).

REFERENCES

[1] Kim, Hyeonmin, Jaehyun Cho, and Jinkyun Park. "Application of a deep learning technique to the development of a fast accident scenario identifier." *IEEE Access* 8 (2020): 177363-177373.

[2] Ryu, Seunghyoung, et al. "Probabilistic deep learning model as a tool for supporting the fast simulation of a thermal–hydraulic code." *Expert Systems with Applications* 200 (2022): 116966.

[3] Fix, Evelyn, and Joseph Lawson Hodges. "Discriminatory analysis. Nonparametric discrimination: Consistency properties." International Statistical Review/Revue Internationale de Statistique 57.3 (1989): 238-247.