Electronic Preprint for Journal of Information Processing Vol.29

Regular Paper

Flexible Imputation Method for Sensor Data based on Programming by Example: APREP-S

Hiroko Nagashima[1,a)] Yuka Kato[1,b)]

Received: May 11, 2020, Accepted: November 5, 2020

Abstract: The quantity of data available for analysis, including data collected by sensors and wearable devices, has been increasing hugely. However, to obtain accurate analysis results, data pre-processing such as outlier detection, handling of missing data, and preparing data recorded by different measuring instruments in different units, is essential. Considering that the pre-processing task consumes 80% of analyst resources, we previously proposed a method to address this problem. The method integrates machine learning based on Bayesian inference with human knowledge by using programming by example approach. However, in situations in which the process of generating the model and the process of updating the model are executed at different sites, the previous method is problematic in two ways: 1) all sites have to use the same features defined when the model is generated, and 2) a helpful process to generate new training data from features without using inference data when updating the model, is not available. This prompted us to propose APREP-S, which has flexible feature processes and a process for updating the model using a clustering method. We evaluate the accuracy of the imputation and the similarity of the trends by comparing APREP-S with the original data and other existing methods. The results show that APREP-S can return the most optimal methods with both accuracy and similarity.

Keywords: pre-processing, sensor data, PBE (programming by example), Bayesian inference, Internet of Things

1. Introduction

Recent years have seen an increase in the quantity and kinds of data available for analysis, including sensor data and wearable device data. Examples of analysis using data collected by sensors are customer trend analysis for shopping malls, autonomous behavior analysis for robots, and production management of smart factories. Because data analysis can support the knowledge and experience of senior experts, data analysis focuses on improving the productivity of non-experts in a factory and managing the overall utilization rate of the factory. The analysis of sensor data, in particular, tends to involve outliers and missing data rather than typical time series data because data are received over a network and because sensors are often powered by batteries. This requires the analyst to use pre-processing, such as replacing or deleting missing data, checking the time interval, or transforming the data format. The pre-processing step required for typical project analysis consumes 80% of the project resources [1]. Moreover, the number of business analysts is insufficient because an analyst needs to understand the particular business to ensure optimal pre-processing. This business understanding refers to the definition of analysis goals, criteria, and related knowledge.

This led us to previously propose APREP-S (Automated Pre-processing for sensor data) [3], [4] to reduce the resources that need to be devoted to pre-processing. This is a method that integrates machine learning based on Bayesian inference with human knowledge using programming by example (PBE) approach - details of the proposed APREP-S are provided in Section 3, and PBE approach is explained in Section 1.2. The previous APREP-S already contains a support function to enable an analyst to update the model, which can be accomplished by using the features obtained from inference; however, two problems remain: 1) all sites have to use the same features defined when the model is generated, and 2) a helpful process to generate new training data from features without using inference data when updating the model, is not available. In other words, the previous method is less flexible in situations in which the process of generating the model and that of updating the model are executed at different sites. In addition, the analyst needs to devise a rule to generate training data for updates. In the business environment, an IT engineer in the IT department is often responsible for generating the common model and distributing it to project sites. Although we often utilize the network and connect IT systems such as Industry 4.0, the number of IT engineers to generate a site-specific model is insufficient [5]. Therefore, the project expert based on the project site needs a method to update the distributed model to ensure it corresponds with site-specific features. An example is provided in Section 1.1. In this paper, the term "IT engineer" refers to an engineer with machine learning skills and who is based in a model generating site such as an IT department, and "project expert" refers to an experienced person with project knowledge, i.e., business understanding, who is based at the project site. The "project expert" has less knowledge of machine learning than the IT engineer. Our aim was to enhance the previous APREP-S by solving these problems. Specifically, we propose the workflow that would

[1] Tokyo Woman's Christian University, Suginami, Tokyo 167–8585, Japan
[a)] h18m001@cis.twcu.ac.jp
[b)] yuka@lab.twcu.ac.jp

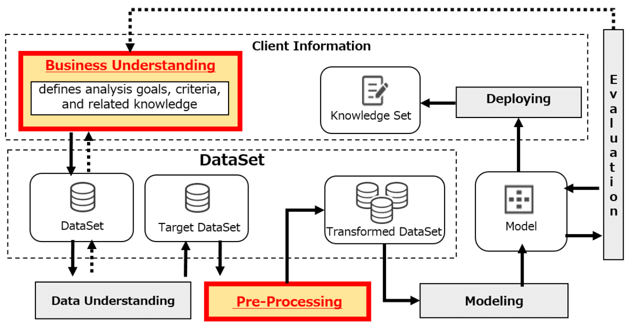


Fig. 1 Data Mining Framework (the drawing is based on Ref. [2] Fig.5).

enable site-specific features to be selected at each project site, and a method to perform classification by using a clustering method.

The contributions of this paper are as follows.

- Propose flexible feature processes by enhancing the previous APREP-S to provide a method that is available even if the site at which the model is generated differs from the site at which the model is updated
- Propose a clustering process to assist the analyst in the model updating phase, even when data with a trend similar to the inference data are not prepared
- Verify APREP-S with three existing imputation methods to assess its effectiveness against outliers and missing data from two perspectives: the accuracy from the sum-of-squared errors and the similarity of the trends to the original data.

1.1 Business Understanding and Model Update

An understanding of the business is essential knowledge for suitable pre-processing. A well-known data mining framework is the cross industry standard process for data mining (CRISP-DM) [6]. Based thereupon, we proposed a data mining framework named “APREP-DM (automated pre-processing for data mining) [2].” An overview of the APREP-DM framework is shown in **Fig. 1**. Note that business understanding is located before pre-processing.

In the business environment, IT engineers would work in the IT department and project experts would work at the project site such as in marketing or manufacturing; thus, an IT engineer and a project expert are not standard to work at the same site. Although it would be the best approach to assign an IT engineer to every project and every site, this would be difficult because the number of IT engineers is limited. A Cyber-Physical System such as Industry 4.0 [7] would enable the project expert to use machine learning models at the project site because the IT engineer would be able to generate the machine learning models and deliver them over the network to other sites. However, IT engineers are too busy to update and maintain the model on time to adjust it to reflect the different features of each site, such as the site climate, behavior of employees on the floor, and project rules. Therefore, a project expert would need to be able to maintain the model without help from an IT engineer. Hence, we need to find a way to enable the model to be updated by project experts. An example based on a factory is shown in **Fig. 2**. An IT engineer generates the flexibility model at the model generating site at first, and a project expert updates that model to adjust the features of each

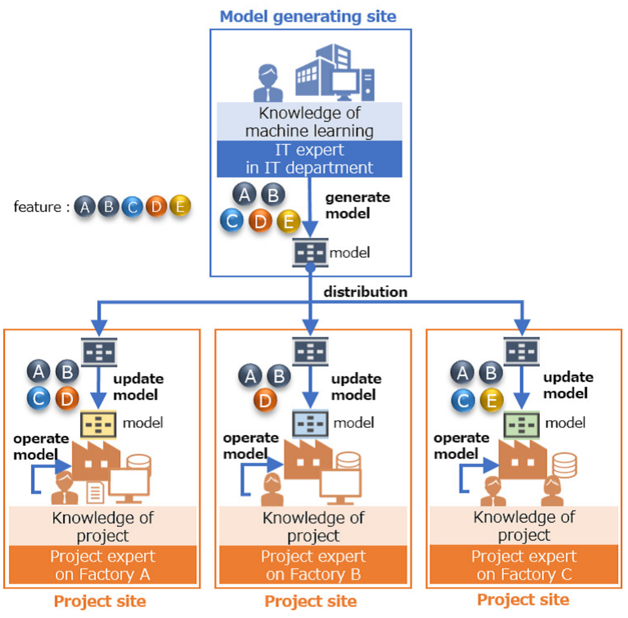
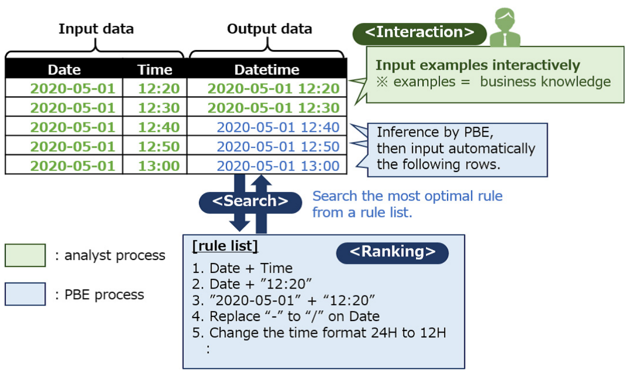


Fig. 2 Example of model generating site and project site.


Input data
Output data
<Interaction>
Input examples interactively
※ examples = business knowledge
Inference by PBE, then input automatically the following rows.
<Search>
Search the most optimal rule from a rule list.
: analyst process
: PBE process
[rule list]
<Ranking>
1. Date + Time
2. Date + ”12:20”
3. ”2020-05-01” + “12:20”
4. Replace “-” to “/” on Date
5. Change the time format 24H to 12H
:

Date	Time	Datetime
2020-05-01	12:20	2020-05-01 12:20
2020-05-01	12:30	2020-05-01 12:30
2020-05-01	12:40	2020-05-01 12:40
2020-05-01	12:50	2020-05-01 12:50
2020-05-01	13:00	2020-05-01 13:00

Fig. 3 Inferring *Datetime* values using PBE.

site at the project site. Specifically, the initial model is generated by using all the features, *A*, *B*, *C*, *D*, and *E*, as input and these might be used at each project site. This model is distributed to each project site, after which the project expert updates the model by using site-specific features as input, e.g., *A*, *B*, *C*, and *D* in Factory A.

1.2 Integration of Human Knowledge and Machine Learning

We used PBE approach for integrating human knowledge into the machine learning model. This approach determines a rule from one or more input examples; then, it infers an output interactively by using the optimal rule. As the quantity of data has increased, PBE approaches, such as FlashFill [8], have become the preferred method for transforming big data.

PBE approach consists of three main processes: 1) a search algorithm that efficiently retrieves matching rules from input data provided by analysts, 2) a ranking program that selects the optimal process from the example input by the analyst, and 3) an interaction model to improve the analyst’s operation. An example of PBE approach is shown in **Fig. 3**. The example is based on a table with three columns: the existing columns are *Date*, and

Time, and the new column is *Datetime*. First, an analyst inputs *Date* as "2020-05-01," *Time* as "12:20," and *Datetime* as "2020-05-01 12:20." This means that the analyst inputs training data of which the output is "2020-05-01 12:20" if the input values are "2020-05-01" and "12:20." After the analyst inputs only the first row, PBE might return "2020-05-01 12:20" because PBE judges that the rule is a fixed data string "2020-05-01 12:20." Next, the analyst inputs new data for which *Date* is "2020-05-01," *Time* is "12:30," and *Datetime* is "2020-05-01 12:30" in the second row. Then, PBE can return "2020-05-01 12:40" if the input *Data* is "2020-05-01" and *Time* is "12:40," because the rule is updated to specify that the data is not a fixed string, but returns a value stored in the *Date* column. The list of rules with a ranking is updated as the analyst interactively inputs new data. Thus, the analyst can interactively insert human knowledge, and the subsequent lines are automatically filled with inferential data.

The overall structure of the paper takes the form of five sections, including this introduction. Section 2 presents a brief overview of the relevant literature. Section 3 explains our proposed APREP-S and includes an overview of the previous method. Section 4 evaluates APREP-S, and contains the results and a discussion of the evaluation. The main conclusion is described in Section 5.

2. Related Work

Various imputation approaches exist. We focus on three methods: calculating from the rules, inferring from a time-series trend, and inferring training data using machine learning. These approaches are described in the following sections.

2.1 Imputation Method using a Rule

Well-known imputation methods for calculating from the rules are the "Listwise method" (also known as the "complete-case method"), "multi imputation," and "single imputation."

"Listwise method" deletes all rows of missing data. Although, it is one of the most well-known methods for handling missing data, the quantity of data decreases. "Multi imputation" is a method for estimating the standard deviation for the population of the missing data using the provisional imputation value generated from the distribution of the missing value of the target imputation data. For improved accuracy, the number of sets is suggested to be 20 or more [9]. Furthermore, analysts commonly do not have sufficient data in a practical business situation. This makes it difficult for analysts to use the multi-imputation method [10]. "Single imputation" can calculate all the imputation values automatically if a rule is defined in advance. However, it cannot change the rules of the imputation values. Therefore, it lacks the scope of customization. One of the most familiar methods of single imputation is "spline interpolation" [11], which enables smooth curves to be drawn through equally spaced data. The method fits a polynomial to the specified data points and obtains a curve that passes through all the specified points.

2.2 Inference of Time-Series Analysis (GAM)

The "generalized additive model" (GAM) is an established method for time-series analysis that generates a nonlinear function by adding multiple functions [12]. For time-series analysis, it is essential to apply nonlinear trends to account for the periodicity and variance in human behavior, seasonality, and time-sensitive trends. One of the most familiar methods based on GAM is Prophet, which is a regression model for inferring time-series data [13]. Prophet is expressed as

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t, \tag{1}$$

and the equation is linear regression if $g(t)$, $s(t)$, and $h(t)$ are all linear functions. Particularly, t is the target time of imputation, $y(t)$ is the target imputation value, $g(t)$ is a non-periodic trend-modeling function for time-series data, $s(t)$ is a periodic function for seasonally changing data, $h(t)$ is a function that accounts for the effects of holidays, and ϵ_t represents any idiosyncratic changes that are not accommodated by the model. In Prophet, it is normally distributed as a parametric assumption.

2.3 Inference of Machine Learning (RNN)

A recurrent neural network (RNN) is an extension of a conventional feedforward neural network, and is able to accept a variable-length sequence as input. The RNN handles the variable-length sequence by having a recurrent hidden state whose activation at each time is dependent on that of the previous time. For time-series data, the accuracy of inferences can be improved by considering the input data as part of a series, rather than independently of one another. An important feature of an RNN is that the output of a hidden unit can use the output of the last layer. "Long short-term memory" (LSTM) is an RNN method. LSTM can calculate long term time-series data short learning performance [14].

3. Proposed Method - APREP-S

We previously proposed "APREP-S," which conducts automatic pre-processing by integrating human knowledge with machine learning via the PBE approach [3], [4]. In this study, we enhance the previous APREP-S and propose a workflow to select site-specific features at each project site, and use clustering as the classification method. APREP-S classifies the target imputation area with features inputted by the project expert. We first present an overview of the previous APREP-S and then describe the proposed method in this paper in detail. In this paper, the term "target imputation area" refers to a range that needs continuous imputation. As mentioned in Section 1, the term "IT engineer" signifies an engineer with machine learning skills at the model generating site, and "project expert" refers to an experienced person with project knowledge on the project site. The "project expert" has less knowledge of machine learning than the IT engineer.

3.1 Previous APREP-S

The previous APREP-S contains several imputation methods, and enables an analyst to select an optimal imputation method by checking the imputation value and likelihood of the target imputation area. Three main phases exist: the model training, model operating, and model updating phases. The model training phase is used to generate the initial model. The model updating phase is used to update the APREP-S model. The model operating phase is used for the inference and calculation of the imputation values using the APREP-S model. These three phases of the previous APREP-S are as follows.

In the model training phase, the previous APREP-S defines features on each target imputation area. After that, the previous APREP-S generates the model which has two parameters, α and $\boldsymbol{\beta}$. The APREP-S model is

$$\boldsymbol{y}(x_d) = \alpha + \boldsymbol{\beta}\, x_d \quad (1 \leq d \leq D) \tag{2}$$

where $x_1, x_2, ... \in \boldsymbol{x}$ (x is finite) is a set of features for the APREP-S model, and D is the size of the target imputation area. The parameter α and $\boldsymbol{\beta}$ are Gaussian distributions, and $\boldsymbol{\beta}$ is a Q-by-K matrix, where Q is the number of features and K is the element number of $\boldsymbol{M}$ which is a set of the imputation method defined in previous APREP-S. The previous APREP-S calculates the likelihood based on Bayesian inference, for each imputation method included in the previous APREP-S. Because $\boldsymbol{M}$ is a set of discrete values, $\boldsymbol{p}$ can be calculated from the proportion of the likelihood of each $m_i \in \boldsymbol{M}$. The posterior probability, $p(m_k|\boldsymbol{y})$, is calculated based on Bayesian inference:

$$\begin{aligned} p(m_k|\boldsymbol{y}) &= \frac{p(\boldsymbol{y}|m_k)p(m_k)}{\sum_{i=1}^{K} p(\boldsymbol{y}|m_i)p(m_i)} \\ &= \frac{\exp(\boldsymbol{y}(x_k))}{\sum_{i=1}^{K} \exp(\boldsymbol{y}(x_i))} \end{aligned} \tag{3}$$

where $p(\boldsymbol{y}|m_k)$ is the prior distribution of each method, which can be calculated from training data tr_m in the model training phase and tr_m is a matrix of the selected number of methods. For example, if the number of methods is three, we first select $method_1$ and then select $method_3$, $tr\_m = [[1, 0, 0], [0, 0, 1]]$. The likelihood function is

$$C(\boldsymbol{M}|\boldsymbol{y}) = \prod_{k=1}^{K} (y_k^{u_k}) \tag{4}$$

where u_k denotes the probability of m_k. $p(m_k|\boldsymbol{y})$ is a normalized exponential function derived from $\sum_{i=1}^{K} p(m_i|\boldsymbol{y}) = \sum_{i=1}^{K} u_i = 1$. Hence, the likelihood (Eq. (3)) equals a proportion $\boldsymbol{P}$. $m_k \in \boldsymbol{M}$, satisfies $\boldsymbol{y}$, and this $\boldsymbol{y}$ is also a parameter of the likelihood function of the previous APREP-S. In addition, the previous APREP-S generates models of methods defined inside the previous APREP-S if necessary.

In the model operating phase, the previous APREP-S infers the optimal methods using the model generated in the model training phase. The previous APREP-S calculates the proportion of each method, and selects the method on target imputation area. Last, the previous APREP-S returns the complementary data.

In the model updating phase, the analyst first checks the previously selected method and the value of each imputation are. If the analyst uses a similar approach to obtain the inference data, the previous APREP-S provides an interface for the analyst to easily select a method. This interface has a function to compare the inference values with similar data and data calculated by other imputation methods.

3.2 Proposed Method

We enhance the previous APREP-S to help a project expert to update the APREP-S model in case the IT engineer and project expert are based on different sites. We propose the flexible customization of the model in the model updating phase by clarifying the feature settings in the model training phase. The workflow of the proposed method is shown in **Fig. 4**. We define two types of analyst: an IT engineer and a project expert. The model generating site is where the IT engineer generates the APREP-S model and the other imputation models defined in APREP-S. The project site is where the project expert uses and updates the APREP-S model. We describe their three phases in detail.

The model training phase occurs at the model generating site. The IT engineer inputs training data to APREP-S, and then APREP-S searches the target imputation area by using the same process as the previous APREP-S. Then, APREP-S defines all features that might be used at every site, e.g., the number of target imputation areas, weather data, and activity data of employees. Although certain sites might not have all of these features because sensors may either not have been prepared or it may not be possible to install the sensors because of the construction of buildings, the project expert at the project site can flexibly select the site-specific features from features defined at the model generating site. After defining the features, APREP-S generates the APREP-S model using these features. Other imputation models are also defined in APREP-S. The models generated in the model training phase are used in the model operating phase.

At the project site, the model operating phase and the model updating phase are implemented. The model operating phase is the same as that of the previous APREP-S. APREP-S infers each of the imputation values and the proportion of the likelihood of using the models generated in the model training phase, and sends them to the project expert. In the model updating phase, the project expert first checks the complementary data generated in the model operating phase as an output of APREP-S. Then, the project expert inputs the method data selected in each imputation area and the features for inferring imputation data to APREP-S. An image of this interface is shown in **Fig. 5**. This is an example with four features: weather code, temperature, humidity, and line speed. The number of clusters is three. The first upper part is for inputting the site-specific data into APREP-S. The features upload when the project expert clicks the "upload" button. If the project expert has feature data, the project expert uploads all site-specific feature data. After that, the project expert inputs the number of clusters and clicks the "run" button. Here, "cluster num" refers to the number of clusters APREP-S is required to return. The interface shows the trend of inputted data and shows the chart classified by the cluster method. The project expert selects the optimal method on each cluster on the lower display, where these trends are visualized.

On the other side, APREP-S classifies the features it receives via the interface. APREP-S defines a Hidden Markov Model (HMM) [15], [16] by the clustering method. This is because the HMM rapidly returns the number of clusters, thereby matching the interaction of the PBE process. Although one of the well-known clustering methods is K-Means [17], it cannot process time-series data. Therefore, we utilize HMM as a representative method for clustering time-series data.

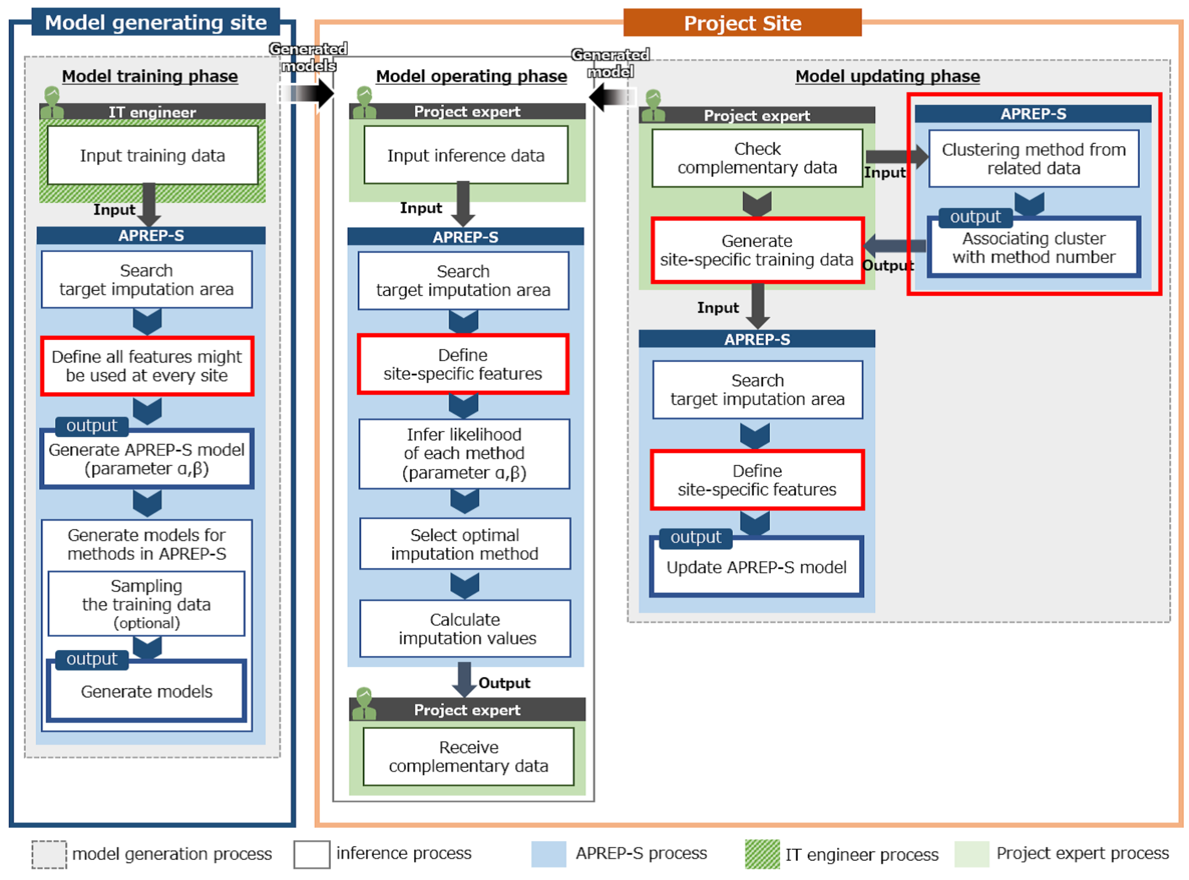
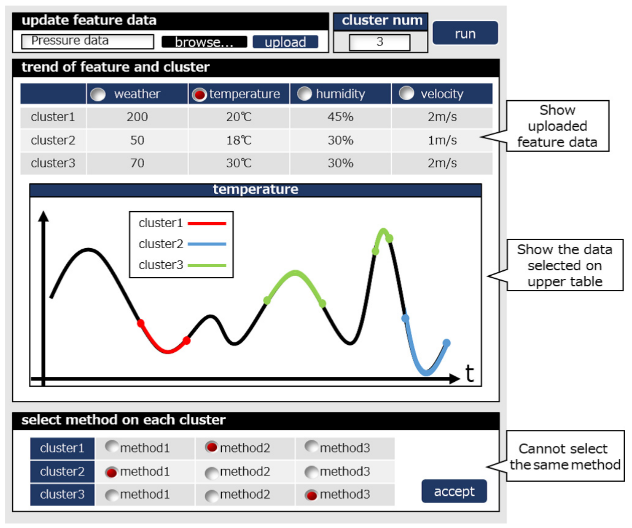

Model generating site
Generated models
Model training phase
IT engineer
Input training data
Input
APREP-S
Search target imputation area
Define all features might be used at every site
output
Generate APREP-S model (parameter α,β)
Generate models for methods in APREP-S
Sampling the training data (optional)
output
Generate models
Project Site
Generated model
Model operating phase
Project expert
Input inference data
Input
APREP-S
Search target imputation area
Define site-specific features
Infer likelihood of each method (parameter α,β)
Select optimal imputation method
Calculate imputation values
Output
Project expert
Receive complementary data
Model updating phase
Project expert
Check complementary data
Input
Generate site-specific training data
Output
APREP-S
Clustering method from related data
output
Associating cluster with method number
Input
APREP-S
Search target imputation area
Define site-specific features
output
Update APREP-S model
model generation process
inference process
APREP-S process
IT engineer process
Project expert process

Fig. 4 workflow of APREP-S: Processes enclosed in a red frame denote the proposed processes, those in a blue frame denote a model generating site, and those in an orange frame denote a project site.


update feature data
Pressure data
browse...
upload
cluster num
3
run
trend of feature and cluster
Show uploaded feature data
temperature
cluster1
cluster2
cluster3
t
Show the data selected on upper table
select method on each cluster
Cannot select the same method
accept

	weather	temperature	humidity	velocity
cluster1	200	20℃	45%	2m/s
cluster2	50	18℃	30%	1m/s
cluster3	70	30℃	30%	2m/s

cluster1	method1	● method2	method3
cluster2	● method1	method2	method3
cluster3	method1	method2	● method3

Fig. 5 Image of interface for the clustering of method: the red solid circle next to the name of an item and method denotes that they have been selected. The chart area displays selected data listed in the upper table. Temperature data are currently selected for display.

HMM is a statistical Markov model in which the system being modeled is assumed to be a Markov process, and HMM has a sequence of observable variables $\mathbf{X}$ and a sequence of internal hidden states $\mathbf{Z}$. $\mathbf{X}$ is generated by $\mathbf{Z}$. The model uses three parameters: the start probability vector $\boldsymbol{\pi}$, a transition probability matrix $\mathbf{A}$, and the emission probability of an observation, which can be any distribution with parameters $\boldsymbol{\theta}$ conditioned on the current hidden state. In the proposed APREP-S, $\mathbf{X}$ is the inference data and the list of each method number deduced by the APREP-S model using the inference data. APREP-S infers $\boldsymbol{\pi}$, $\mathbf{A}$, and $\boldsymbol{\theta}$, and returns the clustering result. After generating the new site-specific training data, the processes are the same as those of the previous APREP-S; i.e., APREP-S searches the target imputation area, defines features, and updates the APREP-S model.

4. Evaluation

We evaluated APREP-S to achieve the following: 1) comparing the model updated using the site-specific features with the model updated using all features defined in the model training phase, and 2) comparing the performance of APREP-S, which generated new learning data using the HMM in the model updating phase, with that of existing imputation methods. For trajectory analysis and behavioral analysis in a case of a factory, we used human activity data.

4.1 Evaluation Settings

4.1.1 Evaluation Dataset

We measured the accelerometer (without g), GPS, and pressure data for two activities by using a smartphone: 1) walking around our university campus, 2) ascending 120 stairs from the first floor to the 6th floor of the university building. The sensor was placed on the left side of the waist. In this evaluation, we use the accelerometer data as inference data, and the GPS data and pressure data as features. The walking data were col-

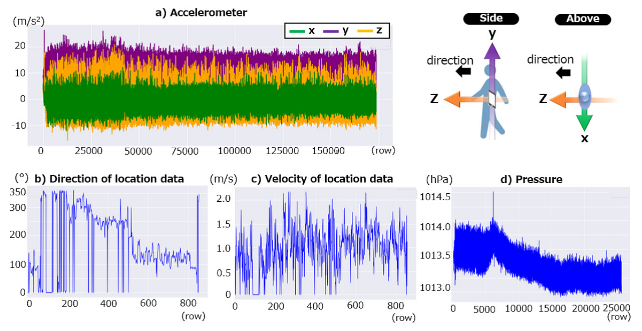


Fig. 6 Walking data measured by phyphox: 25,000 rows of accelerometer values were collected in approximately one minute, 200 rows of GPS data were obtained in approximately 3.3 minutes, and 5,000 rows of pressure data were recorded in approximately 2.8 minutes.

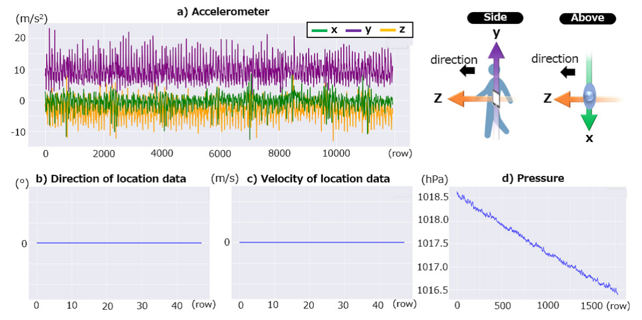


Fig. 7 Ascending stairs data measured by phyphox: 2,000 rows of accelerometer collected in approximately 10 seconds, 10 rows of GPS data, and 500 rows of pressure data measured in approximately 15 seconds.

lected during a period of approximately 14.5 minutes, and generated 177,884 rows of accelerometer data, 872 rows of GPS data, and 26,281 rows of pressure data. When ascending the stairs, data were recorded for approximately 1.5 minutes, and generated 16,132 rows of accelerometer data, 64 rows of GPS data, and 2,382 rows of pressure data. We discarded the first and last 10 seconds of data, respectively, to set the sensor on the waist. The recorded data are shown in **Fig. 6** and **Fig. 7**. We used the x- and y-axes of the accelerometer data as the target imputation data. As the features of the accelerometer data, we used the velocity and direction to update the model of the walking data, and used the pressure data to update the model of the data collected when ascending the stairs. We configured the missing data on the x- and y-axes of the accelerometer data. Let the probability of the occurrence of missing data and the number of continuous missing data points depend on the Gaussian distribution.

$$\mathcal{N}(e;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(e-\mu)^2}{2\sigma^2}\right\} \quad (5)$$

where μ is the mean, and σ^2 is the variance of the Gaussian distribution. The probability of missing a consecutive number is $\mathcal{N}(0, 200)$ for every $\mathcal{N}(0, 2000)$ times. As a result, the accelerometer x- and y-axes data for walking generated 1,582 and 1,374 rows of the target imputation area, respectively. The accelerometer x- and y-axes data for ascending the stairs generated 385 and 138 rows of the target imputation area.

We measured the sensor data using phyphox [18]. Phyphox is a smartphone application that records data by using the built-in sensors of the smartphone such as the accelerometer, magnetometer, gyroscope, light sensor, pressure meter, proximity sensor, microphone, and GPS.

Table 1 Unit of Sensor Data.

accelerometer	m/s^2
height	m
velocity	m/s
direction	°
horizontal accuracy	m
vertical accuracy	m
pressure	hPa

Table 2 Imputation Methods $\boldsymbol{M}$.

m_1	mean
m_2	fbprophet
m_3	GRU
m_4	spline interpolation

Table 3 Features using Each Activity for the APREP-S model.

walking	ascending stairs
(1)(2)(3)(4) (5)(6)	(1)(2)(3)(4) (7)

* the numbers represent the feature number in Section 4.1.3.

In phyphox, the x-axis points to the right side of the screen while looking at the screen in portrait orientation. The y−axis is upwards along the long side of the screen. The z-axis is perpendicular to the screen, and is positive in the direction of the screen. The accelerometer data is composed of three axes, x, y, and z with the unit of m/s^2. The GPS data from the satellite is composed of the latitude, longitude, height (m), velocity (m/s), direction (°), horizontal accuracy (m), and vertical accuracy (m). The pressure data is composed only of pressure (hPa) and is designed to determine the user's vertical position within a building, which is approximately 0.1 hPa = 0.1 mbar. These units are summarized in **Table 1**.

4.1.2 Imputation Method Defined in APREP-S

APREP-S includes four imputation methods: $m_1, m_2, m_3, m_4 \in \boldsymbol{M}$, as listed in **Table 2**. m_1 is the mean value of the addition and division by two of each value in each one before and after the target imputation area,

$$f(v) = \frac{1}{2}(v_b + v_a). \quad (6)$$

where v_b and v_a are just before and just after values of the target imputation area, respectively. m_2 is fbprophet [13], a well-known GAM method published by Facebook (describe in A.1 of the Appendix), m_3 is a gated recurrent unit (GRU) [19] of one of the RNN architectures. This method learns to encode a variable-length sequence into a fixed-length vector representation and to decode a given fixed-length vector representation back into a variable-length sequence. m_4 is spline interpolation.

4.1.3 Evaluation Features

The total number of features in this evaluation is seven. All features (1)–(7) are used for the initial model in the model training phase, (5) and (6) are used for updating the walking data model as specific features of walking data, and (7) is used for updating the ascending stairs model as specific feature of ascending stairs. The features that were used for each activity are listed in **Table 3**.

(1) the continuous number of rows in the target imputation area
(2) the gradient between target imputation areas,

$$(v_b - v_a)/n \quad (7)$$

where v_b and v_a are the just before and just after values of the target imputation area, and n is the number of rows in the target imputation area.

Table 4 Results of E by sum-of-squares error (Eq.(8)).

	data	model	feature	APREP-S	Mean	Fbprophet	GRU	Spline
walking	x-axis (1,582 rows)	initial	1,2,3,4,5,6	11,478 (7.26)	14,579 (9.22)	9,447 (5.97)	18,992 (12.01)	13,364 (8.45)
		update	1,2,3,4,5,6	9,384 (5.93)				
			all	9,384 (5.93)				
	y -axis (1,374 rows)	initial	1,2,3,4,5,6	22,902 (16.67)	17,193 (12.51)	21,570 (15.70)	45,757(33.33)	33,521 (24.40)
		update	1,2,3,4,5,6	20,329 (14.80)				
			all	23,333 (33.96)				
ascending stairs	x-axis (385 rows)	initial	1,2,3,4,7	957 (2.49)	415 (1.08)	957 (2.49)	2,613 (6.79)	747 (1.94)
		update	1,2,3,4,7	918 (2.38)				
			all	918 (2.38)				
	y-axis (138 rows)	initial	1,2,3,4,7	535 (3.88)	801 (5.80)	535 (3.88)	1,308 (9.48)	1,230 (8.91)
		update	1,2,3,4,7	535 (3.88)				
			all	535 (3.88)				

* the value in parentheses is the mean of the sum-of-squares error per row.
* refer to Section 4.1.3 for the number in the feature column.

(3) The gradient trend of before and after the target imputation area by comparing the before area trend with the after area trend. Input "1" if both trends are positive, input "-1" if the trends are opposite.
(4) the continuous number of rows in the target imputation area
(5) the velocity of GPS data
(6) the direction of GPS data
(7) the pressure data

4.2 Evaluation Method

We evaluated APREP-S by assessing the accuracy and the similarity of the trend in the imputation values. We first compare the accuracy of APREP-S with that of existing methods, and then compare the similarity of the trend of the original data with the existing method of which the accuracy is higher than that of APREP-S. Their details are described as follows.

(1) Accuracy: calculating the sum-of-squares error (E) of the original data and the results of APREP-S and existing methods. A smaller E indicates a higher accuracy.

$$E = \frac{1}{2}\sum_{n=1}^{N}(org_n - v_n)^2 \tag{8}$$

where N is the number of target imputation rows, org is the original value, and v_n is the value according to APREP-S or the existing methods it is being compared to.

(2) Similarity: comparing the trend of APREP-S with the original data by using the k-shape. The data distributed in the same cluster as the original data exhibits a similar trend. Details of the k-shape are provided in A.2 of the Appendix.

In this study, we compare APREP-S with the four existing methods: 1) the mean value of the addition and division by two of each value in each one before and after of the target imputation area, 2) fbprophet as a representative GAM, 3) GRU as a representative RNN, and 4) spline interpolation as a representative single imputation. We generate these models by using the same configurations as those defined in APREP-S. These configurations are described in Section 4.3.

4.3 Evaluation Procedure

In the model training phase, first, we generate the initial training data using all features in Section 4.1.3. We define m_1 (mean) if the number of continuous rows equal 1, m_3 (spline interpolation) if the number of continuous rows is less than 100, and the other is m_2 (fbprophet). APREP-S generates the model from these training data. APREP-S infers the optimal imputation method on each target imputation area, and APREP-S calculates the imputation values by each imputation method. Fbprophet (m_2) is used to train the model by using 3,000 rows of data beforehand. Here, it is configured that the periodicity is a second, and the Fourier order of the x- and y-axes is 800 and 1,000 for the walking data, respectively. For the ascending stairs data, the periodicity is configured to be a second, and the Fourier order of the x- and y-axes is 600 and 600, respectively. The GRU (m_3) is TensorFlow's KerasAPI. To generate a model, it extracted the training data from 10,000 rows of data. The step size is 50, and the batch size is 25. The number of hidden units is 200, and the number of training iterations is 100. Spline Interpolation (m_4) is an interpolate library of the SciPy API. Moreover, in the training algorithm of the APREP-S model, the two parameters α and β depend on Gaussian distribution $\mathcal{N}(0, 2)$. We used the PyMC3 library, and configured NUTS as a step method with 4,000 steps.

In the model updating phase, APREP-S performs the clustering by HMM. HMM is a GaussianHMM of the hmmlearn library. The number of clusters is eight which is twice the number of methods, the covariance type is full, and the maximum number of iterations is 300.

4.4 Evaluation Result and Discussion

The results of the accuracy of the imputation values by the sum-of-squares error are presented in **Table 4**. In the feature column, the numbers are the feature number, and "all" means all features are used. The number refers to the site-specific features, as described in Section 4.1.3. In the model column, "initial" means the result of the initial model of the model training phase, and "update" means the result of the updated model of the model updating phase.

The results for the site-specific features and "all" were the same, except for the y-axis of walking. This is attributed to HMM returning the same cluster group. Moreover, data collected for a short period, such as ascending stairs, were distributed in only four clusters of x-axis data and two clusters of y-axis data in spite of the specified number of clusters being eight. For the y-axis of walking, the accuracy of the result of the activity-specific features is higher than "all." Furthermore, the results for the y-axis of

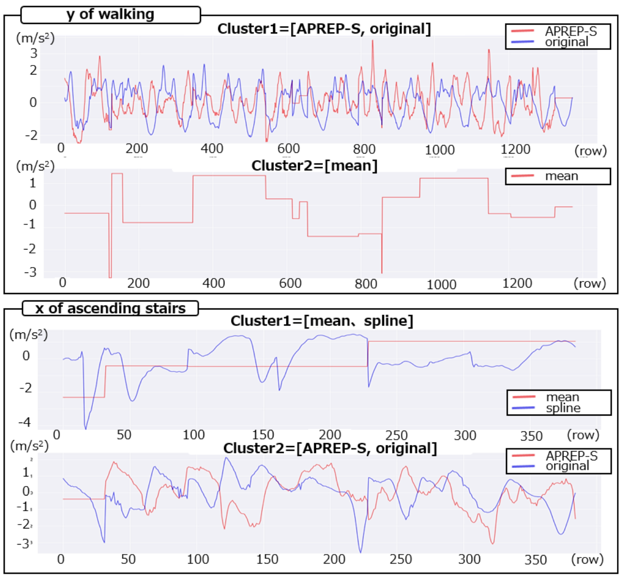


Fig. 8 Result of k-shape for the *y*-axis of walking and the *x*-axis of ascending stairs data. Both are extracted only by the target imputation area using site-specific features.

walking and the *x*-axis of ascending stairs data are less accurate than the mean. We evaluated the similarity of the trend by the k-shape for the *y*-axis of walking and the *x*-axis of ascending stairs. In addition, because the accuracy of the spline is superior to that of APREP-S for the *x*-axis of ascending stairs, the spline was also added. The result is shown in **Fig. 8**. For the *y*-axis of walking data, cluster1 = [APREP-S, original], and cluster2 = [mean]. For the *x*-axis of ascending stairs, cluster1 = [mean, spline], and cluster2 = [APREP-S, original]. The result of APREP-S is distributed in the same cluster as the original data. The similarity of the trend in comparison with the original data is worse than with APREP-S.

In terms of the overall accuracy and the similarity, we consider APREP-S to be the most optimal method when the site at which the model is updated differs from that at which it is generated. The result shows that the accuracy of APREP-S can be maintained by using clustering methods in the model updating phase.

5. Conclusion

In this study, we proposed a new imputation method that is an improvement on the previous APREP-S. The new method is more flexible as a result of clarifying the process of defining the features. We also improved the update process of APREP-S when data with a trend similar to that of the inference data do not exist.

The conclusions of this paper are as follows:

- APREP-S was found to be an efficient method if the site at which the model is generated differs from that at which the model is updated, because it maintains the accuracy when the site-specific features are selected in the model updating phase.
- A clustering method was defined by using the feature data of the inference of the imputation data when APREP-S updates the model. We concluded that APREP-S returned the imputation value with superior accuracy and similarity trend comparing with the other existing imputation method.

The proposed method, APREP-S, has several methods and selects the optimal method for the target imputation areas. We consider that APREP-S will be more suitable to use with short- and long-term data, in condition that we define multiple imputation models on various periodicities to a method. This is because the model to be generated depends on the periodicity of the sensor and the surrounding environment. The results of this study suggested that the accuracy would improve by having several models for each imputation method. In addition, these models can efficiently analyze complex types of data, such as human activity and the manufacture of product components. As the next step, we plan to examine the case where APREP-S has some models generated from the same imputation method.

Acknowledgments A part of this work was supported by JSPS KAKENHI Grant Number JP20K11776.

References

[1] Gulwani, S. and Jain, P.: Programming by Examples: PL meets ML, *Microsoft Research* (2017) (online), available from ⟨https://www.microsoft.com/en-us/research/publication/programming-examples-pl-meets-ml/⟩.

[2] Nagashima, H. and Kato, Y.: APREP-DM: A Framework for Automating the Pre-Processing of a Sensor Data Analysis based on CRISP-DM, *PerFoT'19 - International Workshop on Pervasive Flow of Things* (2019).

[3] Nagashima, H. and Kato Y.: Recommendation of Imputing Value for Sensor Data based on Programming by Example, *Journal of Information Processing*, Vol.28 (2020).

[4] Nagashima, H. and Kato, Y.: Data Imputation Method based on Programming by Example: APREP-S, *2019 IEEE International Conference on Big Data* (*Big Data*), pp.4412–4421, IEEE (2019).

[5] Siemens: Transforming into a Digital Company, Siemens, available from ⟨https://new.siemens.com/jp/en/company/topic-areas/casestudy-konicaminolta.html⟩ (accessed 2020-05-01).

[6] Cross Industry Standard Process for Data Mining Consortium: CRISP-DM by Smart Vision Europe, Cross Industry Standard Process for Data Mining Consortium (online), available from ⟨http://crisp-dm.eu/reference-model/⟩ (accessed 2019-12-14).

[7] Lee, J. and Bagheri, B. and Kao, H.-A.: A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems, *Manufacturing Letters*, Vol.3, pp.18–23 (online), DOI: 10.1016/j.mfglet.2014.12.001 (2015).

[8] Gulwani, S.: Automating String Processing in Spreadsheets Using Input-output Examples, *Proc. 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, *POPL '11*, pp.317–330, ACM (online), DOI: 10.1145/1926385.1926423 (2011).

[9] Graham, J.W. and Olchowski, A.E. and Gilreath, T.D.: How Many Imputations are Really Needed? Some Practical Clarifications of Multiple Imputation Theory, *Prevention Science*, Vol.8, No.3, pp.206–213 (online), DOI: 10.1007/s11121-007-0070-9 (2007).

[10] Graham, J.W.: Missing Data Analysis: Making It Work in the Real World, *Annual Review of Psychology*, Vol.60, No.1, pp.549–576 (online), DOI: 10.1146/annurev.psych.58.110405.085530 (2009).

[11] Mckinley, S. and Levine, M.: Cubic Spline Interpolation, *Coll. Redw.*, Vol.45 (1999).

[12] Hastie, T. and Tibshirani, R.: Generalized Additive Models, *Statistical Science*, Vol.1, No.3, pp.297–310 (online), available from ⟨https://www.jstor.org/stable/2245459⟩ (1986).

[13] Taylor, S.J. and Letham, B.: Forecasting at scale (online), DOI: 10.7287/peerj.preprints.3190v2 (2017).

[14] Hochreiter, S. and Schmidhuber, J.: Long Short-Term Memory, *Neural Computation*, Vol.9, No.8, pp.1735–1780 (online), DOI: 10.1162/neco.1997.9.8.1735 (1997).

[15] Bishop, C.M.: *Pattern recognition and machine learning*, Information science and statistics, Springer (2006).

[16] Nielsen, A.: *Practical Time Series Analysis - Prediction with Statistics and Machine Learning*, O'Reilly Media, Inc. (2019).

[17] Cover, T. and Hart, P.: Nearest neighbor pattern classification, *IEEE Trans. Information Theory*, Vol.13, No.1, pp.21–27 (online), DOI: 10.1109/TIT.1967.1053964 (1967).

[18] Staacks, S.: phyphox - physical phone experiments, 2nd Institute of Physics of the RWTH Aachen University (online), available from ⟨https://phyphox.org/⟩ (accessed 2020-04-24).

[19] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation (2014) (online), available from ⟨http://arxiv.org/abs/1406.1078⟩.

[20] Facebook's Core Data Science team: Prophet, Facebook (online), available from ⟨https://facebook.github.io/prophet/⟩ (accessed 2019-07-15).

[21] Berndt, D.J. and Clifford, J.: Using Dynamic Time Warping to Find Patterns in Time Series, *Proc. 3rd International Conference on Knowledge Discovery and Data Mining*, *AAAIWS'94*, Vol.3, pp.359–370, AAAI Press (1994) (online), available from ⟨http://dl.acm.org/citation.cfm?id=3000850.3000887⟩.

[22] Oehmcke, S., Zielinski, O., and Kramer, O.: kNN ensembles with penalized DTW for multivariate time series imputation, *2016 International Joint Conference on Neural Networks* (*IJCNN*), pp.2774–2781, IEEE (online), DOI: 10.1109/IJCNN.2016.7727549 (2016).

[23] Paparrizos, J. and Gravano, L.: k-Shape: Efficient and Accurate Clustering of Time Series, *Proc. 2015 ACM SIGMOD International Conference on Management of Data - SIGMOD '15*, pp.1855–1870, ACM Press (online), DOI: 10.1145/2723372.2737793 (2015).

Appendix

A.1 Fbprophet

Fbprophet is one of the methods of GAM. It is based on Prophet, a library of Python and R, and it is published by Facebook as OSS. The advantage of Fbprophet is its strength in terms of analyzing nonlinear trends, and offers the feature that data, including outliers and missing values, can be used as input values [13], [20]. The equation of fbprophet is expressed as follows;

$$g(t) = \frac{C(t)}{1 + \exp(-(k + \boldsymbol{a}(t)^{\mathrm{T}}\boldsymbol{\delta})(t - (m + \boldsymbol{a}(t)^{\mathrm{T}}\boldsymbol{\gamma})))} \quad \text{(A.1)}$$

where $g(t)$ is the same function as GAM in Section 2.2. Now,

$$\gamma_j = \left(s_j - m - \sum_{i<j} \gamma_l\right)\left(1 - \frac{k + \sum_{i<j} \delta_l}{k + \sum_{i\le j} \delta_l}\right) \quad \text{(A.2)}$$

$$a_j(t) = \begin{cases} 1, & \text{if} \quad \mathrm{t \le s_j} \\ 0, & \text{otherwise} \end{cases} \quad \text{(A.3)}$$

where $C(t)$ is a time-varying capacity, s_j is the changing points, k is the base rate at time t, and $\delta \in \mathbb{R}^s$ is a vector of rate adjustments.

$$s(t) = \sum_{n-1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right)\right) \quad \text{(A.4)}$$

where $s(t)$ is the same function as GAM, a and b are parameters, and P is the regular period we expect the time-series to have, e.g., $P = 365{:}25$ for yearly data or $P = 7$ for weekly data.

$$h(t) = Z(t)\boldsymbol{\kappa} \quad \text{(A.5)}$$

$$Z(t) = [\mathbf{1}(t \in D_1), ..., \mathbf{1}(t \in D_L)]. \quad \text{(A.6)}$$

where $h(t)$ is the same function as GAM.

A.2 K-shape

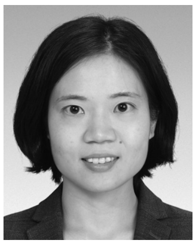
Examples of existing clustering methods for comparing the similarity of data are K-Means [17], dynamic time warping [21], KNN [22], and k-shape [23]. The k-shape method considers the shape of the time series in clustering tasks, in contrast to traditional methods such as K-Means. This method processes the observations in time-series data as independent attributes. In general, we consider the invariance of data before clustering, e.g., amplitude scaling, time-shifting, data length scaling, and occlusion. The k-shape method focuses on amplitude scaling invariance and time-shifting invariance. Moreover, it does not depend on the domain form because it calculates cross-correlation using normalized data as the distance measure for the similarity of data when clustering. It uses an independent method, named "shape-based distance" (SBD), by considering scaling and shifting with normalized data. SBD is expressed as

$$SBD(\boldsymbol{x}, \boldsymbol{y}) = 1 - \max_{\omega}\left(\frac{CC_\omega(\boldsymbol{x}, \boldsymbol{y})}{\sqrt{R_0(\boldsymbol{x}, \boldsymbol{x}) \cdot R_0(\boldsymbol{y}, \boldsymbol{y})}}\right) \quad \text{(A.7)}$$

where $\boldsymbol{x} = (x_1, ..., x_m)$, $\boldsymbol{y} = (y_1, ..., y_m)$ are sequences, and

$$CC_\omega(\boldsymbol{x}, \boldsymbol{y}) = R_{\omega-m}(\boldsymbol{x}, \boldsymbol{y}), \quad \omega \in 1, 2, ..., 2m-1 \quad \text{(A.8)}$$

$$R_k(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \sum_{l=1}^{m-k} x_{l+k} \cdot y_l, & k \ge 0 \\ R_{-k}(\boldsymbol{y}, \boldsymbol{x}), & k < 0 \end{cases} \quad \text{(A.9)}$$



Hiroko Nagashima received her B.Sc. from Tokyo Woman's Christian University in 2009, and her Master of Technology degree from Advanced Institute of Industrial Technology in 2016. She is a Ph.D. student at Tokyo Woman's Christian University. Her current research focuses on data analysis techniques for IoT systems. She is a member of IPSJ and IEEE.

Yuka Kato received her B.Sc from the University of Tokyo in 1989 and her M.E. and Ph.D. from the University of Electro-Communications in 1999 and 2002. From 1989 to 1998, she was with NTT and engaged in research on traffic control in ATM networks. She was a research associate at the University of Electro-Communications from 2002 to 2006, an associate professor and a professor at Advanced Institute of Industrial Technology from 2006 to 2014. Since 2014, she is a professor at Tokyo Woman's Christian University. Her research interests include information networks, network robots, and mathematical models for robotics. She is a member of IPSJ, RSJ, ACM, and IEEE.