Received 28 August 2023, accepted 21 September 2023, date of publication 19 October 2023, date of current version 25 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3326073

RESEARCH ARTICLE

Unsupervised Tensor Based Feature Extraction From Multivariate Time Series

KIYOTAKA MATSUE[1,2] **AND MAHITO SUGIYAMA**[1,3], **(Member, IEEE)**
[1]National Institute of Informatics, Tokyo 101-8430, Japan
[2]Toshiba Infrastructure Systems & Solutions Corporation, Kawasaki, Kanagawa 212-0013, Japan
[3]Department of Advanced Studies, The Graduate University for Advanced Studies, SOKENDAI, Hayama, Kanagawa 240-0193, Japan

Corresponding author: Kiyotaka Matsue (matsue@nii.ac.jp)

This work was supported in part by JSPS, KAKENHI, Japan, under Grant JP21H03503; and in part by JST, CREST, Japan, under Grant JPMJCR22D3.

ABSTRACT Clustering and outlier detection for multivariate time series are essential tasks in data mining fields and many algorithms have been developed for this purpose. However, these tasks remain challenging because both *time-wise* and *variable-wise* associations should be taken into account to treat multivariate time series appropriately. We propose a tensor based feature extraction method called UFEKT, which focuses on subsequences to account for the time-wise association and constructs a feature vector for each subsequence by applying tensor decomposition to account for the variable-wise association. This method is simple and can be used as an effective means of preprocessing for clustering and outlier detection algorithms. We show empirically that UFEKT leads to superior performance on various popularly used clustering algorithms such as K-means and DBSCAN and outlier detection algorithm such as the κ-nearest neighbor and LOF.

INDEX TERMS Feature extraction, multivariate time series, tensor decomposition, Tucker decomposition, clustering, outlier detection, unsupervised learning.

I. INTRODUCTION

In recent years, Internet of Things (IoT) devices with sensors, such as thermometers, hygrometers, or barometers have made it easy to collect large amounts of time series data. Once those data have been collected, they can offer a variety of services [1], [2], [3]. For instance, time series data collected by devices installed in office buildings are used for monitoring the status of their facilities and detecting the state transition of equipment. When multiple sensor devices are installed in a neighborhood, values taken by those devices are expected to be similar within a certain period of time. More specifically, if thermometers are placed in a room and many people gather in the room to attend a meeting, a lively debate could raise the temperature of the room. If building facilities can detect the state transition caused before and during the meeting, the facilities will control themselves to keep people comfortable. Air conditioners might control the temperature in the room or try to take in outside air to stable carbon dioxide concentration in the room. Thus, analyzing time series data or extracting features from time series data will give us various information that can be used for services to keep us comfortable or reduce energy of facilities.

In this paper, we focus on *extracting features* from *multivariate time series* for clustering and outlier detection, which takes the association between time series into account. Fig. 1a provides an example of clusters of multivariate time series data. There are two time series composed of sine waves and straight lines with noise. A combination in an orange dash-dotted frame has two subsequences (top and bottom) and they are composed of sine waves and a straight line. Furthermore, combinations in a green dashed frame are composed of sine waves and a straight line at the top, and only sine waves at the bottom. Both orange and green similar combinations can be seen a few times in the series and they can be categorized into two clusters - orange and green clusters - by collecting similar subsequences together. These clusters might not be able to be categorized if we focus on each time series separately, so it is necessary to take the association between subsequences into account to solve the subsequent time series clustering problem. Furthermore, Fig. 1b shows an example of an outlier detection task. As in

The associate editor coordinating the review of this manuscript and approving it for publication was Jon Atli Benediktsson.

Fig. 1a, there are two time series composed of sine waves and a straight line with noise. The combination in an orange frame (No.1) has two subsequences (top and bottom) and is composed of sine waves and lines. Similar combinations often occur in the series, while the combination in an orange frame (No.2), which is composed of only sine waves in bottom plot, occurs only once in the series; this is deemed to be an outlier. Thus, finding suitable clusters and detecting outliers from multivariate time series are known to be the difficult problems in multivariate time series analysis.

We propose an unsupervised feature extraction algorithm for multivariate time series, called UFEKT (unsupervised feature extraction using kernel method and Tucker decomposition). UFEKT use two main steps to extract features from multiple time series: (1) it constructs multiple kernel matrices from multivariate time series using the radial basis function (RBF) kernel to take the *time-wise* association into account, and (2) it stacks them to make a single tensor to take the *variable-wise* association into account. To get feature representations, UFEKT employs Tucker decomposition [4] using the higher-order orthogonal iteration (HOOI) algorithm, which decomposes a given tensor into one core tensor and multiple factor matrices. HOOI can generate orthogonal factor matrices using singular value decomposition (SVD), which is one of the well-known methods of obtaining factorized matrices. Since every column in factor matrices is expected to have an orthogonal feature of multivariate time series, each row vector can be treated as the resulting representation in the form of a feature vector for the corresponding subsequence of a multivariate time series.

We examine the effectiveness of UFEKT if it is combined with clustering or outlier detection. As for clustering, we use 102 real-world datasets collected from UCR Time Series Classification datasets (UCR) [5] and apply the standard clustering algorithms such as K-means (KMeans) [6], [7], density-based spatial clustering of applications with noise (DBSCAN) [8], agglomerative hierarchical clustering (AHC) [9], or Gaussian mixture model (GMM) [10] to the feature vectors obtained by feature representation methods including UFEKT. For the outlier detection task, we use nine synthetic datasets and six real-world datasets. Similar to the clustering task, we apply the standard existing outlier detection algorithms such as κth-nearest neighbor (κNN) [11], local outlier factor (LOF) [12], one-class support vector machine (OCSVM) [13], or isolation forest (IForest) [14]. We discuss more details of experimental conditions and results of clustering and outlier detection tasks in Sections IV and V.

To summarize, our main contributions are as follows:

- We propose a new algorithm called UFEKT that can extract features from multivariate time series.
- We empirically show the effectiveness of combination of UFEKT and downstream tasks of clustering or outlier detection using 102 datasets for clustering and 15 datasets for outlier detection.
- We provide a hyperparameter tuning algorithm for UFEKT that can work in an unsupervised manner.

We presented the UFEKT algorithm at DSAA 2021 and examined the effectiveness for outlier detection [15]. In this paper, we firstly extend it for clustering and show the effectiveness of UFEKT when combined with not only outlier detection but also clustering algorithms.

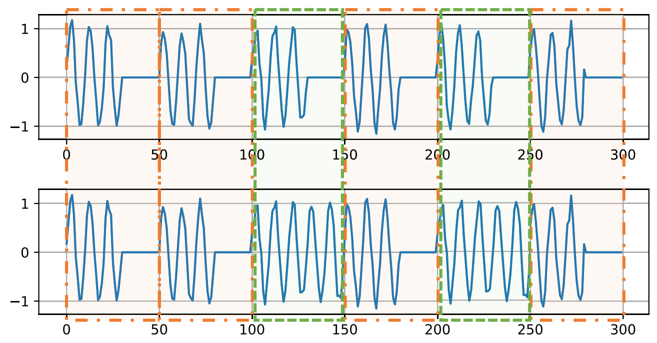
II. RELATED WORK

A. FEATURE EXTRACTION FROM TIME SERIES

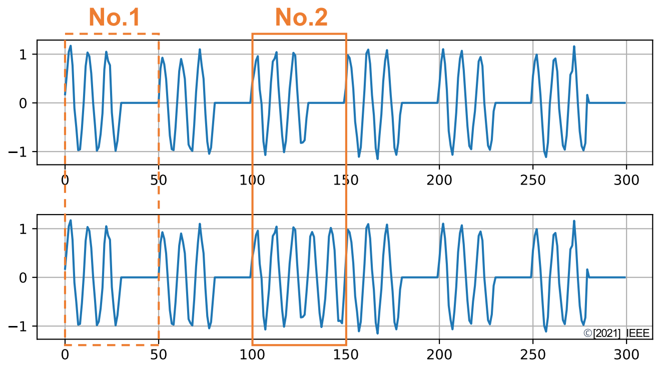
Several algorithms for feature extraction from (univariate) time series have been developed. For instance, discrete Fourier transform (DFT) [16], discrete wavelet transform (DWT) [16], [17], and discrete cosine transformation (DCT) are known as feature representation methods of time series data and are widely used in the signal processing field [18]. Other methods using trend statistics are also well-known techniques for extracting features from time series data [19]. However, those methods cannot be directly applied to multivariate time series. Kernel methods are used for extracting features from multivariate time series [20], [21]. They construct a kernel matrix using radial basis function (RBF) kernel [20] or linear kernel [21] from the set of subsequences of given time series. However, since their approaches only use the integrated signal across multiple time series, they cannot treat combinatorial association of time series. In contrast, UFEKT can treat each time series separately, so it is expected to keep information about association between time series in the resulting output.

B. CLUSTERING FOR TIME SERIES

A number of clustering algorithms of time series have been developed and widely used in many areas, such as biology, climate, energy consumption, environmental analysis, finance, and medicine [18]. These algorithms are mainly categorized into two types: *whole time series clustering* and *subsequence time series clustering*. A basic procedure of both whole and subsequence time series clustering is composed of the two following steps: (1) Representing raw time series as feature vectors under certain model parameters, and (2) making clusters using a non-time series clustering algorithm by measuring distance between the resulting feature vectors. For instance, DFT, DWT, DCT, singular value decomposition (SVD), and symbolic aggregate approximation (SAX) are widely used as representation methods for time series data. In addition, Euclidean distance, dynamic time warping (DTW), and Kullback–Leibler divergence [22] are popular choices of distance measurements for time series data. It is possible to perform clustering by combining those methods of distance measurements and non-time series clustering algorithms such as a distance-based algorithm (such as K-means), a density-based algorithm (for example, DBSCAN), a hierarchical-based algorithm (AHC), or a model-based algorithm (GMM).



(a) An example of subsequence time series clustering for multivariate time series data. There are two time series composed of sine waves and lines with noise. The combination in each orange dash-dotted frame has two subsequences (top and bottom) and is composed of sine waves and lines, while green dashed frames compose of sine waves and lines in top and sine waves only in the bottom. In this scenario, we can make two clusters by collecting orange subsequences and green subsequences.



(b) An example of outlier detection for multivariate time series data. There are two time series composed of sine waves and lines with noise. The combination in an orange frame No.1 has two subsequences (up and bottom) and is composed of sine waves and lines. Similar combination often occur in the series, while combination in an orange frame No.2, which is composed of only sine waves in the bottom plot, occurs only once in the series; this is deemed to be an outlier.

FIGURE 1. Examples of clustering and outlier detection for multivariate time series data.

In addition to the above, various algorithms of time series clustering have been developed, including those using Toeplitz matrices, shapelets, and incremental clustering [18], [23], [24], [25], [26], [27].

Keogh et al. pointed out that subsequence time series clustering has less meaning now [28]. They created 90 instances from three patterns of subsequences with adding Gaussian noise and concatenated them to make a single long time series, followed by performing subsequence time series clustering using K-means. They showed that the resulting cluster center becomes in the form of sine waves, and this phenomenon has also been theoretically analyzed [29]. However, it is not directly related to our study because the UFEKT algorithm transforms time series subsequences to feature vectors using kernels and clustering is performed not on the original subsequences, but on the obtained feature vectors; hence, cluster centers do not become sine waves.

C. OUTLIER DETECTION FOR TIME SERIES

Over the last 10 years, considerable attention has been given to developing outlier detection methods for multivariate time series [20], [30], [31]. For example, an algorithm using sparse representation has been studied [32]. This algorithm requires labeled data as it treats supervised outlier detection, while our method is fully unsupervised. In addition, [32] does not focus on combinatorial outliers.

Moreover, a large number of outlier detection algorithms using neural networks have been studied [21], [33], [34], [35], [36], [37], [38], [39]. A representative method, the multi-scale convolutional recurrent encoder-decoder (MSCRED), based on attention based convolutional long short-term memory (LSTM) [21], has been proposed to detect outliers from multivariate time series using constructed kernel matrices. While these neural network based techniques have grown in popularity recently, they require ground-truth inliers to train neural networks, which are not available in the fully unsupervised setting that we focus on in this paper. Moreover, they have many parameters to be tuned and such parameter tuning is fundamentally difficult in unsupervised learning.

There are a number of unsupervised outlier detection methods for non-time series data [37], [38], [40], [41], [42], [43], [44], [45]. Local outlier factor (LOF) [12], κth-nearest neighbor (κNN) [11], ORCA [46], one-class support vector machine (OCSVM) [13], isolation forest (IForest) [14], and sampling based outlier detection [47] are a part of well-known algorithms. These algorithms fundamentally assume time-wise independence between points; that is, they are not appropriate for time series. Since our method generates feature vectors from multivariate time series that incorporate such time-wise association, the above outlier detection algorithms can be applied directly to the obtained feature vectors, which is an advantage of our method.

III. FEATURE REPRESENTATION METHODS

A. PROBLEM SETTING

Assume that a multivariate time series is given as a matrix $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{P\times T}$ with P variables indexed from 1 to P and the length T of time series, where each row vector $\mathbf{x}^{(p)} = (x_{p1}, x_{p2}, \ldots, x_{pT}) \in \mathbb{R}^T$ represents a time series of a variable p and each column vector $\mathbf{x}_t = (x_{1t}, x_{2t}, \ldots, x_{Pt})^\top \in \mathbb{R}^P$ represents a multivariate vector at a time stamp t. A submatrix $\mathbf{X}_t \in \mathbb{R}^{P\times w}$, which is a part of $\mathbf{X}$ with respect to time stamps from t to $t + w - 1$, is given as

$$\mathbf{X}_t := \begin{bmatrix} x_{1t} & \cdots & x_{1(t+w-1)} \\ x_{2t} & \cdots & x_{2(t+w-1)} \\ \vdots & \ddots & \vdots \\ x_{Pt} & \cdots & x_{P(t+w-1)} \end{bmatrix}, \tag{1}$$

which represents a subsequence at t with its length of w. Each row $\mathbf{x}_t^{(p)} = (x_{pt}, x_{p(t+1)}, \ldots, x_{p(t+w-1)}) \in \mathbb{R}^w$ is a subsequence of the p-th time series. We assume that each extracted subsequence $\mathbf{X}_t \in \mathbb{R}^{P\times w}$ from the multivariate time series is represented as some R-dimensional feature vector $\mathbf{f}_t \in \mathbb{R}^R$.

In the clustering task, we can obtain clusters of subsequences by applying some non-time series clustering algorithm to the collection of feature vectors. Let $C_1, \ldots, C_K$ be K clusters of subsequences, which satisfy the following conditions:

$$C_i \cap C_j = \emptyset, \ \forall i, j \in \{1, \ldots, K\}, \tag{2}$$

$$C_1 \cup \cdots \cup C_K = \{\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_{T-w+1}\}, \tag{3}$$

that is, clusters are disjoint with each other and every feature vectors must be assigned to one of the clusters.

Similarly, we can apply outlier detection to R-dimensional feature vectors $\mathbf{f}_t \in \mathbb{R}^R$. For example, if we employ the κth-nearest neighbor detector (κNN) [11], the outlierness score $q(\mathbf{f}_t)$ of each subsequence is defined as

$$q(\mathbf{f}_t) := d(\mathbf{f}_t; \mathbf{S}_{\text{fvec}}), \tag{4}$$

$$\mathbf{S}_{\text{fvec}} := \{\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_{T-w+1}\}, \tag{5}$$

where $d(\mathbf{f}_t; \mathbf{S}_{\text{fvec}})$ is the Euclidean distance from $\mathbf{f}_t$ to its κth-nearest neighbor in the set $\mathbf{S}_{\text{fvec}}$.

B. A KERNEL METHOD FOR FEATURE REPRESENTATION

There is a representative kernel based method [20] for outlier detection from multivariate time series, which uses the page rank (PR) algorithm to represent features of multivariate time series. In the following, we denote by PRK (PageRank kernel) this kernel based method using the PR algorithm. PRK extracts feature vectors from multivariate time series. More precisely, it constructs a state transition probability matrix converted from a kernel matrix calculated by the RBF kernel, and the state transition probability matrix is used for the PR algorithm to detect outliers.

Given a multivariate time series $\mathbf{X} \in \mathbb{R}^{P\times T}$ with P variables with the length T, the PR algorithm constructs a kernel matrix $K \in \mathbb{R}^{(T-w+1)\times(T-w+1)}$ such that

$$k_{ij} := \exp\left\{-\frac{\sum_{p=1}^{P}\sum_{s=0}^{w-1}(x_{i+s}^{(p)} - x_{j+s}^{(p)})^2}{\sigma^2}\right\}, \tag{6}$$

for each $i, j \in \{1, 2, \ldots, T-w+1\}$, where w is the length of each subsequence. Hence, each row vector $\mathbf{k}_i \in \mathbb{R}^{(T-w+1)}$ in the kernel matrix represents a feature vector that considers pairwise associations between subsequences of multivariate time series.

C. OVERVIEW OF THE UFEKT APPROACH

Although PRK can be an effective way to represent features of multivariate time series, since each element of the kernel matrix is the summed-up value across variables, it may lose information of association among variables. To address this issue, UFEKT makes a tensor rather than a summed-up matrix, and then decomposes it to obtain feature vectors. UFEKT consists of two steps: constructing a tensor by stacking kernel matrices generated from multivariate time series and extracting features from one of the factor matrices obtained by tensor decomposition. This is described in details below.

D. CONSTRUCTING KERNEL MATRICES FROM MULTIVARIATE TIME SERIES

We extract subsequences from time series and UFEKT makes multiple kernel matrices to measure the similarity between subsequences for each time series. Unlike PRK, UFEKT obtains one kernel matrix from one univariate time series, that is, the number of obtained kernel matrices from multivariate time series is the same as that of variables.

First, UFEKT extracts feature vectors from each univariate time series $\mathbf{x}^{(p)}$. Given two subsequences of p-th univariate time series $\mathbf{x}_i^{(p)}, \mathbf{x}_j^{(p)} \in \mathbb{R}^w$ with the length w, when the RBF kernel is employed, the similarity $k_{ij}^{(p)}$ between $\mathbf{x}_i^{(p)}$ and $\mathbf{x}_j^{(p)}$ is defined as

$$k_{ij}^{(p)} := \exp\left\{-\frac{\sum_{s=0}^{w-1}(x_{p(i+s)} - x_{p(j+s)})^2}{\sigma^2}\right\}, \tag{7}$$

where σ is a parameter. Every $k_{ij}^{(p)}$ takes a value in (0, 1]. The RBF kernel for similarity computation was first employed in [20] and UFEKT also follows the strategy. The resulting kernel matrix $\mathbf{K}^{(p)} \in \mathbb{R}^{(T-w+1)\times(T-w+1)}$ becomes a non-negative square matrix given as

$$\mathbf{K}^{(p)} := \begin{bmatrix} k_{11}^{(p)} & \cdots & k_{1(T-w+1)}^{(p)} \\ k_{21}^{(p)} & \cdots & k_{2(T-w+1)}^{(p)} \\ \vdots & \ddots & \vdots \\ k_{(T-w+1)1}^{(p)} & \cdots & k_{(T-w+1)(T-w+1)}^{(p)} \end{bmatrix}. \tag{8}$$

Each row vector is denoted as $\mathbf{k}_t^{(p)} = (k_{t1}^{(p)}, k_{t2}^{(p)}, \ldots, k_{t(T-w+1)}^{(p)}) \in \mathbb{R}^{(T-w+1)}$ and T is the length of time series. Each row vector $\mathbf{k}_t^{(p)}$ in the kernel matrix $\mathbf{K}^{(p)}$ can be viewed as a kernel vector representation of the subsequence $\mathbf{x}_t^{(p)}$, and it incorporates association between a subsequence $\mathbf{x}_t^{(p)}$ and all the other subsequences. The time complexity of constructing each $\mathbf{K}^{(p)}$ is $\mathcal{O}(T^2)$; hence, the total time complexity becomes $\mathcal{O}(T^2P)$ when there are P times series.

E. EXTRACTING FEATURE FROM KERNEL MATRICES

Second, UFEKT constructs a tensor from the set of obtained kernel matrices to take into account the association between multiple time series. Given P kernel matrices $\mathbf{K}^{(1)}, \mathbf{K}^{(2)}, \ldots, \mathbf{K}^{(P)} \in \mathbb{R}^{(T-w+1)\times(T-w+1)}$ generated from P time series by (8), a tensor is constructed by stacking them; that is, it becomes a three-dimensional tensor $\mathcal{K} \in \mathbb{R}^{P\times(T-w+1)\times(T-w+1)}$. The tensor incorporates information about association between subsequences in multivariate time

Algorithm 1 The UFEKT Algorithm

```
Input: $\mathbf{X} \in \mathbb{R}^{P \times T}$, $w, \sigma, R \in \mathbb{R}$
Output: $\mathbf{f}_1, \ldots, \mathbf{f}_{T-w+1} \in \mathbb{R}^R$
    // Construct kernel matrices from multivariate time series
 1: for $p = 1$ to $P$ do
 2:     for $(i, j) = (1, 1)$ to $(T - w + 1, T - w + 1)$ do
 3:         $k_{ij}^{(p)} \leftarrow \exp\{-\sum_{s=0}^{w-1}(x_{p(i+s)} - x_{p(j+s)})^2/\sigma^2\}$
 4:     end for
 5: end for
    // Feature Extraction from Kernel Matrices
 6: Construct $\mathcal{K} \in \mathbb{R}^{P \times (T-w+1) \times (T-w+1)}$ from $\mathbf{K}^{(1)}, \ldots, \mathbf{K}^{(P)}$
 7: $(\mathcal{C}, \mathbf{F}^{(1)}, \mathbf{F}^{(2)}, \mathbf{F}^{(3)}) \leftarrow$ Tucker$(\mathcal{K}, [P, R, R])$
 8: for $i = 1$ to $T - w + 1$ do
 9:     $\mathbf{f}_i \leftarrow i$th-row vector of $\mathbf{F}^{(2)}$
10: end for
11: return $\mathbf{f}_1, \ldots, \mathbf{f}_{T-w+1}$
```

series. To extract features from the tensor, UFEKT uses Tucker decomposition [4].

Given a tensor $\mathcal{K} \in \mathbb{R}^{M \times N \times P}$, Tucker decomposition decomposes it into one core tensor $\mathcal{C} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ and three factor matrices $\mathbf{F}^{(1)} \in \mathbb{R}^{M \times R_1}$, $\mathbf{F}^{(2)} \in \mathbb{R}^{N \times R_2}$, and $\mathbf{F}^{(3)} \in \mathbb{R}^{P \times R_3}$. The optimization problem for decomposing the tensor $\mathcal{K}$ is formulated as

$$\min_{\mathcal{C}, \mathbf{F}^{(1)}, \mathbf{F}^{(2)}, \mathbf{F}^{(3)}} \|\mathcal{K} - \mathcal{C} \times_1 \mathbf{F}^{(1)} \times_2 \mathbf{F}^{(2)} \times_3 \mathbf{F}^{(3)}\|, \tag{9}$$

where each entry k'_{mnp} of the term $\mathcal{C} \times_1 \mathbf{F}^{(1)} \times_2 \mathbf{F}^{(2)} \times_3 \mathbf{F}^{(3)}$ is defined as

$$k'_{mnp} := \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} c_{r_1 r_2 r_3} f^{(1)}_{mr_1} f^{(2)}_{nr_2} f^{(3)}_{pr_3}. \tag{10}$$

Several algorithms to solve the optimization problem in (9) have been already developed, and UFEKT adopts *Higher-Order Orthogonal Iteration* (HOOI) [48], [49], which is based on singular-value decomposition (SVD) for matrices. By applying HOOI to the obtained tensor of kernel, the tensor $\mathcal{K} \in \mathbb{R}^{P \times (T-w+1) \times (T-w+1)}$ can be decomposed into one core tensor $\mathcal{C} \in \mathbb{R}^{P \times R \times R}$ and three factor matrices $\mathbf{F}^{(1)} \in \mathbb{R}^{P \times P}$, $\mathbf{F}^{(2)} \in \mathbb{R}^{(T-w+1) \times R}$, $\mathbf{F}^{(3)} \in \mathbb{R}^{(T-w+1) \times R}$, where ranks of the core tensor are given as $[P, R, R]$ in advance as hyperparameters. The computational cost for performing Tucker decomposition using HOOI for a third-order tensor is known to be $\mathcal{O}(T^3R + TR^4 + R^6)$ [50], [51], where it assumes that the rank and size of a tensor are $\mathcal{C} \in \mathbb{R}^{R \times R \times R}$ and $\mathcal{K} \in \mathbb{R}^{T \times T \times T}$, respectively.

After decomposing the given tensor, UFEKT focuses on one of the factor matrices, $\mathbf{F}^{(2)}$ or $\mathbf{F}^{(3)}$, and extracts its row vectors as the resulting feature vectors of subsequences in the original multivariate time series. UFEKT consistently uses $\mathbf{F}^{(2)}$ to construct feature vectors as there is usually no significant difference between $\mathbf{F}^{(2)}$ and $\mathbf{F}^{(3)}$. This is why kernel matrices before constructing a tensor are always symmetric and similar features are incorporated in their decomposed factor matrices. More precisely, given a tensor of kernels $\mathcal{K} = (k_{mnp}) \in \mathbb{R}^{M \times N \times P}$, SVD used in HOOI is performed against matrices $\mathbf{K}_M \in \mathbb{R}^{M \times (N \times P)}$, $\mathbf{K}_N \in \mathbb{R}^{N \times (P \times M)}$, and $\mathbf{K}_P \in \mathbb{R}^{P \times (M \times N)}$, which are extracted from the tensor $\mathcal{K}$. If ranks to $[R, R, P]$ are set, the similar factor matrices $\mathbf{F}_M \in \mathbb{R}^{M \times R}$ and $\mathbf{F}_N \in \mathbb{R}^{N \times R}$ are obtained from $\mathbf{K}_M$ and $\mathbf{K}_N$, respectively, because both m-th row vectors $\mathbf{k}_{Mm}$ in $\mathbf{K}_M$ and $\mathbf{k}_{Nm}$ in $\mathbf{K}_N$ are composed of the similar elements except for the order - that is, $k_{mnp} \simeq k_{nmp}$ - always holds. However, factor matrices $\mathbf{F}'_M$ and $\mathbf{F}'_N$, which are eventually obtained by HOOI, could be slightly different from each other due to the iteration of SVD in the framework of HOOI, they are still expected to be similar with each other. Therefore, their matrices are not distinguished and the matrix $\mathbf{F}^{(2)}$ is consistently used in UFEKT. A symmetric Tucker-2 decomposition could be used instead of the original Tucker decomposition, as Tucker-2 decomposition assumes that $\mathbf{F}^{(2)}$ and $\mathbf{F}^{(3)}$ are the same. However, the original Tucker decomposition algorithm is employed in UFEKT since it is expected that there is no significant difference between the original Tucker and Tucker-2 due to the above reasons (equivalence of SVD) and factor matrices might be used for other tasks, such as feature selection.

As a result, row vectors $\mathbf{f}_1, \ldots, \mathbf{f}_{(T-w+1)} \in \mathbb{R}^R$ can be obtained from the factor matrix $\mathbf{F}^{(2)}$. Furthermore, column vectors of $\mathbf{F}^{(2)}$ are considered to be almost orthogonal with each other because HOOI uses the SVD algorithm to decompose a tensor. Therefore, it is considered that row vectors of $\mathbf{F}^{(2)}$ represent feature vectors of multivariate time series, where both time-wise and variable-wise associations are taken into account, and these feature vectors can be applied to a variety of applications like clustering or outlier detection. The pseudo code of UFEKT is summarized in Algorithm 1.

F. HOW TO DETERMINE THE RANKS FOR TUCKER DECOMPOSITION

In UFEKT, ranks of a core tensor in tensor decomposition must be determined in advance. As we apply the feature vectors extracted by UFEKT to clustering or outlier detection, automatic parameter selection such as grid search via cross-validation cannot be used because of the unsupervised nature. As mentioned above, a three-dimensional rank $[R_1, R_2, R_3]$ is required to decompose a three-dimensional tensor into one core tensor and three factor matrices. Assume that R_1 indicates an axis in the direction of variables and R_2, R_3 indicate axes in the direction of time series. The first rank R_1 is recommended to be the same value as the number of variables because a decomposed factor matrix is not used in our method and compression on this direction is not required. We recommend setting $R_2 = R_3$ because only one of two factor matrices would be used to construct feature vector representation in UFEKT. Hence, only one parameter R_2 should be determined in UFEKT.

Algorithm 2 Rank Selection of a Core Tensor for UFEKT (especially for Outlier detection)

```
Input: $\mathcal{K} \in \mathbb{R}^{P\times(T-w+1)\times(T-w+1)}$, $\kappa \in \mathbb{N}$, $i\_min \in \mathbb{N}$
Output: $R_{\mathrm{opt}} \in \mathbb{R}$
 1: Prepare $\mathbf{d} \in \mathbb{R}^R$
 2: for $i = 1$ to $T-w+1$ do
 3:     $(\mathcal{C}_i, \mathbf{F}_i^{(1)}, \mathbf{F}_i^{(2)}, \mathbf{F}_i^{(3)}) \leftarrow \mathrm{Tucker}(\mathcal{K}, [P, i, i])$
 4:     $\mathcal{F}_i \leftarrow \emptyset$
 5:     for $j = 1$ to $T-w+1$ do
 6:         $\mathbf{f}_j \leftarrow j$th row vector of $\mathbf{F}_i^{(2)}$;   $\mathcal{F}_i \leftarrow \mathcal{F}_i \cup \{\mathbf{f}_j\}$
 7:     end for
 8:     Prepare $\mathbf{s} \in \mathbb{R}^{T-w+1}$
 9:     for $j = 1$ to $T-w+1$ do
10:         $\mathbf{s}_j \leftarrow d^{\kappa}(\mathbf{f}_j; \mathcal{F}_i)$
11:     end for
12:     $\mathbf{d}_i \leftarrow (\max_j \mathbf{s}_j)/(\sum_j \mathbf{s}_j)$
13: end for
14: $R_{\mathrm{opt}} \leftarrow \mathrm{argmax}_{i\geq i\_min}\mathbf{d}_i$
15: return $R_{\mathrm{opt}}$
```

In clustering, we use a constant value, $R_2 = R_3 = 8$, which is not too small or large for most datasets, because essential information of time series must be included in lower dimensional space of feature vectors and unnecessary information might be included in higher dimensions in clustering.

By contrast, it is difficult to determine the rank for outlier detection because information about outliers must be contained in feature vectors in a higher dimensional space. Therefore, we propose an algorithm using the κth nearest neighbor (κNN) distance to decide appropriate ranks. For each rank setting, we decompose a tensor and compute the κNN distance for each row vector in $\mathbf{F}^{(2)}$. We then normalize each distance by dividing it by the summation of all distances to compare distances across different ranks. Finally, we adopt the rank with the largest distance for outlier detection as it is expected that outliers may be well distinguished. Since the effect of outliers are considered not to appear if the rank is too small, we seek ranks greater than or equal to i_min, which is a hyper parameter. In our experiments, we set $\kappa = 5$ and $i\_min = 10$, and we will show one of experimental results in Fig. 2, which means that this proposed heuristic strategy gives good rank setting in terms of the performance in outlier detection in practice. The details of our algorithm to find an appropriate rank of a core tensor is shown in Algorithm 2.

IV. EXPERIMENTS FOR CLUSTERING

To examine the effectiveness of extracted feature vectors, we empirically evaluated them on real-world datasets for clustering and outlier detection, and synthetic datasets for outlier detection. We compared UFEKT with two other features representation algorithms, the PageRank kernel (PRK) and Subsequence (SS), under four different clustering

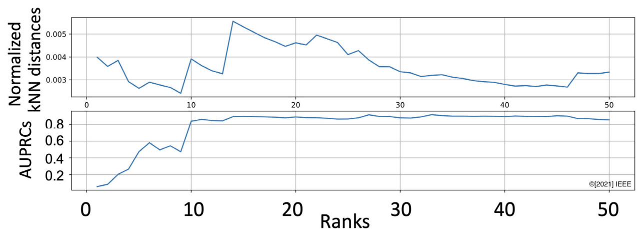


FIGURE 2. Rank dependencies of AUPRCs. The x-axes indicate ranks of a core tensor and the y-axes indicate normalized κNN distances for the top plot and AUPRCs for the bottom plot, respectively. The top plot is obtained by Algorithm 2, while the bottom plot is obtained by the grid search, which requires the ground truth labels that are not available in practice.

TABLE 1. Hyperparameters of our experiments about UCR real-world datasets. The length of each subsequence, which coincides with the window size, can be calculated from the percentages shown in the table. Since the percentages indicate the ratio of the length of time series, if a time series has 1,000 time stamps and [8, 16, 32] are given as parameters, the lengths of a subsequence would be 80, 160, and 320, respectively. We simulated three times for each dataset while varying the percentages (that is, 8, 16, and 32), and chose the percentage that has the best NMI score for our evaluation. The rank is used for Tucker decomposition in UFEKT. In general, three parameters of ranks are needed for three-way tensor, however, UFEKT requires only one rank as a hyperparameter because one of the three parameters is decided by the number of parameters and the other two always take the same value.

Name of Datasets	Percentages of Subsequence	Ranks for Tucker Decomposition
EOGHorizontalSignal	[8, 16, 32]	[1251, 8, 8]
EOGVerticalSignal	[8, 16, 32]	[1251, 8, 8]
FiftyWords	[8, 16, 32]	[271, 8, 8]
GestureMidAirD1	[8, 16, 32]	[361, 8, 8]
InlineSkate	[8, 16, 32]	[1883, 8, 8]
MelbournePedestrian	[1/32, 1/16, 1/8]	[25, 8, 8]
SemgHandGenderCh2	[8, 16, 32]	[1501, 8, 8]
SemgHandMovementCh2	[8, 16, 32]	[1501, 8, 8]
SemgHandSubjectCh2	[8, 16, 32]	[1501, 8, 8]
SyntheticControl	[8, 16, 32]	[61, 8, 8]
UWaveGestureLibraryX	[1/32, 1/16, 1/8]	[316, 8, 8]
UWaveGestureLibraryY	[1/32, 1/16, 1/8]	[316, 8, 8]
UWaveGestureLibraryZ	[1/32, 1/16, 1/8]	[316, 8, 8]
Other than the above	[8, 16, 32]	[# of variables, 8, 8]

algorithms, KMeans, DBSCAN, AHC, and GMM, which are popular and widely used in data analysis.

A. THE PAGERANK KERNEL (PRK)

As mentioned in Section III-B, the PageRank kernel (PRK) has been proposed in the outlier detection method PR [20]. It constructs a state transition probability matrix converted from a kernel matrix calculated by the RBF kernel. Given a multivariate time series $\mathbf{X} \in \mathbb{R}^{(P\times T)}$ with P variables with the length T, the kernel matrix $K \in \mathbb{R}^{(T-w+1)\times(T-w+1)}$ is defined as Equation (6). In our experiments, we use PRK for comparison of our evaluation.

B. THE SUBSEQUENCE (SS)

Subsequence, which we denote by SS, is widely used in extracting features from time series. Given $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{P\times T}$ with P variables with the length T, we can obtain a subsequence $\mathbf{x}_{ss}$ by summing up every variable at each time t

TABLE 2. Summary of UCR datasets for clustering.

Items	Values
The numbers of datasets	102
The lengths of time series	40 - 4478
The numbers of variables	16 - 2835
The numbers of class labels	2 - 52

as

$$\mathbf{x}_{ss} := \left(\sum_{p=1}^{P} x_1^{(p)}, \ldots, \sum_{p=1}^{P} x_T^{(p)} \right). \tag{11}$$

When we define each element of $\mathbf{x}_{ss}$ at time t as $x_t^{ss} := \sum_{p=1}^{P} x_t^{(p)}$, the subsequence matrix $\mathbf{X}_{ss} \in \mathbb{R}^{(T-w+1)\times w}$ with the window size w can be defined as $x_{ss,ij} = x_{i+j-1}^{ss}$. We treat each row in $\mathbf{X}_{ss}$ as a feature vector representation of the corresponding subsequence, and apply clustering algorithms to it.

C. EXPERIMENTAL ENVIRONMENT

We used CentOS release 6.10 with 4 × 22-Core model 2.20 GHz Intel Xeon CPU E7-8880 v4 processors and 3.18 TB memory. All methods are implemented in Python 3.7.6 and all experiments are also performed in the same platform.

D. DATASETS

We employed the time series classification datasets offered by the University of California, Riverside, called *UCR Time Series Classification datasets (UCR)*[1] for our experiments. These datasets are widely used as experimental real-world time series datasets for a classification and a clustering problem. We collected 102 out of 128 UCR datasets because some long time series datasets led to extremely high computational costs. Since all of the datasets are provided for a classification problem - that is, training and test datasets are offered separately - we concatenated them into one time series for each dataset and then applied their concatenated time series to a clustering problem. Its summary of the UCR datasets used in our experiments is shown in Table 1 and Table 2. The details are also shown in Table 3.

E. EVALUATION METRICS

There are some evaluation metrics about clustering such as cross entropy, J-Measure, and NMI [26]. To evaluate the results of our experiments we adopted *Normalized Mutual Information (NMI)*, which is widely used for evaluating accuracy of a clustering task. NMI is defined as follows:

$$NMI(U, V) := \frac{MI(U, V)}{\frac{1}{2}(H(U) + H(V))}, \tag{12}$$

[1]http://www.timeseriesclassification.com/index.php

TABLE 3. Summary of real-world time series datasets used in clustering tasks. TS and CL in column names stand for Time Series and Class Labels, respectively.

	Len. of TS	# of vars.	# of CL
ACSF1	200	1,461	10
Adiac	781	177	37
AllGestureWiimoteX	1,000	501	10
AllGestureWiimoteY	1,000	501	10
AllGestureWiimoteZ	1,000	501	10
ArrowHead	211	252	3
Beef	60	471	5
BeetleFly	40	513	2
BirdChicken	40	513	2
BME	180	129	3
Car	120	578	4
CBF	930	129	3
Chinatown	363	25	2
Coffee	56	287	2
Computers	500	721	2
CricketX	780	301	12
CricketY	780	301	12
CricketZ	780	301	12
DiatomSizeReduction	322	346	4
DistalPhalanxOutAgeGrp	539	81	3
DistalPhalanxOutCorrect	876	81	2
DistalPhalanxTW	539	81	6
DodgerLoopDay	158	289	7
DodgerLoopGame	158	289	2
DodgerLoopWeekend	158	289	2
Earthquakes	461	513	2
ECG200	200	97	2
ECGFiveDays	884	137	2
EOGHorizontalSignal	724	1,251	12
EOGVerticalSignal	724	1,251	12
FaceFour	112	351	4
FiftyWords	905	271	50
Fish	350	464	7
Fungi	204	202	18
GestureMidAirD1	338	361	26
GestureMidAirD2	338	361	26
GestureMidAirD3	338	361	26
GesturePebbleZ1	304	456	6
GesturePebbleZ2	304	456	6
GunPoint	200	151	2
GunPointAgeSpan	451	151	2
GunPointMaleVersusFemale	451	151	2
GunPointOldVersusYoung	451	151	2
Ham	214	432	2
Haptics	463	1,093	5
Herring	128	513	2
HouseTwenty	159	2,001	2
InlineSkate	650	1,883	7
InsectEPGRegularTrain	311	602	3
InsectEPGSmallTrain	266	602	3
InsectWingbeatSound	2,200	257	11
ItalyPowerDemand	1,096	25	2
LargeKitchenAppliances	750	721	3

TABLE 3. ***(Continued.)*** **Summary of real-world time series datasets used in clustering tasks. TS and CL in column names stand for Time Series and Class Labels, respectively.**

Lightning2	121	638	2
Lightning7	143	320	7
Meat	120	449	3
MedicalImages	1,141	100	10
MelbournePedestrian	3,633	25	10
MiddlePhalanxOutAgeGrp	554	81	3
MiddlePhalanxOutCorrect	891	81	2
MiddlePhalanxTW	553	81	6
MoteStrain	1,272	85	2
OliveOil	60	571	4
OSULeaf	442	428	6
PhalangesOutlinesCorrect	2,658	81	2
PickupGestureWiimoteZ	100	362	10
PigAirwayPressure	312	2,001	52
PigArtPressure	312	2,001	52
PigCVP	312	2,001	52
Plane	210	145	7
PowerCons	360	145	2
ProximalPhalanxOutAgeGrp	605	81	3
ProximalPhalanxOutCorct	891	81	2
ProximalPhalanxTW	605	81	6
RefrigerationDevices	750	721	3
Rock	70	2,845	4
ScreenType	750	721	3
SemgHandGenderCh2	900	1,501	2
SemgHandMovementCh2	900	1,501	6
SemgHandSubjectCh2	900	1,501	5
ShakeGestureWiimoteZ	100	386	10
ShapeletSim	200	501	2
SmallKitchenAppliances	750	721	3
SmoothSubspace	300	16	3
SonyAIBORobotSurface1	621	71	2
SonyAIBORobotSurface2	980	66	2
Strawberry	983	236	2
SwedishLeaf	1,125	129	15
Symbols	1,020	399	6
SyntheticControl	600	61	6
ToeSegmentation1	268	278	2
ToeSegmentation2	166	344	2
Trace	200	276	4
TwoLeadECG	1,162	83	2
UMD	180	151	3
UWaveGestureLibraryX	4,478	316	8
UWaveGestureLibraryY	4,478	316	8
UWaveGestureLibraryZ	4,478	316	8
Wine	111	235	2
WordSynonyms	905	271	25
Worms	258	901	5
WormsTwoClass	258	901	2

$$MI(U,V) := \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log\left(\frac{N|U_i \cap V_j|}{|U_i||V_j|}\right), \quad (13)$$

$$H(U) := -\sum_{i=1}^{|U|} P(i)\log\left(P(i)\right), \quad (14)$$

$$H(V) := -\sum_{j=1}^{|V|} P'(j)\log\left(P'(j)\right), \quad (15)$$

where MI and H stand for the mutual information and the entropy, respectively. The other notations - U, V, and N - indicate a set of estimated cluster numbers, a set of class labels, and the number of samples, respectively. In addition, P means a probability and defines as $P(i,j) = |U_i \cap V_j|/N$, $P(i) = |U_i|/N$, and $P(j) = |V_j|/N$. In our experiments, N indicates the number of all subsequences obtained by UFEKT. $|U_i|$ and $|V_j|$ are the number of subsequences belonging to the cluster i and the number of class labels belonging to the class j. The NMI score takes values between zero and one, and higher is better.

F. HYPERPARAMETERS OF CLUSTERING ALGORITHMS

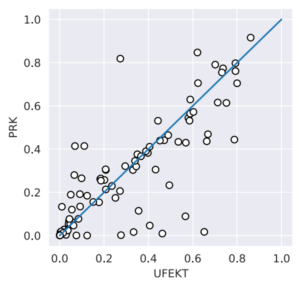
Clustering algorithms generally require hyperparameters to be determined. However, determining the best value for a hyperparameter is one of the well-known problems because clustering is unsupervised and grid search via cross-validation cannot be used. In our scenario, we use simple heuristics to determine them because our goal is to evaluate not clustering algorithms but feature representation, so it is sufficient if clustering algorithms are fairly used.

We use the ground-truth values for the number of clusters required in K-means, AHC, and GMM for fair comparison. In DBSCAN, we use the elbow method [52] to determine ϵ, which is known as the standard heuristic to obtain the suitable values. In following the rule, we set the ϵ to the maximum point of second order differentiation of sorted distances from nearest neighbors. In addition, the value of *MinPts* is set to five for every dataset, which is a default value of DBSCAN in the scikit-learn library implemented in Python. Table 4 provides a summary of the hyperparameters.

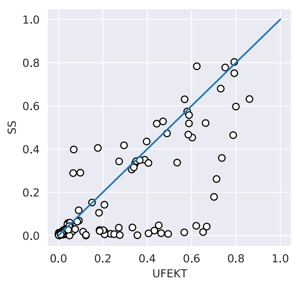
G. EXPERIMENTAL RESULTS

We performed three feature representation algorithms - UFEKT, PRK, and SS - combined with four clustering algorithms - KMeans, DBSCAN, AHC, and GMM - resulting in 12 combinations of feature extraction and clustering algorithms. The results are shown in Fig. 3, Table 5, Table 7 and Table 6. TS and CLs in the table stand for Time Series and Class Labels, respectively. Furthermore, the numbers of datasets that take the best NMI scores per each clustering algorithm are shown in Table 5.

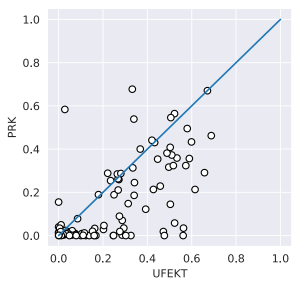
Fig. 3a and Fig. 3b show the results of NMIs obtained by KMeans. Each dot in the scatter plots indicates each UCR dataset; that is, there are 102 dots in the figure. The x-axis and y-axis indicate NMI scores obtained by UFEKT and PRK in Fig. 3a, and by UFEKT and SS in the Fig. 3b, respectively. A straight blue line means the border of NMI scores. If a dot is located under the line, UFEKT is superior to the other method. As for the results of the NMI scores obtained by KMeans shown in Fig. 3a and Fig. 3b, the results of UFEKT are considered to be almost the same as those by PRK, while it is superior to results by SS because many



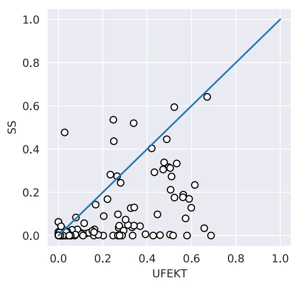
(a) The results of KMeans using UFEKT and PRK.



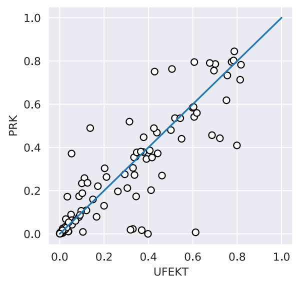
(b) The results of KMeans using UFEKT and SS.



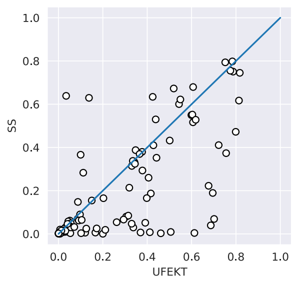
(c) The results of DBSCAN using UFEKT and PRK.



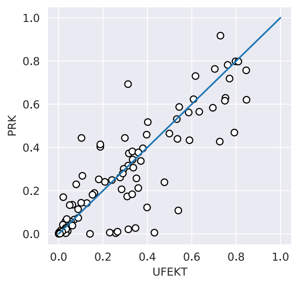
(d) The results of DBSCAN using UFEKT and SS.



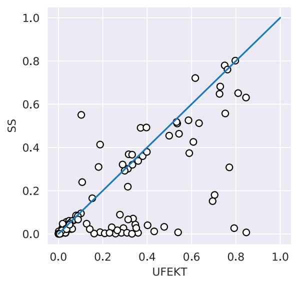
(e) The results of AHC using UFEKT and PRK.



(f) The results of AHC using UFEKT and SS.



(g) The results of GMM using UFEKT and PRK.



(h) The results of GMM using UFEKT and SS.

FIGURE 3. **Scatter plots of NMIs for UCR datasets. Each dot indicates each dataset. A blue straight line in every plot means the border of NMI scores. If a dot is located under the blue line, UFEKT is superior to the other method.**

TABLE 4. **Hyperparameters and their values used in our methods for clustering and outlier detection.**

Hyperparameters	Descriptions	Values (Clustering)	Values (Outlier Detection)
w	Length of subsequence (SS; window size)	variable	2
σ	Used in RBF kernel for UFEKT and PRK	variable	1
R	A rank used in Tucker decomposition for UFEKT	8	-
ϵ	Used in DBSCAN	variable	-
$MinPts$	Used in DBSCAN	5	-
Number of clusters	Used in KMeans, AHC, and GMM	variable	-
i_min	Used in rank selection for UFEKT	-	10
κ	Used in κNN	-	5

dots are located around the straight blue line on PRK and under the line on SS. The numbers of datasets that take the best NMI scores obtained by KMeans are 41 for UFEKT and 42 for PRK, as shown in Table 5. This means that UFEKT might be slightly inferior to PRK; however, it is considered that UFEKT has almost the same capability of PRK. Table 7 shows the average of NMI scores calculated from all datasets. When we focus on KMeans, the average of NMI scores obtained from UFEKT is higher than the other scores from PRK and SS; that is, UFEKT is superior to the others. Moreover, the results obtained by DBSCAN, AHC, and GMM, using UFEKT are superior to those by PRK and SS for all datasets, as shown in Fig. 3 and Table 7. Those results lead to the conclusion that UFEKT has higher potential to extract features from multivariate time series than PRK and SS.

TABLE 5. **The count of the best NMI scores for each clustering algorithm across all datasets. Total counts are equal to the number of datasets. Best counts are denoted in bold.**

	UFEKT	PRK	SS	Total
KMeans	41	**42**	19	102
DBSCAN	**76**	17	9	102
AHC	**43**	39	20	102
GMM	**47**	40	15	102

V. EXPERIMENTS FOR OUTLIER DETECTION

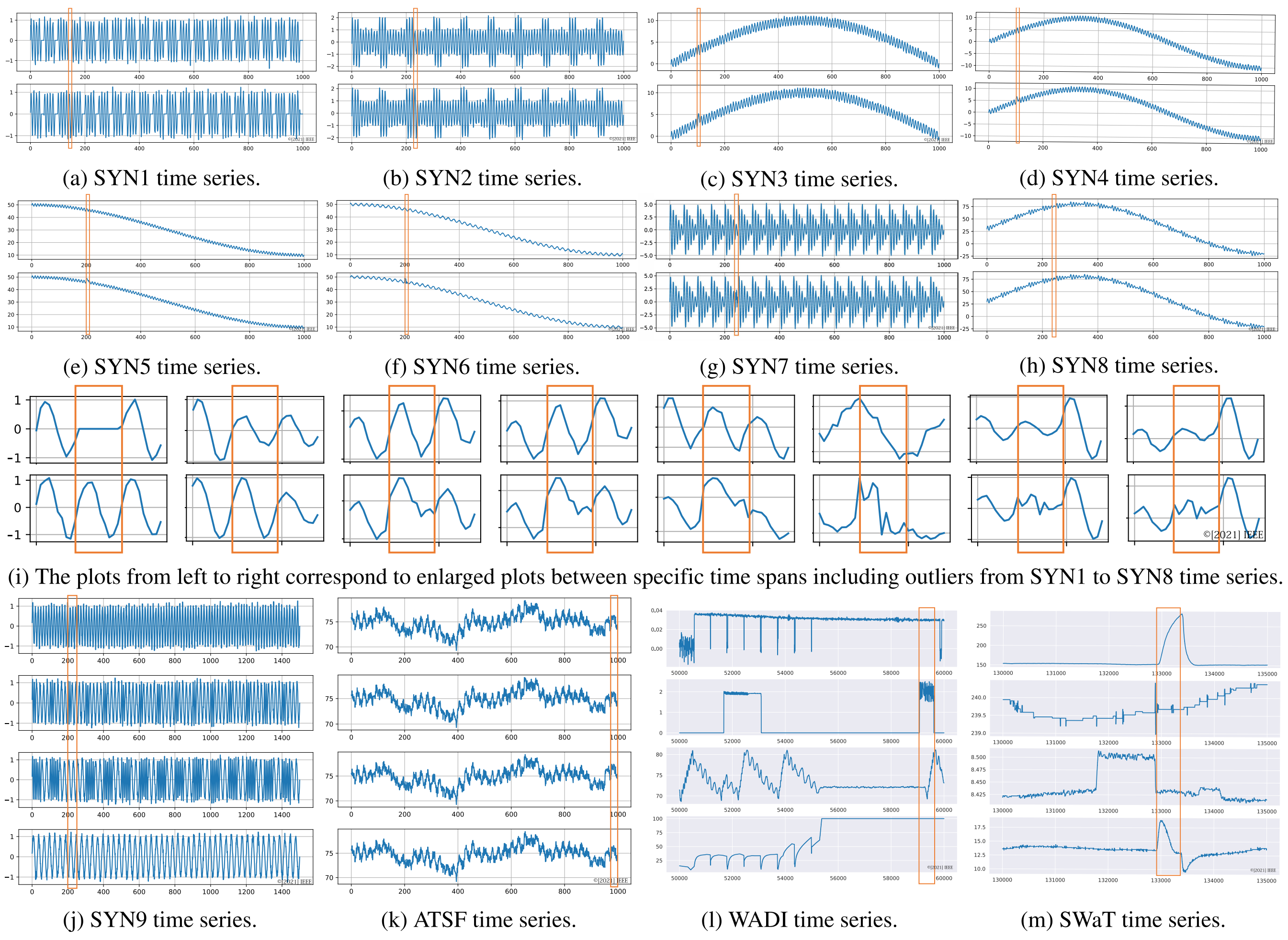
Similar to the clustering tasks, to examine the effectiveness of extracted feature vectors, we empirically evaluated the effectiveness of extracted feature vectors by UFEKT on synthetic and real-world datasets for outlier detection. We compared UFEKT with two other feature representation

TABLE 6. The NMI scores for UCR datasets. Best scores are denoted in bold.

	KMeans			DBSCAN			AHC			GMM		
Name of Datasets	UFEKT	PRK	SS	UFEKT	PRK	SS	UFEKT	PRK	SS	UFEKT	PRK	SS
ACSF1	0.701	**0.791**	0.179	0.599	0.433	0.129	0.695	0.755	0.189	0.704	0.763	0.180
Adiac	0.330	0.303	0.306	0.202	0.028	0.000	**0.331**	0.305	0.315	0.313	0.316	0.301
AllGestureWiimoteX	0.580	0.544	0.574	0.533	0.359	0.333	**0.599**	0.584	0.550	0.586	0.562	0.526
AllGestureWiimoteY	0.588	0.564	0.519	0.510	0.373	0.273	**0.603**	0.588	0.552	0.544	0.587	0.463
AllGestureWiimoteZ	0.603	0.572	0.453	**0.657**	0.291	0.034	0.606	0.541	0.516	0.608	0.623	0.426
ArrowHead	0.208	0.303	0.018	0.368	**0.400**	0.044	0.391	0.347	0.052	0.293	0.301	0.027
Beef	0.398	0.382	0.435	**0.497**	0.316	0.316	0.378	0.447	0.293	0.317	0.373	0.368
BeetleFly	0.273	**0.819**	0.343	0.447	0.354	0.098	0.428	0.751	0.410	0.313	0.693	0.218
BirdChicken	**0.432**	0.305	0.023	0.393	0.122	0.006	0.337	0.272	0.030	0.021	0.170	0.040
BME	**0.567**	0.089	0.015	0.000	0.015	0.006	0.120	0.109	0.006	0.127	0.142	0.047
Car	0.005	0.004	0.002	**0.025**	0.003	0.000	0.001	0.004	0.002	0.002	0.003	0.004
CBF	0.042	0.067	0.057	0.000	0.040	0.017	0.051	**0.090**	0.062	0.070	0.066	0.064
Chinatown	0.589	0.629	0.557	0.339	0.539	0.520	0.520	0.535	0.673	**0.752**	0.629	0.557
Coffee	0.621	**0.847**	0.045	0.332	0.677	0.038	0.506	0.763	0.009	**0.847**	0.621	0.007
Computers	0.235	0.229	0.008	**0.326**	0.000	0.127	0.262	0.197	0.054	0.079	0.229	0.085
CricketX	0.048	0.053	0.059	**0.122**	0.000	0.009	0.045	0.061	0.041	0.045	0.046	0.048
CricketY	0.040	0.053	0.052	**0.138**	0.000	0.013	0.044	0.049	0.058	0.036	0.061	0.056
CricketZ	0.052	0.056	0.059	**0.160**	0.000	0.000	0.055	0.064	0.050	0.048	0.049	0.061
DiatomSizeReduction	0.014	0.018	0.017	**0.041**	0.018	0.000	0.011	0.018	0.016	0.016	0.014	0.024
DistalPhalanxOutAgeGrp	0.272	0.206	0.036	0.221	0.288	0.168	**0.305**	0.212	0.080	0.284	0.206	0.005
DistalPhalanxOutCorrect	0.031	0.013	0.007	0.027	0.016	0.024	0.040	0.011	0.010	**0.041**	0.013	0.023
DistalPhalanxTW	0.053	0.064	0.033	**0.100**	0.000	0.013	0.054	0.063	0.034	0.056	0.055	0.023
DodgerLoopDay	0.623	0.705	0.783	0.504	0.145	0.313	0.607	**0.795**	0.679	0.617	0.730	0.721
DodgerLoopGame	0.111	**0.414**	0.018	0.335	0.313	0.000	0.054	0.372	0.003	0.188	0.403	0.008
DodgerLoopWeekend	0.069	0.414	0.398	0.250	0.189	0.436	0.138	0.490	**0.629**	0.188	0.414	0.413
Earthquakes	0.029	0.003	0.013	0.015	0.005	0.007	0.014	0.004	0.014	**0.032**	0.003	0.005
ECG200	0.000	0.000	0.005	**0.009**	0.002	0.000	0.002	0.002	0.004	0.000	0.001	0.002
ECGFiveDays	0.471	0.441	0.528	0.340	0.186	0.046	0.438	0.469	0.530	**0.536**	0.439	0.511
EOGHorizontalSignal	0.345	0.320	0.334	0.272	0.259	0.036	0.343	0.357	0.332	0.371	0.337	**0.491**
EOGVerticalSignal	0.001	0.002	0.013	**0.031**	0.023	0.000	0.003	0.006	0.000	0.003	0.008	0.001
FaceFour	0.055	0.120	0.038	0.086	0.078	0.028	0.086	0.078	0.070	0.062	**0.135**	0.060
FiftyWords	0.340	0.345	0.315	0.286	0.002	0.002	0.335	**0.354**	0.338	0.334	0.344	0.320
Fish	0.048	0.054	0.044	**0.074**	0.010	0.000	0.052	0.049	0.035	0.052	0.046	0.049
Fungi	0.736	**0.774**	0.359	0.688	0.462	0.000	0.756	0.733	0.374	0.771	0.719	0.308
GestureMidAirD1	0.350	0.376	0.344	0.294	0.035	0.023	0.348	0.377	**0.388**	0.332	0.382	0.366
GestureMidAirD2	0.389	0.390	0.351	0.246	0.000	0.000	0.377	0.379	0.380	0.380	**0.396**	0.360
GestureMidAirD3	0.366	0.367	0.349	0.287	0.070	0.000	0.366	**0.381**	0.370	0.360	0.377	0.338
GesturePebbleZ1	0.066	0.280	0.289	**0.314**	0.148	0.049	0.112	0.258	0.283	0.107	0.269	0.240
GesturePebbleZ2	0.099	0.265	0.291	0.204	0.046	0.090	0.100	0.234	**0.366**	0.181	0.252	0.310
GunPoint	0.021	0.028	0.004	0.011	0.050	0.001	0.028	**0.068**	0.009	0.024	0.030	0.011
GunPointAgeSpan	**0.494**	0.233	0.007	0.000	0.155	0.063	0.461	0.270	0.003	0.477	0.239	0.033
GunPointMaleVersusFemale	0.535	0.433	0.338	0.028	0.583	0.477	**0.722**	0.443	0.411	0.590	0.433	0.374
GunPointOldVersusYoung	0.788	0.444	0.465	0.488	0.381	0.445	**0.799**	0.409	0.472	0.103	0.444	0.551
Ham	0.356	0.114	0.001	0.303	0.000	0.074	0.166	0.079	0.006	**0.540**	0.108	0.007
Haptics	0.017	0.018	0.020	**0.057**	0.008	0.000	0.014	0.025	0.015	0.020	0.022	0.017
Herring	0.004	0.003	0.012	0.017	0.003	0.000	0.007	0.004	**0.020**	0.002	0.005	0.012
HouseTwenty	0.178	0.154	**0.405**	0.268	0.211	0.098	0.345	0.173	0.324	0.399	0.122	0.379
InlineSkate	0.023	0.025	0.025	**0.105**	0.009	0.007	0.023	0.019	0.017	0.021	0.026	0.028
InsectEPGRegularTrain	0.489	0.464	0.473	0.522	0.564	**0.594**	0.501	0.480	0.432	0.499	0.464	0.454
InsectEPGSmallTrain	0.443	0.531	0.518	**0.671**	0.670	0.641	0.618	0.560	0.528	0.533	0.531	0.518
InsectWingbeatSound	0.015	0.020	0.013	**0.060**	0.002	0.003	0.017	0.015	0.016	0.015	0.015	0.012

TABLE 6. ***(Continued.)*** **The NMI scores for UCR datasets. Best scores are denoted in bold.**

ItalyPowerDemand	0.003	0.005	0.000	**0.012**	0.002	0.006	0.006	0.004	0.009	0.005	0.002	0.001
LargeKitchenAppliances	0.208	0.213	0.006	0.472	0.018	0.306	0.315	**0.519**	0.084	0.359	0.212	0.005
Lightning2	0.028	0.024	0.027	**0.061**	0.022	0.026	0.039	0.011	0.030	0.031	0.058	0.025
Lightning7	0.091	**0.192**	0.117	0.163	0.033	0.028	0.088	0.174	0.148	0.102	0.144	0.095
Meat	0.452	0.439	0.047	0.580	0.495	0.000	0.702	**0.786**	0.069	0.402	0.517	0.040
MedicalImages	0.029	0.026	0.028	**0.077**	0.005	0.004	0.032	0.032	0.030	0.031	0.032	0.026
MelbournePedestrian	0.800	0.705	0.597	0.503	0.408	0.005	0.814	0.713	0.617	**0.846**	0.757	0.631
MiddlePhalanxOutAgeGrp	0.207	0.306	0.143	0.234	0.254	0.282	0.203	0.304	0.165	**0.337**	0.306	0.071
MiddlePhalanxOutCorrect	0.004	0.016	0.001	**0.041**	0.012	0.018	0.016	0.019	0.007	0.010	0.016	0.002
MiddlePhalanxTW	0.062	0.046	0.018	**0.116**	0.012	0.057	0.056	0.042	0.030	0.062	0.038	0.023
MoteStrain	0.002	0.001	0.002	**0.017**	0.002	0.001	0.001	0.001	0.003	0.000	0.001	0.000
OliveOil	**0.036**	0.029	0.026	0.000	0.000	0.000	0.026	0.033	0.024	0.029	0.024	0.022
OSULeaf	0.712	0.616	0.262	0.506	0.546	0.212	0.677	**0.791**	0.223	0.695	0.584	0.152
PhalangesOutlinesCorrect	0.007	0.007	0.004	**0.021**	0.006	0.005	0.007	0.007	0.001	0.008	0.007	0.001
PickupGestureWiimoteZ	0.861	0.916	0.632	0.433	0.430	0.293	0.787	0.844	0.751	0.729	**0.917**	0.682
PigAirwayPressure	0.752	0.614	0.777	0.563	0.034	0.189	0.752	0.618	**0.794**	0.750	0.616	0.780
PigArtPressure	0.793	0.761	0.751	0.268	0.263	0.000	0.775	**0.796**	0.755	0.762	0.782	0.760
PigCVP	0.792	0.797	**0.803**	0.573	0.324	0.079	0.784	0.802	0.798	0.797	0.798	0.802
Plane	0.092	**0.134**	0.069	0.005	0.034	0.014	0.097	0.107	0.090	0.088	0.114	0.083
PowerCons	**0.652**	0.017	0.016	0.276	0.019	0.000	0.035	0.172	0.639	0.019	0.042	0.047
ProximalPhalanxOutAgeGrp	0.183	0.261	0.105	0.265	0.285	0.275	**0.293**	0.276	0.066	0.277	0.261	0.089
ProximalPhalanxOutCorct	0.020	0.028	0.007	**0.035**	0.008	0.017	0.026	0.021	0.017	0.032	0.028	0.005
ProximalPhalanxTW	0.037	0.024	0.028	**0.056**	0.002	0.000	0.031	0.030	0.014	0.034	0.022	0.024
RefrigerationDevices	0.463	0.009	0.011	**0.561**	0.000	0.177	0.330	0.023	0.047	0.432	0.006	0.012
Rock	0.405	0.411	0.336	0.280	0.287	0.244	0.442	0.373	0.352	0.299	**0.444**	0.292
ScreenType	0.251	0.174	0.006	**0.458**	0.229	0.002	0.412	0.202	0.008	0.309	0.173	0.005
SemgHandGenderCh2	0.124	0.184	0.000	0.274	0.089	0.046	0.200	0.130	0.001	**0.332**	0.183	0.001
SemgHandMovementCh2	0.585	0.531	0.467	0.589	0.356	0.170	0.543	0.535	0.601	**0.634**	0.565	0.512
SemgHandSubjectCh2	0.569	0.430	0.630	0.615	0.213	0.234	0.550	0.440	0.622	**0.726**	0.427	0.648
ShakeGestureWiimoteZ	0.731	0.755	0.680	0.421	0.441	0.403	**0.817**	0.783	0.745	0.810	0.798	0.651
ShapeletSim	0.007	0.019	0.004	0.000	0.012	0.063	**0.613**	0.007	0.004	0.258	0.003	0.001
SmallKitchenAppliances	0.406	0.046	0.010	**0.523**	0.058	0.176	0.416	0.354	0.187	0.347	0.027	0.043
SmoothSubspace	**0.663**	0.436	0.520	0.427	0.213	0.000	0.425	0.489	0.634	0.397	0.459	0.492
SonyAIBORobotSurface1	0.003	0.005	0.005	**0.020**	0.006	0.000	0.000	0.002	0.003	0.005	0.004	0.004
SonyAIBORobotSurface2	0.000	0.002	0.004	**0.008**	0.003	0.001	0.000	0.001	0.001	0.001	0.001	0.001
Strawberry	0.050	0.189	0.000	0.180	**0.189**	0.003	0.102	0.188	0.003	0.161	0.189	0.002
SwedishLeaf	0.085	0.077	0.065	**0.153**	0.020	0.022	0.091	0.087	0.062	0.089	0.074	0.067
Symbols	0.016	0.012	0.012	**0.049**	0.000	0.000	0.013	0.016	0.015	0.016	0.011	0.016
SyntheticControl	0.668	0.469	0.042	0.517	0.324	0.000	0.687	0.457	0.039	**0.792**	0.469	0.027
ToeSegmentation1	0.124	0.000	0.003	0.168	0.000	0.143	**0.398**	0.000	0.166	0.141	0.000	0.022
ToeSegmentation2	0.075	0.000	0.029	0.079	0.000	0.084	**0.370**	0.017	0.007	0.266	0.010	0.017
Trace	0.044	**0.077**	0.027	0.012	0.000	0.043	0.071	0.060	0.032	0.037	0.069	0.022
TwoLeadECG	0.001	0.001	0.001	0.009	**0.018**	0.000	0.001	0.001	0.003	0.006	0.001	0.000
UMD	0.295	0.321	**0.417**	0.342	0.245	0.130	0.406	0.386	0.260	0.289	0.280	0.321
UWaveGestureLibraryX	0.185	**0.264**	0.025	0.001	0.001	0.001	0.172	0.221	0.025	0.240	0.249	0.031
UWaveGestureLibraryY	0.202	0.257	0.025	0.002	0.001	0.002	0.125	0.236	0.024	**0.351**	0.257	0.027
UWaveGestureLibraryZ	0.186	0.254	0.021	0.001	0.001	0.001	0.211	**0.263**	0.019	0.209	0.241	0.003
Wine	0.010	**0.133**	0.005	0.111	0.001	0.000	0.041	0.054	0.046	0.050	**0.133**	0.043
WordSynonyms	0.151	0.155	0.153	0.160	0.000	0.016	0.150	0.160	0.155	0.153	**0.181**	0.165
Worms	0.333	0.016	0.037	**0.477**	0.000	0.339	0.319	0.019	0.214	0.314	0.021	0.067
WormsTwoClass	0.276	0.002	0.002	0.248	0.000	**0.536**	0.105	0.009	0.065	0.231	0.006	0.005



(a) SYN1 time series. (b) SYN2 time series. (c) SYN3 time series. (d) SYN4 time series.

(e) SYN5 time series. (f) SYN6 time series. (g) SYN7 time series. (h) SYN8 time series.

(i) The plots from left to right correspond to enlarged plots between specific time spans including outliers from SYN1 to SYN8 time series.

(j) SYN9 time series. (k) ATSF time series. (l) WADI time series. (m) SWaT time series.

FIGURE 4. Examples of synthetic and real-world datasets. Orange solid areas indicate time spans including combinatorial outliers.

TABLE 7. The average of NMI scores across all datasets. Best scores are denoted in bold.

	UFEKT	PRK	SS
KMeans	**0.273**	0.255	0.182
DBSCAN	**0.243**	0.145	0.100
AHC	**0.282**	0.267	0.203
GMM	**0.276**	0.253	0.180

algorithms, the PageRank kernel (PRK) and Subsequence (SS), under four different outlier detection algorithms - κNN, LOF, OCSVM, and IForest - which are popular and widely used in data analysis. Our experimental environments about the hardware and software platforms are also same as the ones for clustering tasks, which is mentioned in Section IV-C.

A. DATASETS

We prepared nine types of synthetic multivariate time series datasets and six types of real-world multivariate time series datasets. Each synthetic dataset includes two or more time series. Outlierness behavior occurs in only one of the time

TABLE 8. Summary of datasets for outlier detection. Datasets SYN$*$ are synthetic, and the others are real-world datasets.

# of Datasets	# of Variables	Range of Outliers	Length of Time Series
SYN1	2	140-150	1,000
SYN2	2	231-240	1,000
SYN3	2	101-110	1,000
SYN4	2	101-110	1,000
SYN5	2	201-210	1,000
SYN6	2	201-210	1,000
SYN7	2	241-250	1,000
SYN8	2	241-250	1,000
SYN9	4	201-250	1,500
ATSF8	8	951-1,000	1,000
ATSF16	16	951-1,000	1,000
ATSF32	32	951-1,000	1,000
ATSF64	64	951-1,000	1,000
WADI	93	9,054-9,644	10,000
SWaT	39	2,918-3,380	5,000

series, shown as an orange solid area in Figures from Fig. 4a to Fig. 4j. All nine synthetic datasets are composed of sine waves or straight lines, and Gaussian noise was added to

TABLE 9. Area under precision-recall curve (AUPRC) for synthetic datasets. OD, FR, PR, PRK and SS stand for Outlier Detection, Feature Representation, PageRank, PageRank Kernel and SubSequence, respectively. Mean±S.D. in ten trials are shown in the table. Best scores are denoted in bold.

OD		κNN			LOF		-
FR	UFEKT	PRK	SS	UFEKT	PRK	SS	-
SYN1	**0.878** ±0.018	0.815 ±0.010	0.486 ±0.028	0.871 ±0.017	0.736 ±0.034	0.258 ±0.017	
SYN2	0.715 ±0.096	**0.734** ±0.048	0.575 ±0.055	0.699 ±0.095	0.062 ±0.031	0.380 ±0.122	
SYN3	**0.988** ±0.029	0.625 ±0.167	0.016 ±0.005	0.961 ±0.066	0.118 ±0.041	0.011 ±0.002	
SYN4	**0.964** ±0.034	0.573 ±0.111	0.010 ±0.002	0.920 ±0.057	0.114 ±0.073	0.011 ±0.003	
SYN5	**0.992** ±0.017	0.046 ±0.009	0.007 ±0.001	0.933 ±0.172	0.006 ±0.000	0.011 ±0.003	-
SYN6	0.026 ±0.012	0.063 ±0.019	0.182 ±0.051	0.096 ±0.061	0.007 ±0.001	0.139 ±0.043	
SYN7	0.872 ±0.026	0.779 ±0.057	0.010 ±0.001	**0.884** ±0.026	0.047 ±0.029	0.060 ±0.069	
SYN8	0.118 ±0.040	0.012 ±0.001	0.006 ±0.000	0.010 ±0.002	0.007 ±0.000	0.007 ±0.000	
SYN9	**0.425** ±0.035	0.368 ±0.041	0.033 ±0.001	0.248 ±0.035	0.069 ±0.020	0.035 ±0.002	
Average	**0.664**	0.446	0.147	0.625	0.129	0.101	-

OD		OCSVM			IForest		PR
FR	UFEKT	PRK	SS	UFEKT	PRK	SS	PRK
SYN1	0.841 ±0.026	0.006 ±0.000	0.007 ±0.000	0.717 ±0.050	0.017 ±0.000	0.013 ±0.001	0.618 ±0.012
SYN2	0.570 ±0.127	0.006 ±0.000	0.018 ±0.000	0.153 ±0.132	0.006 ±0.000	0.019 ±0.001	0.431 ±0.101
SYN3	0.908 ±0.066	0.006 ±0.000	0.008 ±0.000	0.168 ±0.293	0.006 ±0.000	0.011 ±0.001	0.826 ±0.098
SYN4	0.799 ±0.069	0.006 ±0.000	0.007 ±0.000	0.019 ±0.019	0.006 ±0.000	0.007 ±0.000	0.798 ±0.061
SYN5	0.827 ±0.277	0.006 ±0.000	0.010 ±0.000	0.629 ±0.241	0.006 ±0.000	0.008 ±0.001	0.030 ±0.010
SYN6	0.011 ±0.001	0.007 ±0.001	0.008 ±0.000	0.012 ±0.001	0.007 ±0.000	0.014 ±0.005	**0.197** ±0.042
SYN7	0.737 ±0.072	0.006 ±0.000	0.006 ±0.000	0.109 ±0.218	0.007 ±0.000	0.006 ±0.000	0.751 ±0.024
SYN8	0.029 ±0.003	0.012 ±0.003	0.014 ±0.000	0.050 ±0.013	0.051 ±0.016	0.006 ±0.000	**0.411** ±0.082
SYN9	0.048 ±0.010	0.020 ±0.001	0.032 ±0.001	0.058 ±0.013	0.030 ±0.002	0.032 ±0.001	0.287 ±0.035
Average	0.53	0.008	0.012	0.213	0.015	0.013	0.483

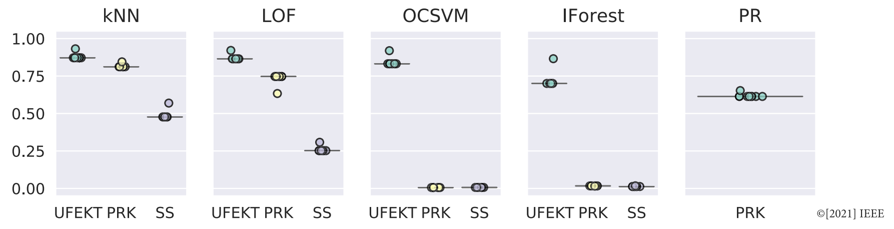
TABLE 10. Area under precision-recall curve (AUPRC) for real-world datasets. Mean±S.D. in 10 trials are shown in the table; however, WADI and SWaT are performed only once.

OD		κNN			LOF		-
FR	UFEKT	PRK	SS	UFEKT	PRK	SS	-
ATSF8	0.877 ±0.038	0.199 ±0.028	0.040 ±0.001	**0.972** ±0.011	0.187 ±0.030	0.067 ±0.001	
ATSF16	0.856 ±0.039	0.260 ±0.038	0.039 ±0.001	0.938 ±0.034	0.066 ±0.009	0.064 ±0.001	
ATSF32	0.765 ±0.057	0.186 ±0.018	0.039 ±0.001	0.705 ±0.053	0.032 ±0.002	0.063 ±0.001	-
ATSF64	**0.672** ±0.059	0.089 ±0.005	0.038 ±0.001	0.434 ±0.029	0.028 ±0.000	0.063 ±0.000	
WADI	0.090 ±0.000	0.047 ±0.000	0.128 ±0.000	0.069 ±0.000	0.064 ±0.000	0.055 ±0.000	
SWaT	0.179 ±0.000	0.085 ±0.000	0.118 ±0.000	0.080 ±0.000	0.070 ±0.000	0.100 ±0.000	
Average	0.573	0.144	0.067	0.533	0.074	0.069	-

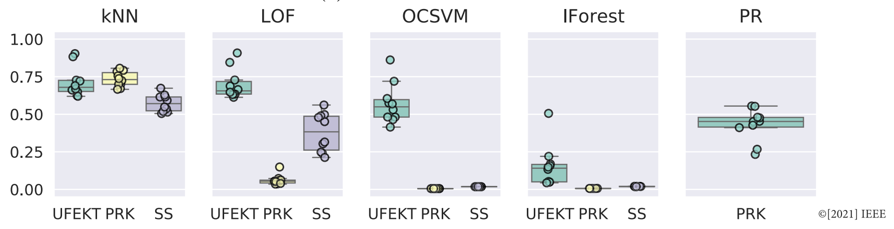
OD		OCSVM			IForest		PR
FR	UFEKT	PRK	SS	UFEKT	PRK	SS	PRK
ATSF8	0.960 ±0.011	0.026 ±0.000	0.034 ±0.000	0.882 ±0.051	0.029 ±0.000	0.035 ±0.001	0.094 ±0.006
ATSF16	**0.957** ±0.016	0.026 ±0.000	0.034 ±0.000	0.784 ±0.124	0.030 ±0.000	0.034 ±0.000	0.072 ±0.002
ATSF32	**0.825** ±0.041	0.027 ±0.000	0.034 ±0.000	0.608 ±0.147	0.030 ±0.000	0.034 ±0.000	0.057 ±0.002
ATSF64	0.543 ±0.060	0.032 ±0.003	0.034 ±0.000	0.414 ±0.159	0.034 ±0.000	0.034 ±0.000	0.044 ±0.002
WADI	0.186 ±0.000	0.032 ±0.000	0.751 ±0.000	0.224 ±0.000	0.033 ±0.000	**0.628** ±0.000	0.061 ±0.000
SWaT	0.230 ±0.000	0.118 ±0.000	**0.881** ±0.000	0.310 ±0.000	0.054 ±0.000	0.365 ±0.000	0.087 ±0.000
Average	**0.617**	0.044	0.295	0.537	0.035	0.188	0.069

every data point, where the noise was generated by Gaussian distribution with zero mean and 0.1 standard deviation $\mathcal{N}(0, 0.1^2)$. We generated each synthetic dataset 10 times with random noise.

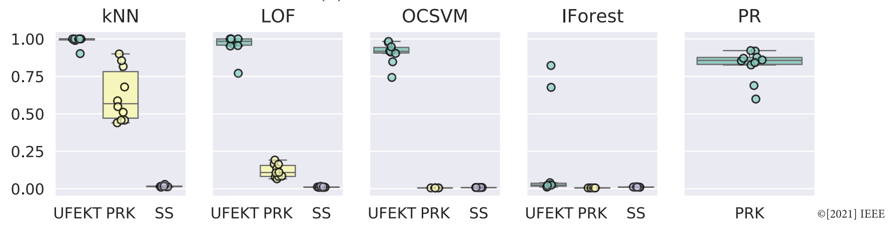
Eight figures (Fig. 4a to Fig. 4h) illustrate synthetic multivariate time series datasets, each of which is composed of two time series. Each time series has 1,000 time stamps, and outliers with the length of 10 or 11 time stamps are injected. The SYN1 time series in Fig. 4a is composed of sine waves and straight line. The SYN2 time series in Fig. 4b is composed of two sine waves with different amplitude. Datasets from SYN3 to SYN6 illustrated in Fig. 4c, Fig. 4d, Fig. 4e, and Fig. 4f are composed of sine waves, and their averages in subsequence sway over time. Moreover, a phase shift occurs between their time series in SYN6. The SYN7 time series in Fig. 4g is composed of sine waves with different amplitude. The SYN8 time series is almost the same as SYN7 except for changes of their averages over time. Enlarged plots between specific time spans that include outliers from SYN1 to SYN8 are shown in Fig. 4i. The SYN9 dataset in Fig. 4j has a set of four time series and each time series has 1,500 time stamps. Their wavelengths of the top and the bottom time



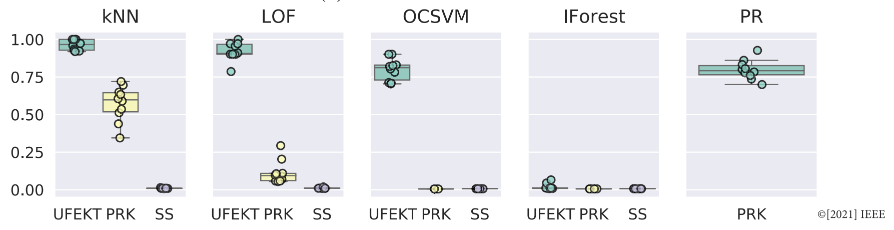
(a) AUPRCs for SYN1.



(b) AUPRCs for SYN2.



(c) AUPRCs for SYN3.



(d) AUPRCs for SYN4.

FIGURE 5. AUPRCs for synthetic datasets (SYN1/2/3/4). PR, PRK, and SS stand for PageRank, PageRank Kernel, and SubSequence, respectively. We generated each dataset 10 times and each circle in the plot corresponds to each trial of the respective method.

series are 15 and 30 time stamps, respectively. The second time series is combined with two wavelengths of 10 and 20 time stamps. Similarly, the third time series is combined with two wavelengths of ten and twenty-five. Consequently, all of four time series are composed of different wavelengths.

Real-world datasets *ambient temperature system failure* (ATSF) are shown in Fig. 4k, which come from the Numenta Anomaly Benchmark (NAB) v1.1 publicly available[2] [53] and record ambient temperature in an office setting measured every hour. Since it is hard to find ground truth combinatorial outliers in multivariate time series from real-world datasets, we collected a univariate time series and artificially simulated combinatorial outliers on it. Time series we extracted have successive 1,000 time stamps out of 7,267 where it corresponds between November 1, 2013 and December 13, 2013, and we created 8, 16, 32, and 64 variants by adding noise generated by Gaussian distribution with $\mathcal{N}(0, 0.1^2)$. Furthermore, we artificially injected outliers in the range from 951 to 1,000 by adding about one percent values of the original values to only a single variable. For example, such outliers simulate the abnormal drift of a temperature sensor. Similar to synthetic datasets, we repeated the generation of each ATSF dataset 10 times by adding random noise.

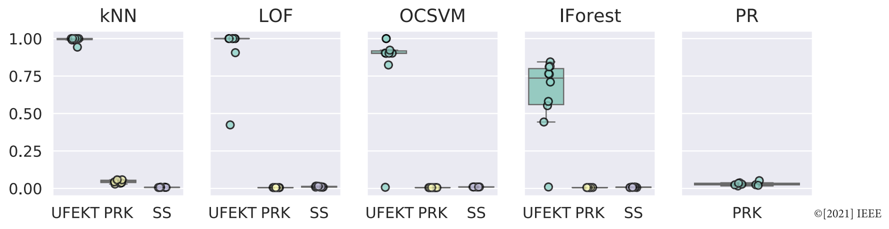
Furthermore, we employed other types of multivariate time series:[3] *Water Distribution* (WADI)[4] illustrated in Fig. 4l and itSecure Water Treatment (SWaT)[5] illustrated in Fig. 4m. These real-world multivariate time series datasets are also publicly available and outliers have been already included in them. We extracted 10,000 successive time stamps out of 172,801 time stamps between 50,001 and 60,000 from WADI

[2]https://github.com/numenta/NAB/tree/master/data

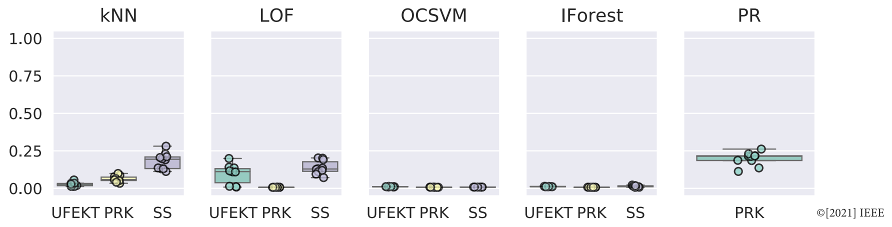
[3]iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design.

[4]https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_wadi/

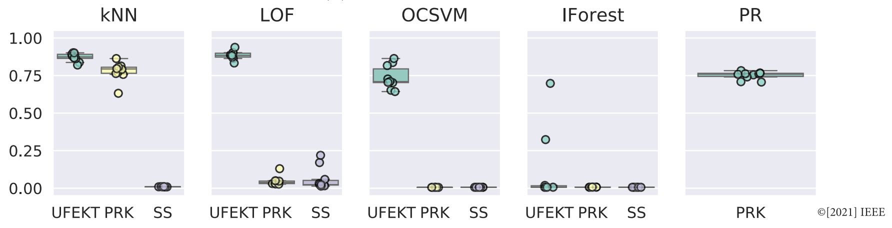
[5]https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_swat/



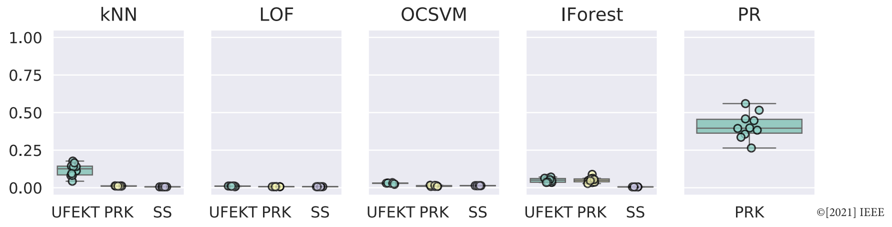
(a) AUPRCs for SYN5.



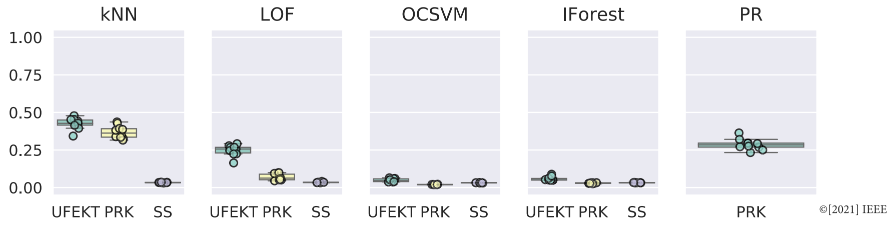
(b) AUPRCs for SYN6.



(c) AUPRCs for SYN7.



(d) AUPRCs for SYN8.



(e) AUPRCs for SYN9.

FIGURE 6. AUPRCs for synthetic datasets (SYN5/6/7/8/9). PR, PRK, and SS stand for PageRank, PageRank Kernel, and SubSequence, respectively. We generated each dataset 10 times and each circle in the plot corresponds to each trial of the respective method.

and then 5,000 out of 449,919 time stamps between 130,000 and 135,000 from SWaT. Their subsets include some outliers that can be obviously and visually identified as outliers. Note that, in comparison with ATSF, we did not artificially inject any outliers in both WADI and SWaT datasets.

A summary of datasets for outlier detection is shown in Table 8.

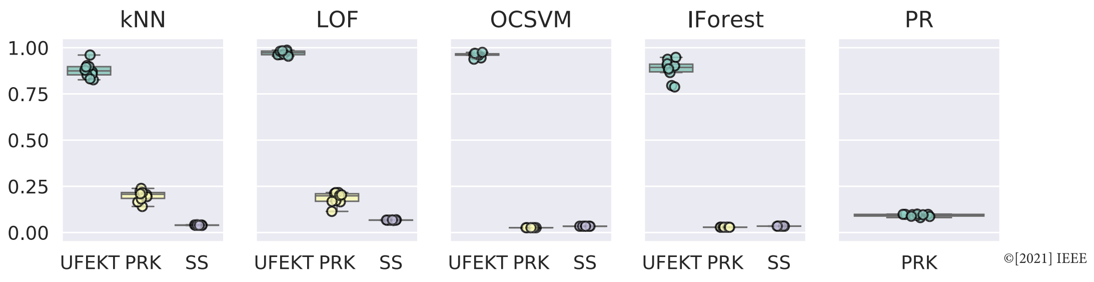
B. EVALUATION METRICS

The effectiveness of outlier detection was evaluated by the *area under precision-recall curve* (AUPRC) [40]. The precision and recall used in AUPRC are defined as follows:

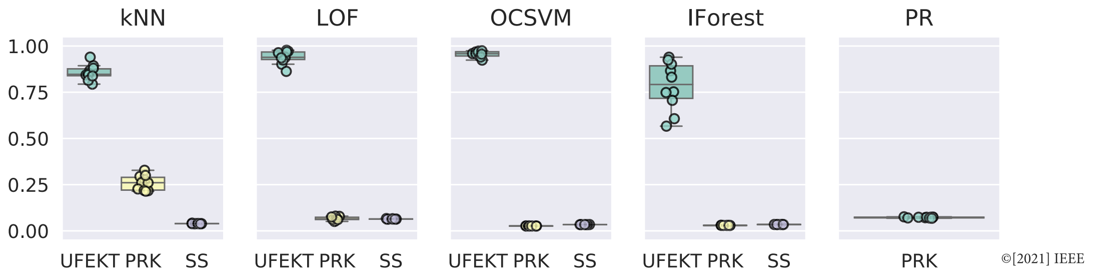
$$Precision(t) := \frac{|S(t) \cap G|}{|S(t)|}, \tag{16}$$

$$Recall(t) := \frac{|S(t) \cap G|}{|G|}, \tag{17}$$

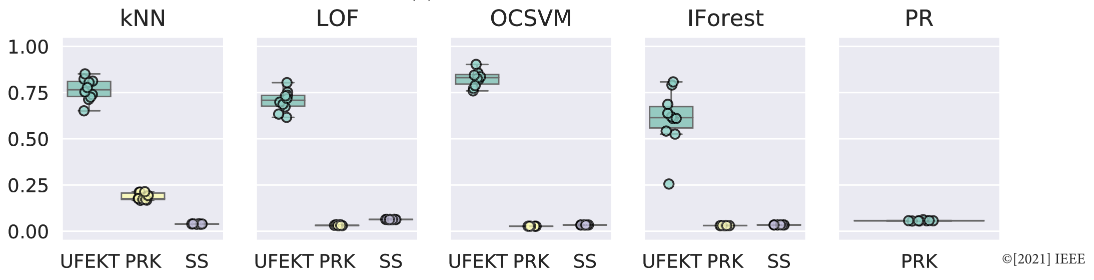
where $S(t)$, G, and t indicate the datasets that infer to be outliers, dataset of ground truth, and a given threshold, respectively. Once a threshold t is set to any specific value, the values of precision and recall are calculated; for example, AUPRC can be obtained by calculating while changing the



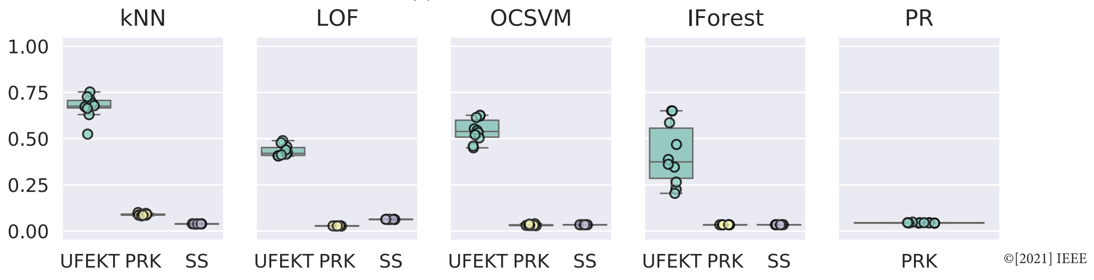
(a) AUPRCs for ATSF8.



(b) AUPRCs for ATSF16.



(c) AUPRCs for ATSF32.



(d) AUPRCs for ATSF64.

FIGURE 7. AUPRCs for real-world datasets (ATSF8/16/32/64). PR, PRK, and SS stand for PageRank, PageRank Kernel, and SubSequence, respectively. We generated each dataset 10 times and each circle in the plot corresponds to each trial of the respective method.

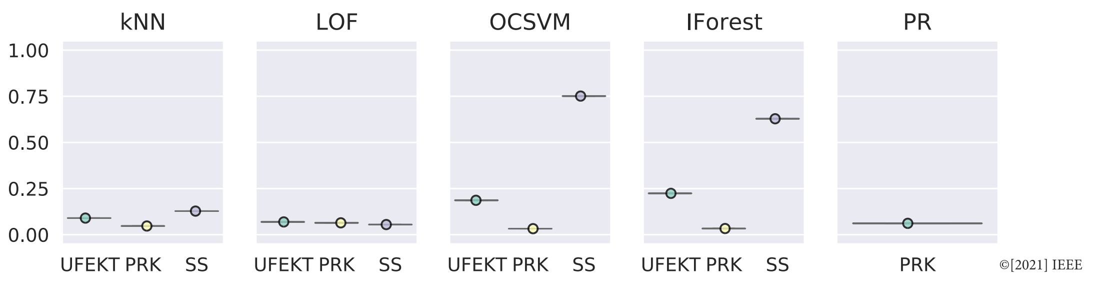
threshold t. The AUPRC score takes values between zero and one, and higher is better.

C. HYPERPARAMETERS OF UFEKT FOR OUTLIER DETECTION ALGORITHMS

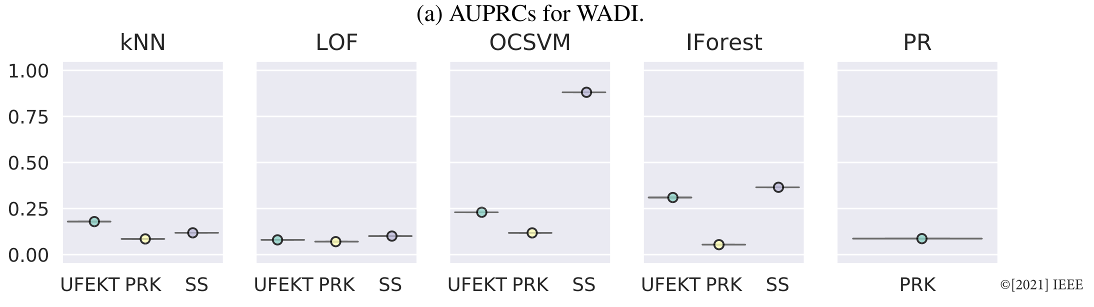
Some hyperparameters are shown in Table 4. Length of subsequence w, which is also known as window size, and σ, which is known to be an appropriate value for normalized datasets, are set to fixed values of two and one, respectively. The value of length of subsequence was the default setting and it becomes low resolution (that is, low AUPRCs) if the length increases further. Moreover, we set $i_{min} = 10$ used in rank selection for UFEKT and $\kappa = 5$, which is the default setting used in κNN.

D. EXPERIMENTAL RESULTS

Similar to experiments of clustering, we performed three feature representation algorithms - UFEKT, PageRank kernel (PRK), and SubSequence (SS) - combined with four outlier detection algorithms, κNN, LOF, OCSVM, and IForest, resulting in 12 combinations of feature extraction and outlier detection in total. In addition, we also tried to perform one of the popular algorithms called Prophet [54], which is a forecasting procedure for univariate time series. However,



(a) AUPRCs for WADI.



(b) AUPRCs for SWaT.

FIGURE 8. AUPRCs for real-world datasets (WADI and SWaT). PR, PRK, and SS stand for PageRank, PageRank Kernel, and SubSequence, respectively.

we could not get results of Prophet due to high computational cost and it was hard to correctly determine the date and time information for our datasets, which is required for Prophet as additional input.

Each parameter that we used in the algorithms is shown in Table 4. We set $\kappa = 5$, which is the default setting, and set $\sigma = 1$, which is known to be an appropriate value for normalized datasets. The window size was set to be two to avoid low resolution and to improve accuracy of AUPRCs. If the window size increases further, it becomes low resolution, resulting in low AUPRCs.

Results are summarized in Table 9 and Table 10. OD, FR, PR, PRK, and SS in the table stand for Outlier Detection, Feature Representation, PageRank, PageRank Kernel, and SubSequence, respectively. All datasets except for WADI and SWaT were created 10 times with adding Gaussian noises. Therefore, each method are performed 10 times for each dataset except for WADI and SWaT. In addition to the tables, we show plots of the results of every dataset in Fig. 5, Fig. 6, Fig. 7, and Fig. 8.

1) SYN DATASETS

Results of SYN datasets are shown in Table 9. Our algorithm, UFEKT, is superior to the other feature representation algorithms PageRank Kernel (PRK) and SubSequence (SS) in all synthetic datasets except for SYN2, SYN6, and SYN8, and shows the best performance on average. In comparison with PRK, kernels used in our algorithm do not sum up elements in a row of a kernel matrix that represents association between subsequences. Therefore, it is considered that UFEKT has high capability of feature representations. Moreover, by using the Gaussian kernel, it is expected that UFEKT can reduce noise and extract features easier than SS can. Furthermore, it is also interesting that κNN, which is an outlier detection algorithm, also tends to show better results than the other algorithms. The reasons why κNN, which is a distance-based algorithm, has a good result is that outlier points tend to be placed far away from normal points in a kernel space.

2) ATSF DATASETS

The results of ATSF datasets are also shown in Table 10. UFEKT is superior to PRK and SS in all ATSF datasets. Those ATSF datasets are prepared for detecting outliers that cannot be found if we look at each multivariate time series separately. In the case of ATSF8, there are eight similar time series with artificially injected noise. The results of UFEKT remain highly accurate even if the number of variables are increased. This is one of characteristics of UFEKT.

3) WADI AND SWaT DATASETS

Results of the WADI and SWaT datasets are shown in Table 10. Their results are different from the other datasets because SS is superior to our algorithm UFEKT. Those datasets have different tendencies to other datasets, SYN and ATSF, because each time series has a unique shape shown in Fig.4l and Fig. 4m, and the differences between variables are not dominant. This means that feature extraction and outlier detection to these datasets are fundamentally difficult.

E. RANK DEPENDENCIES OF AUPRCs

Finally, we examined the effectiveness of our parameter selection algorithm in Algorithm 2, which tries to find a good choice of a parameter, rank of a core tensor, used in UFEKT.

TABLE 11. Comparison of AUPRCs between our algorithm in Algorithm 2 and grid search for all ranks. The latter can be possible only when the ground-truth is given.

Datasets	Our algorithm	Optimal value
SYN1	0.878±0.018	0.914±0.009
SYN2	0.715±0.096	0.943±0.053
SYN3	0.988±0.029	0.999±0.003
SYN4	0.964±0.034	0.994±0.008
SYN5	0.992±0.017	0.999±0.003
SYN6	0.026±0.012	0.245±0.057
SYN7	0.872±0.026	0.905±0.029
SYN8	0.118±0.004	0.271±0.076
SYN9	0.425±0.035	0.485±0.039
ATSF8	0.877±0.038	0.950±0.026
ATSF16	0.856±0.039	0.928±0.016
ATSF32	0.765±0.057	0.893±0.020
ATSF64	0.672±0.059	0.852±0.019
WADI	0.090	0.172
SWaT	0.179	0.228

We compare the AUPRCs obtained by Algorithm 2 and that by the optimal rank obtained by the grid search. Note that the latter is possible only when the ground truth labels for outliers are given and it is not possible in practice. Fig. 2 shows the rank dependencies of AUPRCs in our experiments. Our algorithm indicates that the rank should be set as 14 as it takes the highest normalized κNN distance, and the bottom plot shows that the corresponding AUPRC is a good choice. Table 11 shows that our algorithm is almost successful in finding ranks for all datasets, and the loss of AUPRCs is marginal. These results show that our heuristic rank selection algorithm is effective in the task of outlier detection.

VI. CONCLUSION

In this paper, we have proposed a new algorithm, called *Unsupervised Feature Extraction using Kernel Method and Tucker Decomposition (UFEKT)*, that can extract feature vectors from multivariate time series using a kernel method and Tucker decomposition in an unsupervised manner. The proposal is a simple algorithm and can be combined with any downstream data mining tasks. To empirically examine the effectiveness of the strategy of combining feature representation and applications, such as clustering and outlier detection, we have examined the effectiveness of UFEKT using real-world and synthetic multivariate time series datasets, and shown that using UFEKT is more effective than directly using subsequences in both clustering and outlier detection. Furthermore, one of the hyperparameters of UFEKT - the rank value for tensor decomposition - can be easily determined without using any specific algorithm in clustering. Our results indicate that feature vectors obtained by UFEKT are flexible in the sense that they can be combined with many existing data mining algorithms for various applications.

REFERENCES

[1] B. Ballinger, J. Hsieh, A. Singh, N. Sohoni, J. Wang, G. Tison, G. Marcus, J. Sanchez, C. Maguire, J. Olgin, and M. Pletcher, "DeepHeart: Semi-supervised sequence learning for cardiovascular risk prediction," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 2079–2086.

[2] Z. Chen et al., "A CRISPR/Cas12a-empowered surface plasmon resonance platform for rapid and specific diagnosis of the omicron variant of SARS-CoV-2," *Nat. Sci. Rev.*, vol. 9, no. 8, Aug. 2022, Art. no. nwac104.

[3] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[4] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, Sep. 1966.

[5] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. (Jul. 2015). *The UCR Time Series Classification Archive*. [Online]. Available: https://www.cs.ucr.edu/~eamonn/time_series_data/

[6] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 128–137, Mar. 1982.

[7] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, 1967, pp. 281–297.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Int. Conf. Knowl. Discovery Data Mining (KDD)*, 1996, pp. 226–231.

[9] W. H. E. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *J. Classification*, vol. 1, no. 1, pp. 7–24, Dec. 1984.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[11] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: Algorithms and applications," *VLDB J. Int. J. Very Large Data Bases*, vol. 8, nos. 3–4, pp. 237–253, Feb. 2000.

[12] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2000, pp. 93–104.

[13] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," in *Proc. Int. Joint Conf. Neural Netw.*, 2003, pp. 1741–1745.

[14] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 413–422.

[15] K. Matsue and M. Sugiyama, "Unsupervised tensor based feature extraction and outlier detection for multivariate time series," in *Proc. IEEE 8th Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2021, pp. 1–12.

[16] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Proc. Int. Conf. Found. Data Org. Algorithms*, in Lecture Notes in Computer Science, vol. 730, 1993, pp. 69–84.

[17] K. Kawagoe and T. Ueda, "A similarity search method of time series data with combination of Fourier and wavelet transforms," in *Proc. 9th Int. Symp. Temporal Represent. Reasoning*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 86–92.

[18] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, "Time-series clustering—A decade review," *Inf. Syst.*, vol. 53, pp. 16–38, Oct. 2015.

[19] W. Liu and L. Shao, "Research of SAX in distance measuring for financial time series data," in *Proc. 1st Int. Conf. Inf. Sci. Eng.*, Dec. 2009, pp. 935–937.

[20] H. Cheng, P.-N. Tan, C. Potter, and S. Klooster, "Detection and characterization of anomalies in multivariate time series," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2009, pp. 413–424.

[21] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 33, 2019, pp. 1409–1416.

[22] R. Dahlhaus, "On the Kullback–Leibler information divergence of locally stationary processes," *Stochastic Processes Appl.*, vol. 62, no. 1, pp. 139–168, Mar. 1996.

[23] D. Hallac, S. Vare, S. Boyd, and J. Leskovec, "Toeplitz inverse covariance-based clustering of multivariate time series data," in *Proc. Data Mining Knowl. Discovery (SIGKDD)*, 2017, pp. 215–223.

[24] J. Zakaria, A. Mueen, and E. Keogh, "Clustering time series using unsupervised-shapelets," in *Proc. IEEE 12th Int. Conf. Data Mining*, Dec. 2012, pp. 785–794.

[25] Y. Kakizawa, R. H. Shumway, and M. Taniguchi, ''Discrimination and clustering for multivariate time series,'' *J. Amer. Stat. Assoc.*, vol. 93, no. 441, pp. 328–340, Mar. 1998.

[26] S. Zolhavarieh, S. Aghabozorgi, and Y. W. Teh, ''A review of subsequence time series clustering,'' *Sci. World J.*, vol. 2014, pp. 1–19, Jul. 2014.

[27] S. Aghabozorgi, M. R. Saybani, and T. Y. Wah, ''Incremental clustering of time-series by fuzzy clustering,'' *J. Inf. Sci. Eng.*, vol. 28, no. 4, pp. 671–688, 2012.

[28] J. Lin, E. Keogh, and W. Truppel, ''Clustering of streaming time series is meaningless,'' in *Proc. 8th ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery*, Jun. 2003, pp. 56–65.

[29] T. Ide, ''Why does subsequence time-series clustering produce sine waves?'' in *Knowledge Discovery in Databases: PKDD 2006*. Berlin, Germany: Springer, 2006, pp. 211–222.

[30] J. Lee, H. S. Choi, Y. Jeon, Y. Kwon, D. Lee, and S. Yoon, ''Detecting system anomalies in multivariate time series with information transfer and random walk,'' in *Proc. 5th IEEE/ACM Int. Conf. Big Data Comput., Appl. Technol.*, Dec. 2018, pp. 71–80.

[31] K. Matsue and M. Sugiyama, ''Unsupervised feature extraction from multivariate time series for outlier detection,'' *Intell. Data Anal.*, vol. 26, no. 6, pp. 1451–1467, Nov. 2022.

[32] N. Takeishi and T. Yairi, ''Anomaly detection from multivariate time-series with sparse representation,'' in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2014, pp. 2651–2656.

[33] Z. Li, W. Chen, and D. Pei, ''Robust and unsupervised KPI anomaly detection based on conditional variational autoencoder,'' in *Proc. IEEE 37th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Nov. 2018, pp. 1–9.

[34] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, ''Robust anomaly detection for multivariate time series through stochastic recurrent neural network,'' in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2828–2837.

[35] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, ''LSTM-based encoder–decoder for multi-sensor anomaly detection,'' in *Proc. Int. Conf. Mach. Learn. Anomaly Detection Workshop*, 2016, pp. 1–5.

[36] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, ''Deep structured energy based models for anomaly detection,'' in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, vol. 3, 2016, pp. 1742–1751.

[37] C. Zhou and R. C. Paffenroth, ''Anomaly detection with robust deep autoencoders,'' in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 665–674.

[38] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, ''Deep autoencoding Gaussian mixture model for unsupervised anomaly detection,'' in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–19.

[39] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, ''USAD: Unsupervised anomaly detection on multivariate time series,'' in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 3395–3404.

[40] C. C. Aggarwal, *Outlier Analysis*. Cham, Switzerland: Springer, 2017.

[41] C. C. Aggarwal and S. Sathe, *Outlier Ensembles*. Cham, Switzerland: Springer, 2017.

[42] N. N. R. R. Suri, M. N. Murty, and G. Athithan, *Outlier Detection: Techniques and Applications* (Intelligent Systems Reference Library), vol. 155. Cham, Switzerland: Springer, 2019.

[43] V. Chandola, A. Banerjee, and V. Kumar, ''Anomaly detection,'' *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[44] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, ''Outlier detection for temporal data: A survey,'' *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 9, pp. 2250–2267, Sep. 2014.

[45] H. Wang, M. J. Bah, and M. Hammad, ''Progress in outlier detection techniques: A survey,'' *IEEE Access*, vol. 7, pp. 107964–108000, 2019.

[46] S. D. Bay and M. Schwabacher, ''Mining distance-based outliers in near linear time with randomization and a simple pruning rule,'' in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2003, pp. 29–38.

[47] M. Sugiyama and K. M. Borgwardt, ''Rapid distance-based outlier detection via sampling,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1–9.

[48] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, ''On the best rank-1 and rank-(R_1, R_2,. . ., R_N) approximation of higher-order tensors,'' *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1324–1342, 2000.

[49] L. De Lathauwer, B. De Moor, and J. Vandewalle, ''A multilinear singular value decomposition,'' *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, 2000.

[50] M. Ishteva, L. D. Lathauwer, P.-A. Absil, and S. Van Huffel, ''Differential-geometric Newton method for the best rank-(R_1, R_2, R_3) approximation of tensors,'' *Numer. Algorithms*, vol. 51, no. 2, pp. 179–194, 2009.

[51] M. Ishteva, P.-A. Absil, S. Van Huffel, and L. De Lathauwer, ''Best low multilinear rank approximation of higher-order tensors, based on the Riemannian trust-region scheme,'' *SIAM J. Matrix Anal. Appl.*, vol. 32, no. 1, pp. 115–135, Jan. 2011.

[52] C. Yuan and H. Yang, ''Research on K-value selection method of K-means clustering algorithm,'' *J*, vol. 2, no. 2, pp. 226–235, Jun. 2019.

[53] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, ''Unsupervised real-time anomaly detection for streaming data,'' *Neurocomputing*, vol. 262, pp. 134–147, Nov. 2017.

[54] S. J. Taylor and B. Letham, ''Forecasting at scale,'' *PeerJ*, pp. 1–25, Sep. 2017.

KIYOTAKA MATSUE received the M.Sc. degree in applied physics from the Tokyo Institute of Technology, Tokyo, Japan, in 1998, and the Ph.D. degree in informatics from The Graduate University for Advanced Studies, SOKENDAI, Kanagawa, Japan, in 2022. From 1998 to 2017, he was a Software Engineer for wireless networks and a Researcher of machine learning with Toshiba Corporation. Since 2017, he has been a Researcher with Toshiba Infrastructure Systems & Solutions Corporation, and studied machine learning and data mining using data obtained from the IoT devices installed in buildings, factories, and infrastructure systems. Since 2022, he has been a Visiting Researcher with the National Institute of Informatics. His research interests include feature extractions from multivariate time series, unsupervised learning, and data mining.

MAHITO SUGIYAMA (Member, IEEE) received the Ph.D. degree in informatics from Kyoto University, in 2012, under the supervision of Prof. Dr. Akihiro Yamamoto. He has been an Associate Professor with the National Institute of Informatics, since 2017. Prior to joining the faculty, he was a Research Scientist with the Machine Learning and Computational Biology Research Group of Prof. Dr. Karsten Borgwardt, Max Planck Institutes Tübingen, Germany, from 2012 to 2014, and an Assistant Professor with Osaka University, from 2014 to 2017. His research interests include artificial intelligence, machine learning, and data mining.