A Study on Unsupervised Feature Extraction for Multivariate Time Series

by

MATSUE Kiyotaka

Dissertation

submitted to the Department of Informatics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

The Graduate University for Advanced Studies, SOKENDAI
March 2022

A dissertation submitted to Department of Informatics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Advisory Committee

Assoc. Prof. SUGIYAMA Mahito (Chair)	National Institute of Informatics, The Graduate University for Advanced Studies, SOKENDAI
Prof. Emeritus HAYAMI Ken	National Institute of Informatics, The Graduate University for Advanced Studies, SOKENDAI
Prof. UNO Takeaki	National Institute of Informatics, The Graduate University for Advanced Studies, SOKENDAI
Assoc. Prof. MIZUNO Takayuki	National Institute of Informatics, The Graduate University for Advanced Studies, SOKENDAI
Assoc. Prof. KOBAYASHI Ryota	The University of Tokyo

Acknowledgements

My dissertation would not have been possible without the support of many people. I would like to express my gratitude to them.

First and foremost, I wish to express my sincere gratitude to my supervisor, Assoc. Prof. SUGIYAMA Mahito for the continuous support of my research. My scientific papers would not be accepted to an international conference and an academic journal without his technical support. Besides my supervisor, I am extremely grateful to advisory committee, Prof. Emeritus HAYAMI Ken, Prof. UNO Takeaki, Assoc. Prof. MIZUNO Takayuki, and Assoc. Prof. KOBAYASHI Ryota for their insightful and helpful comments. In addition, my special thanks are due to secretary TOKUDA Hiroko for helping me in many things such as procurement of supplies, accounting procedures, and so on.

Furthermore, I am most grateful to my managers and my colleagues belonging to Infrastructure Systems Research and Development Center in Toshiba Infrastructure Systems & Solutions Corporation. In particular, my managers, KONDO Koichi, KAWAMURA Takuya and NAKATANI Hiroshi, gave me an opportunity to pursue a PhD.

Moreover, I would like to thank all the members in Sugiyama laboratory, ABE Hiroto, MUHAMMAD Masykur, MIZUGUCHI Makoto, YAMADA Masatsugu, LUO Simon Junming, CHEEMA Prasad, PÂRȚACHI Profir-Petru, AHMED Md. Sohel, GHALAMKARI Kazu, KANOH Ryuichi, and KAWAKAMI Yuhi. They gave me a lot of invaluable advice in the Lab. Seminar and the Lunch Seminar.

Last but certainly not least, I would like to express my gratitude to my family members, MATSUE Yasuto, MATSUE Shizuko, MATSUE Hisako, MIYAKAWA Sadayuki, MIYAKAWA Keiko, MATSUE Yasuko and MATSUE Keika, for supporting me through my life.

MATSUE Kiyotaka
Kawasaki, March 2022

Abstract

It has been a while since Internet of things (IoT) devices that measure various types of events such as temperature, voltage and pressure are used in many systems, and the amount of data collected by such devices is increasing year by year. Accordingly, a variety of services using those collected data have been produced in a lot of fields and utilization of the data has become much more important than ever. Considering an office building as one of the examples, collected data by sensors installed on walls or ceilings, which measure temperature, humidity, or carbon dioxide concentration, can be used in an air conditioning control system or a lighting system. In the case of sensing at multiple locations in a building, the sensors located in the same local area are expected to record similar values. In this situation, if a sensor is broken, different patterns of values from others might be recorded, which indicates that the sensor should be replaced with a new one immediately. However, finding such broken sensors is difficult because anomalous behavior of broken sensors may emerge combinatorially together with other healthy sensors, and the combinatorial relationship between sensors must be taken into account. Since those data collected by multiple sensors are in the form of multivariate time series, it is essential to extract features encoding association between multiple time series, and there are heavy demands particularly for industrial fields.

Once data taken by sensors are collected, features extracted from the data that properly takes relationships between multivariate time series into account can be used in various data science applications such as outlier detection and clustering. Since finding useful feature vector representation from time series is one of crucial tasks in those fields, a lot of methods to extract association between time stamps have been developed so far such as Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), and Discrete Cosine Transformation (DCT). These methods are widely used in signal processing fields and the methods are commonly targeted to univariate time series data, that is, they cannot be directly applied to multivariate time series even though they are widely seen in the real-world. Therefore, extraction of features with considering association between multivariate time series remains a challenging task because both time-wise and variable-wise associations should be taken into account. Although some algorithms using machine learning technology like deep learning are becoming popular among outlier detection tasks nowadays, which can implicitly take such time-wise and variable-wise associations into account, they commonly need ground truth inlier (normal) time series that do not include any outliers (anomalous patterns) to train

a model. As one of the examples, there is an autoencoder method that can detect outliers by calculating reconstruction errors. A model made by the autoencoder is expected to correctly decode inliers and wrongly decode outliers if only inliers are used in its training, and the difference of reconstruction errors makes it possible to discriminate outliers from inliers. However, if outliers exist in a training dataset, a model trained by the autoencoder may overlook outliers because they can be also correctly decoded, resulting in the suboptimal performance. To date, only few unsupervised algorithms have been proposed that do not require any labeled time series data, although such unsupervised algorithms are of high importance in practice. To address this issue, we focus on unsupervised feature extraction that can be used for various downstream tasks including outlier detection and clustering.

In this dissertation, we present unsupervised feature extraction algorithms for multivariate time series, called UFEKS (Unsupervised Feature Extraction using Kernel and Stacking) and UFEKT (Unsupervised Feature Extraction using Kernel Method and Tucker Decomposition). UFEKS (1) constructs a kernel matrix for the set of subsequences from each time series and (2) concatenates all matrices horizontally. Feature representation is obtained as row vectors in the concatenated matrix in a fully unsupervised manner, which can be used in subsequent machine learning problems. Likewise, UFEKT (1) constructs a kernel matrix from subsequences of each time series to account for time-wise association and (2) constructs a single tensor by stacking the kernel matrices and performs Tucker decomposition to account for variable-wise association. Tucker decomposition is one of the well-known tensor decomposition techniques and it decomposes the constructed tensor of a kernel into one core tensor and three factor matrices. Feature representation is obtained as a row vector in one of the decomposed factor matrices. In the decomposition process, ranks of a tensor must be given as one of the hyper-parameters. Although finding the best values of hyper-parameters in an unsupervised learning is known as a difficult task, we present an algorithm to find appropriate values of the ranks heuristically. The whole process of UFEKT is also fully unsupervised and can be used for subsequent machine learning tasks.

After we describe our new algorithms in detail, we empirically evaluate our algorithms and show experimental results in two tasks of outlier detection and clustering. Nine synthetic and six real-world datasets are used for outlier detection and 102 real-world datasets are used for clustering. Our methods are compared with two well-established existing feature extraction methods, the subsequence-based method (SS) and the page rank kernel-based method (PRK). Furthermore, we discuss reasons why our algorithms are superior to the existing methods using the principal component analysis (PCA). Finally, we summarize main findings of this dissertation and discuss future work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Internet of things (IoT) devices, composed of many sensors such as temperature, humidity, and pressure, are installed in various types of systems, and multivariate time series data collected by those sensors are used in a wide range of fields. For instance, in the task of facility maintenance for a building, it may be possible to know the best timing of replacement for broken facilities by monitoring and analyzing collected data. Although this process, called *Condition-Based Maintenance* (CBM) [24, 45], is well known technology in industrial fields, this task is still challenging as we need to use many multivariate time series to find relationship between sensors. In the case of sensing at multiple locations in a building, sensors located in the same local area are expected to record similar values. If one sensor takes different values from others, the sensor might be broken and need to be replaced to the new one. However, it is hard to find sensors taking different values because we need to find different combinatorial relationships between sensors. Therefore, a new algorithm to solve these problems is still in great demand.

1.2 Problem Settings and Our Approaches

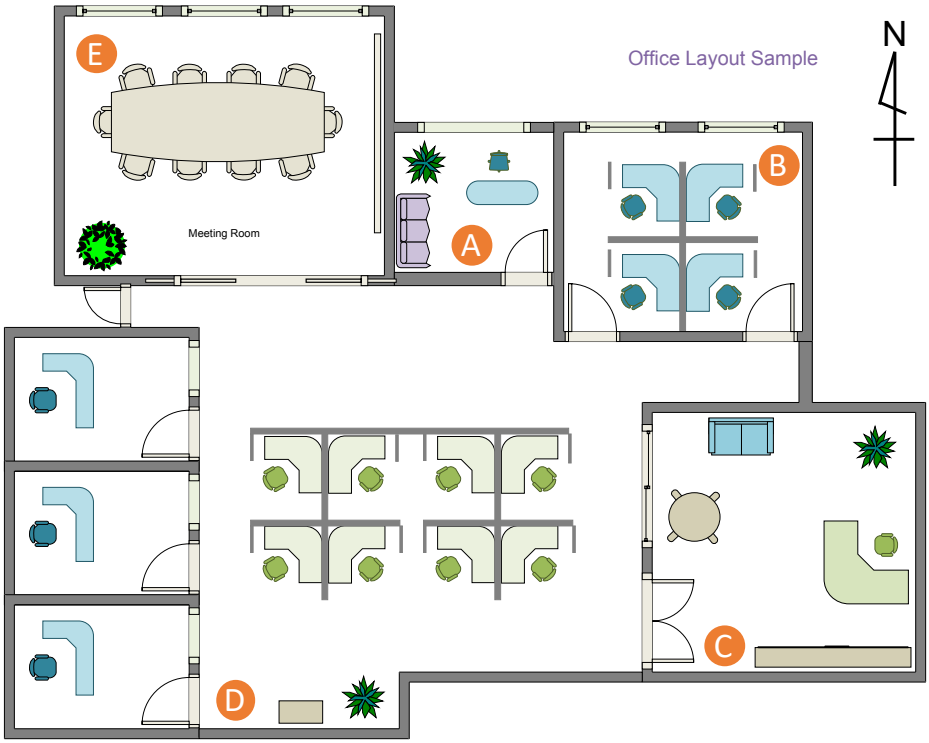


Figure 1.1: This is an example of an office layout[1]. There are five sensors denoted from A to E installed in the office rooms and each sensor record temperature measured every hour at each room.

In this dissertation, we consider *association between multiple time series*. An example of association is shown in Fig. 1.1. It shows a floor map in an office building and five temperature sensors denoted from A to E are installed in each room. All sensors record temperature measured every hour in the rooms. Examples of time series measured by their sensors are shown in Figure 1.2. To clarify the difference between time series, small values are added to each data as an offset. All the time series seem to have similar waveform, however, the time series collected from the sensor E seem to rise slightly on the afternoon of January 12th, 2022, whereas the others drop. If building managers find such a trend, they are probably considered as signals of maintenance for equipment and they can start to inspect it in more detail, whether or not the sensor E should be calibrated, the filters of the air conditioner installed in the room E must be cleaned for clogging, and the air conditioner require to be replaced with a new one. Finding a different combination from multivariate time series data is one of the most important tasks in a number of sensor monitoring tasks including CBM.

[1]This figure can be obtained from *https://www.edrawsoft.com/template-office-layout.html*, which is provided as a free customizable office layout template. The sensor labels and direction symbol are added by the author.

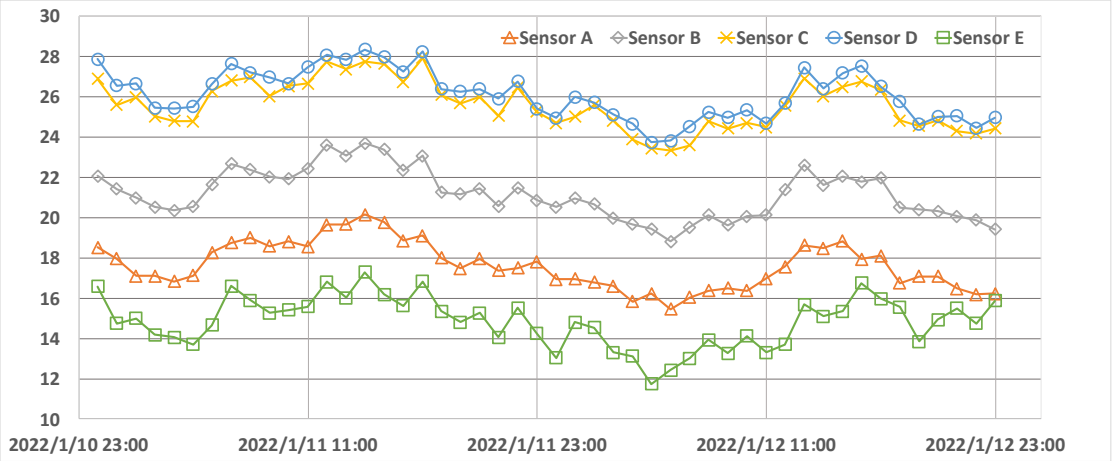


Figure 1.2: An example of multivariate time series data collected by sensors installed in office rooms shown in Figure. 1.1. They are fluctuations of temperature measured every hour in each room. To clearly display each data, some values are added to the data as offset. The temperature measured by the sensor E seems to rise slightly on the afternoon of January 12th, 2022, whereas the others drop.

Moreover, we show the other easy-to-understand example of association between multiple time series in Figure 1.3. There two time series from the corresponding variables composed of lines and sine waves with noise. In an orange frame, the time series is composed of line and sine wave (up and bottom), while the other subsequences out of the orange frame are composed of the combination of only lines or sine waves. Hence only the subsequence in the orange frame has a different combination. This is an example of a combinatorial outlier, and finding such outliers is fundamentally difficult as *they cannot be found if we look at each time series separately*.

Therefore, we focus on the task of extracting feature vector representation from multivariate time series data that can incorporate combinatorial association between two or more time series. This approach is unsupervised, hence it enables us to apply general existing machine learning algorithms to multivariate time series.

Furthermore, we try to apply the extracted feature vector representation to a clustering problem because the representation is not limited to only an outlier detection task.

1.3 Proposed Algorithms

To extract feature vectors from multivariate time series, we propose two new algorithms, called UFEKS (*Unsupervised Feature Extraction using Kernel and Stacking*) and UFEKT (*Unsupervised Feature Extraction using Kernel Method and Tucker Decomposition*) [36]. A basic procedure of the proposed algorithm, UFEKS, is as follows:

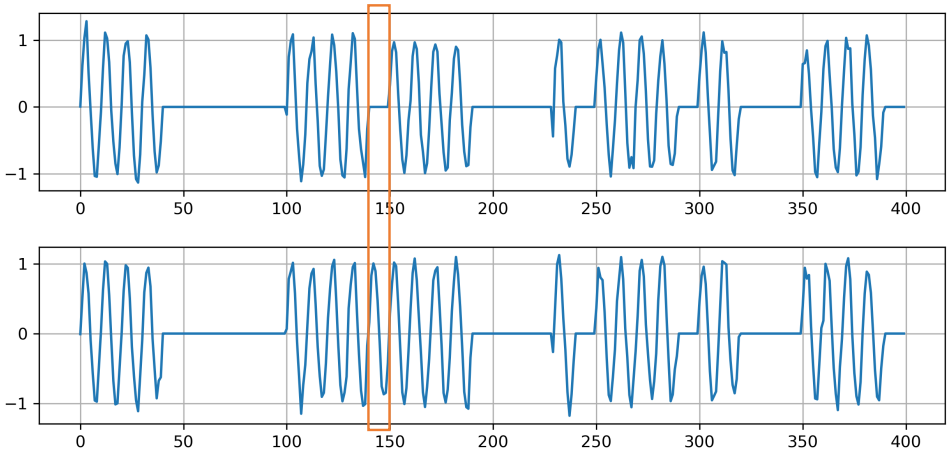


Figure 1.3: An example of association for multivariate time series data. There are time series data of two variables (that is, two time series) composed of lines and sine waves with noise. Combination in an orange frame has two subsequences (up and bottom), which are composed of a line and a sine wave, while subsequences other than the orange frame are composed of similar shapes between up and bottom subsequences. Hence, the orange frame is deemed to be an combinatorial outlier in the time series.

1. Divide given time series into a set of its subsequences,
2. Make kernel matrices from each subsequence using the RBF kernel,
3. Place the all kernel matrices across in one row and horizontally concatenate them into one kernel matrix,
4. Extract row vectors in the concatenated matrix as feature vectors.

The basic procedure of UFEKT is also as follows:

1. Divide given time series into a set of its subsequences,
2. Make kernel matrices from each subsequence using the RBF kernel,
3. Stack all the kernel matrices and make a three-way tensor from their kernel matrices,
4. Decompose the tensor into one core tensor and three factor matrices,
5. Extract row vectors in one of the factor matrices as feature vectors.

Each row vector obtained by UFEKS or UFEKT is corresponded to a point in the multidimensional Euclidean space. This means that our approach allows us to use existing standard algorithms of non-time series for an outlier detection task or a clustering task with considering combinatorial association between time series. We examine their proposed methods using unsupervised outlier detection and clustering algorithms. Although both the row vectors obtained by UFEKS and UFEKT have features, each has advantages and disadvantages. The vectors obtained by UFEKS tend to have a high dimensions, while the vectors by UFEKT can be low dimensions using Tucker decomposition. It means that, if we use the feature vectors by UFEKT for outlier detection or clustering, it might be possible to get high accuracy of the results, however, it would have high computational cost. Moreover, it should be noted that *our proposed method does not use any labeled data* for detecting outliers from multivariate time series since many algorithms about outlier detection use labeled data even if they are mentioned to unsupervised learning.

1.4 Contributions

To summarize, the main contributions of our works are:

- Our proposed methods UFEKS and UFEKT can extract features from multivariate time series by incorporating combinatorial association between time series.
- The obtained features using UFEKS and UFEKT can be applied to a variety of applications such as outlier detection, clustering, and other data mining tasks.
- In outlier detection, the proposed method can detect outliers that cannot be found if we look at each of the multivariate time series separately.
- The UFEKS and UFEKT have good experimental results for outlier detection scenario both synthetic and real-world datasets. Furthermore, the UFEKT have also good experimental results for clustering scenarios for real-world datasets.

1.5 Outline

First, we mention related work about feature extraction from univariate and multivariate time series datasets. In these fields, it is difficult to find literature related to feature extraction from time series because methods or techniques are often mentioned in literature about applications like outlier detection using extracted features. Therefore, we cited some literature about applications such as outlier detection from time series and non-time series data, and clustering, respectively, in Chapter 2.

Second, our proposed algorithms to extract features from multivariate time series are mathematically formulated in Chapter 3. In addition, one of the parameters for UFEKT, the

rank, which is used for Tucker decomposition, must be decided in an unsupervised scenario. The rank selection algorithm that we have proposed is mentioned in the same chapter. The algorithm employs a heuristic strategy, however, it gives good rank settings in terms of the performance in outlier detection in practice. It is discussed in the same chapter.

To evaluate our algorithms, two feature representation algorithms, the *PageRank Kernel* (PRK) and *SubSequence* (SS), are employed for comparison with our algorithms. After explanations about procedures of their algorithms, the experimental results applied to an outlier detection task and a clustering task are shown in Chapter 4 and Chapter 5, respectively. In their experiments, we prepared nine types of synthetic and six types of real-world multivariate time series datasets for an outlier detection task, and 102 real-world multivariate time series datasets for a clustering task. Their datasets are explained in the same chapters. In addition, to evaluate our algorithms, *κth-Nearest Neighbor* (κNN), *Local Outlier Factor* (LOF), *One-class Support Vector Machine* (OCSVM), and *Isolation Forest* (IForest) are used for an outlier detection task, and *K-means* (KMeans), *Density-based Spatial Clustering of Applications with Noise* (DBSCAN), *Agglomerative Hierarchical Clustering* (AHC), and *Gaussian Mixture Model* (GMM) used for a clustering task. Furthermore, *Area under Precision and Recall Curve* (AUPRC) and *Normalized Mutual Information* (NMI) are used as metrics for outlier detection and clustering. Those algorithms and metrics are also mentioned in the same chapters. In discussion parts, we discuss reasons why our algorithms take higher scores than existing methods by considering Euclidean distances between subsequences analyzed with *Principal Component Analysis* (PCA).

Finally, we summarize the main findings of our studies and mention future work in Chapter 6.

Chapter 2

Related Work

2.1 Feature Extraction from Time Series

Several algorithms about feature extraction from time series for outlier detection have been developed so far. Some representative algorithms for representation of time series data are shown in Table 2.1 [3]. The *Discrete Fourier Transform* (DFT), *Discrete Wavelet Transform* (DWT), and *Discrete Cosine Transformation* (DCT) are known as the methods to represent features of time series data and widely used in a signal processing field. Other methods like statistics are also well known techniques to extract features from time series data. However, some methods shown in Table 2.1 cannot be directly applied to multivariate time series. As another point of view, kernel methods are used for extraction of features from multivariate time series [12, 54]. They construct a kernel matrix using *radial basis function* (RBF) kernel [12] or linear kernel [54] from a time series. However, since their approaches use only the integrated signal across multiple time series, they cannot treat combinatorial association of time series. In contrast, our proposal treats each time series separately when we apply kernels, hence we can treat such combinatorial effects.

2.2 Outlier Detection from Univariate Time Series

Outlier detection for time series have been actively studied and a number of methods have been proposed [25, 1, 44, 12, 28, 48, 38, 39]. In particular for outlier detection from a univariate time series, *autoregressive moving average* (ARMA) and an *autoregressive integrated moving average* (ARIMA) have been commonly used [44]. They can find outliers from differences between predicted values by ARMA or ARIMA and actual values. However, they are considered to be sensitive to noise, resulting in increasing false positives when the noise level is severe [44, 54]. Furthermore, a lot of algorithms we cited cannot directly be applied to outlier detection problems for multivariate time series. *Dynamic time warping* (DTW) is another representative method, which measures similarity between two time series by aligning them [40, 7, 37]. DTW is widely used in a variety of applications because it

Table 2.1: Representative Methods for Representation of Time Series Data cited from [3].

Representation Methods	Abbreviation of Methods
Discrete Fourier Transform	DFT
Discrete Wavelet Transform	DWT
Discrete Cosine Transformation	DCT
Piece-waise Aggregate Approximation	PAA
Symbolic Approximation	SAX
PageRank Kernel	PRK
Subsequence	SS
Statistics (mean, variance, etc.)	STATS

is robust to different frequencies or lengths. DTW can be used for univariate time series, however, it cannot directly measure the association between three or more time series.

2.3 Outlier Detection from Multivariate Time Series

2.3.1 General Methods

In terms of outlier detection from multivariate time series, several algorithms have been proposed [12, 28, 48]. TAKEISHI and YAIRI [48] have proposed an algorithm using sparse representation. This method is designed in a supervised manner and requires labeled data. Moreover, they do not focus on combinatorial association between time series and may not detect combinatorial outliers. Furthermore, although tensor decomposition technique have been used for outlier detection [55, 30, 18], where anomaly detection using Tucker decomposition has been proposed, they do not focus on multivariate time series.

2.3.2 Neural Network Based Methods

Nowadays, a number of outlier detection methods for time series have been proposed based on neural networks [35, 53, 56, 29, 57, 46, 54, 5]. One of the outlier detection methods for multivariate time series is *multi-scale convolutional recurrent encoder-decoder* (MSCRED), which is an algorithm using *attention-based convolutional long-short time memory* (LSTM) [54]. It extracts features and detects outliers from multivariate time series by constructing a kernel matrix. Many algorithms based on neural networks are becoming popular recently and widely used in the real world. However, most of the neural network based models have many parameters to be tuned, which is fundamentally difficult in the unsupervised setting. Furthermore, they are often designed as supervised, hence ground-truth labels are required to train their models to perform outlier detection.

2.4 Outlier Detection from Non-Time Series

Numerous algorithms have been proposed so far for outlier detection for non-time series data [1, 2, 42, 11, 19, 51, 56, 57]. Representative algorithms include *local outlier factor* (LOF) [9], *κth-nearest neighbor* (κNN) [26], ORCA [6], *one-class support vector machine* (OCSVM) [33, 43], *isolation forest* (IForest) [31], and sampling-based outlier detection [47]. Although these algorithms have been widely used in a variety of fields, they fundamentally assume i.i.d. data and cannot be applied to time series directly. Since our method generates feature vectors from multivariate time series with incorporating such time-wise association, the above outlier detection algorithms can be directly applied to the obtained feature vectors, which is an advantage of our method.

2.5 Clustering for Time Series

2.5.1 Clustering Applications

A clustering is one of the applications that is widely used in various categories such as biology, energy, finance, medicines, and so on. Some representative clustering applications are shown in Table 2.2.

2.5.2 Clustering Algorithms

Considering time series clustering, it can be classified to three types of clustering categories, whole time series clustering, subsequence time series clustering, and time point clustering [3]. Many algorithms have been focused on whole time series clustering and developed so far. In general, many algorithms have two steps to find clusters, feature extraction or decision of model parameters, and usage of existing clustering algorithms for non-time series. Some representative algorithms for whole time series clustering are shown in Table 2.3. Subsequence time series clustering and time point clustering are commonly applied to univariate time series, that is, there are few algorithms applied to multivariate time series. Therefore we focus on the area and evaluate capabilities of clustering using features extracted from our algorithm UFEKT.

Table 2.2: Representative Clustering Applications for Time Series Data cited from [3].

Category	Clustering Application
Aviation/Astronomy	Astronomical data - pre-processing for outlier detection
Biology	Multiple gene expression profile alignment for microarray time-series data clustering
Climate	Discovery of climate indices
Energy	Discovering energy consumption pattern
Finance	Finding seasonality patterns
Medicine	Detecting grain activity
Psychology	Analysis of human behavior in psychological domain
Robotics	Forming prototypical representations of the robot's experiences

Table 2.3: Representative Whole Time Series Clustering Algorithms cited from [3].

Clustering Algorithms	Representation Method	Distance Measurement
Modified relocation clustering	Raw time series	Euclidean
Fuzzy C-Means	Raw time series	Euclidean and two cross correlation-based
Agglomerative hierarchical	Raw time series	J divergence
Agglomerative hierarchical	Raw time series	Root mean square
Fuzzy clustering	Raw time series	Euclidean
Agglomerative hierarchical	Raw time series	Gaussian moedels of data errors
K-Means	Discrete Wavelet Transform	Euclidean
K-Means and fuzzy C-Means	Symbolic Aggregate Approximation	Euclidean and symmetric version of Kullback-Liebler
K-Means, Hierarchical, and RS	Raw time series	DTW
K-Means	Derivative time series segment approximation	DTW
Bayesian Hierarchical Clustering	Gaussian process data model	Dirichlet process model
Hybrid, K-Medoids, and Hierarchical	Piece-wise aggregate approximation	Euclidean distance and DTW

Chapter 3

Algorithms

3.1 UFEKS: Unsupervised Feature Extraction using Kernel and Stacking

We formulate our method UFEKS in Section 3.1, which extracts feature vectors from multivariate time series, and introduce its application to outlier detection in Chapter 4.

3.1.1 Extraction of features from multivariate time series

Many algorithms to extract features from time series have focused on its subsequences [54, 12, 48, 28]. In this thesis, we follow the idea of using subsequences and extract features based on the similarity between subsequences. We use a kernel method to measure the similarity between subsequences as it is widely used in data analysis for time series and its effectiveness is well known.

Assume that there are P variables indexed from 1 to P. Given a multivariate time series with the length T as a matrix $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{P \times T}$, where each row vector $\mathbf{x}^{(p)} = (x_{p1}, x_{p2}, \cdots, x_{pT}) \in \mathbb{R}^T$ represents a time series of a variable p and each column vector $\mathbf{x}_t = (x_{1t}, x_{2t}, \cdots, x_{Pt})^\top \in \mathbb{R}^P$ 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 denoted as

$$\mathbf{X}_t = \begin{bmatrix} x_{1t} & x_{1(t+1)} & \cdots & x_{1(t+w-1)} \\ x_{2t} & x_{2(t+1)} & \cdots & x_{2(t+w-1)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{Pt} & x_{P(t+1)} & \cdots & x_{P(t+w-1)} \end{bmatrix}, \tag{3.1}$$

where each row $\mathbf{x}_t^{(p)} = (x_{pt}, x_{p(t+1)}, \cdots, x_{p(t+w-1)}) \in \mathbb{R}^w$ represents a subsequence at t of the p-th time series with length w. Examples of subsequences from a univariate time series are shown in Figure 3.1.

First, we consider extraction of feature vectors from a 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. We use

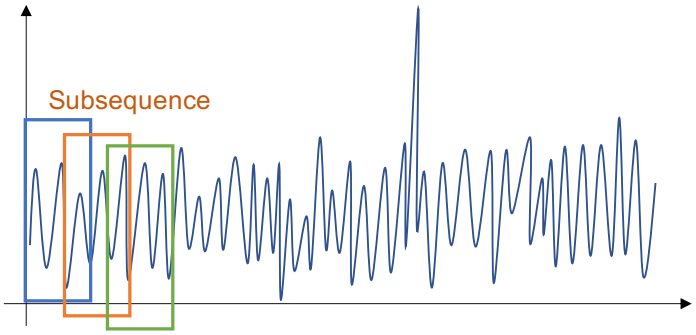


Figure 3.1: Examples of a subsequence from a univariate time series.

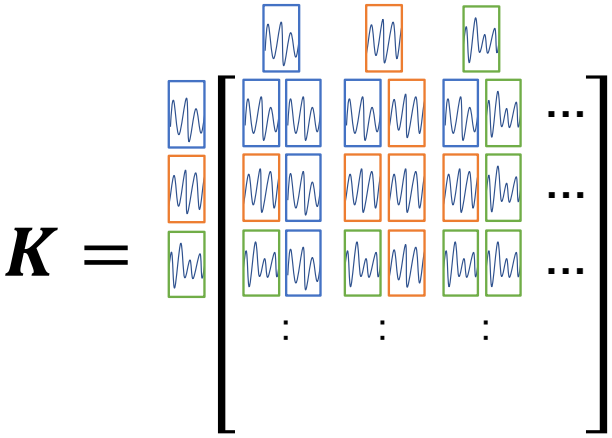
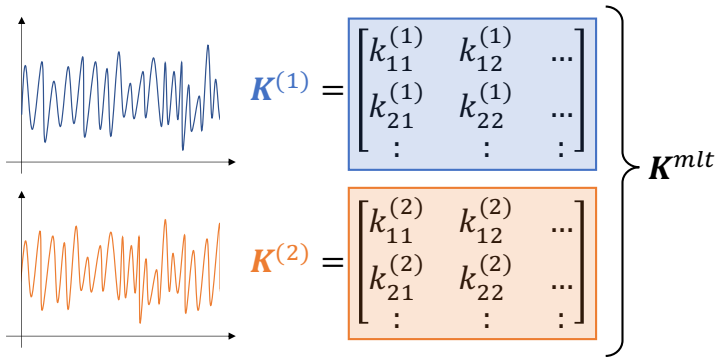


Figure 3.2: An image of a kernel matrix. Each element in the kernel matrix shows the relationship between subsequences.

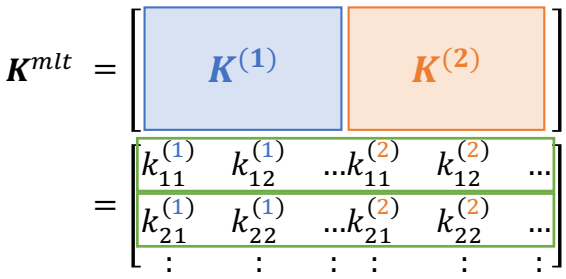
the RBF kernel to obtain the similarity between them, which is given as

$$k_{ij}^{(p)} = \exp\left\{-\frac{\sum_{s=0}^{w-1}(x_{p(i+s)} - x_{p(j+s)})^2}{\sigma^2}\right\}, \tag{3.2}$$

where $\sigma \in \mathbb{R}$ is a parameter. Every $k_{ij}^{(p)}$ takes a value in $(0, 1]$. The RBF kernel for similarity computation between subsequences was first employed in [12] and we follow this strategy. As shown in Figure 3.2, each element in the kernel matrix shows the relationship between subsequences. For example, diagonal elements in the kernel matrix are a relationship between the same subsequences themselves, and the other elements are relationships between different sequences. If they are similar to each other, their elements take close to one, on the other hand, if they are different, their elements take close to zero. So, this kernel matrix is a non-negative symmetric matrix and each element takes value between zero and one. The resulting kernel matrix $\mathbf{K}^{(p)} \in \mathbb{R}^{(T-w+1)\times(T-w+1)}$ becomes a non-negative square matrix



(a) Kernel matrices $\mathbf{K}^{(1)}$ and $\mathbf{K}^{(2)}$ from each time series



(b) A concatenated kernel matrix $\mathbf{K}^{mlt}$

Figure 3.3: An image of a concatenated kernel matrix generated from two time series. Each row vector in the concatenated kernel matrix represents associations between the two time series.

given as

$$\mathbf{K}^{(p)} = \begin{bmatrix} k_{11}^{(p)} & k_{12}^{(p)} & \cdots & k_{1(T-w+1)}^{(p)} \\ k_{21}^{(p)} & k_{22}^{(p)} & \cdots & k_{2(T-w+1)}^{(p)} \\ \vdots & \vdots & \ddots & \vdots \\ k_{(T-w+1)1}^{(p)} & k_{(T-w+1)2}^{(p)} & \cdots & k_{(T-w+1)(T-w+1)}^{(p)} \end{bmatrix}, \tag{3.3}$$

where we denote each row vector as $\mathbf{k}_t^{(p)} = (k_{t1}^{(p)}, k_{t2}^{(p)}, \cdots, k_{t(T-w+1)}^{(p)}) \in \mathbb{R}^{(T-w+1)}$ and T is the length of the time series. Each row $\mathbf{k}_t^{(p)}$ of the kernel matrix $\mathbf{K}^{(p)}$ is a feature vector representation of the subsequence $\mathbf{x}_t^{(p)}$, and it incorporates association between $\mathbf{x}_t^{(p)}$ and all the other subsequences.

Now we extend our feature vector representation for univariate time series to multivariate time series. First we generate kernel matrices $\mathbf{K}^{(1)}, \mathbf{K}^{(2)}, ..., \mathbf{K}^{(P)}$ for all variables 1, 2,. . . , P. Then, we horizontally concatenate all kernel matrices with each other and generate a single matrix. The resulting matrix $\mathbf{K} \in \mathbb{R}^{(T-w+1)\times(T-w+1)P}$ for multivariate time series is

Algorithm 1 The UFEKS algorithm

Input: $\mathbf{X} \in \mathbb{R}^{P \times T}, w, \sigma, R \in \mathbb{R}$

Output: $\mathbf{k}_1, \ldots, \mathbf{k}_{T-w+1} \in \mathbb{R}^{(T-w+1)P}$

// Construct kernel matrices from multivariate time series

for $p = 1$ to P **do**

  for $(i, j) = (1, 1)$ to $(T - w + 1, T - w + 1)$ **do**

    $k_{ij}^{(p)} \leftarrow \exp\{ - \sum_{s=0}^{w-1} (x_{p(i+s)} - x_{p(j+s)})^2 / \sigma^2 \}$

  end for

end for

Construct $\mathbf{K} \in \mathbb{R}^{(T-w+1) \times (T-w+1)P}$ from $\mathbf{K}^{(1)}, \ldots, \mathbf{K}^{(P)}$ by concatenating each matrix

// Feature Extraction from Kernel Matrices

for $i = 1$ to $T - w + 1$ **do**

  $\mathbf{k}_i \leftarrow i$th-row vector of $\mathbf{K}$

end for

return $\mathbf{k}_1, \ldots, \mathbf{k}_{T-w+1}$

given as

$$\mathbf{K} := [\mathbf{K}^{(1)}, \cdots, \mathbf{K}^{(P)}] \tag{3.4}$$
$$= \begin{bmatrix} k_{11}^{(1)} & \cdots & k_{1(T-w+1)}^{(1)} & \cdots & k_{11}^{(P)} & \cdots & k_{1(T-w+1)}^{(P)} \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ k_{(T-w+1)1}^{(1)} & \cdots & k_{(T-w+1)(T-w+1)}^{(1)} & \cdots & k_{(T-w+1)1}^{(P)} & \cdots & k_{(T-w+1)(T-w+1)}^{(P)} \end{bmatrix}$$

and its row vector $\mathbf{k}_t \in \mathbb{R}^{(T-w+1)P}$ is given as

$$\mathbf{k}_t = \left(k_{t1}^{(1)}, \cdots, k_{t(T-w+1)}^{(1)}, \cdots, k_{t1}^{(P)}, \cdots, k_{t(T-w+1)}^{(P)} \right). \tag{3.5}$$

This matrix $\mathbf{K}$ is considered to be the set of feature vectors $\{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_{T-w+1}\}$. We treat each row $\mathbf{k}_t$ as a feature vector representation of the corresponding multivariate subsequence $\mathbf{X}_t$, which is expected to encode the association between variables with respect to the subsequence from t to $t + w - 1$. An image of concatenated kernel matrices are shown in Figure 3.3. The pseudo code of our algorithm is summarized in Algorithm 1.

3.2 UFEKT: Unsupervised Feature Extraction using Kernel Method and Tucker Decomposition

3.2.1 Feature Extraction Algorithm

Although UFEKS can extract features from multivariate time series, if datasets include a lot of variables, its row vector in a feature matrix obtained by UFEKS might be a high dimensional vector and makes it difficult to represent features of time series. To overcome this issue, we propose a new method *Unsupervised Feature Extraction using Kernel Method and Tucker Decomposition* (UFEKT). The UFEKT is composed of two steps: construction of kernel matrices from multivariate time series and feature extraction from the kernel matrices via tensor decomposition. After formulating those steps, we also formulate outlier detection and clustering as applications for UFEKT.

Constructing kernel matrices from multivariate time series

To extract features from time series, most of algorithms have focused on subsequences, which means they are sequences derived from original time series [54, 12, 48, 28]. In this thesis, we follow the idea of using subsequences and extract features based on the similarity between subsequences. A kernel method to measure the similarity between subsequences is employed because it is widely used in data analysis for time series and its effectiveness is well known in this area [12, 54].

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 $\mathbf{x}_t = (x_{1t}, x_{2t}, \ldots, x_{Pt})^\mathsf{T} \in \mathbb{R}^P$ 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

$$\mathbf{X}_t = \begin{bmatrix} x_{1t} & \cdots & x_{1(t+w-1)} \\ x_{2t} & \cdots & x_{2(t+w-1)} \\ \vdots & \ddots & \vdots \\ x_{Pt} & \cdots & x_{P(t+w-1)} \end{bmatrix}, \tag{3.6}$$

where it 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.

First, we consider extracting feature vectors from a 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 for similarities between subsequences, their similarities $k_{ij}^{(p)}$ are

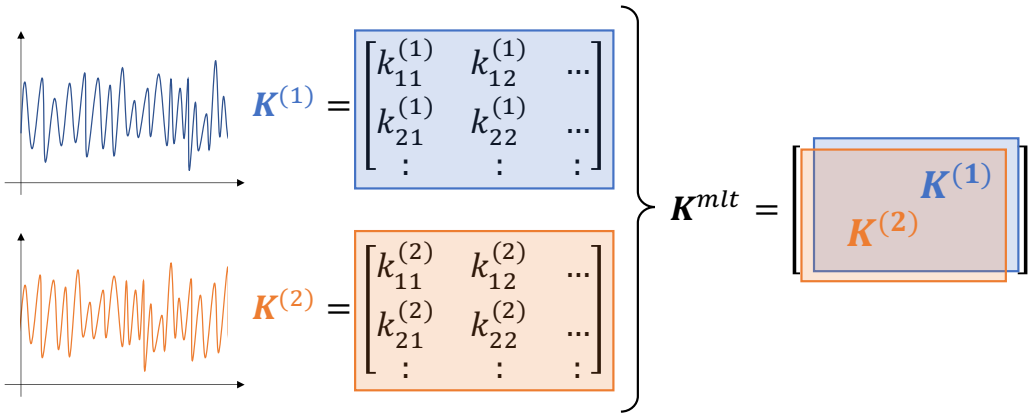


Figure 3.4: An image of a tensor of kernel matrices. Each kernel matrix is stacked and a three-way tensor is generated.

defined as

$$k_{ij}^{(p)} = \exp\left\{-\frac{\sum_{s=0}^{w-1}(x_{p(i+s)} - x_{p(j+s)})^2}{\sigma^2}\right\}, \tag{3.7}$$

where σ is a parameter. Since the σ must be changed depending on the input datasets, we show one of the way to get suitable values of the σ, which is the mean sum of squares of differences between two subsequences for every variable. In this case, the σ take different values for each variable. More details are shown in Algorithm 4. Every $k_{ij}^{(p)}$ takes a value in $(0, 1]$. The RBF kernel for similarity computation was first employed in [12] and we follow the strategy. The resulting kernel matrix $\mathbf{K}^{(p)} \in \mathbb{R}^{(T-w+1)\times(T-w+1)}$ becomes non-negative square matrix given as

$$\mathbf{K}^{(p)} = \begin{bmatrix} k_{11}^{(p)} & \cdots & k_{1(T-w+1)}^{(p)} \\ k_{21}^{(p)} & \cdots & k_{2(T-w+1)}^{(p)} \\ \vdots & \ddots & \vdots \\ k_{(T-w+1)1}^{(p)} & \cdots & k_{(T-w+1)(T-w+1)}^{(p)} \end{bmatrix}. \tag{3.8}$$

We denote each row vector 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 $O(T^2)$, hence the total time complexity becomes $O(T^2P)$ when there are P times series.

Feature extraction from kernel matrices

Second, we construct a tensor from the obtained kernel matrices to take association between multiple time series into account. Given P kernel matrices $\mathbf{K}^{(1)}, \mathbf{K}^{(2)}, \ldots, \mathbf{K}^{(P)} \in$

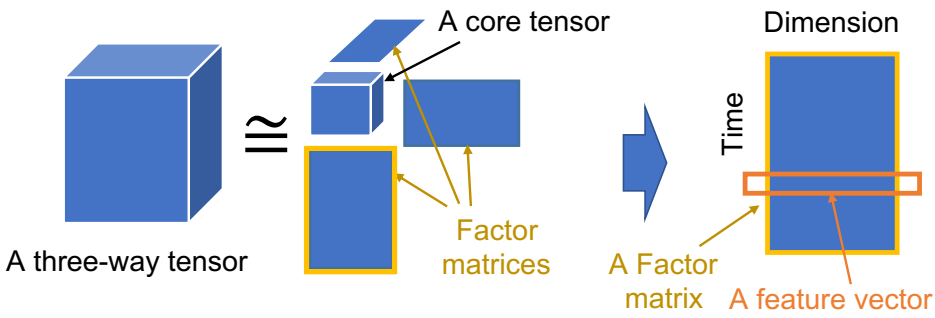


Figure 3.5: An image of Tucker decomposition. A three-way tensor is decomposed into one core tensor and three factor matrices. After the decomposition, resulting each row vector in a factor matrix is regarded as a feature vector for multivariate time series.

$\mathbb{R}^{(T-w+1)\times(T-w+1)}$ generated from P time series by (3.8), a tensor is constructed by stacking them; that is, it becomes 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 series. To extract features from the tensor, we propose to use Tucker decomposition [50]. Although CANDECOMP/PARAFAC (CP) decomposition [27, 10, 21] is a well-known alternative to tensor decomposition, we consider that Tucker decomposition is more suitable in our task for the following reason. In CP decomposition, a third-order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ is decomposed into three matrices $\mathbf{A} \in \mathbb{R}^{I\times J}$, $\mathbf{B} \in \mathbb{R}^{J\times K}$, and $\mathbf{C} \in \mathbb{R}^{K\times I}$ in the form of $\mathcal{X} \approx \|\lambda; \mathbf{A}, \mathbf{B}, \mathbf{C}\| = \sum_{r=1}^{R} \lambda_r \ \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ under appropriate column-wise normalization, where $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^R$ and the symbol "$\circ$" denotes the vector outer product. In this case, information of outliers tend to appear in the vectors belonging to a higher-order rank of λ_r and it is difficult to detect them in an unsupervised manner. In contrast, in Tucker decomposition, information of outliers is encoded in a factor matrix, hence it is easy to extract vectors from it.

As shown in Figure 3.5, if a tensor $\mathcal{K} \in \mathbb{R}^{M\times N\times P}$ is given, Tucker decomposition decomposes it into one core tensor $C \in \mathbb{R}^{R_1\times R_2\times R_3}$ 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 we wish to solve for decomposing the tensor $\mathcal{K}$ is formulated as

$$\min_{C,\mathbf{F}^{(1)},\mathbf{F}^{(2)},\mathbf{F}^{(3)}} \|\mathcal{K} - C \times_1 \mathbf{F}^{(1)} \times_2 \mathbf{F}^{(2)} \times_3 \mathbf{F}^{(3)}\|, \tag{3.9}$$

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

$$k'_{mnp} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} c_{r_1 r_2 r_3} f^{(1)}_{mr_1} f^{(2)}_{nr_2} f^{(3)}_{pr_3}. \tag{3.10}$$

Several algorithms to solve the optimization problem in (3.9) have been already developed so far. We adopt the *higher-order orthogonal iteration* (HOOI) [16, 15], which is based on singular-value decomposition (SVD) for matrices. The SVD is a well known algorithm for

Algorithm 2 The Higher-order orthogonal iteration (HOOI) algorithm for UFEKT

Input: $\mathcal{K} \in \mathbb{R}^{P\times(T-w+1)\times(T-w+1)}, R_1, R_2, R_3 \in \mathbb{R}$

Output: $C \in \mathbb{R}^{R_1\times R_2\times R_3}, \mathbf{F}^{(1)} \in \mathbb{R}^{P\times R_1}, \mathbf{F}^{(2)} \in \mathbb{R}^{(T-w+1)\times R_2}, \mathbf{F}^{(3)} \in \mathbb{R}^{(T-w+1)\times R_3}$

// Initialize $\mathbf{F}^{(1)} \in \mathbb{R}^{P\times R_1}, \mathbf{F}^{(2)} \in \mathbb{R}^{(T-w+1)\times R_2}, \mathbf{F}^{(3)} \in \mathbb{R}^{(T-w+1)\times R_3}$ using HOSVD

repeat
  $N \leftarrow 3$
  for $n = 1, \ldots, N$ **do**
    $\mathcal{Y} \leftarrow \mathcal{K} \times_1 \mathbf{F}^{(1)\mathsf{T}} \cdots \times_{n-1} \mathbf{F}^{(n-1)\mathsf{T}} \times_{n+1} \mathbf{F}^{(n+1)\mathsf{T}} \cdots \times_N \mathbf{F}^{(n)\mathsf{T}}$
    $\mathbf{F}^{(n)} \leftarrow R_n$ leading left singular vectors of $\mathbf{Y}_{(n)}$
  end for
until fit ceases to improve or maximum iterations exhausted
$C \leftarrow \mathcal{K} \times_1 \mathbf{F}^{(1)\mathsf{T}} \times_2 \mathbf{F}^{(2)\mathsf{T}} \times_3 \mathbf{F}^{(3)\mathsf{T}}$
return $C, \mathbf{F}^{(1)}, \mathbf{F}^{(2)}, \mathbf{F}^{(3)}$

Algorithm 3 The Higher-order SVD (HOSVD) algorithm

Input: $\mathcal{K} \in \mathbb{R}^{P\times(T-w+1)\times(T-w+1)}, R_1, R_2, R_3 \in \mathbb{R}$

Output: $C \in \mathbb{R}^{R_1\times R_2\times R_3}, \mathbf{F}^{(1)} \in \mathbb{R}^{P\times R_1}, \mathbf{F}^{(2)} \in \mathbb{R}^{(T-w+1)\times R_2}, \mathbf{F}^{(3)} \in \mathbb{R}^{(T-w+1)\times R_3}$

$N \leftarrow 3$
for $n = 1, \ldots, N$ **do**
  $\mathbf{F}^{(n)} \leftarrow R_n$ leading left singular vectors of $\mathbf{K}_{(n)}$
end for
$C \leftarrow \mathcal{K} \times_1 \mathbf{F}^{(1)\mathsf{T}} \times_2 \mathbf{F}^{(2)\mathsf{T}} \times_3 \mathbf{F}^{(3)\mathsf{T}}$
return $C, \mathbf{F}^{(1)}, \mathbf{F}^{(2)}, \mathbf{F}^{(3)}$

decomposing a single matrix into three matrices: one diagonal matrix and two orthogonal matrices. Furthermore, the *higher-order SVD* (HOSVD) is used as initialization and inputs into HOOI. Although the HOSVD also decompose a tensor into one core tensor and three factor matrices using SVD, it is known for that their outputs are not optimal [27]. Therefore, to overcome the issue, the HOOI has been developed by utilizing results of HOSVD. The procedures of the HOOI and the HOSVD algorithms are shown in Algorithm 2 and Algorithm 3. The HOOI also makes it possible to decompose a tensor into one core tensor and multiple factor matrices with SVD. By applying HOOI to the obtained tensor of kernel we mentioned, the tensor $\mathcal{K} \in \mathbb{R}^{P\times(T-w+1)\times(T-w+1)}$ can be decomposed into one core tensor $C \in \mathbb{R}^{P\times R\times R}$ 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 parameters. The computational cost for performing Tucker decomposition using HOOI for a third order tensor is known to be $O(T^3R + TR^4 + R^6)$ [23, 22] where it assumes ranks and size of a tensor are $C \in \mathbb{R}^{R\times R\times R}$ and $\mathcal{K} \in \mathbb{R}^{T\times T\times T}$, respectively.

After decomposing the given tensor, our idea is to focus on one of the factor matrices,

Algorithm 4 How to decide the σ used in UFEKT

Input: $\mathbf{X} \in \mathbb{R}^{P \times T}, w \in \mathbb{R}$

Output: $\mathbf{\Sigma} \in \{ \sigma^{(1)}, \sigma^{(2)}, \ldots, \sigma^{(P)} \}$

$\mathbf{\Sigma} \leftarrow \emptyset$
for $p = 1$ to P **do**
  $X \leftarrow \emptyset$
  for $(i, j) = (1, 1)$ to $(T - w + 1, T - w + 1)$ **do**
    $\Delta x_{ij} \leftarrow \sum_{s=0}^{w-1} (x_{p(i+s)} - x_{p(j+s)})^2$
    $X \leftarrow X \cup \Delta x_{ij}$
  end for
  $(\sigma^{(p)})^2 \leftarrow Mean\{ X \}$
  $\Sigma \leftarrow \Sigma \cup \sigma^{(p)}$
end for
return Σ

Algorithm 5 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
for $p = 1$ to P **do**
  for $(i, j) = (1, 1)$ to $(T - w + 1, T - w + 1)$ **do**
    $k_{ij}^{(p)} \leftarrow \exp\{ - \sum_{s=0}^{w-1} (x_{p(i+s)} - x_{p(j+s)})^2 / \sigma^2 \}$
  end for
end for
// Feature Extraction from Kernel Matrices
Construct $\mathcal{K} \in \mathbb{R}^{P \times (T-w+1) \times (T-w+1)}$ from $\mathbf{K}^{(1)}, \ldots, \mathbf{K}^{(P)}$
$(C, \mathbf{F}^{(1)}, \mathbf{F}^{(2)}, \mathbf{F}^{(3)}) \leftarrow \text{Tucker}(\mathcal{K}, [P, R, R])$
for $i = 1$ to $T - w + 1$ **do**
  $\mathbf{f}_i \leftarrow i$th-row vector of $\mathbf{F}^{(2)}$
end for
return $\mathbf{f}_1, \ldots, \mathbf{f}_{T-w+1}$

$\mathbf{F}^{(2)}$ or $\mathbf{F}^{(3)}$, and extract its row vectors as the resulting feature vectors of subsequences in the original multivariate time series. We will consistently use $\mathbf{F}^{(2)}$ 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

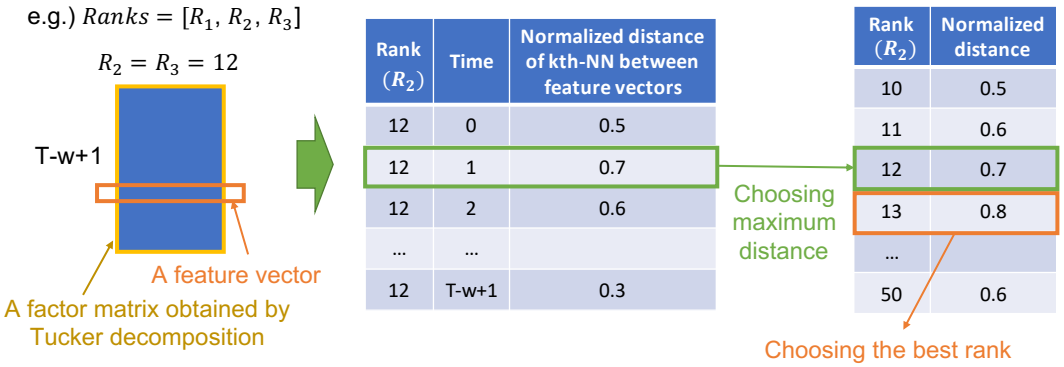


Figure 3.6: An example of rank selection.

the tensor $\mathcal{K}$. If we set ranks to $[R, R, P]$, we obtain exactly the same factor matrices $\mathbf{F}_M \in \mathbb{R}^{M \times R}$ and $\mathbf{F}_N \in \mathbb{R}^{N \times R}$ 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 same elements except for the order, that is, $k_{mnp} = k_{nmp}$ always holds. Although factor matrices $\mathbf{F}'_M$ and $\mathbf{F}'_N$, which are eventually obtained by HOOI, could be slightly different with each other due to the iteration of SVD in the framework of HOOI, they are still expected to be similar with each other. We therefore do not distinguish them and consistently use $\mathbf{F}^{(2)}$ in our algorithm. 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, we employ the original Tucker decomposition algorithm in our proposal 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, e.g., feature selection, in our future work.

As a result, we obtain row vectors $\mathbf{f}_1, \ldots, \mathbf{f}_{(T-w+1)} \in \mathbb{R}^R$ 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, we consider 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 such as outlier detection and clustering. The pseudo codes of our algorithm are summarized in Algorithm 5 and Algorithm 4.

3.2.2 Rank Selection Algorithm

In UFEKT, ranks of a core tensor in tensor decomposition must be determined in advance. We provide a guideline of how to determine it in the following. We again emphasize that we are considering the setting of unsupervised learning, and automatic parameter selection such as grid search via cross-validation cannot be used. To decompose a three-dimensional

Algorithm 6 Rank selection of a core tensor for UFEKT

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: $R_{\text{opt}} \in \mathbb{R}$

Prepare $\mathbf{d} \in \mathbb{R}^{R}$
for $i = 1$ to $T - w + 1$ **do**
  $(C_i, \mathbf{F}_i^{(1)}, \mathbf{F}_i^{(2)}, \mathbf{F}_i^{(3)}) \leftarrow \text{Tucker}(\mathcal{K}, [P, i, i])$
  $\mathcal{F}_i \leftarrow \emptyset$
  for $j = 1$ to $T - w + 1$ **do**
    $\mathbf{f}_j \leftarrow j$th row vector of $\mathbf{F}_i^{(2)}$; $\mathcal{F}_i \leftarrow \mathcal{F}_i \cup \{\mathbf{f}_j\}$
  end for
  Prepare $\mathbf{s} \in \mathbb{R}^{T-w+1}$
  for $j = 1$ to $T - w + 1$ **do**
    $\mathbf{s}_j \leftarrow d^{\kappa}(\mathbf{f}_j; \mathcal{F}_i)$
  end for
  $\mathbf{d}_i \leftarrow (\max_j \mathbf{s}_j)/(\sum_j \mathbf{s}_j)$
end for
$R_{\text{opt}} \leftarrow \text{argmax}_{i \geq i\_min} \mathbf{d}_i$
return R_{opt}

tensor into one core tensor and three factor matrices, a three-dimensional rank $[R_1, R_2, R_3]$ is required. 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 in this direction is not required. We recommend to set $R_2 = R_3$ because only one of two factor matrices will be used to construct feature vector representation in UFEKT. Hence, only one parameter R_2 should be determined in UFEKT.

To determine the rank R_2, we propose to use the κth nearest neighbor (κNN) distance. For each rank setting, we decompose a tensor, and we 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.

To get the best rank, first of all, we perform Tucker decomposition while changing ranks. After each decomposition, we measure normalized distance between feature vectors in a factor matrix and regard the longest distance as a representative distance of the rank. This slide shows the case where R2 is 12. To make the right table, choose the maximum distance from the middle table, it becomes 0.7 in this case, and set the value in the table. After making the right table, choose the best rank by selecting the longest normalized distance. This method does not guarantee to output the best rank, however, our experiments show that favorable results can be obtained.

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 parameter. As we will mention in Chapter 4, we set $\kappa = 5$ and $i\_min = 10$ in our experiments. 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 6.

Chapter 4

Outlier Detection

4.1 Feature Extraction from Multivariate Time Series

To evaluate our algorithms, UFEKS and UFEKT, we introduce two representative feature extraction algorithms, *PageRank kernel* (PRK) and *SubSequence* (SS).

4.1.1 UFEKS for Feature Extraction

To evaluate the effectiveness of our algorithm UFEKS, we propose to apply the UFEKS to the problem of unsupervised outlier detection from multivariate time series. When we use our method for a multivariate time series $\mathbf{X}$, we can extract feature vectors for subsequences in the kernel matrix $\mathbf{K}$. Then we can directly perform outlier detection on the extracted vectors to find outlier subsequences.

4.1.2 UFEKT for Feature Extraction

As the same as UFEKS above, we also apply our algorithm UFEKT to an unsupervised outlier detection problem. Given a multivariate time series $\mathbf{X}$, feature vectors of subsequences in the factor matrix $\mathbf{F}^{(2)}$ extracted by UFEKT will be used to detect outliers (anomalous subsequences) using existing outlier detection techniques.

4.1.3 PageRank Kernel (PRK) for Feature Extraction

The PageRank kernel, which we denote by PRK, has been proposed in the outlier detection method PR [12], which is considered to be the state-of-the-art technique for unsupervised outlier detection from multivariate time series. PR is a kernel-based method using the PageRank algorithm. 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 PageRank algorithm to detect outliers. Given a multivariate time series $\mathbf{X} \in \mathbb{R}^{(P \times T)}$ with P variables with the length T, PR constructs a kernel matrix $K \in \mathbb{R}^{(T-w+1)\times(T-w+1)}$

such that

$$k_{ij} = \exp\left\{-\frac{\sum_{p=1}^{P}\sum_{s=0}^{w-1}(x_{i+s}^{(p)} - x_{j+s}^{(p)})^2}{\sigma^2}\right\}, \tag{4.1}$$
$$i, j \in \{1, 2, \ldots, T - w + 1\},$$

where w is the length of each subsequence. The difference between this kernel and our kernel function in Equation (3.7) is whether or not values representing associations between subsequences calculated for each variable are summed up. In the case of Equation (4.1), information of association among variables may be lost by its summation.

4.1.4 SubSequence (SS) for Feature Extraction

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

$$\mathbf{x}_{ss} = \left(\sum_{p=1}^{P} x_1^{(p)}, \ldots, \sum_{p=1}^{P} x_T^{(p)}\right). \tag{4.2}$$

When we denote each element of $\mathbf{x}_{ss}$ at time t as $x_t^{ss} = \sum_{p=1}^{P} x_t^{(p)}$, the subsequence based matrix $\mathbf{X}_{ss} \in \mathbb{R}^{(T-w+1)\times w}$ with the window size w is defined as

$$\mathbf{X}_{ss} = \begin{bmatrix} x_1^{ss} & \cdots & x_w^{ss} \\ x_2^{ss} & \cdots & x_{w+1}^{ss} \\ \vdots & \ddots & \vdots \\ x_{T-w+1}^{ss} & \cdots & x_T^{ss} \end{bmatrix}. \tag{4.3}$$

By considering each row in Equation (4.3) to be a multidimensional data point, outliers can be detected by conventional algorithms like κNN. We compare our algorithm with PRK and SS.

4.2 Outlier Detection Methods

We employed five types of outlier detection methods, such as distance-based, density-based, kernel-based, ensemble-based, and pagerank-based algorithms for evaluating our algorithms. Furthermore, we chose representative method from each type, *κth-Nearest Neighbor* (κNN), *Local Outlier Factor* (LOF), *One-class Support Vector Machine* (OCSVM), *Isolation Forest* (IForest), and *PageRank* (PR), and briefly mention their methods.

Table 4.1: Existing methods of outlier detection for non-time series datasets.

Types	Names
A distance-based algorithm	κth-Nearest Neighbor (κNN)
A density-based algorithm	Local Outlier Factor (LOF)
A kernel-based algorithm	One-class Support Vector Machine (OCSVM)
A ensemble-based algorithm	Isolation Forest (IForest)
A PageRank-based algorithm	PageRank (PR)

4.2.1 Distance-based Algorithm

The κth-Nearest Neighbor (κNN) [26, 41] is one of the distance-based algorithm and is widely used in the data mining field. Outlierness to be measured by κNN has been proposed by RAMASWAMY et al. and the score $q(\mathbf{X}_t)$ is defined as:

$$q(\mathbf{X}_t) := d^{\kappa}(\mathbf{k}_t; \mathcal{S}_{\text{fvec}}), \tag{4.4}$$

where the $d^{\kappa}(\mathbf{k}_t; \mathcal{S}_{\text{fvec}})$ is Euclidean distance from $\mathbf{k}_t \in \mathbb{R}^{(T-w+1)P}$, which is the feature vector representation of $\mathbf{X}_t$ to its κth-nearest neighbor in the set $\mathcal{S}_{\text{fvec}} = \{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_{T-w+1}\}$.

4.2.2 Density-based Algorithm

There is Local Outlier Factor (LOF) [9] as one of the representative distance-based algorithms. The LOF is defined as the ratio of the reachability density of data point $\mathbf{k}_t$ and the score $q(\mathbf{X}_t)$ is defined as:

$$q(\mathbf{X}_t) := \left(|N^{\kappa}(\mathbf{k}_t)|^{-1} \sum_{\mathbf{k}_t^{'} \in N^{\kappa}(\mathbf{k}_t)} \rho(\mathbf{k}_t^{'}) \right) \rho(\mathbf{k}_t)^{-1}, \tag{4.5}$$

$$\rho(\mathbf{k}_t) := |N^{\kappa}(\mathbf{k}_t)| \left(\sum_{\mathbf{k}_t^{'} \in N^{\kappa}(\mathbf{k}_t)} max \left\{ d^{\kappa}(\mathbf{k}_t, \mathbf{X}_t), d(\mathbf{k}_t, \mathbf{k}_t^{'}) \right\} \right)^{-1}, \tag{4.6}$$

where $N^{\kappa}(\mathbf{k}_t)$ is the set of κth-nearest neighbors of data point $\mathbf{k}_t$.

4.2.3 Kernel-based Algorithm

The One-class Support Vector Machine (OCSVM) [33, 43] classifies given datasets into inliers and outliers by introducing a hyperplane between them. If a data point is located in the outlier space, the data point would be decided as an outlier. The score $q(\mathbf{X}_t)$ of a vector $\mathbf{k}_t$ with a feature map Φ is defined as:

$$q(\mathbf{X}_t) := \rho - (\omega \cdot \Phi(\mathbf{k}_t)), \tag{4.7}$$

where the offset ρ and the weight vector $\boldsymbol{\omega}$ are optimized by the quadratic programming:

$$\min_{\boldsymbol{\omega}\in\mathcal{F},\boldsymbol{\xi}\in\mathbb{R}^n,\rho\in\mathbb{R}} \frac{1}{2}||\boldsymbol{\omega}||^2 + \frac{1}{\nu n}\sum_{i=1}^{T-w+1}\xi_i - \rho, \tag{4.8}$$

$$\text{subject to } (\boldsymbol{\omega}\cdot\Phi(\mathbf{k}_i)) \geq \rho - \xi_i, \ \ \xi_i \geq 0, \tag{4.9}$$

with a regularization parameter ν.

4.2.4 Ensemble-based Algorithm

The Isolation Forest (IForest) [31] is known for one of the ensemble-based algorithms and a random forest-like method. The outlierness of a data point $\mathbf{k}_t$ is measured by path length $h(\mathbf{k}_t)$ on a tree and the score $q(\mathbf{X}_t)$ is defined as:

$$q(\mathbf{X}_t) := 2^{-\overline{h(\mathbf{k}_t)}/c(s)}, \tag{4.10}$$

$$c(s) := 2H(s-1) - \frac{2(s-1)}{T-w+1}, \tag{4.11}$$

where $\overline{h(\mathbf{k}_t)}$, s and H indicate the average of $h(\mathbf{k}_t)$, the depth of the tree, and the harmonic number, respectively.

4.2.5 PageRank-based Algorithm

The PageRank-based algorithm (PR) is proposed as outlier detection algorithm for time series data [12]. After making a kernel matrix defined in Equation (4.1), PR constructs a weighted graph $G = (V, E)$, where V corresponds to the set of subsequences, and use the PageRank algorithm on the graph to quantify the outlierness of each subsequence, where anomalous subsequences will receive low score in the method.

4.3 Evaluation Metrics

For evaluating accuracy of our algorithms, we introduce *area under precision recall curve* (AUPRC) as an evaluation metric. AUPRC is widely used for evaluating accuracy especially for outlier detection problems. Precision and Recall are defined as follows:

$$Precision(t) := \frac{|S(t)\cap G|}{|S(t)|}, \tag{4.12}$$

$$Recall(t) := \frac{|S(t)\cap G|}{|G|}, \tag{4.13}$$

where $S(t)$ and G indicate a set of data points that are inferred to be outliers with a threshold t and a set of data points with ground truth labels, respectively. Once they are calculated, it is possible to plot a curve between the precision and the recall by varying the threshold t. An AUPRC score corresponds to an area under the curve. The score takes values between zero and one, and higher is better.

Table 4.2: A package list of Python that we used in our experiments. Some packages called from the listed packages are not written here.

Package	Version
matplotlib	3.2.1
networkx	2.5
numpy	1.18.4
pandas	1.1.4
scikit-learn	0.23.1
scipy	1.4.1
seaborn	0.11.0
statsmodels	0.12.1
tensorly	0.5.0

Table 4.3: Parameters for algorithms. The κ and σ are used for κNN and PRK, respectively. The length of subsequences, or the window size, is used in not only SS but all algorithms to convert a given time-series to a set of subsequences.

Name	Value
κ for κ-th nearest neighbor (κNN)	5
σ for PageRank kernel (PRK)	1
Length of subsequence or window size of time series	2

4.4 Experimental Environment

We used CentOS release 6.10 with 4x 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 performed on the same platform. A package list of Python used in our experiments is shown in Table 4.2.

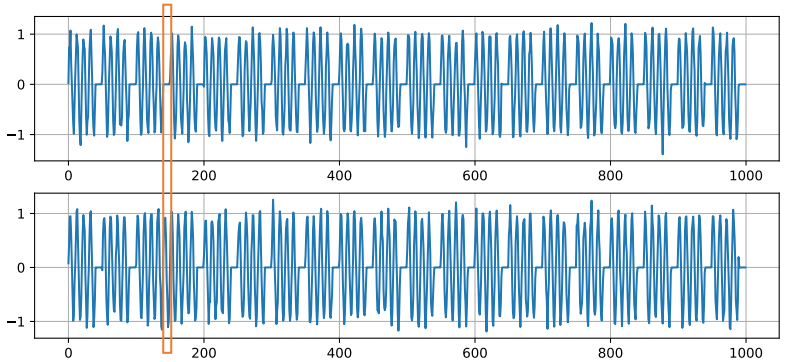
As for hyper parameters, we set $\kappa = 5$, in κth-Nearest Neighbor which is commonly used in literature [6, 8], and set $\sigma = 1$, which is known to be an appropriate value for normalized datasets. The length of subsequences or the window size was set to be two to avoid low resolution and to improve accuracy of AUPRCs. If the window size increases further, it becomes low resolution, resulting in low AUPRCs. We show each parameter that we used in the algorithms in Table 4.3.

4.5 Datasets

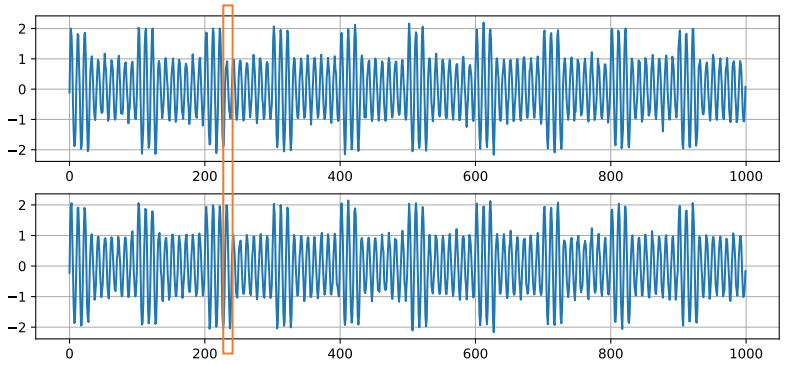
4.5.1 Synthetic Datasets

We prepared nine types of synthetic multivariate time series datasets. Each synthetic dataset includes two or more time series. Outlierness behavior occurs in only one of time series shown as an orange solid area in Figures from 4.1a to 4.4. All the nine synthetic datasets are composed of sine waves or straight lines, and Gaussian noise were added to every data point, where the noise were generated by Gaussian distribution with zero mean and 0.1 standard deviation $\mathcal{N}(0, 0.1^2)$. We generated each synthetic dataset ten times with random noise.

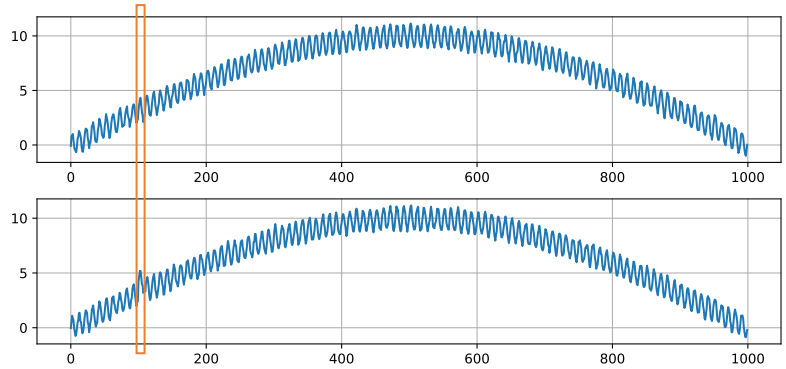
Eight figures from Figure 4.1a to Figure 4.2d illustrate synthetic multivariate time series datasets, each of which is composed of two time series. Each time series has 1,000 time stamps, and outliers with the length of ten or eleven time stamps are injected. SYN1 time series in Figure 4.1a is composed of sine waves and straight lines. SYN2 time series in Figure 4.1b is composed of two sine waves with different amplitudes. Datasets from SYN3 to SYN6 illustrated in Figure 4.1c, Figure 4.1d, Figure 4.2a, and Figure 4.2b are composed of sine waves, and their averages in subsequence are swaying over time. Moreover, a phase shift occurs between their time series in SYN6. SYN7 time series in Figure 4.2c is composed of sine waves with different amplitudes. 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 Figure 4.3a and Figure 4.3b. The SYN9 dataset in Figure 4.4 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 series are fifteen and thirty, respectively. The second time series is combined with two wavelengths of ten and twenty. 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.



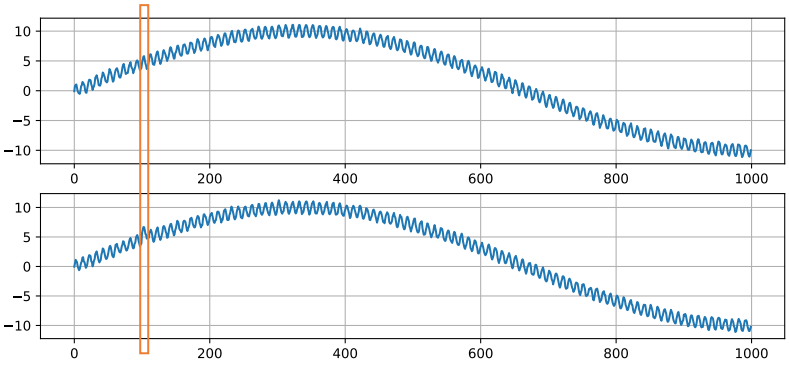
(a) SYN1 time series.



(b) SYN2 time series.

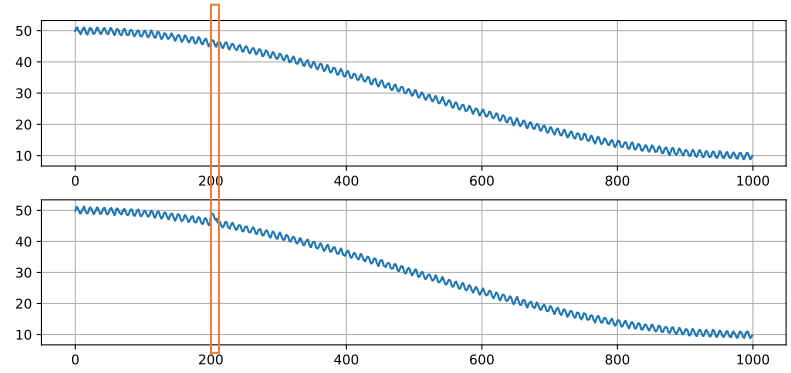


(c) SYN3 time series.

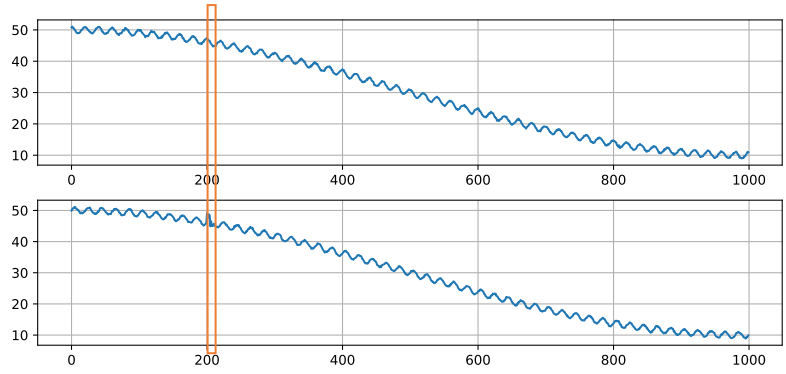


(d) SYN4 time series.

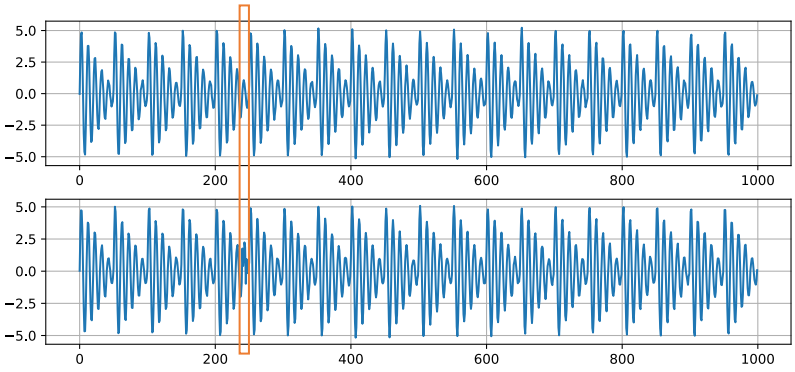
Figure 4.1: Synthetic datasets (SYN1-SYN4). Orange solid areas indicate time spans including combinatorial outliers.



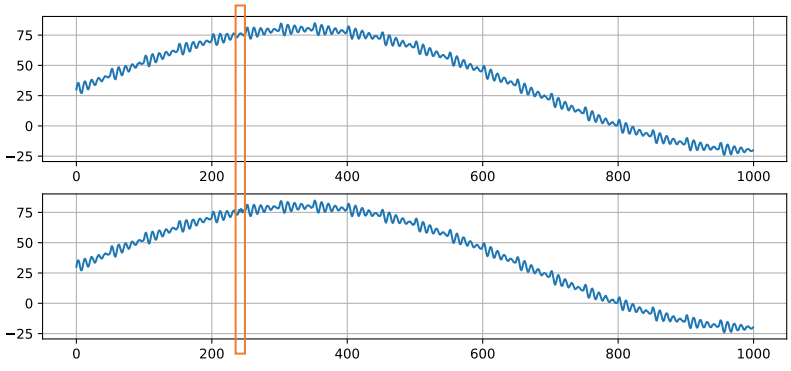
(a) SYN5 time series.



(b) SYN6 time series.

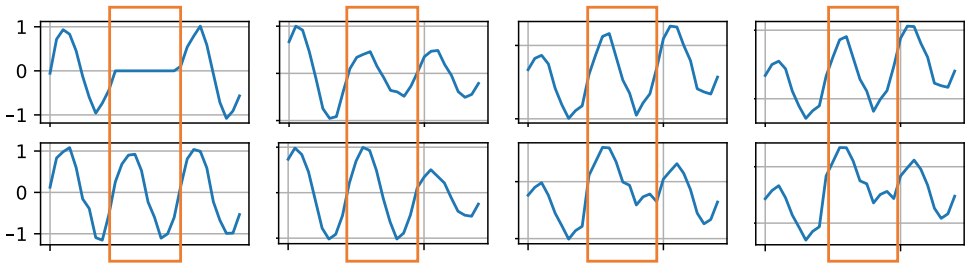


(c) SYN7 time series.

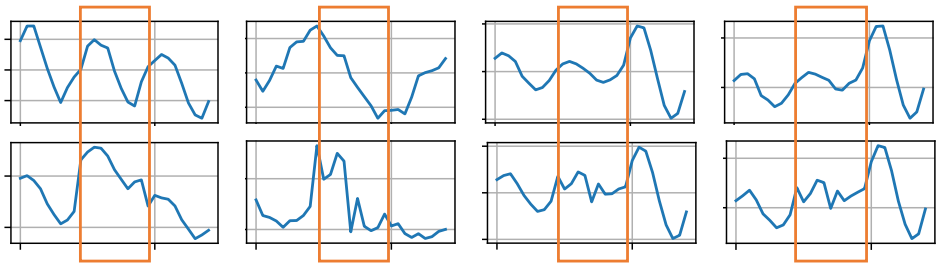


(d) SYN8 time series.

Figure 4.2: Synthetic datasets (SYN5-SYN8). Orange solid areas indicate time spans including combinatorial outliers.



(a) The plots from left to right correspond to enlarged plots between specific time spans that include outliers from SYN1 to SYN4 time series.



(b) The plots from left to right correspond to enlarged plots between specific time spans that include outliers from SYN5 to SYN8 time series.

Figure 4.3: Synthetic datasets (Enlarged SYN1-SYN8). Orange solid areas indicate time spans including combinatorial outliers.

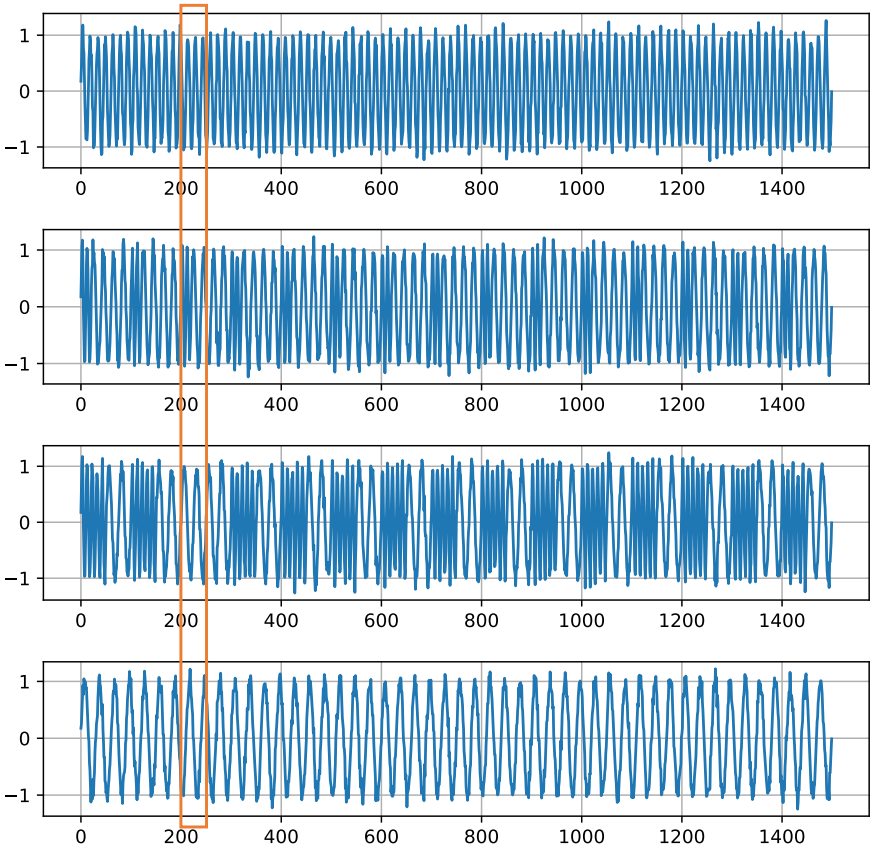


Figure 4.4: Synthetic datasets (SYN9). Orange solid areas indicate time spans including combinatorial outliers.

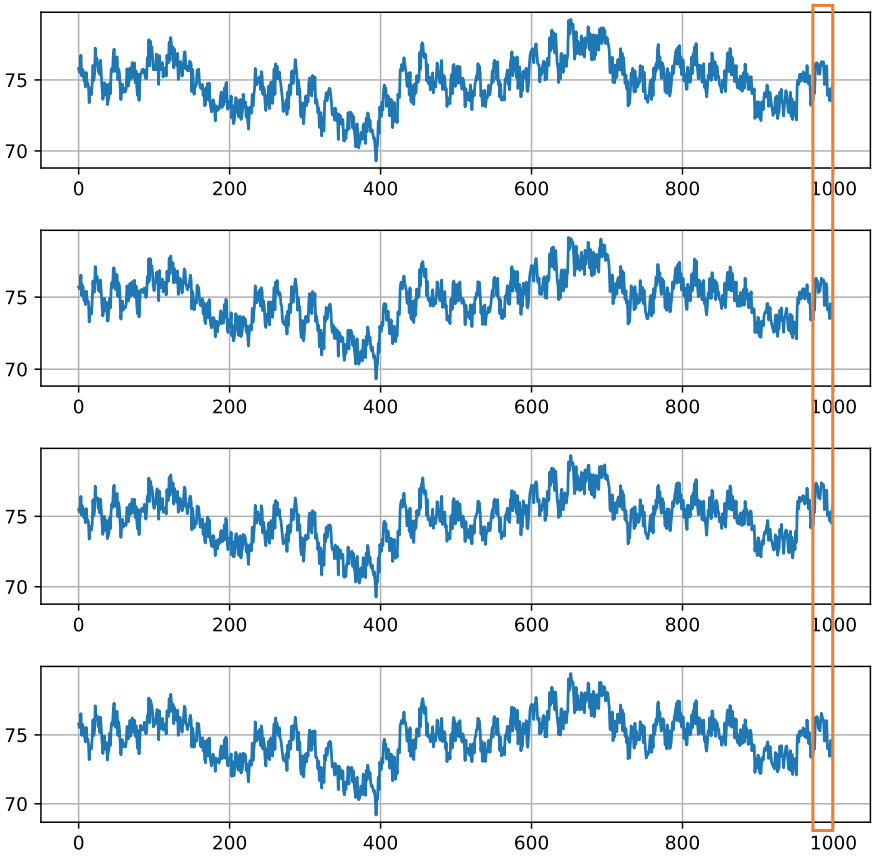
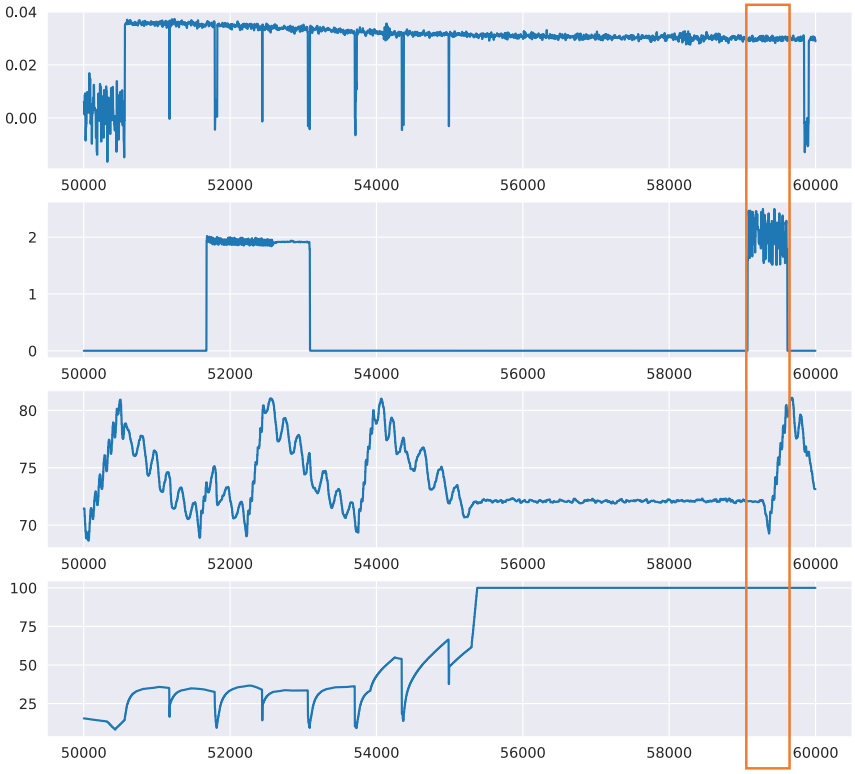


Figure 4.5: Real-world datasets (ATSF). Orange solid areas indicate time spans including combinatorial outliers.

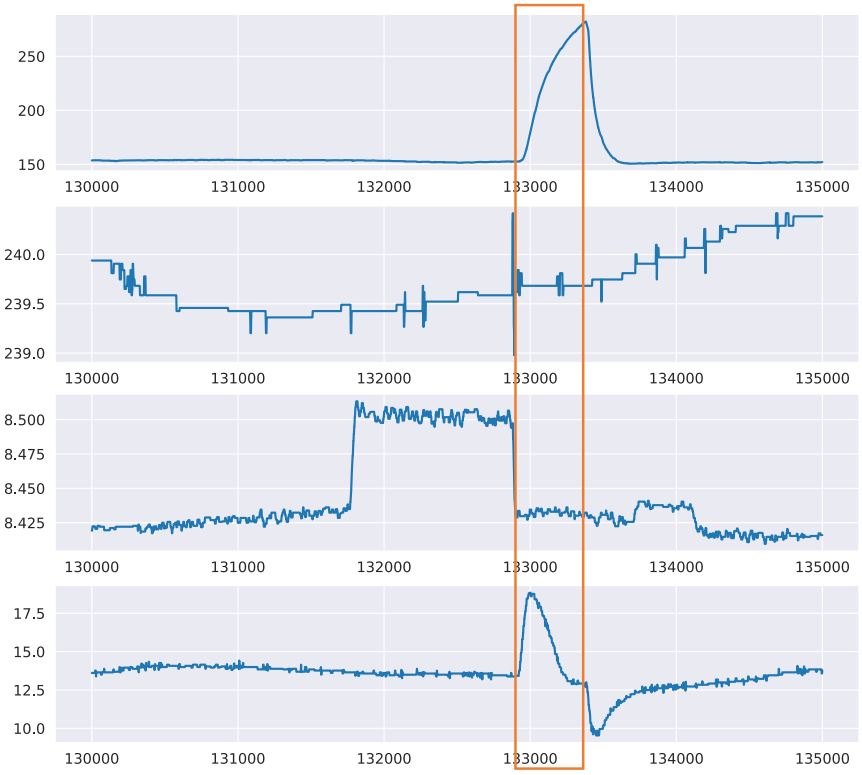
4.5.2 Real-world Datasets

We prepared six types of real-world multivariate time series datasets. Real-world datasets *ambient temperature system failure* (ATSF) are shown in Figure 4.5, which come from the Numenta Anomaly Benchmark (NAB) v1.1 publicly available[1] [4] 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 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 with 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. Such outliers simulate, for example, abnormal drift of a temperature sensor. Similar to synthetic datasets, we repeated generating each ATSF dataset ten times with adding random noise.

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



(a) WADI time series.



(b) SWaT time series.

Figure 4.6: Examples of real-world datasets (WADI and SWaT). Orange solid areas indicate time spans including combinatorial outliers.

Table 4.4: Summary of synthetic and real-world datasets.

Name 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

Furthermore, we employed another types of multivariate time series [2] called *Water Distribution* (WADI) [3] illustrated in Figure 4.6a and *Secure Water Treatment* (SWaT) [4] illustrated in Figure 4.6b. 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 and successive 5,000 time stamps 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 do not artificially inject any outliers in both WADI and SWaT datasets.

4.6 Experimental Results and Discussion for UFEKS

4.6.1 Experimental Results

Results are summarized in Figure 4.7, Figure 4.8, and Tables 4.5 and Tables 4.6. OD, FR, PRK, and SS in the figures stand for Outlier Detection, Feature Representation, PageRank Kernel, and SubSequence, respectively. The datasets except for WADI and SWaT are prepared

[2] iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design
[3] https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_wadi/
[4] https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_swat/

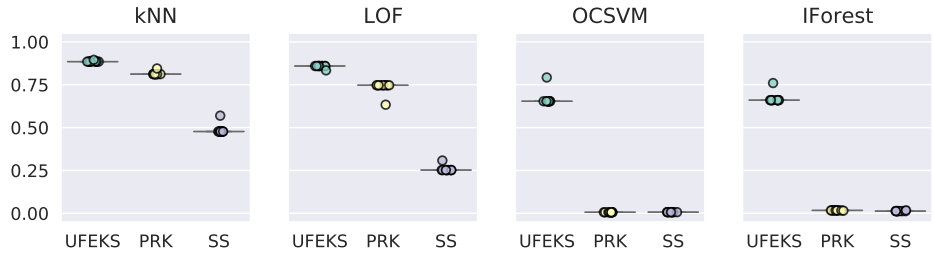
Table 4.5: Area under precision-recall curve (AUPRC) obtained by UFEKS for synthetic datasets. OD, FR, PRK and SS stand for Outlier Detection, Feature Representation, PageRank Kernel and SubSequence respectively. Averages±StandardDeviations in ten trials are shown in the table. Best scores are denoted in bold.

OD		κNN			LOF	
FR	UFEKS	PRK	SS	UFEKS	PRK	SS
SYN1	**0.886** ±0.003	0.815 ±0.010	0.486 ±0.028	0.857 ±0.008	0.736 ±0.034	0.258 ±0.017
SYN2	**0.913** ±0.054	0.734 ±0.048	0.575 ±0.055	0.803 ±0.043	0.062 ±0.031	0.380 ±0.122
SYN3	**0.980** ±0.013	0.625 ±0.167	0.016 ±0.005	0.953 ±0.028	0.118 ±0.041	0.011 ±0.002
SYN4	**0.956** ±0.028	0.573 ±0.111	0.010 ±0.002	0.901 ±0.017	0.114 ±0.073	0.011 ±0.003
SYN5	**0.990** ±0.005	0.046 ±0.009	0.007 ±0.001	0.957 ±0.028	0.006 ±0.000	0.011 ±0.003
SYN6	**0.237** ±0.068	0.063 ±0.019	0.182 ±0.051	0.023 ±0.009	0.007 ±0.001	0.139 ±0.043
SYN7	0.853 ±0.012	0.779 ±0.057	0.010 ±0.001	0.867 ±0.031	0.047 ±0.029	0.060 ±0.069
SYN8	**0.739** ±0.074	0.012 ±0.001	0.006 ±0.000	0.006 ±0.000	0.007 ±0.000	0.007 ±0.000
SYN9	**0.375** ±0.026	0.368 ±0.041	0.033 ±0.001	0.140 ±0.020	0.069 ±0.020	0.035 ±0.002
Average	**0.770**	0.446	0.147	0.612	0.129	0.101
OD		OCSVM			IForest	
FR	UFEKS	PRK	SS	UFEKS	PRK	SS
SYN1	0.668 ±0.042	0.006 ±0.000	0.007 ±0.000	0.670 ±0.030	0.017 ±0.000	0.013 ±0.001
SYN2	0.600 ±0.036	0.006 ±0.000	0.018 ±0.000	0.303 ±0.143	0.006 ±0.000	0.019 ±0.001
SYN3	0.418 ±0.105	0.006 ±0.000	0.008 ±0.000	0.017 ±0.005	0.006 ±0.000	0.011 ±0.001
SYN4	0.508 ±0.109	0.006 ±0.000	0.007 ±0.000	0.014 ±0.001	0.006 ±0.000	0.007 ±0.000
SYN5	0.545 ±0.114	0.006 ±0.000	0.010 ±0.000	0.022 ±0.009	0.006 ±0.000	0.008 ±0.001
SYN6	0.008 ±0.002	0.007 ±0.001	0.008 ±0.000	0.007 ±0.000	0.007 ±0.000	0.014 ±0.005
SYN7	**0.894** ±0.013	0.006 ±0.000	0.006 ±0.000	0.374 ±0.049	0.007 ±0.000	0.006 ±0.000
SYN8	0.331 ±0.075	0.012 ±0.003	0.014 ±0.000	0.162 ±0.026	0.051 ±0.016	0.006 ±0.000
SYN9	0.117 ±0.021	0.020 ±0.001	0.032 ±0.001	0.070 ±0.020	0.030 ±0.002	0.032 ±0.001
Average	0.454	0.008	0.012	0.182	0.015	0.013

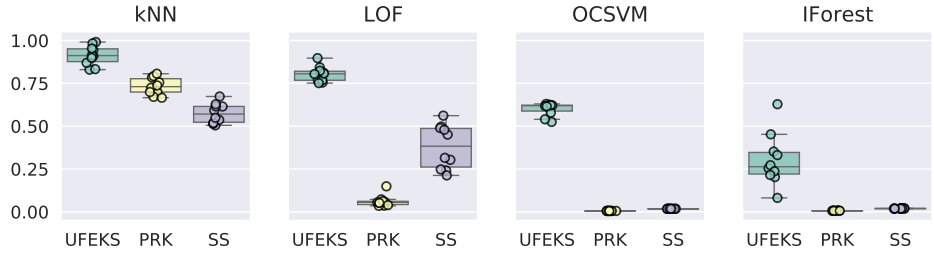
ten patterns for each as we inject Gaussian noises and Averages ± StandardDeviations in ten trials are shown in the table. Best scores are denoted in bold.

Synthetic datasets (SYN)

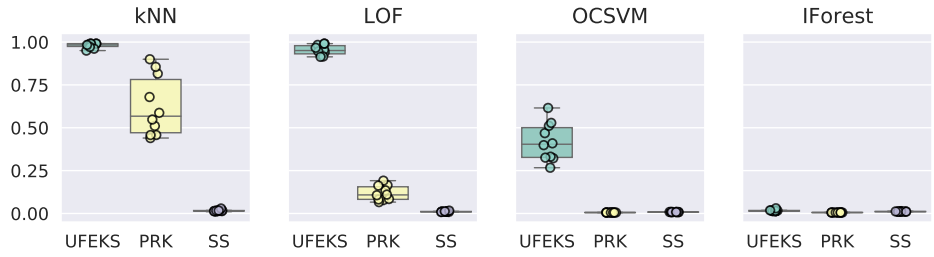
Figures from Figure 4.7a to Figure 4.9 show results of synthetic datasets. There are four plots in each figure and each title in their plots shows the name of the corresponding outlier detection algorithm. Our algorithm, UFEKS, is superior to the other feature representation algorithms PageRank Kernel (PRK) and SubSequence (SS) in all synthetic datasets. In comparison with PRK and SS, our kernel matrix used in the algorithm does not sum up elements in a row of the kernel matrix that represents association between subsequences, while PRK and SS sum up them. Therefore, it is considered that UFEKS has high capability



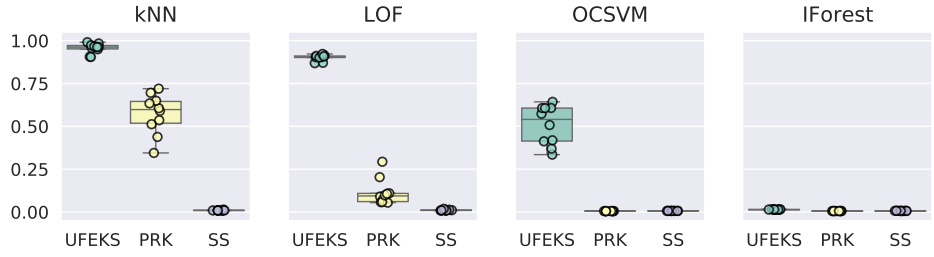
(a) SYN1 time series.



(b) SYN2 time series.



(c) SYN3 time series.



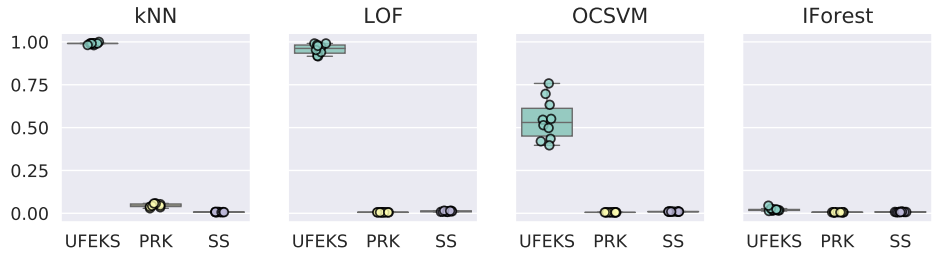
(d) SYN4 time series.

Figure 4.7: AUPRCs for SYN1, SYN2, SYN3, ans SYN4 by UFEKS.

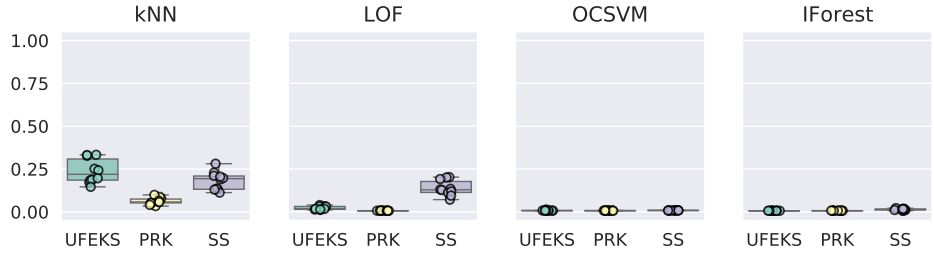
of representing features. Moreover, by using Gaussian kernel, it is expected that UFEKS can reduce noise and extract features easier than SS. Furthermore, it is also interesting that κNN, which is an outlier detection algorithm, also tends to produce better results than the other algorithms except for SYN8 datasets. Their details are shown in Table 4.5.

Real-world datasets (ATSF)

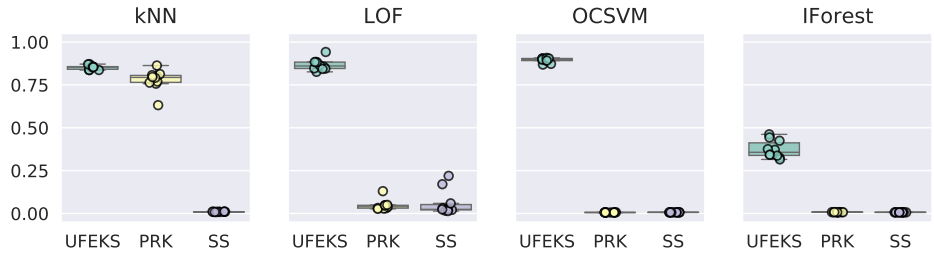
Figures from Figure 4.10a to Figure 4.10d show results of ATSF datasets. A detail of the results is shown in Table. 4.6. The results tend to be similar to synthetic datasets, that is,



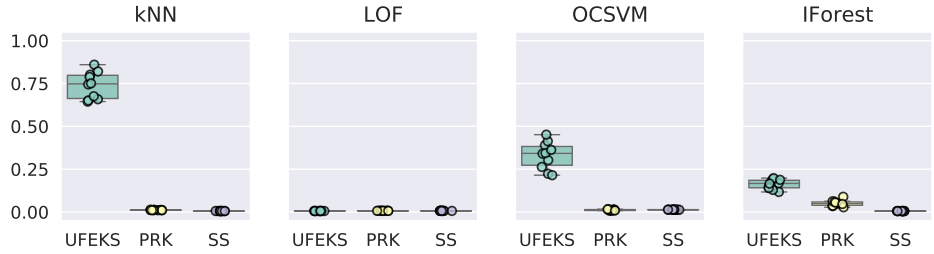
(a) SYN5 time series.



(b) SYN6 time series.



(c) SYN7 time series.



(d) SYN8 time series.

Figure 4.8: AUPRCs for SYN5, SYN6, SYN7, and SYN8 by UFEKS.

combinations of κNN and UFEKS have resulted in high accuracy for all datasets except for ATSF8.

Real-world datasets (WADI and SWaT)

Figure 4.11a and Figure 4.11b show results for WADI and SWaT datasets. These results are different from those for the other datasets because OCSVM and IForest have better results than κNN. Moreover, our algorithm, UFEKS, is not better than SS.

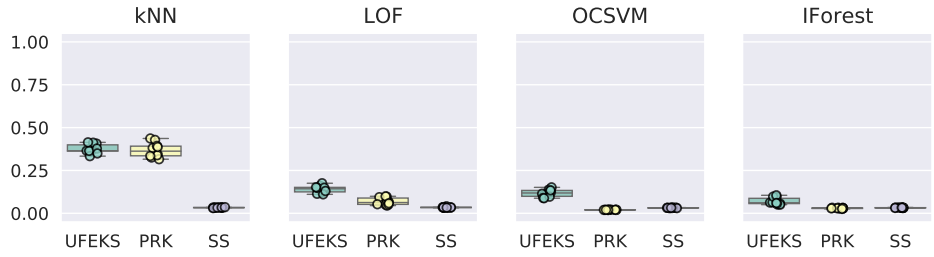


Figure 4.9: AUPRCs for SYN9 by UFEKS.

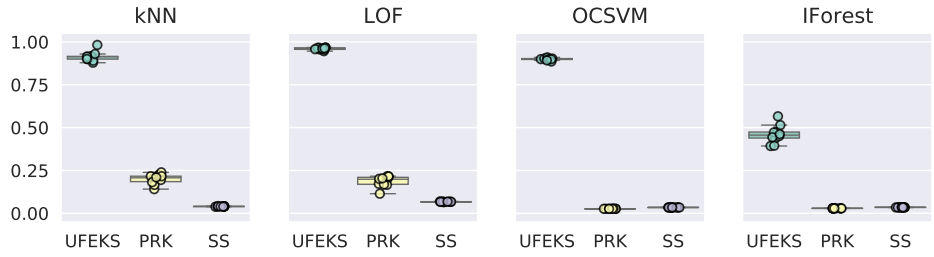
Table 4.6: Area under precision-recall curve (AUPRC) obtained by UFEKS for real-world datasets. OD, FR, PRK and SS stand for Outlier Detection, Feature Representation, PageRank Kernel and SubSequence respectively. Averages±StandardDeviations in ten trials are shown in the table, however, WADI and SWaT are performed only once. Best scores are denoted in bold.

OD		κNN			LOF	
FR	UFEKS	PRK	SS	UFEKS	PRK	SS
ATSF8	0.914 ±0.027	0.199 ±0.028	0.040 ±0.001	**0.960** ±0.006	0.187 ±0.030	0.067 ±0.001
ATSF16	**0.868** ±0.018	0.260 ±0.038	0.039 ±0.001	0.630 ±0.012	0.066 ±0.009	0.064 ±0.001
ATSF32	**0.614** ±0.027	0.186 ±0.018	0.039 ±0.001	0.176 ±0.010	0.032 ±0.002	0.063 ±0.001
ATSF64	**0.306** ±0.015	0.089 ±0.005	0.038 ±0.001	0.080 ±0.003	0.028 ±0.000	0.063 ±0.000
WADI	0.092	0.047	0.128	0.057	0.064	0.055
SWaT	0.143	0.085	0.118	0.088	0.070	0.100
Average	0.489	0.144	0.067	0.332	0.074	0.069
OD		OCSVM			IForest	
FR	UFEKS	PRK	SS	UFEKS	PRK	SS
ATSF8	0.899 ±0.006	0.026 ±0.000	0.034 ±0.000	0.461 ±0.049	0.029 ±0.000	0.035 ±0.001
ATSF16	0.821 ±0.026	0.026 ±0.000	0.034 ±0.000	0.247 ±0.042	0.030 ±0.000	0.034 ±0.000
ATSF32	0.329 ±0.026	0.027 ±0.000	0.034 ±0.000	0.132 ±0.021	0.030 ±0.000	0.034 ±0.000
ATSF64	0.189 ±0.003	0.032 ±0.003	0.034 ±0.000	0.113 ±0.024	0.034 ±0.000	0.034 ±0.000
WADI	0.249	0.032	**0.751**	0.172	0.033	0.628
SWaT	0.783	0.118	**0.881**	0.643	0.054	0.365
Average	**0.545**	0.044	0.295	0.295	0.035	0.188

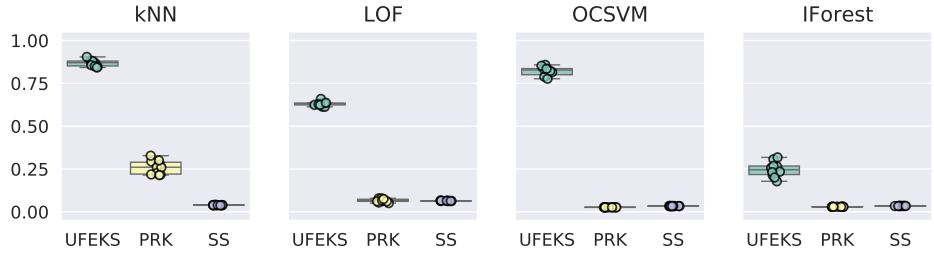
4.6.2 Discussion - Principal Component Analysis (PCA)

Synthetic Datasets (SYN)

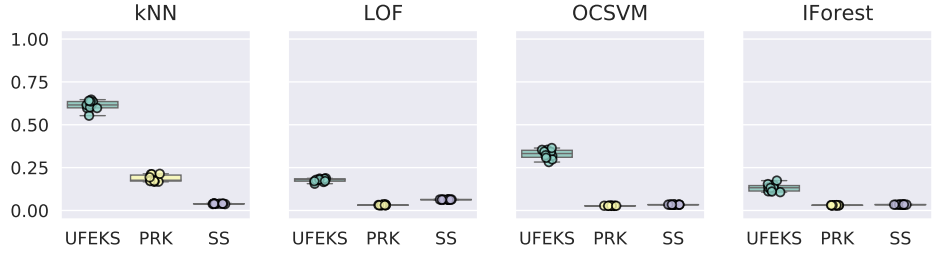
As shown in Subsection 4.6.1, κNN had good results. One of the reasons is that outlier points tend to place far away from normal points. To analyze the difference between feature representation methods deeper, we apply Principal Component Analysis (PCA) for the obtained feature representations from SYN7 datasets as a representative example. The result obtained by SubSequence (SS) is plotted in Figure 4.12 and those by UFEKS and PageRank



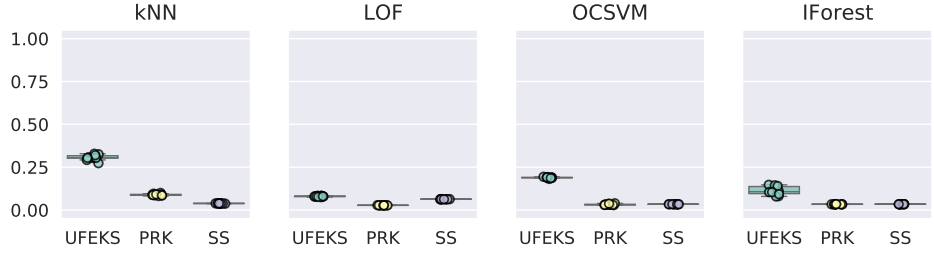
(a) ATSF8 time series.



(b) ATSF16 time series.



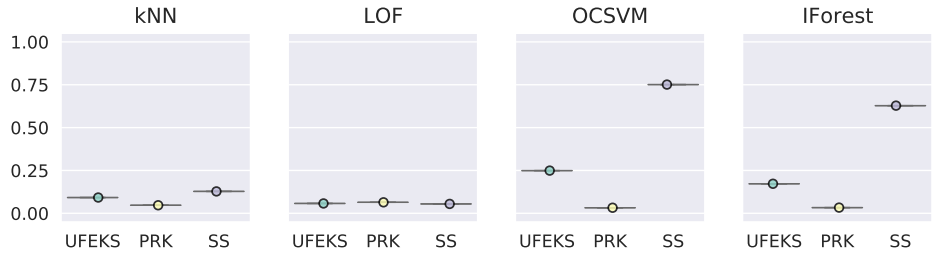
(c) ATSF32 time series.



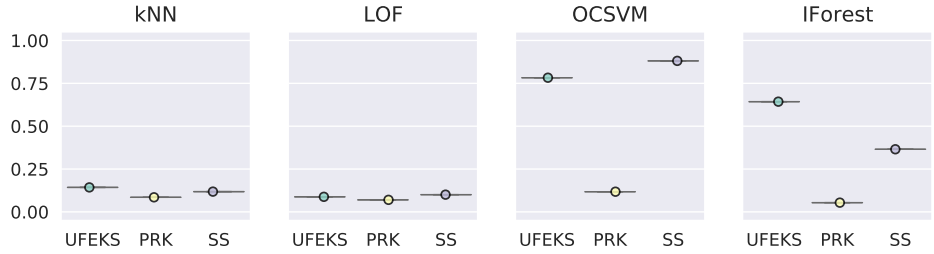
(d) ATSF64 time series.

Figure 4.10: AUPRCs for ATSF8, ATSF16, ATSF32, and ATSF64 by UFEKS.

kernel are in from Figure 4.13 to Figure 4.18. The x- and y-axes in Figure 4.12, Figure 4.13 and Figure 4.16 indicate the first and the second principal components, respectively. Furthermore, we plot the second and third principal components in Figure 4.14 and Figure 4.17, and the second and fourth ones in Figure 4.15 and Figure 4.18. The circles and crosses in all plots denote normal and outlier points, respectively. As the Figure 4.12 shows, it seems to be difficult to detect outliers by an existing algorithm like κNN from this feature representation because most outliers are close to the normal data points. In contrast, we can see some outliers that are apart from normal data points in Figure 4.14 and Figure 4.18. This demonstrates the



(a) WADI time series.



(b) SWaT time series.

Figure 4.11: AUPRCs for WADI and SWaT by UFEKS.

effectiveness of our approach as it means that there is a possibility to detect such outliers by distance-based outlier detection methods from these feature representations. Note that a feature representation matrix from SubSequence (SS) has only two dimensions as we set the length of subsequence to be two.

Real-world Datasets (WADI and SWaT)

As shown in Subsection 4.6.1, the results obtained by our algorithm, UFEKS, is not better than SS. To analyze this reason, we illustrate results of PCA for SWaT datasets in Figure 4.19, Figure 4.20, and Figure 4.21. The x- and y-axes indicate the first and second principal component, the second and the third principal components, and the second and fourth principal components, respectively. The circles and crosses in these plots denote normal and outlier points, respectively. These three plots have different features compared with other PCA plots for synthetic datasets that we have shown before. Outliers and normal data points are aligned with each other. Furthermore, distances between data points seem to be comparatively equal. In this case, detecting outliers using distance-based outlier detection algorithms such as κNN may be difficult. Although it might be possible to detect them by supervised learning, it is out of scope of this thesis. We consider that both WADI and SWaT time series include a variety types of data patterns and it is fundamentally difficult to detect outliers from these datasets in an unsupervised manner.

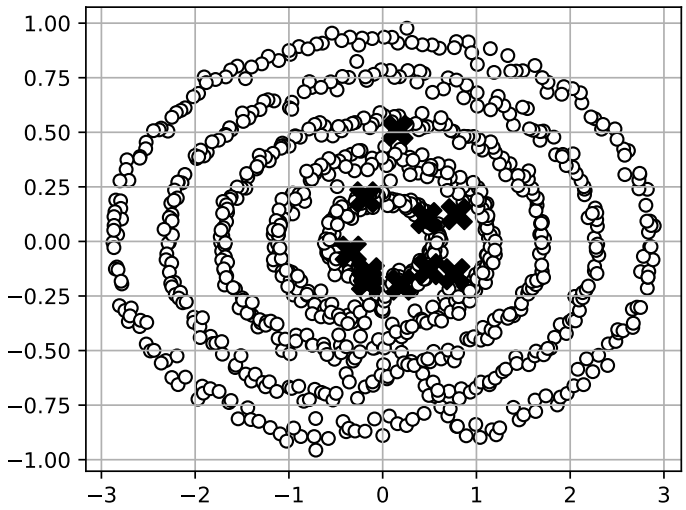


Figure 4.12: A result of PCA that is applied to the feature representation obtained by SubSequence (SS) for the SYN7 time series dataset. The 1st and 2nd principal components are shown here.

4.7 Experimental Results and Discussion for UFEKT

4.7.1 Experimental Results

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 twelve combinations of feature extraction and outlier detection in total. In addition to them, we tried to perform one of the common algorithms called Prophet [49], which is a forecasting procedure for univariate time series. However, we could not get results of Prophet due to high computational cost and it was hard to decide date and time information correctly for our datasets, which is required for Prophet as additional input. The effectiveness of each method was evaluated by the *area under precision-recall curve* (AUPRC) [1]. The AUPRC score takes values between zero and one, and higher is better. Moreover, we show each parameter that we used in the algorithms in Table 4.3. We set $\kappa = 5$, which is 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 4.7 and Table 4.8. OD, FR, PR, PRK, and SS in the tables stand for Outlier Detection, Feature Representation, PageRank, PageRank Kernel, and SubSequence, respectively. All datasets except for WADI and SWaT were created ten

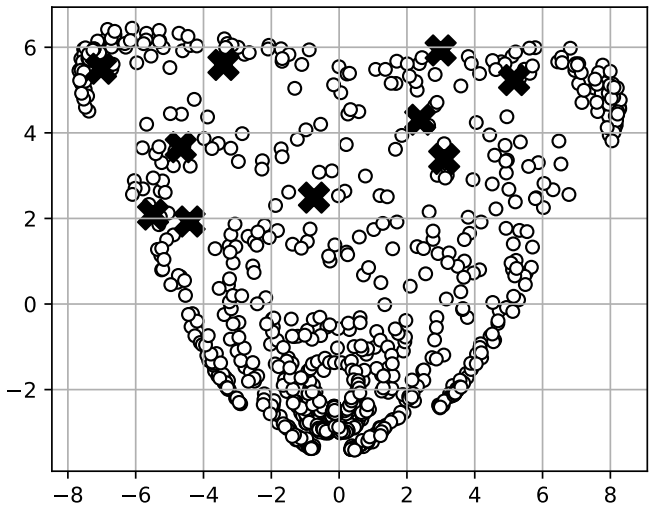


Figure 4.13: A result of PCA that is applied to the feature representation obtained by UFEKS for the SYN7 time series dataset. The 1st and 2nd principal components are shown in this figure.

times with adding Gaussian noises. Therefore, each method are performed ten times for each dataset except for WADI and SWaT. In addition to the tables, we show plots of the results of every dataset in Figure 4.22, Figure 4.23, and Figure 4.24 for synthetic datasets, Figure 4.25 for ATSF datasets, and Figure 4.26 for WADI and SWaT datasets.

Synthetic Datasets (SYN)

Results of SYN datasets are shown in Table 4.7. 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 a 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. 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 distance-based algorithm, have a good result is that outlier points tend to place far away from normal points in a kernel space.

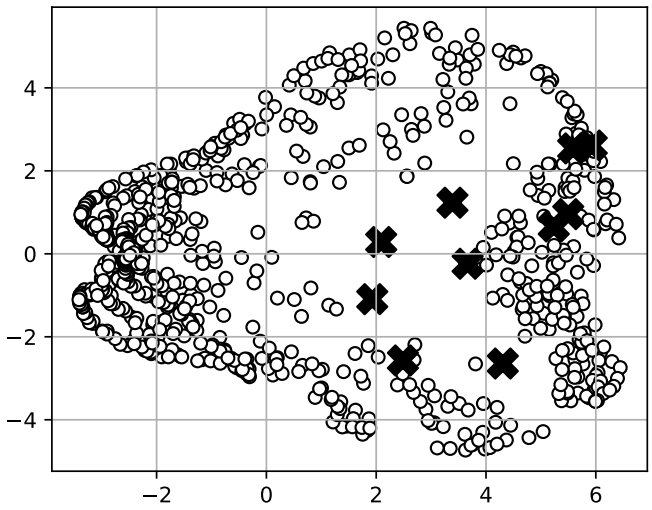


Figure 4.14: A result of PCA that is applied to the feature representation obtained by UFEKS for the SYN7 time series dataset. The 2nd and 3rd principal components are shown in this figure.

Real-world Datasets (ATSF)

Results of ATSF datasets are shown in Table 4.8. 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 of the multivariate time series separately. In the case of ATSF8, there are eight similar time series with artificially injected noise. The results of UFEKT remain high accuracy even if the number of variables are increased. This is one of the characteristics of UFEKT.

Real-world Datasets (WADI and SWaT)

Results of WADI and SWaT datasets are shown in Table 4.8. Their results are different from the other datasets because SS is superior to our algorithm UFEKT. Those datasets have different tendencies compared to other datasets, SYN and ATSF, because each time series has a unique shape shown in Fig.4.6a and Figure 4.6b, and the differences between variables are not dominant. This means that feature extraction and outlier detection to these datasets are fundamentally difficult.

4.7.2 Discussion - Rank Dependencies of AUPRCs

We examine the effectiveness of our parameter selection algorithm in Algorithm 6, which tries to find a good choice of a parameter, rank of a core tensor, used in UFEKT. We compare

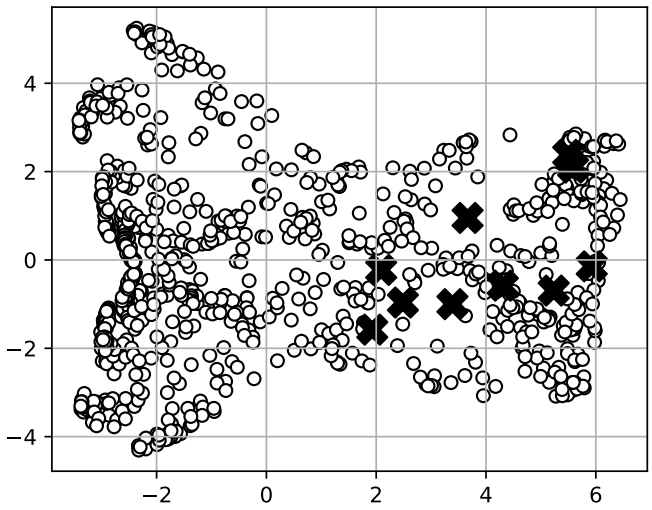


Figure 4.15: A result of PCA that is applied to the feature representation obtained by UFEKS for the SYN7 time series dataset. The 2nd and 4th principal components are shown in this figure.

the AUPRCs obtained by Algorithm 6 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. Figure 4.27 shows 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 4.9 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.

4.7.3 Discussion - Usage of The Rest of Factor Matrices by UFEKT

As we have mentioned above, in the UFEKT, three factor matrices are produced by Tucker decomposition and only one of the matrices is used for extracting feature vectors, that is, the rest of the matrices remained unused. However, one of the remaining matrices in variable direction can be considered to contain information about outliers. To make sure if it is true, we prepared real-world datasets called *ambient temperature system failure* (ATSF) datasets shown in Figure 4.28. The datasets we employed are the same as datasets for outlier detection as we have mentioned except for outlierness. The outliers are located between 950 and 1,000 time stamps at the second plot from the bottom on the left side and the second plot from the top on the right side in the figure, that is, two out of sixteen variables have included outliers. Their outliers are made by adding about one percent values of the original values. After

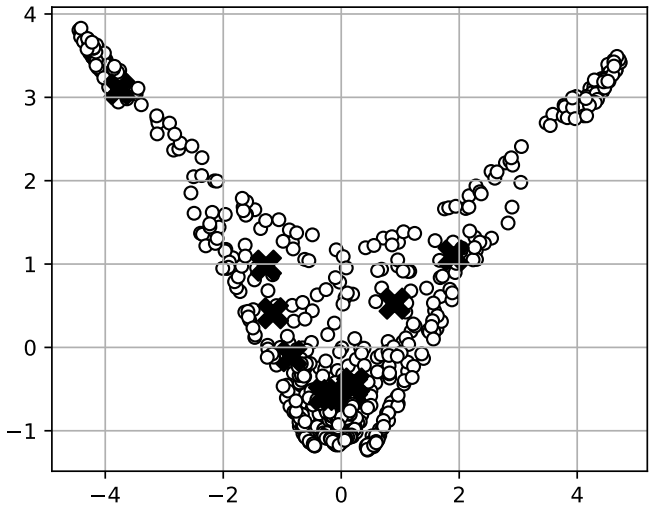


Figure 4.16: A result of PCA that is applied to the feature representation obtained by PRK for the SYN7 time series dataset. The 1st and 2nd principal components are shown in this figure.

applying the datasets to UFEKT, we extract feature vectors from the factor matrix and plot them in Figure 4.29. The upper and bottom plots show the first and the second component of the matrix, respectively. As we can see from the plots, there are two peaks at the sixth and ninth variables, that is, they indicate that the seventh and tenth variables are different from the others. They correspond to variables that include outliers in Figure 4.28. Although this might be one of examples showing that features about outliers are included in the factor matrix, it makes us feel a potential.

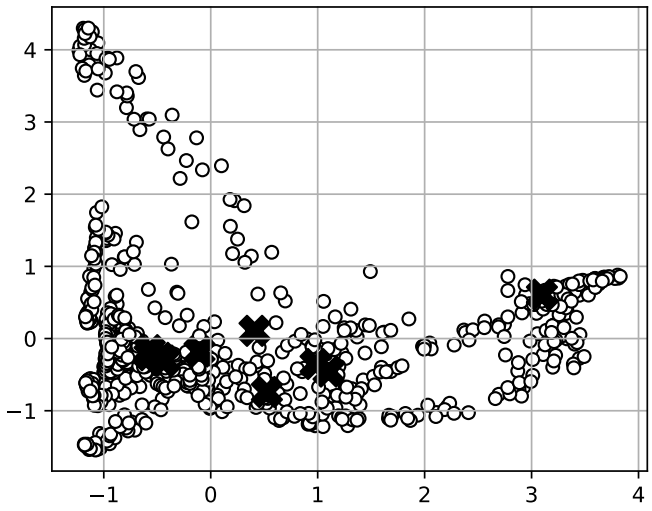


Figure 4.17: A result of PCA that is applied to the feature representation obtained by PRK for the SYN7 time series dataset. The 2nd and 3rd principal components are shown in this figure.

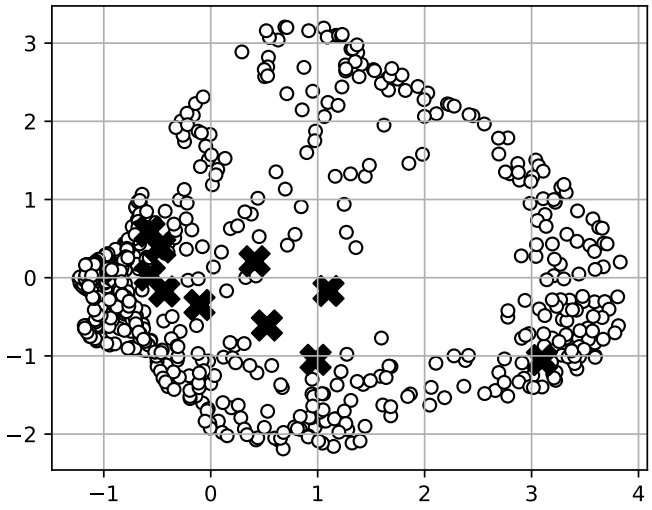


Figure 4.18: A result of PCA that is applied to the feature representation obtained by PRK for the SYN7 time series dataset. The 2nd and 4th principal components are shown in this figure.

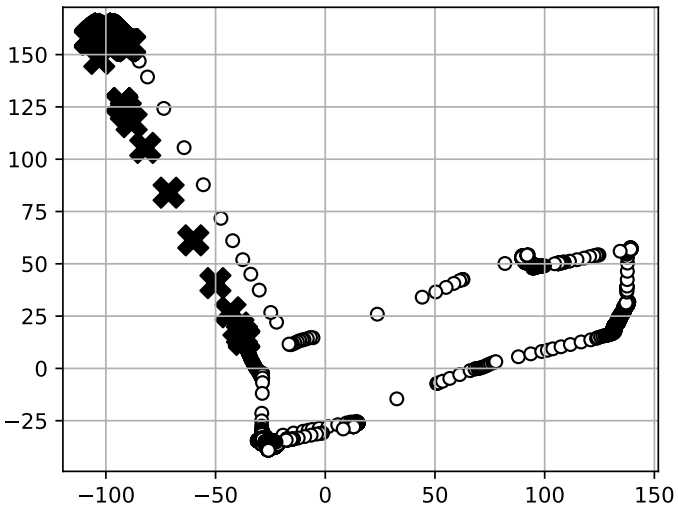


Figure 4.19: A result of PCA that is applied to the feature representation obtained by UFEKS for the SWaT time series dataset. The 1st and 2nd principal components are shown in this figure.

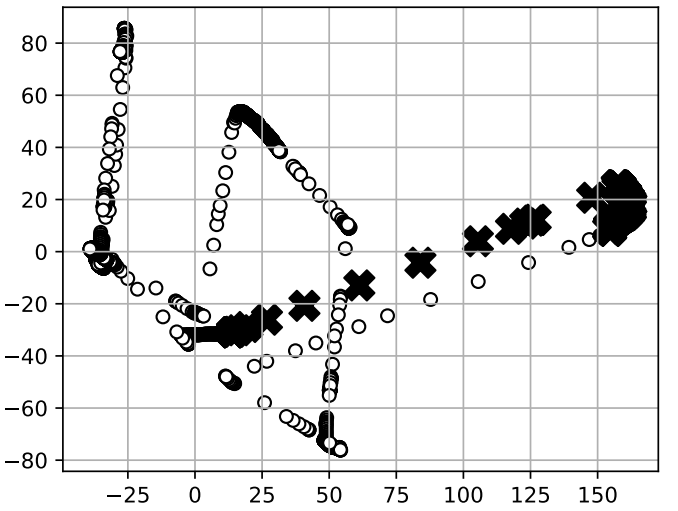


Figure 4.20: A result of PCA that is applied to the feature representation obtained by UFEKS for the SWaT time series dataset. The 2nd and 3rd principal components are shown in this figure.

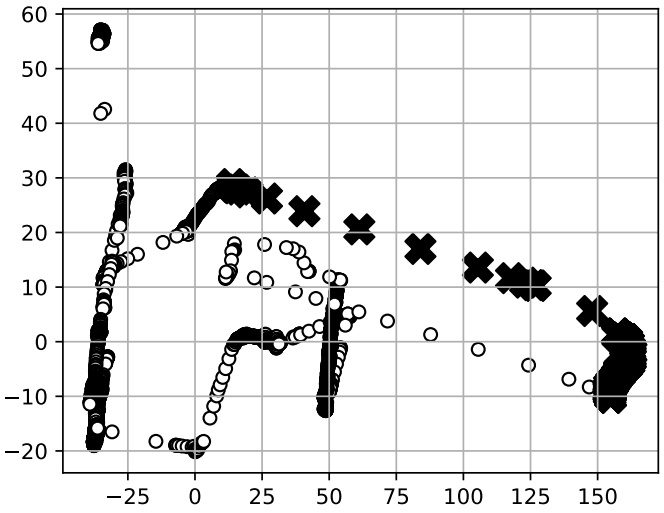


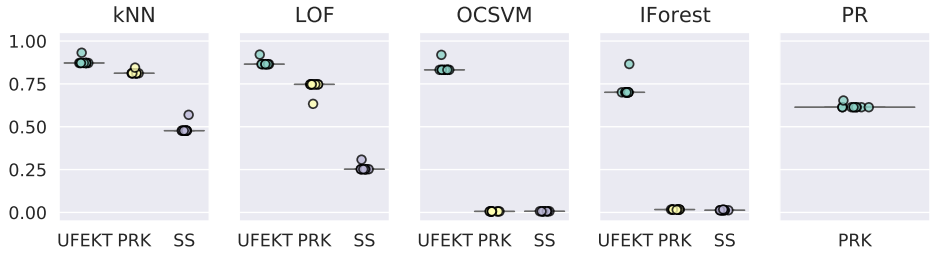
Figure 4.21: A result of PCA that is applied to the feature representation obtained by UFEKS for the SWaT time series dataset. The 2nd and 4th principal components are shown in this figure.

Table 4.7: Area under precision-recall curve (AUPRC) obtained by UFEKT for synthetic datasets. OD, FR, PR, PRK and SS stand for Outlier Detection, Feature Representation, PageRank, PageRank Kernel and SubSequence, respectively. Mean±StandardDeviation in ten trials are shown in the table, however, WADI and SWaT are performed only once. Best scores are denoted in bold.

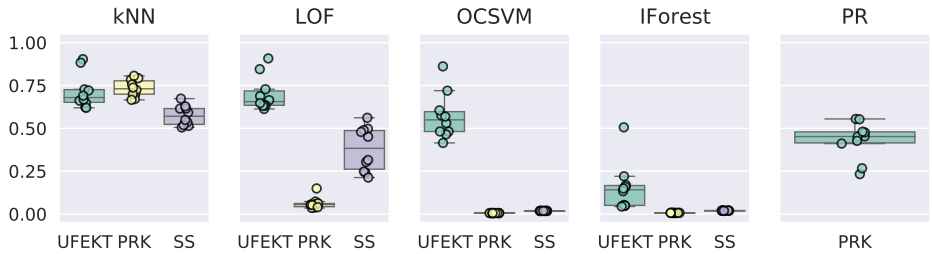
OD		κNN			LOF	
FR	UFEKT	PRK	SS	UFEKT	PRK	SS
SYN1	**0.878** ±0.018	0.815 ±0.010	0.486 ±0.028	0.871 ±0.017	0.736 ±0.034	0.258 ±0.017
SYN2	0.715 ±0.096	**0.734** ±0.048	0.575 ±0.055	0.699 ±0.095	0.062 ±0.031	0.380 ±0.122
SYN3	**0.988** ±0.029	0.625 ±0.167	0.016 ±0.005	0.961 ±0.066	0.118 ±0.041	0.011 ±0.002
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	
FR	UFEKT	PRK	SS	UFEKT	PRK	SS
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
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
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
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
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
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
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
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
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
Average	0.53	0.008	0.012	0.213	0.015	0.013
OD		PR			-	
FR	UFEKT	PRK	SS	-	-	-
SYN1	-	0.618 ±0.012	-	-	-	-
SYN2	-	0.431 ±0.101	-	-	-	-
SYN3	-	0.826 ±0.098	-	-	-	-
SYN4	-	0.798 ±0.061	-	-	-	-
SYN5	-	0.030 ±0.010	-	-	-	-
SYN6	-	**0.197** ±0.042	-	-	-	-
SYN7	-	0.751 ±0.024	-	-	-	-
SYN8	-	**0.411** ±0.082	-	-	-	-
SYN9	-	0.287 ±0.035	-	-	-	-
Average	-	0.483	-	-	-	-

Table 4.8: Area under precision-recall curve (AUPRC) obtained by UFEKT for real-world datasets. OD, FR, PR, PRK and SS stand for Outlier Detection, Feature Representation, PageRank, PageRank Kernel and SubSequence, respectively. Mean±StandardDeviation in ten trials are shown in the table, however, WADI and SWaT are performed only once. Best scores are denoted in bold.

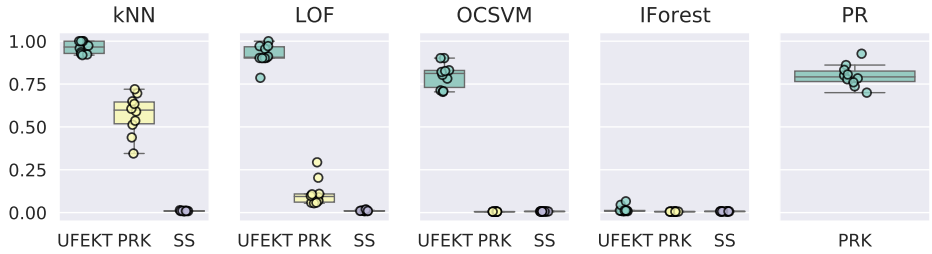
OD		κNN			LOF	
FR	UFEKT	PRK	SS	UFEKT	PRK	SS
ATSF8	0.877 ±0.038	0.199 ±0.028	0.040 ±0.001	**0.972** ±0.011	0.187 ±0.030	0.067 ±0.001
ATSF16	0.856 ±0.039	0.260 ±0.038	0.039 ±0.001	0.938 ±0.034	0.066 ±0.009	0.064 ±0.001
ATSF32	0.765 ±0.057	0.186 ±0.018	0.039 ±0.001	0.705 ±0.053	0.032 ±0.002	0.063 ±0.001
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	
FR	UFEKT	PRK	SS	UFEKT	PRK	SS
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
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
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
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
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
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
Average	**0.617**	0.044	0.295	0.537	0.035	0.188
OD		PR			-	
FR	-	PRK	-	-	-	-
ATSF8	-	0.094 ±0.006	-	-	-	-
ATSF16	-	0.072 ±0.002	-	-	-	-
ATSF32	-	0.057 ±0.002	-	-	-	-
ATSF64	-	0.044 ±0.002	-	-	-	-
WADI	-	0.061 ±0.000	-	-	-	-
SWaT	-	0.087 ±0.000	-	-	-	-
Average	-	0.069	-	-	-	-



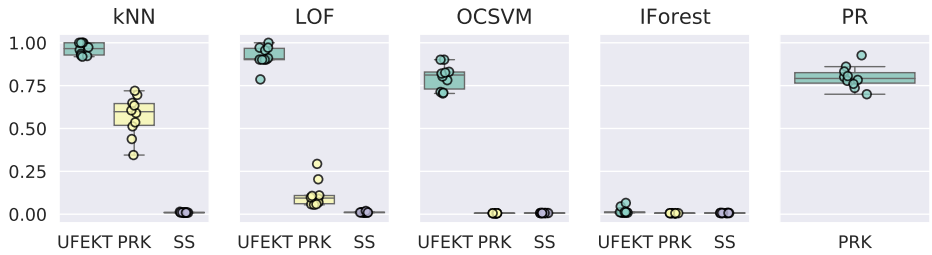
(a) AUPRCs for SYN1.



(b) AUPRCs for SYN2.

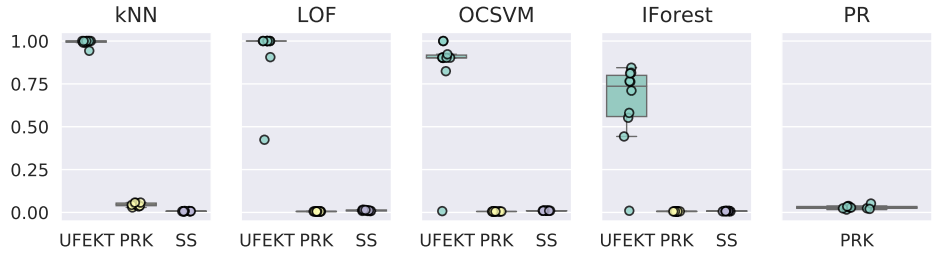


(c) AUPRCs for SYN3.

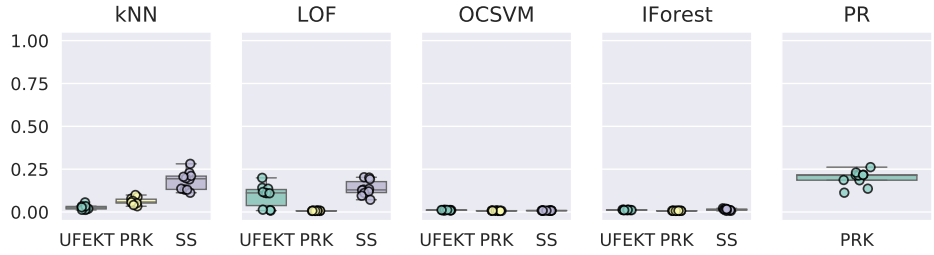


(d) AUPRCs for SYN4.

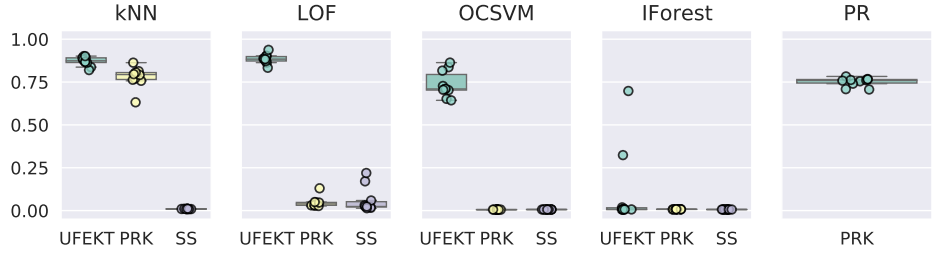
Figure 4.22: AUPRCs for synthetic datasets (SYN1 and SYN4). PR, PRK, and SS stands for PageRank, PageRank Kernel, and SubSequence, respectively. We generated each dataset ten times and each circle in the plot corresponds to each trial of the respective method.



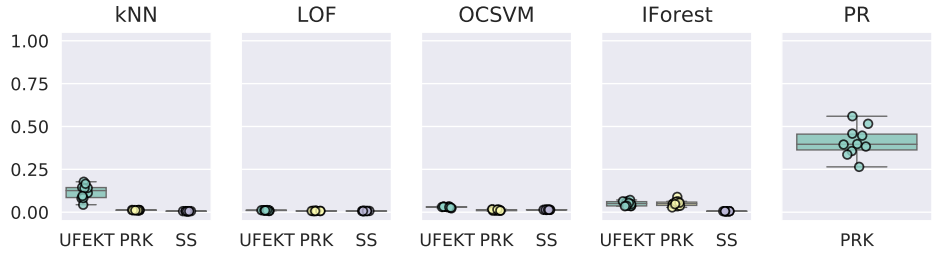
(a) AUPRCs for SYN5.



(b) AUPRCs for SYN6.

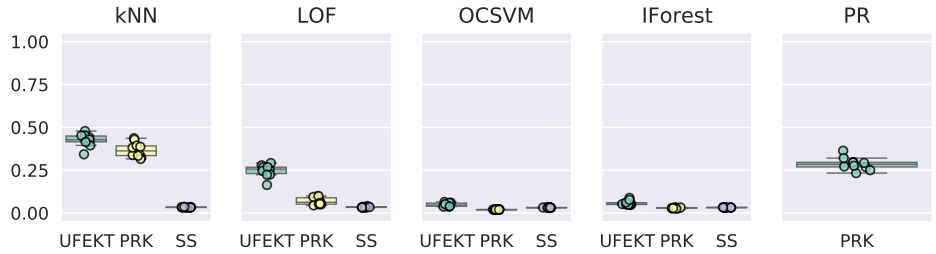


(c) AUPRCs for SYN7.



(d) AUPRCs for SYN8.

Figure 4.23: AUPRCs for synthetic datasets (SYN5 and SYN8). PR, PRK, and SS stands for PageRank, PageRank Kernel, and SubSequence, respectively. We generated each dataset ten times and each circle in the plot corresponds to each trial of the respective method.

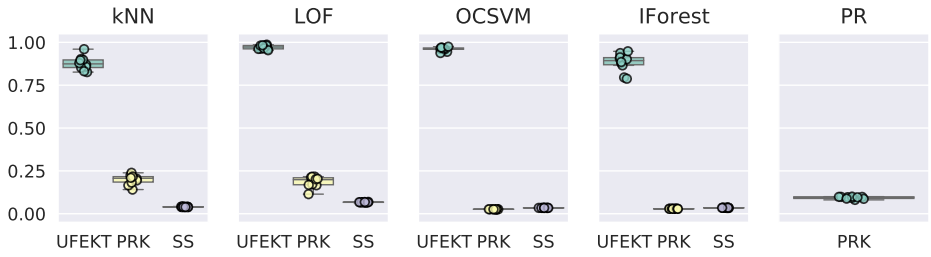


(a) AUPRCs for SYN9.

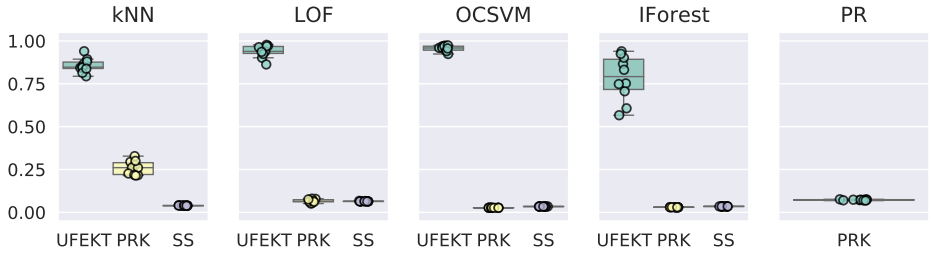
Figure 4.24: AUPRCs for synthetic datasets (SYN9). PR, PRK, and SS stands for PageRank, PageRank Kernel, and SubSequence, respectively. We generated each dataset ten times and each circle in the plot corresponds to each trial of the respective method.

Table 4.9: Comparison of AUPRCs between our algorithm in Algorithm 6 and grid search for all ranks. The latter can be possible only when the ground-truths are given.

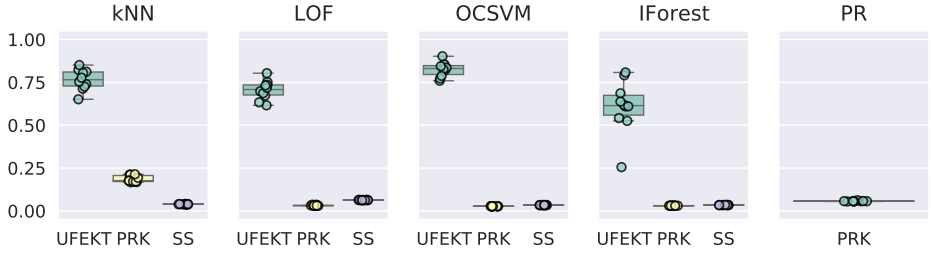
Datasets	Our algorithm	Optimal value
SYN1	0.878±0.018	0.914±0.009
SYN2	0.715±0.096	0.943±0.053
SYN3	0.988±0.029	0.999±0.003
SYN4	0.964±0.034	0.994±0.008
SYN5	0.992±0.017	0.999±0.003
SYN6	0.026±0.012	0.245±0.057
SYN7	0.872±0.026	0.905±0.029
SYN8	0.118±0.004	0.271±0.076
SYN9	0.425±0.035	0.485±0.039
ATSF8	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



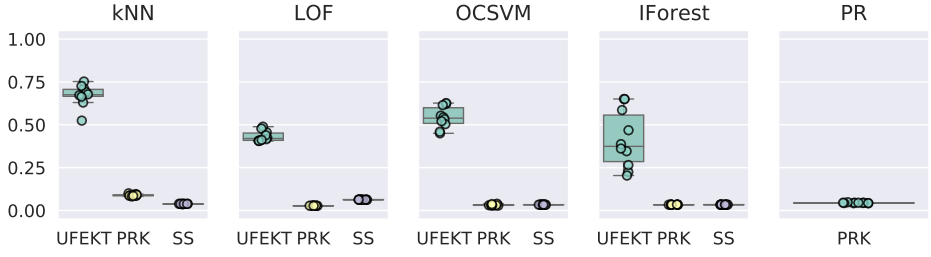
(a) AUPRCs for ATSF8.



(b) AUPRCs for ATSF16.

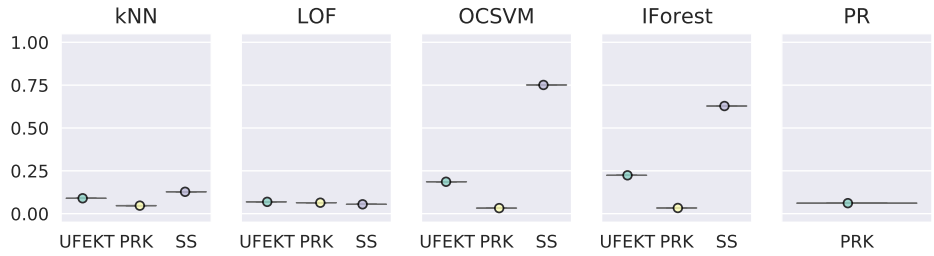


(c) AUPRCs for ATSF32.

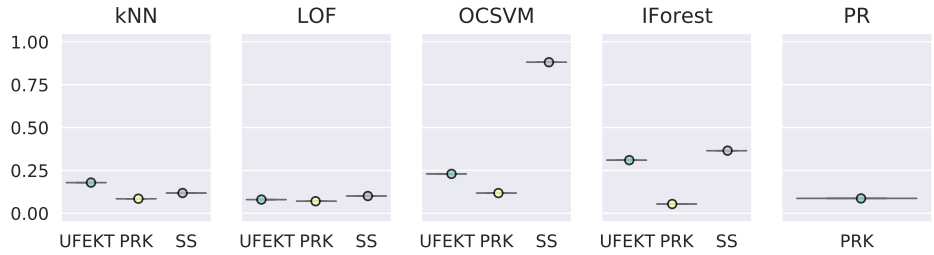


(d) AUPRCs for ATSF64.

Figure 4.25: AUPRCs for real-world datasets (ATSF8, ATSF16, ATSF32 and ATSF64). PR, PRK, and SS stands for PageRank, PageRank Kernel, and SubSequence, respectively. We generated each dataset ten times and each circle in the plot corresponds to each trial of the respective method.



(a) AUPRCs for WADI.



(b) AUPRCs for SWaT.

Figure 4.26: AUPRCs for real-world datasets (WADI and SWaT). PR, PRK, and SS stands for PageRank, PageRank Kernel, and SubSequence, respectively. We generated each dataset ten times and each circle in the plot corresponds to each trial of the respective method.

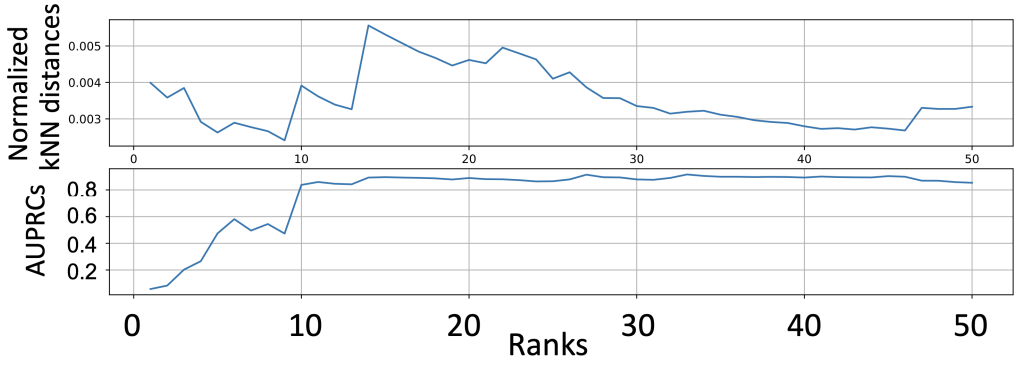
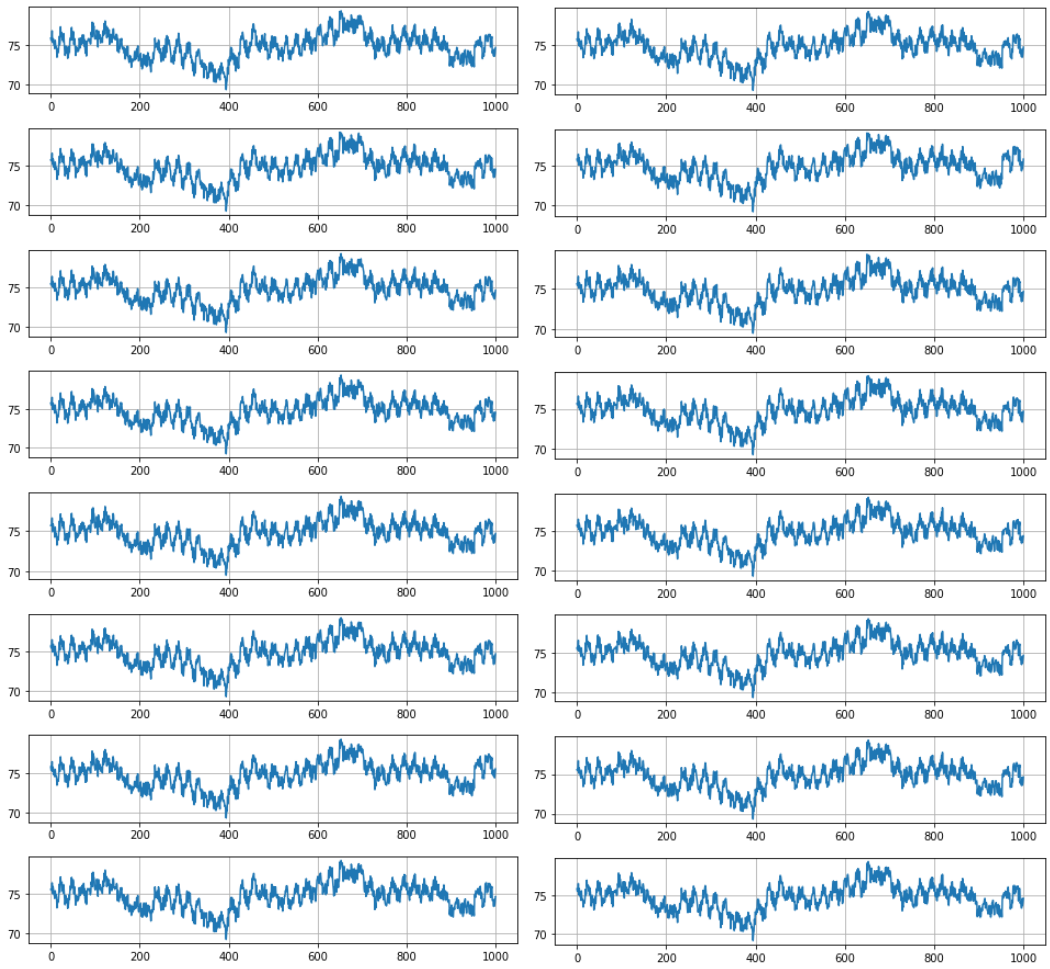
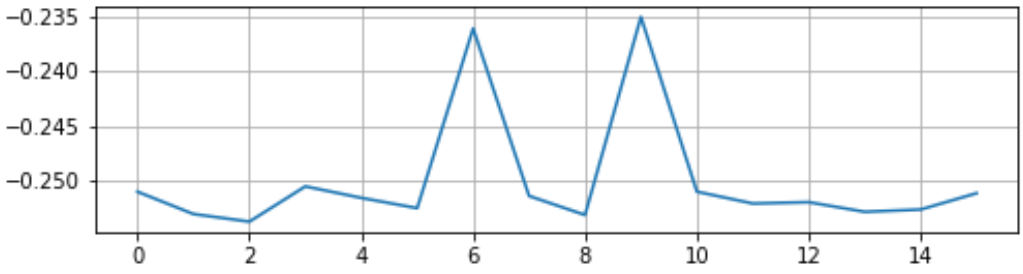


Figure 4.27: Rank dependencies of AUPRCs. x-axes indicate ranks of a core tensor and y-axes indicate normalized κNN distances for the top plot and AUPRCs for the bottom plot, respectively The top plot is obtained by Algorithm 6, while the bottom plot is obtained by the grid search, which requires the ground truth labels that are not available in practice.

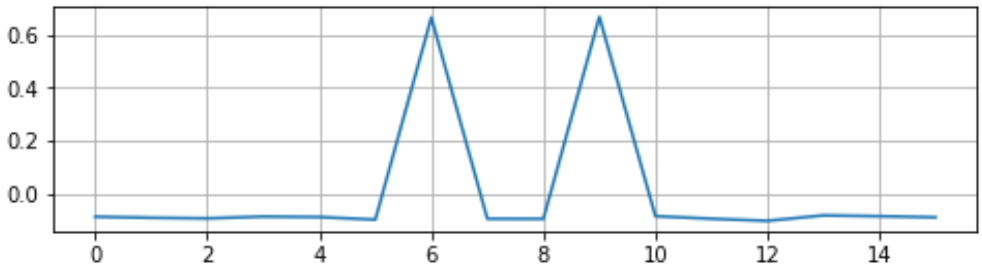


(a) ATSF16 (# 1). (b) ATSF16 (# 2).

Figure 4.28: Real-world Datasets (ATSF16). The datasets are prepared for evaluating whether or not the UFEKT can detect variables that include outliers. The datasets we employed are the same as datasets for outlier detection as we have mentioned except for outlierness. The outliers are located between 950 and 1,000 time stamps at the second plot from the bottom on the left side and the second plot from the top on the right side, that is, two out of sixteen variables have included outliers. Their outliers are made by adding about one percent values of the original values.



(a) The 1st component in the factor matrix.



(b) The 2nd component in the factor matrix.

Figure 4.29: Feature Vectors in The Factor Matrix with A Direction of Variables.

Chapter 5

Clustering

5.1 Clustering for Multivariate Time Series

A basic idea of clustering for multivariate time series using our algorithm UFEKT is similar to the one for the outlier detection problem as we mentioned in Chapter 4. Given a multivariate time series $\mathbf{X} \in \mathbb{R}^{P \times T}$, their feature vectors $\mathbf{f}_t \in \mathbb{R}^R$ extracted by UFEKT can be regarded as non-time series data points in a multidimensional space, where P, T, and R are the number of variables, the length of time series, and the rank used in Tucker decomposition, respectively. Therefore, those vectors can be applied to clustering algorithms for non-time series datasets such as *K-means* (KMeans) [32, 34], *Density-based Spatial Clustering of Applications with Noise* (DBSCAN) [17], *Agglomerative Hierarchical Clustering* (AHC) [14], and *Gaussian Mixture Model* (GMM) [13]. In the following, a clustering problem is briefly summarised as using one of the representative algorithms, K-means.

Assumed a set of K clusters $C = \{C_1, C_2, \ldots, C_k, \ldots, C_K\}$ where C_k is defined as a set of feature vectors $\mathbf{f}_i \in \mathbb{R}^R$ extracted by UFEKT, a clustering problem is defined as follows:

$$C^* = \arg\min_{C} \{SSE(C)\}, \tag{5.1}$$

$$SSE(C) = \sum_{k=1}^{K} \sum_{\mathbf{f}_i \in C_k} ||\mathbf{f}_i - \boldsymbol{\mu}_k||^2, \tag{5.2}$$

$$\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\mathbf{f}_i \in C_k} \mathbf{f}_i, \tag{5.3}$$

where C^* indicates a set of the best clusters, SSE stands for the *Sum of Square Errors*, $\boldsymbol{\mu}_k \in \mathbb{R}^R$ and $|C_k|$ indicate a centroid of all data points in the cluster C_k and the number of feature vectors in the cluster C_k, respectively [52]. Considering that all feature vectors need to be assigned to one of the clusters, the best cluster k^* that belong to feature vectors $\mathbf{f}$ is formularized as follows:

$$k^* = \arg\min_{k=1}^{K} \left\{||\mathbf{f}_i - \boldsymbol{\mu}_k||^2\right\}, \tag{5.4}$$

Algorithm 7 The K-means (KMeans) algorithm

Input: $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_i, \ldots, \mathbf{f}_{(T-w+1)} \in \mathbb{R}^{(T-w+1)}, K, \epsilon \in \mathbb{R}$

Output: $C = \{C_1, C_2, \ldots, C_k, \ldots, C_K\}$

$t \leftarrow 0$

Initialize all centroids with random values: $\boldsymbol{\mu}_1^{(t)}, \boldsymbol{\mu}_2^{(t)}, \ldots, \boldsymbol{\mu}_k^{(t)}, \ldots, \boldsymbol{\mu}_K^{(t)} \in \mathbb{R}^{(T-w+1)}$

repeat

  $t \leftarrow t + 1$

  $C_k \leftarrow \emptyset$ for all $k = 1, \ldots, K$

  for i = 1, . . . , T-w+1 **do**

    $k^* \leftarrow \arg\min_k \left\{ ||\mathbf{f}_i - \boldsymbol{\mu}_k^{(t-1)}||^2 \right\}$

    $C_{k^*} \leftarrow C_{k^*} \cup \{\mathbf{f}_i\}$

  end for

  for k = 1, 2, . . . , K **do**

    $\boldsymbol{\mu}_k^{(t)} \leftarrow \frac{1}{|C_k|} \sum_{\mathbf{f}_i \in C_k} \mathbf{f}_i$

  end for

until $\sum_{k=1}^{K} ||\boldsymbol{\mu}_k^{(t)} - \boldsymbol{\mu}_k^{(t-1)}||^2 \leq \epsilon$

return $C = \{C_1, C_2, \ldots, C_k, \ldots, C_K\}$

where K indicates the number of clusters that must be initialized in advance. In addition, a set of vectors of centroids $\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k, \ldots, \boldsymbol{\mu}_K\}$ must be initialized with random values. The computational cost of K-means is known as $O((T - w + 1)(T - w + 1)K)$ [52].

5.2 Existing Methods for Comparison with Our Algorithm

In the same manner as Chapter 4, we compare our algorithm, UFEKT, with two existing feature representation algorithms, the *PageRank kernel* (PRK) and *SubSequence* (SS), under four different clustering algorithms,*K-means* (KMeans), *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN), *Agglomerative Hierarchical clustering* (AHC), and *Gaussian Mixture Model* (GMM), which are commonly and widely used in data analysis.

5.2.1 PageRank Kernel (PRK) for Feature Extraction

As shown in Chapter 4, when a multivariate time series $\mathbf{X} \in \mathbb{R}^{(P \times T)}$ with P variables with the length T is given, PageRank (PR) constructs a kernel matrix $K \in \mathbb{R}^{(T-w+1) \times (T-w+1)}$ such that

$$k_{ij} = \exp\left\{ -\frac{\sum_{p=1}^{P} \sum_{s=0}^{w-1} (x_{i+s}^{(p)} - x_{j+s}^{(p)})^2}{\sigma^2} \right\}, \tag{5.5}$$

$$i, j \in \{1, 2, \ldots, T - w + 1\},$$

Table 5.1: Existing methods of clustering problem for non-time series datasets.

Types	Names
A centroid-based algorithm	K-means (KMeans)
A density-based algorithm	Density-based Spatial Clustering of Applications with Noise (DBSCAN)
A hierarchical-based algorithm	Agglomerative Hierarchical Clustering (AHC)
A probabilistic model-based algorithm	Gaussian Mixture Model (GMM)

where w is the length of each subsequence. In this case, each row vector $\mathbf{k}_i \in \mathbb{R}^{(T-w+1)}$ would be a feature vector that represents associations between subsequences. Therefore, we apply the vectors in a kernel matrix to existing clustering algorithms for comparison with our algorithm.

5.2.2 SubSequence (SS) for Feature Extraction

In comparison with PRK, the subsequence based approach, which we denote by SS, is also widely used in extracting features from a time series. A subsequence is a sequence extracted from time series. Given $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{P\times T}$ with P variables with the length T, first we obtain a single time series $\mathbf{x}_{ss}$ by summing up every variable at each time t,

$$\mathbf{x}_{ss} = \left(\sum_{p=1}^{P} x_1^{(p)}, \ldots, \sum_{p=1}^{P} x_T^{(p)} \right). \tag{5.6}$$

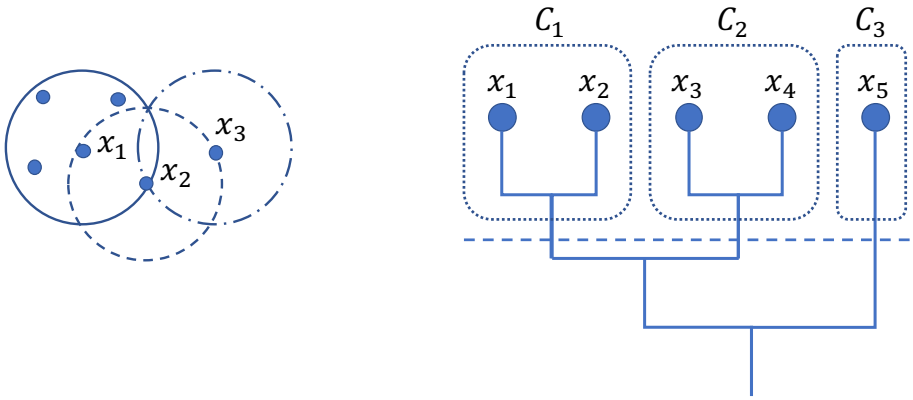
When we denote each element of $\mathbf{x}_{ss}$ at time t as $x_t^{ss} = \sum_{p=1}^{P} x_t^{(p)}$, the subsequence based matrix $\mathbf{X}_{ss} \in \mathbb{R}^{(T-w+1)\times w}$ with the window size w is defined as

$$\mathbf{X}_{ss} = \begin{bmatrix} x_1^{ss} & \cdots & x_w^{ss} \\ x_2^{ss} & \cdots & x_{w+1}^{ss} \\ \vdots & \ddots & \vdots \\ x_{T-w+1}^{ss} & \cdots & x_T^{ss} \end{bmatrix}. \tag{5.7}$$

By considering each row in Equation (5.7) to be a multidimensional data point, it can be applied to clustering algorithms like K-Means. Eventually, our algorithm is compared with two methods, PRK and SS.

5.2.3 Clustering Methods

As shown in Table 5.1, we employed four clustering algorithms such as KMeans, DBSCAN, AHC, and GMM. They are the representative algorithms for a centroid-base algorithm, a density-based algorithm, a hierarchical-based algorithm, and an probabilistic model-based algorithm, respectively [20].



(a) Density-based Spatial Clustering ofApplications with Noise (DBSCAN)

(b) Agglomerative Hierarchical Clustering (AHC).

Figure 5.1: Images of two representative clustering methods, DBSCAN and AHC.

KMeans

One of the most famous clustering algorithms is KMeans. We employed it as a representative of a centroid-based algorithm. The detailed algorithm has already been mentioned above and shown in Algorithm 7.

DBSCAN

The KMeans is suitable for datasets having convex clusters, while it is not suitable for non-convex clusters. Therefore we use DBSCAN as a density-based clustering that can be applicable in non-convex clusters. As shown in Figure 5.1a, DBSCAN define the points $\mathbf{x}_1, \mathbf{x}_2$, and $\mathbf{x}_3$ as a core point (a solid circle), a border point (a dashed circle), and a noise point (a dotted and dashed circle), respectively, if $MinPts$ (the minimum points) is set to five. In case of DBSCAN, there are two main hyper-parameters, ϵ and $MinPts$. Those parameters heavily affect the accuracy of the results of clustering. We will show you how to decide the values of the parameters later.

AHC

An image of AHC is shown in Figure 5.1b. The goal of AHC is to create a tree structure or hierarchy of clusters, which is called the cluster dendrogram. A tree structure is easily understandable for humans and used in various cases. Once a tree is constructed, clusters are generated by deciding a threshold. In the case of Figure 5.1b, three clusters, C_1, C_2, and C_3 are results where the dashed line in the figure indicates threshold for clustering.

GMM

As for GMM, it is assumed that each cluster is characterized by a multivariate normal distribution. Given dataset $\mathbf{x}_i \in \mathbb{R}^d$, a multivariate normal distribution is as follows:

$$f_i(\mathbf{x}) := f(\mathbf{x}|\mu_i, \mathbf{\Sigma}_i) = \frac{1}{(2\pi)^{\frac{d}{2}}(|\mathbf{\Sigma}_i|)^{\frac{1}{2}}} exp\left\{-\frac{(\mathbf{x}-\mu_i)^{\mathrm{T}}\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\mu_i)}{2}\right\} \tag{5.8}$$

where the cluster mean $\mu_i \in \mathbb{R}^d$ and the covariance matrix $\Sigma_i \in \mathbb{R}^{d\times d}$ are hyper parameters. Therefore GMM is given as

$$f(\mathbf{x}) := \sum_{i=1}^{K} f_i(\mathbf{x})P(C_i) = \sum_{i=1}^{K} f(\mathbf{x}|\mu_i, \mathbf{\Sigma}_i)P(C_i) \tag{5.9}$$

where the mixture parameters $P(C_i)$ must satisfy the condition $\sum_{i=1}^{k} P(C_i) = 1$. The model parameters, $\mu_i, \mathbf{\Sigma}_i$, and $P(C_i)$ are often decided by the expectation-maximization (EM) algorithm.

5.3 Evaluation Metrics

For evaluating accuracy of our algorithm, we introduce *Normalized Mutual Information* (NMI) as an evaluation metric. The NMI is widely used for evaluating accuracy of a clustering problem. The NMI is defined as follows:

$$NMI(U,V) := \frac{MI(U,V)}{\frac{1}{2}\left(H(U)+H(V)\right)}, \tag{5.10}$$

$$MI(U,V) := \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right) \tag{5.11}$$

$$= \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N}\log\left(\frac{N|U_i \cap V_j|}{|U_i||V_j|}\right), \tag{5.12}$$

$$H(U) := -\sum_{i=1}^{|U|} P(i)\log\left(P(i)\right), \tag{5.13}$$

$$H(V) := -\sum_{j=1}^{|V|} P'(j)\log\left(P'(j)\right), \tag{5.14}$$

$$P(i,j) := \frac{|U_i \cap V_j|}{N}, \tag{5.15}$$

$$P(i) := \frac{|U_i|}{N}, \tag{5.16}$$

$$P(j) := \frac{|V_j|}{N}, \tag{5.17}$$

where U and V are a set of estimated cluster numbers and a set of class labels, and N is the number of samples. MI, H, and P indicate the Mutual Information, an Entropy, and a

Probability, respectively. 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.

5.4 Experimental Environment

We used CentOS release 6.10 with 4x 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 performed on the same platform. A package list of Python we used in our experiments is shown in Table 4.2. They are the same platform for outlier detection tasks.

Table 5.2 is a hyper parameter list used in clustering algorithms. We set $\kappa = 5$ in κth-Nearest Neighbor which is commonly used in literature [6, 8].

The σ used in UFEKT and PageRank kernels is automatically determined by calculating the mean sum of squares of differences between two subsequences for every variable. Therefore, the σ take a different value for each variable. The details of how to decide them are shown in Algorithm 4.

The ϵ and $MinPts$ (the minimum points) are hyper-parameters for DBSCAN. They indicate the maximum distance between two samples for one to be considered as in the neighborhood of the other, and the number of samples in a neighborhood for a point to be considered as a core point, respectively. Therefore, the ϵ must be changed for every datasets because it strongly affects the accuracy of the results of clustering. Some algorithms to decide the appropriate values of the ϵ have been developed so far. We introduce one of the methods called the elbow rule, which is one of the renowned techniques to get the suitable values. As following the rule, we set the ϵ to the maximum point of second order differentiation of sorted distances from nearest neighbors. More details about the algorithm is shown in Algorithm 8. Moreover, the value of $MinPts$ is set to five, which is a default value of DBSCAN.

Furthermore, the number of clusters is used for some algorithms such as KMeans, AHC, and GMM, as a hyper parameter. For these algorithms, it is one of the most difficult problems to decide the best number of clusters in an unsupervised manner. However, since our aim is to evaluate accuracy of our algorithm that can extract features from multivariate time series, we gave the number of clusters with ground truth to each clustering algorithm as initialized values in advance.

Table 5.3 is a parameter list about the length of subsequence and the rank for Tucker decomposition used in UFEKT. The length of subsequence as known as a window size can be calculated from the percentages shown in the table. Since the percentages indicate a ratio of length of a subsequence, if a time series has 1,000 time stamps and [8, 16, 32] are given as parameters, the lengths of the 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,

Table 5.2: Parameters about experiments for a clustering. The κ and σ are used for κNN and PRK, respectively. The length of subsequences, or the window size, is used in not only SS but all algorithms to convert a given time-series to a set of subsequences.

Name	Value
κ for κ-th Nearest Neighbor (κNN)	5
σ for UFEKT and PageRank kernel (PRK)	variable
ϵ for DBSCAN	variable
$MinPts$ for DBSCAN	5
Number of clusters for KMeans, AHC, and GMM	variable

Algorithm 8 How to decide the value of ϵ for DBSCAN

Input: $\mathbf{f}_1, \ldots, \mathbf{f}_{T-w+1} \in \mathbb{R}^R$

Output: $\epsilon \in \mathbb{R}$

$D, \Delta \leftarrow \emptyset$
for $i = 1$ to $T - w + 1$ **do**
  $d^{(i)} \leftarrow$ Distance from the nearest neighbor of $\mathbf{f}_i$
  $D \leftarrow D \cup d^{(i)}$
end for
$D = \{ d_1, d_2, \cdots, d_{(T-w+1)} \} \leftarrow sort(\ D = \{d^{(1)}, d^{(2)}, \cdots, d^{(T-w+1)}\}\ )$
for $j = 2$ to $(T - w + 1) - 1$ **do**
  $\delta_j \leftarrow$ Second order differentiation(d_{j-1}, d_j, d_{j+1})
  $\Delta \leftarrow \Delta \cup \delta_j$
end for
$I \leftarrow \underset{i \in \{2, \cdots, T-w\}}{argmax} (\Delta)$
$\epsilon \leftarrow d^{(I)}$
return ϵ

and one of the lengths that have the best NMI score was used in our evaluations.

A rank parameter is used for Tucker Decomposition in UFEKT. If the rank is too small, features of time series might not be included in row vectors obtained by UFEKT. In contrast, computational costs become high if they are too large.

5.5 Datasets

We employed the time series classification datasets that are offered by the University of California, Riverside, called *UCR Time Series Classification datasets* (UCR)[1] for our

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

Table 5.3: Parameters of clustering for Real-world datasets (UCR). A length of subsequence as known as a window size can be calculated from the percentages shown in the table. Since the percentages indicate a ratio of 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 one of the percentages that have the best NMI scores for evaluations. A 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 parameter because one of the three parameters is decided by the number of parameters and the rest of two parameters take same values, that is, one parameter is required for UFEKT.

Name of Datasets	Percentages of Subsequence	A Rank for Tucker Decomposition
EOGHorizontalSignal	[8, 16, 32]	8
EOGVerticalSignal	[8, 16, 32]	8
FiftyWords	[8, 16, 32]	8
GestureMidAirD1	[8, 16, 32]	8
InlineSkate	[8, 16, 32]	8
MelbournePedestrian	[1/32, 1/16, 1/8]	8
SemgHandGenderCh2	[8, 16, 32]	8
SemgHandMovementCh2	[8, 16, 32]	8
SemgHandSubjectCh2	[8, 16, 32]	8
SyntheticControl	[8, 16, 32]	8
UWaveGestureLibraryX	[1/32, 1/16, 1/8]	8
UWaveGestureLibraryY	[1/32, 1/16, 1/8]	8
UWaveGestureLibraryZ	[1/32, 1/16, 1/8]	8
Other than the above	[8, 16, 32]	8

experiments. They 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 in that 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. A list of datasets that we employed are shown in Table 5.4, Table 5.5, and Table 5.6.

5.6 Experimental Results and Discussion for UFEKT

5.6.1 Experimental Results

We performed three feature representation algorithms, UFEKT, PageRank kernel (PRK), and Subsequence (SS), combined with four clustering algorithms, KMeans, DBSCAN, AHC, and GMM, resulting in twelve combinations of feature extraction and clustering in total. The effectiveness of each method was evaluated by scores of *the Normalized Mutual Information* (NMI). The NMI score takes values between zero and one, and higher is better. The results are summarised from Table 5.7 to Table 5.10 and from Figure 5.2 to Figure 5.5. The CL, FR, PRK, and SS in the figures stand for Clustering, Feature Representation, PageRank Kernel, and SubSequence, respectively. The more detailed results for all UCR datasets are shown from Figure A.1 to Figure A.26 in Appendix.

The Figure 5.2 shows the results of NMIs obtained by KMeans. The data points in the scatter plots indicate each UCR datasets, that is, there are 102 data points in the figure. The x-axis and y-axis indicate NMI scores obtained by UFEKT and PRK in the Plot 5.2a, respectively, and UFEKT and SS in the Plot 5.2b, respectively. A blue straight line in every plot means a border of NMI scores. If a data point is located under the line, UFEKT is superior to the other method because the inclination of every blue line is one.

As for the results of NMI scores obtained by KMeans shown in Figure 5.2, UFEKT consider to be almost same as PRK and apparently superior to SS because many points seem to be located around the line on PRK and under the line on SS. In fact, 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 5.10. It means that UFEKT might be slightly inferior to PRK, however, it is also considered that UFEKT has almost the same capability of PRK. Moreover, as for the results obtained by DBSCAN, AHC, and GMM, the results obtained by UFEKT are apparently superior to the ones from PRK and SS for all datasets. These results lead to the conclusion that UFEKT has high potential to extract features from multivariate time series.

Table 5.4: Summary of UCR datasets (# 1).

Name of Datasets	Abbreviation of Name	Length of Time Series	Number of Variables	Number of Class Labels
ACSF1	ACSF	200	1461	10
Adiac	Adia	781	177	37
AllGestureWiimoteX	AllX	1000	501	10
AllGestureWiimoteY	AllY	1000	501	10
AllGestureWiimoteZ	AllZ	1000	501	10
ArrowHead	Arro	211	252	3
Beef	BME	60	471	5
BeetleFly	Beef	40	513	2
BirdChicken	Beet	40	513	2
BME	Bird	180	129	3
Car	CBF	120	578	4
CBF	Car	930	129	3
Chinatown	Chin	363	25	2
Coffee	Coff	56	287	2
Computers	Comp	500	721	2
CricketX	CriX	780	301	12
CricketY	CriY	780	301	12
CricketZ	CriZ	780	301	12
DiatomSizeReduction	DiSR	322	346	4
DistalPhalanxOutlineAgeGroup	DiAG	539	81	3
DistalPhalanxOutlineCorrect	DiOC	876	81	2
DistalPhalanxTW	Dist	539	81	6
DodgerLoopDay	DoLD	158	289	7
DodgerLoopGame	DoLG	158	289	2
DodgerLoopWeekend	DoLW	158	289	2
Earthquakes	ECG2	461	513	2
ECG200	ECGF	200	97	2
ECGFiveDays	EOGH	884	137	2
EOGHorizontalSignal	EOGV	724	1251	12
EOGVerticalSignal	Eart	724	1251	12
FaceFour	Face	112	351	4
FiftyWords	Fift	905	271	50
Fish	Fish	350	464	7
Fungi	Fung	204	202	18

Table 5.5: Summary of UCR datasets (# 2).

Name of Datasets	Abbreviation of Name	Length of Time Series	Number of Variables	Number of Class Labels
GestureMidAirD1	GeD1	338	361	26
GestureMidAirD2	GeD2	338	361	26
GestureMidAirD3	GeD3	338	361	26
GesturePebbleZ1	GeZ1	304	456	6
GesturePebbleZ2	GeZ2	304	456	6
GunPoint	GunP	200	151	2
GunPointAgeSpan	GuPA	451	151	2
GunPointMaleVersusFemale	GuPM	451	151	2
GunPointOldVersusYoung	GuPO	451	151	2
Ham	Ham	214	432	2
Haptics	Hapt	463	1093	5
Herring	Herr	128	513	2
HouseTwenty	Hous	159	2001	2
InlineSkate	Inli	650	1883	7
InsectEPGRegularTrain	InRT	311	602	3
InsectEPGSmallTrain	InST	266	602	3
InsectWingbeatSound	InBS	2200	257	11
ItalyPowerDemand	Ital	1096	25	2
LargeKitchenAppliances	Larg	750	721	3
Lightning2	Lig2	121	638	2
Lightning7	Lig7	143	320	7
Meat	Meat	120	449	3
MedicalImages	Medi	1141	100	10
MelbournePedestrian	Melb	3633	25	10
MiddlePhalanxOutlineAgeGroup	MiAG	554	81	3
MiddlePhalanxOutlineCorrect	MiOC	891	81	2
MiddlePhalanxTW	MiTW	553	81	6
MoteStrain	Mote	1272	85	2
OliveOil	OSUL	60	571	4
OSULeaf	Oliv	442	428	6
PhalangesOutlinesCorrect	Phal	2658	81	2
PickupGestureWiimoteZ	Pick	100	362	10
PigAirwayPressure	PigA	312	2001	52

Table 5.6: Summary of UCR datasets (# 3).

Name of Datasets	Abbreviation of Name	Length of Time Series	Number of Variables	Number of Class Labels
PigArtPressure	PigA	312	2001	52
PigCVP	PigC	312	2001	52
Plane	Plan	210	145	7
PowerCons	Powe	360	145	2
ProximalPhalanxOutlineAgeGroup	PrAG	605	81	3
ProximalPhalanxOutlineCorrect	PrOC	891	81	2
ProximalPhalanxTW	PrTW	605	81	6
RefrigerationDevices	Refr	750	721	3
Rock	Rock	70	2845	4
ScreenType	Scre	750	721	3
SemgHandGenderCh2	SeGE	900	1501	2
SemgHandMovementCh2	SeMO	900	1501	6
SemgHandSubjectCh2	SeSU	900	1501	5
ShakeGestureWiimoteZ	Shak	100	386	10
ShapeletSim	Shap	200	501	2
SmallKitchenAppliances	Smal	750	721	3
SmoothSubspace	Smoo	300	16	3
SonyAIBORobotSurface1	Son1	621	71	2
SonyAIBORobotSurface2	Son2	980	66	2
Strawberry	Stra	983	236	2
SwedishLeaf	Swed	1125	129	15
Symbols	Symb	1020	399	6
SyntheticControl	Synt	600	61	6
ToeSegmentation1	Toe1	268	278	2
ToeSegmentation2	Toe2	166	344	2
Trace	Trac	200	276	4
TwoLeadECG	TwoL	1162	83	2
UMD	UMD	180	151	3
UWaveGestureLibraryX	UWaX	4478	316	8
UWaveGestureLibraryY	UWaY	4478	316	8
UWaveGestureLibraryZ	UWaZ	4478	316	8
Wine	Wine	111	235	2
WordSynonyms	Word	905	271	25
Worms	Worm	258	901	5
WormsTwoClass	WoTC	258	901	2

Table 5.7: The NMI scores for UCR datasets (# 1). The CL, FR, PRK and SS stand for Clustering, Feature Representation, PageRank Kernel and SubSequence respectively. Best scores are denoted in bold.

CL	KMeans			DBSCAN			AHC			GMM		
FR	UFEKT	PRK	SS	UFEKT	PRK	SS	UFEKT	PRK	SS	UFEKT	PRK	SS
ACSF	0.701	**0.791**	0.179	0.599	0.433	0.129	0.695	0.755	0.189	0.704	0.763	0.180
Adia	0.330	0.303	0.306	0.202	0.028	0.000	**0.331**	0.305	0.315	0.313	0.316	0.301
AllX	0.580	0.544	0.574	0.533	0.359	0.333	**0.599**	0.584	0.550	0.586	0.562	0.526
AllY	0.588	0.564	0.519	0.510	0.373	0.273	**0.603**	0.588	0.552	0.544	0.587	0.463
AllZ	0.603	0.572	0.453	**0.657**	0.291	0.034	0.606	0.541	0.516	0.608	0.623	0.426
Arro	0.208	0.303	0.018	0.368	**0.400**	0.044	0.391	0.347	0.052	0.293	0.301	0.027
BME	0.398	0.382	0.435	**0.497**	0.316	0.316	0.378	0.447	0.293	0.317	0.373	0.368
Beef	0.273	**0.819**	0.343	0.447	0.354	0.098	0.428	0.751	0.410	0.313	0.693	0.218
Beet	**0.432**	0.305	0.023	0.393	0.122	0.006	0.337	0.272	0.030	0.021	0.170	0.040
Bird	**0.567**	0.089	0.015	0.000	0.015	0.006	0.120	0.109	0.006	0.127	0.142	0.047
CBF	0.005	0.004	0.002	**0.025**	0.003	0.000	0.001	0.004	0.002	0.002	0.003	0.004
Car	0.042	0.067	0.057	0.000	0.040	0.017	0.051	**0.090**	0.062	0.070	0.066	0.064
Chin	0.589	0.629	0.557	0.339	0.539	0.520	0.520	0.535	0.673	**0.752**	0.629	0.557
Coff	0.621	**0.847**	0.045	0.332	0.677	0.038	0.506	0.763	0.009	**0.847**	0.621	0.007
Comp	0.235	0.229	0.008	**0.326**	0.000	0.127	0.262	0.197	0.054	0.079	0.229	0.085
CriX	0.048	0.053	0.059	**0.122**	0.000	0.009	0.045	0.061	0.041	0.045	0.046	0.048
CriY	0.040	0.053	0.052	**0.138**	0.000	0.013	0.044	0.049	0.058	0.036	0.061	0.056
CriZ	0.052	0.056	0.059	**0.160**	0.000	0.000	0.055	0.064	0.050	0.048	0.049	0.061
DiSR	0.014	0.018	0.017	**0.041**	0.018	0.000	0.011	0.018	0.016	0.016	0.014	0.024
DiAG	0.272	0.206	0.036	0.221	0.288	0.168	**0.305**	0.212	0.080	0.284	0.206	0.005
DiOC	0.031	0.013	0.007	0.027	0.016	0.024	0.040	0.011	0.010	**0.041**	0.013	0.023
Dist	0.053	0.064	0.033	**0.100**	0.000	0.013	0.054	0.063	0.034	0.056	0.055	0.023
DoLD	0.623	0.705	0.783	0.504	0.145	0.313	0.607	**0.795**	0.679	0.617	0.730	0.721
DoLG	0.111	**0.414**	0.018	0.335	0.313	0.000	0.054	0.372	0.003	0.188	0.403	0.008
DoLW	0.069	0.414	0.398	0.250	0.189	0.436	0.138	0.490	**0.629**	0.188	0.414	0.413
ECG2	0.029	0.003	0.013	0.015	0.005	0.007	0.014	0.004	0.014	**0.032**	0.003	0.005
ECGF	0.000	0.000	0.005	**0.009**	0.002	0.000	0.002	0.002	0.004	0.000	0.001	0.002
EOGH	0.471	0.441	0.528	0.340	0.186	0.046	0.438	0.469	0.530	**0.536**	0.439	0.511
EOGV	0.345	0.320	0.334	0.272	0.259	0.036	0.343	0.357	0.332	0.371	0.337	**0.491**
Eart	0.001	0.002	0.013	**0.031**	0.023	0.000	0.003	0.006	0.000	0.003	0.008	0.001
Face	0.055	0.120	0.038	0.086	0.078	0.028	0.086	0.078	0.070	0.062	**0.135**	0.060
Fift	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
Fung	0.736	**0.774**	0.359	0.688	0.462	0.000	0.756	0.733	0.374	0.771	0.719	0.308
GeD1	0.350	0.376	0.344	0.294	0.035	0.023	0.348	0.377	**0.388**	0.332	0.382	0.366
GeD2	0.389	0.390	0.351	0.246	0.000	0.000	0.377	0.379	0.380	0.380	**0.396**	0.360
GeD3	0.366	0.367	0.349	0.287	0.070	0.000	0.366	**0.381**	0.370	0.360	0.377	0.338
GeZ1	0.066	0.280	0.289	**0.314**	0.148	0.049	0.112	0.258	0.283	0.107	0.269	0.240
GeZ2	0.099	0.265	0.291	0.204	0.046	0.090	0.100	0.234	**0.366**	0.181	0.252	0.310

Table 5.8: The NMI scores for UCR datasets (# 2). The CL, FR, PRK and SS stand for Clustering, Feature Representation, PageRank Kernel and SubSequence respectively. Best scores are denoted in bold.

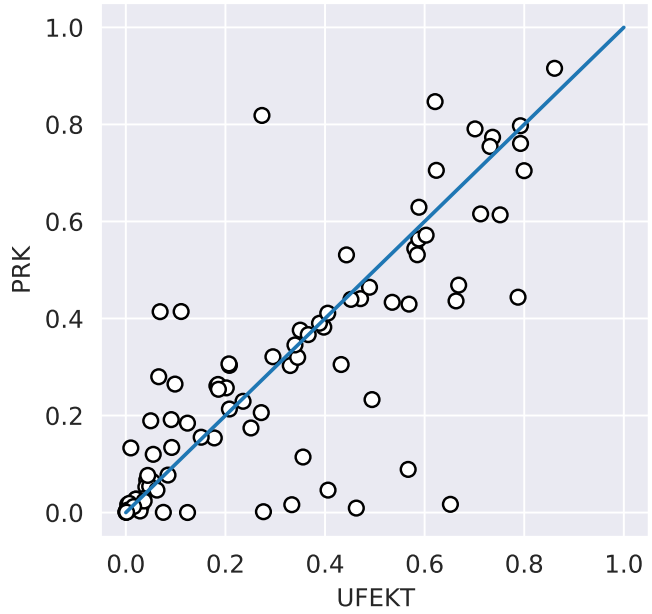
CL	KMeans			DBSCAN			AHC			GMM		
FR	UFEKT	PRK	SS	UFEKT	PRK	SS	UFEKT	PRK	SS	UFEKT	PRK	SS
GunP	0.021	0.028	0.004	0.011	0.050	0.001	0.028	**0.068**	0.009	0.024	0.030	0.011
GuPA	**0.494**	0.233	0.007	0.000	0.155	0.063	0.461	0.270	0.003	0.477	0.239	0.033
GuPM	0.535	0.433	0.338	0.028	0.583	0.477	**0.722**	0.443	0.411	0.590	0.433	0.374
GuPO	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
Hapt	0.017	0.018	0.020	**0.057**	0.008	0.000	0.014	0.025	0.015	0.020	0.022	0.017
Herr	0.004	0.003	0.012	0.017	0.003	0.000	0.007	0.004	**0.020**	0.002	0.005	0.012
Hous	0.178	0.154	**0.405**	0.268	0.211	0.098	0.345	0.173	0.324	0.399	0.122	0.379
Inli	0.023	0.025	0.025	**0.105**	0.009	0.007	0.023	0.019	0.017	0.021	0.026	0.028
InRT	0.489	0.464	0.473	0.522	0.564	**0.594**	0.501	0.480	0.432	0.499	0.464	0.454
InST	0.443	0.531	0.518	**0.671**	0.670	0.641	0.618	0.560	0.528	0.533	0.531	0.518
InBS	0.015	0.020	0.013	**0.060**	0.002	0.003	0.017	0.015	0.016	0.015	0.015	0.012
Ital	0.003	0.005	0.000	**0.012**	0.002	0.006	0.006	0.004	0.009	0.005	0.002	0.001
Larg	0.208	0.213	0.006	0.472	0.018	0.306	0.315	**0.519**	0.084	0.359	0.212	0.005
Lig2	0.028	0.024	0.027	**0.061**	0.022	0.026	0.039	0.011	0.030	0.031	0.058	0.025
Lig7	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
Medi	0.029	0.026	0.028	**0.077**	0.005	0.004	0.032	0.032	0.030	0.031	0.032	0.026
Melb	0.800	0.705	0.597	0.503	0.408	0.005	0.814	0.713	0.617	**0.846**	0.757	0.631
MiAG	0.207	0.306	0.143	0.234	0.254	0.282	0.203	0.304	0.165	**0.337**	0.306	0.071
MiOC	0.004	0.016	0.001	**0.041**	0.012	0.018	0.016	0.019	0.007	0.010	0.016	0.002
MiTW	0.062	0.046	0.018	**0.116**	0.012	0.057	0.056	0.042	0.030	0.062	0.038	0.023
Mote	0.002	0.001	0.002	**0.017**	0.002	0.001	0.001	0.001	0.003	0.000	0.001	0.000
OSUL	**0.036**	0.029	0.026	0.000	0.000	0.000	0.026	0.033	0.024	0.029	0.024	0.022
Oliv	0.712	0.616	0.262	0.506	0.546	0.212	0.677	**0.791**	0.223	0.695	0.584	0.152
Phal	0.007	0.007	0.004	**0.021**	0.006	0.005	0.007	0.007	0.001	0.008	0.007	0.001
Pick	0.861	0.916	0.632	0.433	0.430	0.293	0.787	0.844	0.751	0.729	**0.917**	0.682
PigA	0.752	0.614	0.777	0.563	0.034	0.189	0.752	0.618	**0.794**	0.750	0.616	0.780
PigA	0.793	0.761	0.751	0.268	0.263	0.000	0.775	**0.796**	0.755	0.762	0.782	0.760
PigC	0.792	0.797	**0.803**	0.573	0.324	0.079	0.784	0.802	0.798	0.797	0.798	0.802
Plan	0.092	**0.134**	0.069	0.005	0.034	0.014	0.097	0.107	0.090	0.088	0.114	0.083
Powe	**0.652**	0.017	0.016	0.276	0.019	0.000	0.035	0.172	0.639	0.019	0.042	0.047
PrAG	0.183	0.261	0.105	0.265	0.285	0.275	**0.293**	0.276	0.066	0.277	0.261	0.089
PrOC	0.020	0.028	0.007	**0.035**	0.008	0.017	0.026	0.021	0.017	0.032	0.028	0.005
PrTW	0.037	0.024	0.028	**0.056**	0.002	0.000	0.031	0.030	0.014	0.034	0.022	0.024

Table 5.9: The NMI scores for UCR datasets (# 3). The CL, FR, PRK and SS stand for Clustering, Feature Representation, PageRank Kernel and SubSequence respectively. Best scores are denoted in bold.

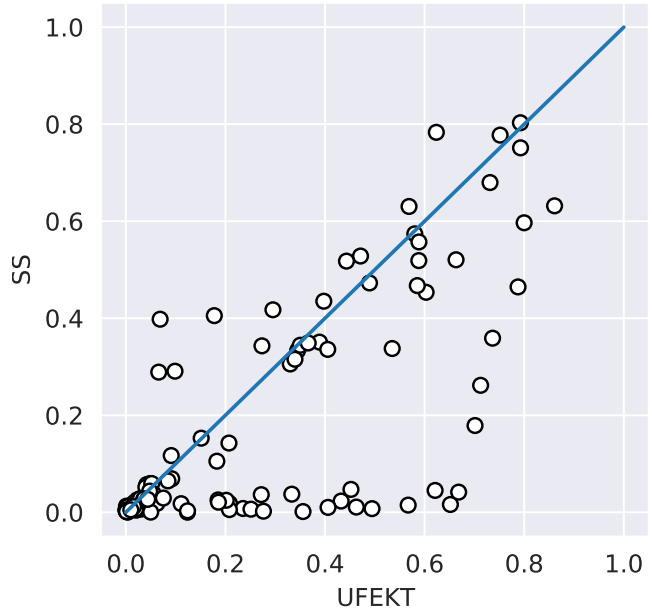
CL	KMeans			DBSCAN			AHC			GMM		
FR	UFEKT	PRK	SS	UFEKT	PRK	SS	UFEKT	PRK	SS	UFEKT	PRK	SS
Refr	0.463	0.009	0.011	**0.561**	0.000	0.177	0.330	0.023	0.047	0.432	0.006	0.012
Rock	0.405	0.411	0.336	0.280	0.287	0.244	0.442	0.373	0.352	0.299	**0.444**	0.292
Scre	0.251	0.174	0.006	**0.458**	0.229	0.002	0.412	0.202	0.008	0.309	0.173	0.005
SeGE	0.124	0.184	0.000	0.274	0.089	0.046	0.200	0.130	0.001	**0.332**	0.183	0.001
SeMO	0.585	0.531	0.467	0.589	0.356	0.170	0.543	0.535	0.601	**0.634**	0.565	0.512
SeSU	0.569	0.430	0.630	0.615	0.213	0.234	0.550	0.440	0.622	**0.726**	0.427	0.648
Shak	0.731	0.755	0.680	0.421	0.441	0.403	**0.817**	0.783	0.745	0.810	0.798	0.651
Shap	0.007	0.019	0.004	0.000	0.012	0.063	**0.613**	0.007	0.004	0.258	0.003	0.001
Smal	0.406	0.046	0.010	**0.523**	0.058	0.176	0.416	0.354	0.187	0.347	0.027	0.043
Smoo	**0.663**	0.436	0.520	0.427	0.213	0.000	0.425	0.489	0.634	0.397	0.459	0.492
Son1	0.003	0.005	0.005	**0.020**	0.006	0.000	0.000	0.002	0.003	0.005	0.004	0.004
Son2	0.000	0.002	0.004	**0.008**	0.003	0.001	0.000	0.001	0.001	0.001	0.001	0.001
Stra	0.050	0.189	0.000	0.180	**0.189**	0.003	0.102	0.188	0.003	0.161	0.189	0.002
Swed	0.085	0.077	0.065	**0.153**	0.020	0.022	0.091	0.087	0.062	0.089	0.074	0.067
Symb	0.016	0.012	0.012	**0.049**	0.000	0.000	0.013	0.016	0.015	0.016	0.011	0.016
Synt	0.668	0.469	0.042	0.517	0.324	0.000	0.687	0.457	0.039	**0.792**	0.469	0.027
Toe1	0.124	0.000	0.003	0.168	0.000	0.143	**0.398**	0.000	0.166	0.141	0.000	0.022
Toe2	0.075	0.000	0.029	0.079	0.000	0.084	**0.370**	0.017	0.007	0.266	0.010	0.017
Trac	0.044	**0.077**	0.027	0.012	0.000	0.043	0.071	0.060	0.032	0.037	0.069	0.022
TwoL	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
UWaX	0.185	**0.264**	0.025	0.001	0.001	0.001	0.172	0.221	0.025	0.240	0.249	0.031
UWaY	0.202	0.257	0.025	0.002	0.001	0.002	0.125	0.236	0.024	**0.351**	0.257	0.027
UWaZ	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
Word	0.151	0.155	0.153	0.160	0.000	0.016	0.150	0.160	0.155	0.153	**0.181**	0.165
Worm	0.333	0.016	0.037	**0.477**	0.000	0.339	0.319	0.019	0.214	0.314	0.021	0.067
WoTC	0.276	0.002	0.002	0.248	0.000	**0.536**	0.105	0.009	0.065	0.231	0.006	0.005
Average	0.273	0.255	0.182	0.243	0.145	0.100	**0.282**	0.267	0.203	0.276	0.253	0.180

Table 5.10: The count of the best NMI scores per each clustering algorithm. Total counts in the table are equal to the number of datasets. Best counts are denoted in bold.

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

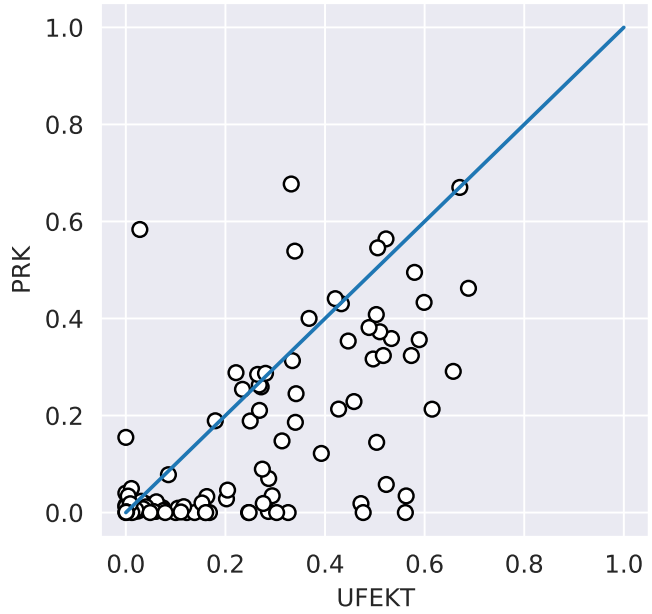


(a) The results were performed by UFEKT and PRK for feature extraction and KMeans for clustering.

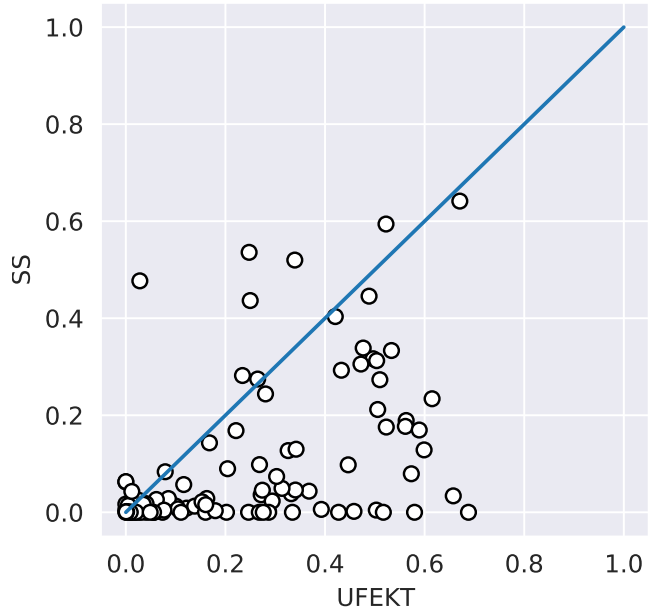


(b) The results were performed by UFEKT and SS for feature extraction and KMeans for clustering.

Figure 5.2: Scatter plots of NMIs for UCR datasets.

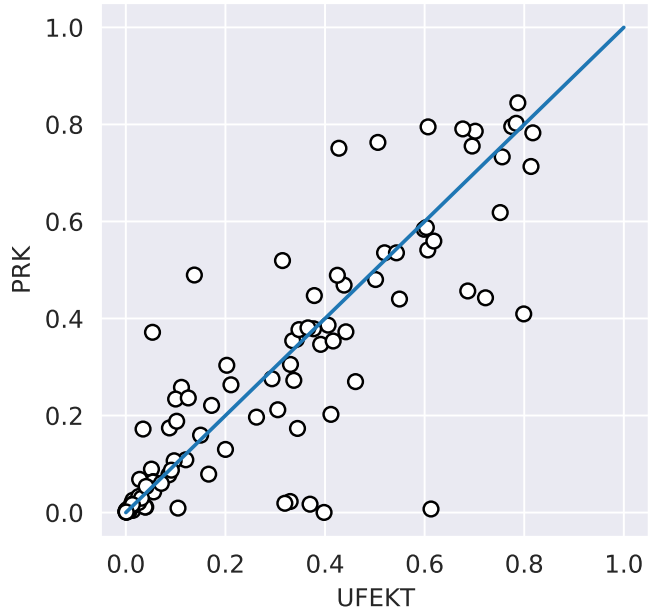


(a) The results were performed by UFEKT and PRK for feature extraction and DBSCAN for clustering.

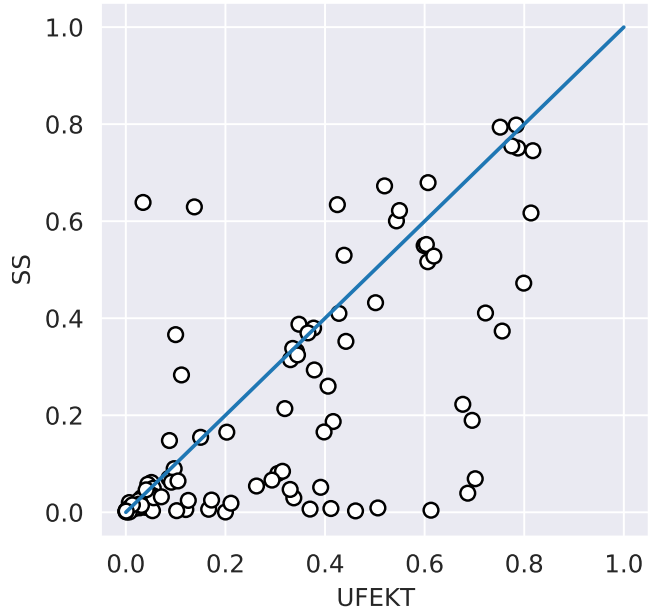


(b) The results were performed by UFEKT and SS for feature extraction and DBSCAN for clustering.

Figure 5.3: Scatter plots of NMIs for UCR datasets.

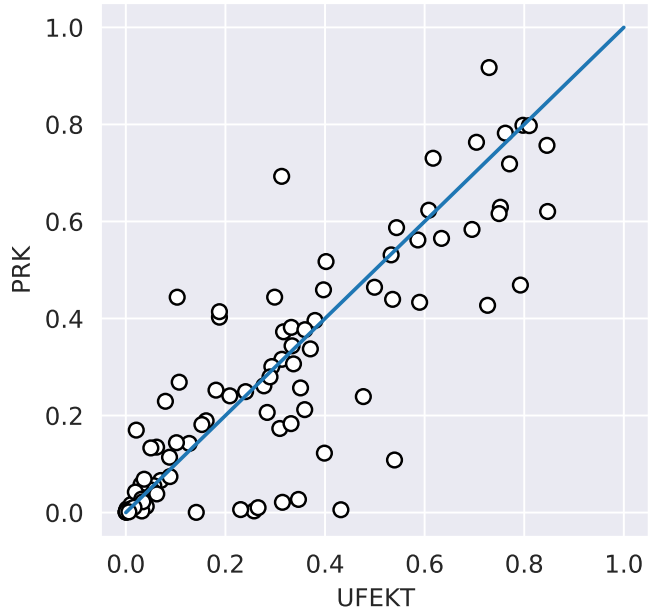


(a) The results were performed by UFEKT and PRK for feature extraction and AHC for clustering.

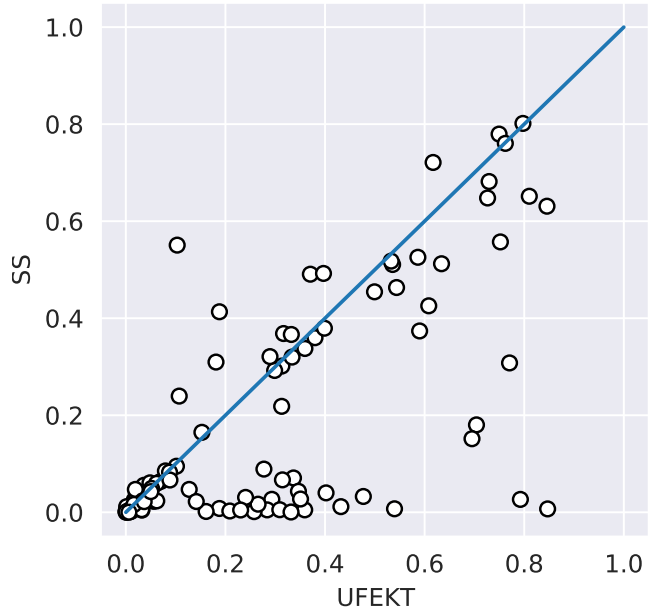


(b) The results were performed by UFEKT and SS for feature extraction and AHC for clustering.

Figure 5.4: Scatter plots of NMIs for UCR datasets.



(a) The results were performed by UFEKT and PRK for feature extraction and GMM for clustering.



(b) The results were performed by UFEKT and SS for feature extraction and GMM for clustering.

Figure 5.5: Scatter plots of NMIs for UCR datasets.

Chapter 6

Conclusion

In this thesis, we proposed two main algorithms UFEKS (*Unsupervised Feature Extraction using Kernel and Stacking*) and UFEKT (*Unsupervised Feature Extraction using Kernel Method and Tucker Decomposition*), which can extract features from multivariate time series. To evaluate both algorithms, we performed experiments such as outlier detection and clustering tasks for synthetic and real-world datasets, and had better results than existing algorithms. After summarizing my thesis, I will mention current limitations and future works.

6.1 Summary

First, we defined combinatorial outliers of multivariate time series as follows:

- Combinatorial outliers are defined as outliers that can be found when we look at multivariate time series simultaneously.

We have proposed two new algorithms that extract features from multivariate time series called *Unsupervised Feature Extraction using Kernel and Stacking* (UFEKS) and *Unsupervised Feature Extraction using Kernel Method and Tucker Decomposition* (UFEKT). The procedures are as follows:

- UFEKS
  - Divide a given time series into a set of its subsequences,
  - Make a kernel matrix from each subsequence using RBF kernel,
  - Place the all kernel matrices across in one row and horizontally concatenate them into one kernel matrix,
  - Extract row vectors in the concatenated matrix as feature vectors.
- UFEKT

  - Divide a given time series into a set of its subsequences,
  - Make a kernel matrix from each subsequence using RBF kernel,
  - Stack the all kernel matrices and make a three-way tensor of a kernel matrix,
  - Decompose the tensor into one core tensor and three factor matrices,
  - Extract row vectors in one of the factor matrices as feature vectors.

When the UFEKT are applied to outlier detection tasks, a value of *rank R*, which is one of the parameters, must be given and we propose the algorithm as follows:

- Rank selection in UFEKT
  - Perform Tucker decomposition while changing ranks,
  - Measure normalized distance between feature vectors in a factor matrix,
  - Regard the longest distance as a representative distance of the rank,
  - Choose the best rank by selecting the representative distance.

This method does not guarantee to output the best rank, however, our experiments showed that favorable results could be obtained.

In addition, to evaluate our algorithms, UFEKS and UFEKT, we applied them to outlier detection tasks, and employed five outlier detection methods for non-time series, such as *κth-Nearest Neighbor* (κNN), *Local Outlier Factor* (LOF), *One-class Support Vector Machine* (OCSVM), *Isolation Forest* (IForest), and *PageRank* (PR) for synthetic and real-world datasets. Our experiments using those datasets showed that our algorithms had an enough potential to extract features from multivariate time series for outlier detection.

Furthermore, we showed the reasons why our algorithm had better results than existing algorithms by analyzing a resulting kernel matrix with the *principal component analysis* (PCA).

Moreover, we also applied our algorithm UFEKT to clustering tasks. As representative clustering algorithms for non-time series, *K-means* (KMeans), *Density-based Spatial Clustering of Applications with Noise* (DBSCAN), *Agglomerative Hierarchical Clustering* (AHC), and *Gaussian Mixture Model* (GMM) are employed for our evaluations. Like the results of outlier detection, they had better results than the existing algorithms.

6.2 Current Limitations and Future Works

Although our algorithms could have better results for some applications we mentioned above, we consider that some challenges remain.

6.2.1 Datasets containing missing values

Datasets that we employed in this dissertation do not contain any missing values. However, when we consider that our algorithms are applied to a real-world system, missing values might be contained in datasets collected by sensors in the system. To overcome this issue, two types of ways might be considered: replacing missing values to complemented values using existing algorithms, or developing a new algorithm that can directly handle missing values. The former might be one of the better ways, however, we consider that our algorithms have capabilities of directory applying to datasets with missing values and interpolating them because both time-wise and variable-wise associations have been already taken into account in our algorithm.

6.2.2 Time series with different lengths

Our algorithms require the same length of multivariate time series as input datasets. Therefore, if collected time series from sensors have different lengths from each other, we need to adjust them to the same lengths before applying them to our algorithms. However, if some similarity measurements between multivariate time series like the Dynamic Time Warping (DTW) can be applied to our algorithms, it can be said that our algorithm supports a variety of time series with different lengths. For instance, considered that two time series with same time period but different sampling interval are given, a kernel matrix that represent association between them can be calculated by DTW substituted for RBF kernel in UFEKT after dividing the time series into the same numbers of subsequences from the time series. This might be worthy of consideration.

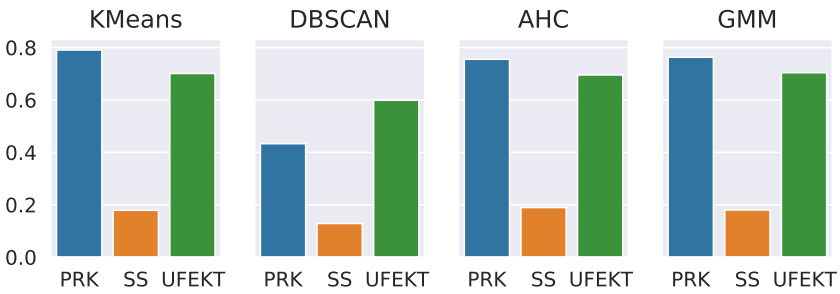
6.2.3 Effective usage of factor matrices

As we have mentioned in this thesis, variables including outliers can be extracted by checking one of the factor matrices obtained by Tucker decomposition. However, the result we showed in this thesis is one of the datasets donated as ATSF16. We need to try to apply the method to a variety of datasets and evaluate accuracy of them. If high accuracy can be obtained from many datasets, UFEKT would be more attractive algorithm because it can be said that it is able to generate two types of features, time-wise and variable-wise features, in one method.

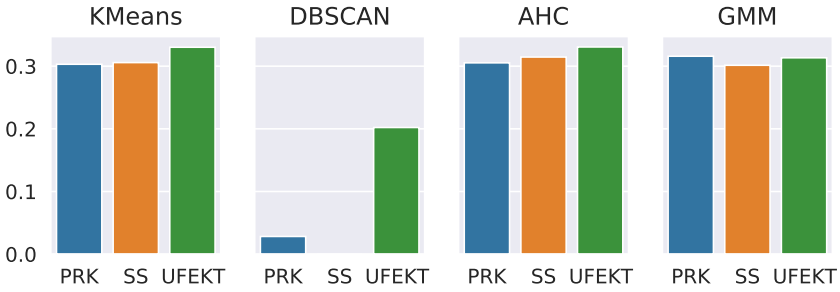
Appendix A

Experimental results for clustering by UFEKT

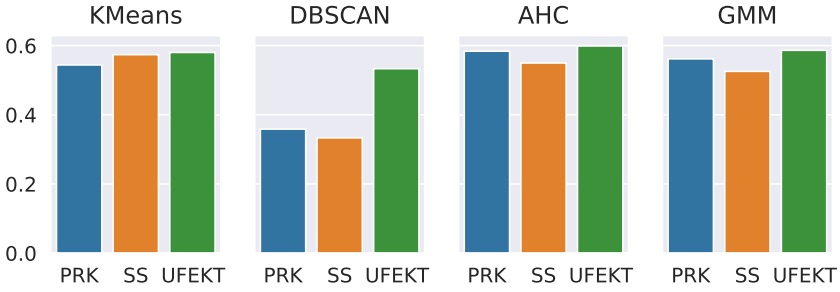
Experimental results of a clustering using UCR datasets are shown here. We performed three feature representation algorithms, UFEKT, PageRank kernel(PRK), and Subsequence (SS), combined with four clustering algorithms, KMeans, DBSCAN, AHC, and GMM, resulting in twelve combinations of feature extraction and clustering in total. The effectiveness of each method was evaluated by *Normalized Mutual Information* (NMI). The NMI score takes values between zero and one, and higher is better.



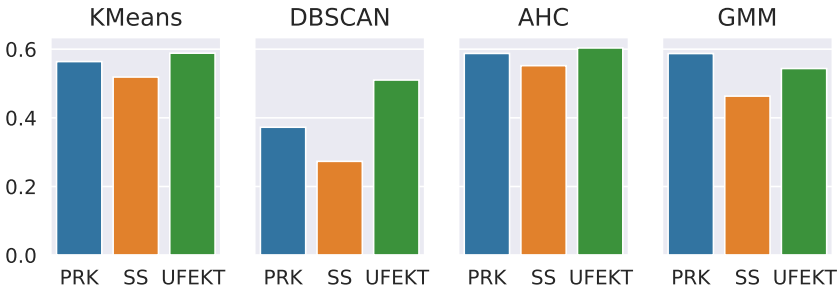
(a) ACSF1



(b) Adiac

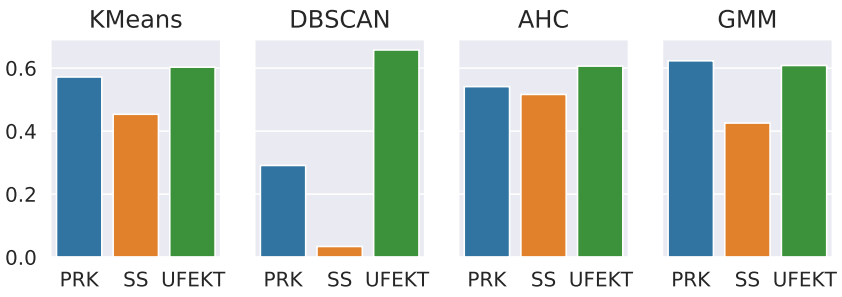


(c) AllGestureWiimoteX

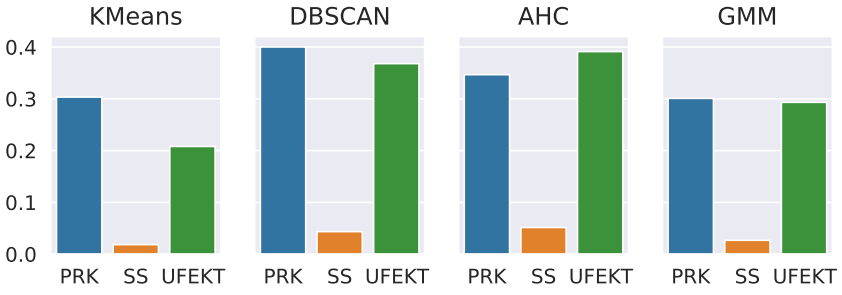


(d) AllGestureWiimoteY

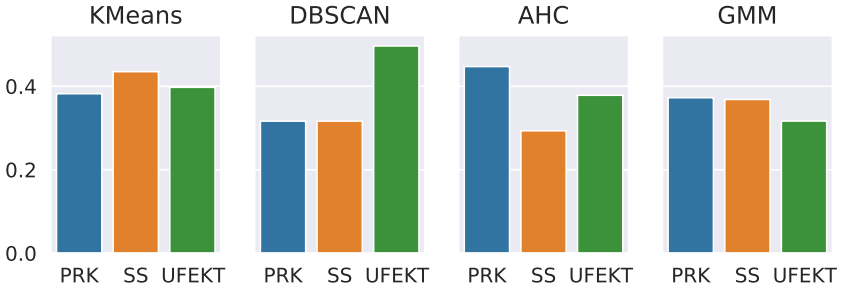
Figure A.1: NMIs for UCR (#1).



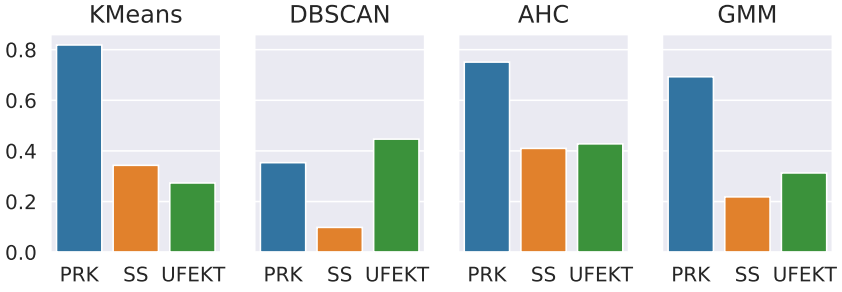
(a) AllGestureWiimoteZ



(b) ArrowHead

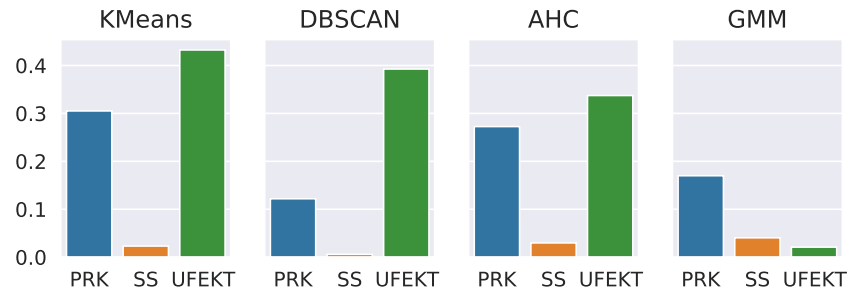


(c) BME

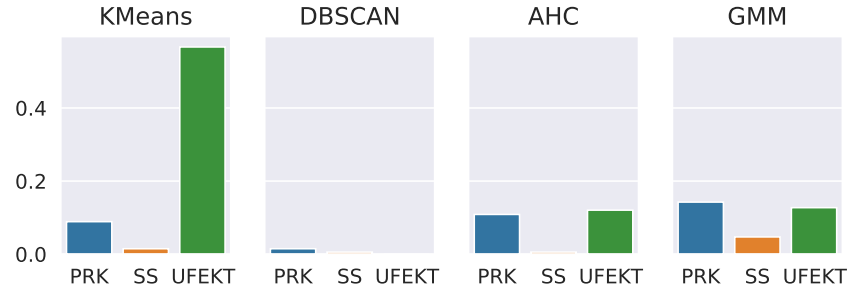


(d) Beef

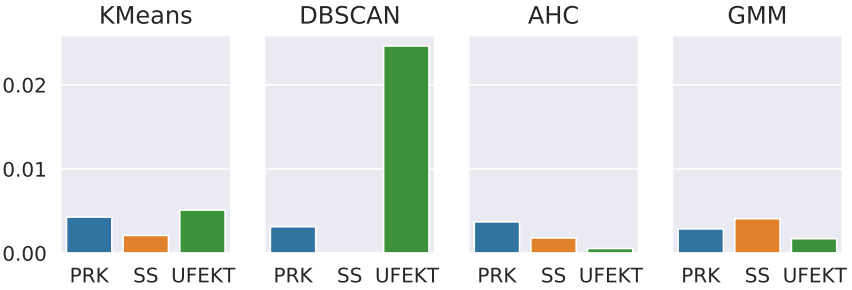
Figure A.2: NMIs for UCR (#2).



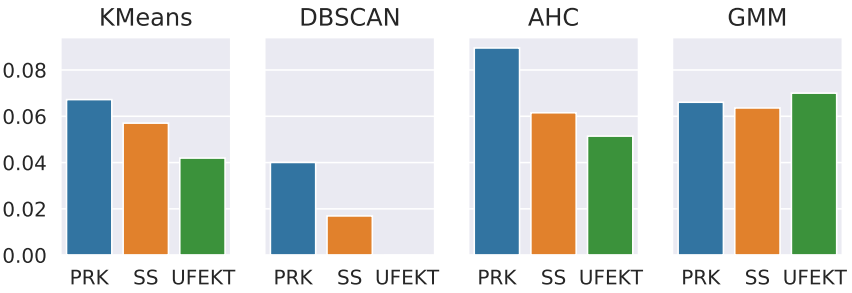
(a) BeetleFly



(b) BirdChicken

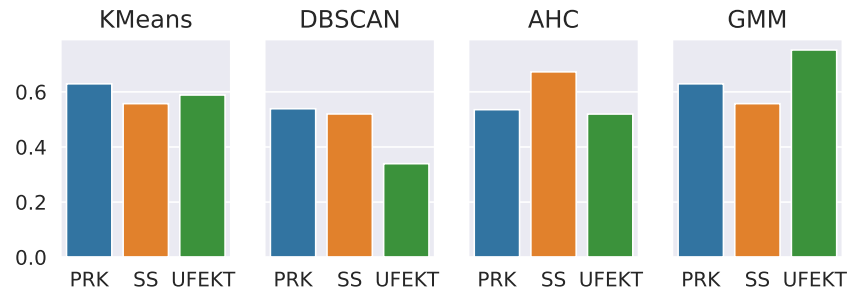


(c) CBF

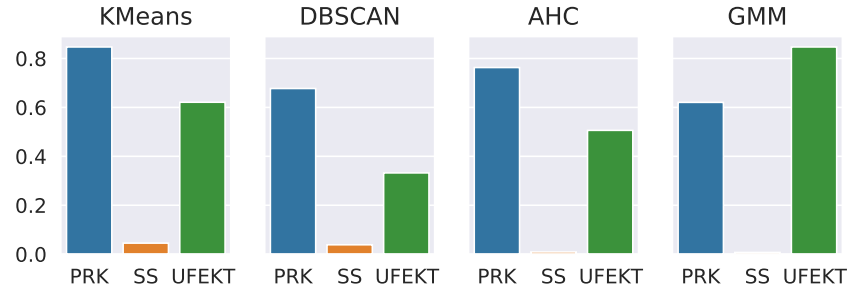


(d) Car

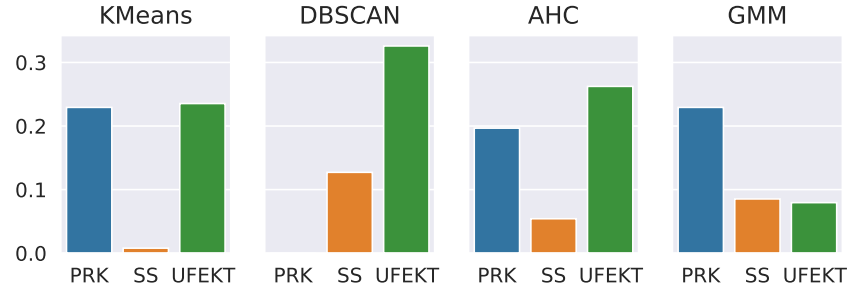
Figure A.3: NMIs for UCR (#3).



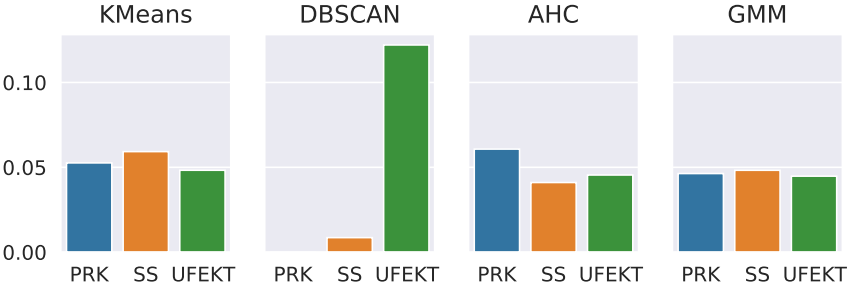
(a) Chinatown



(b) Coffee

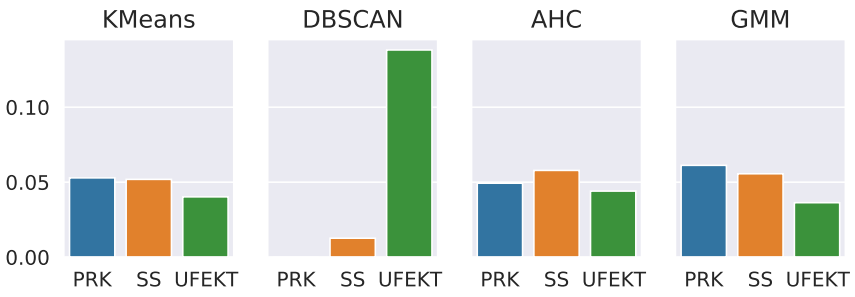


(c) Computers

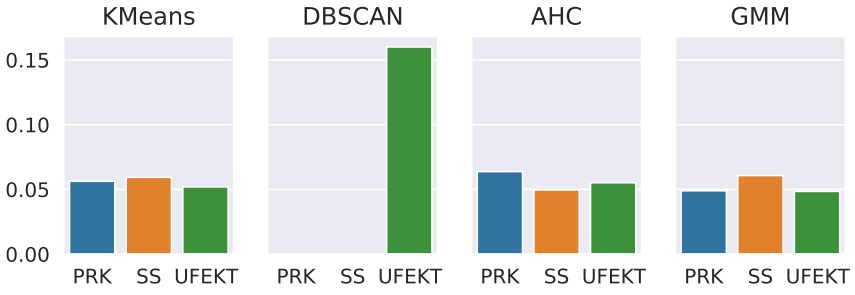


(d) CricketX

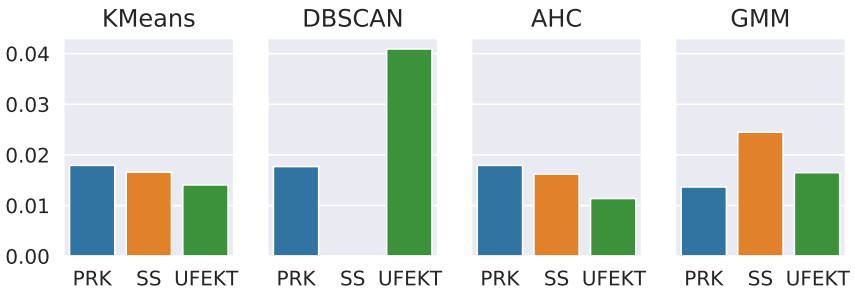
Figure A.4: NMIs for UCR (#4).



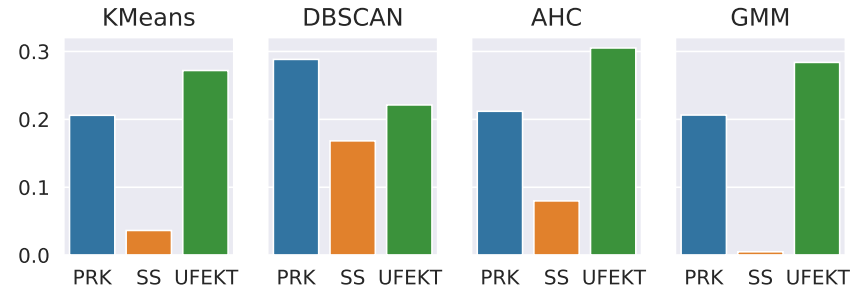
(a) CricketY



(b) CricketZ

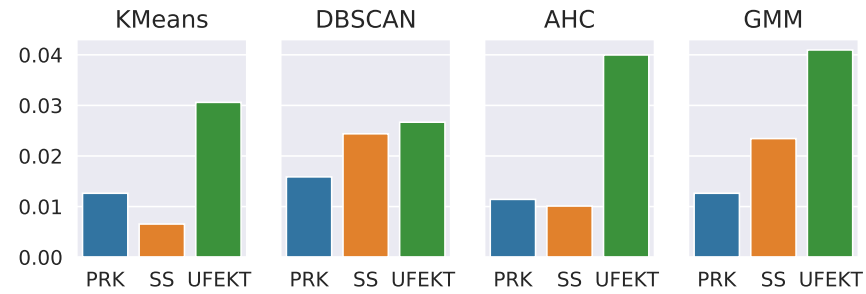


(c) DiatomSizeReduction

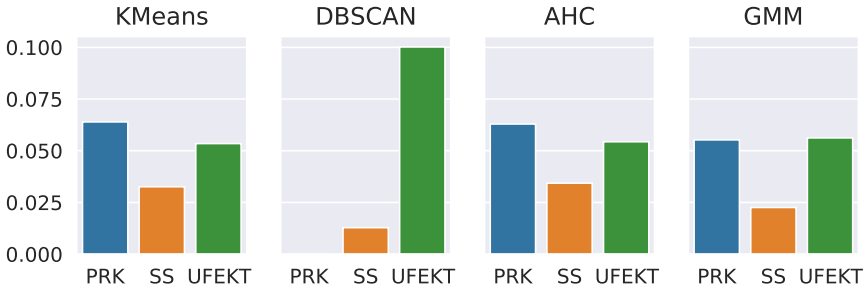


(d) DistalPhalanxOutlineAgeGroup

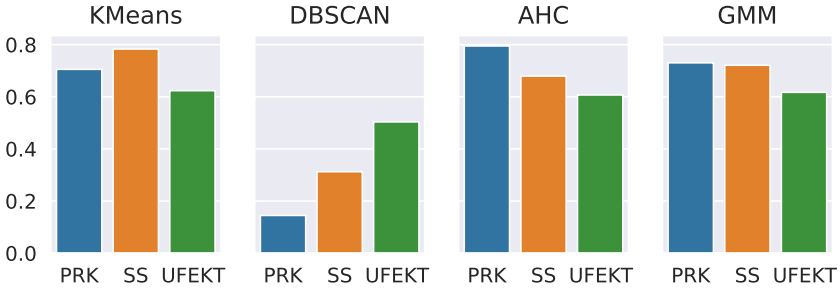
Figure A.5: NMIs for UCR (#5).



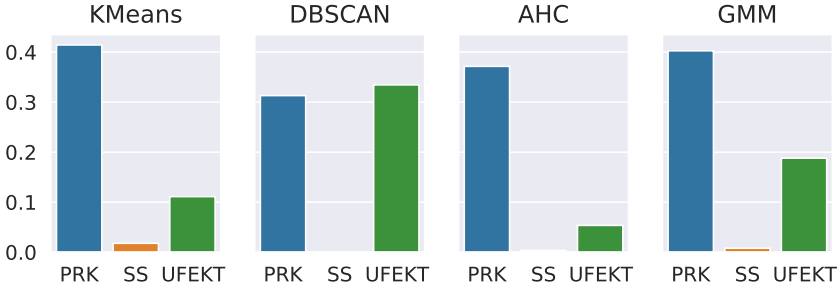
(a) DistalPhalanxOutlineCorrect



(b) DistalPhalanxTW

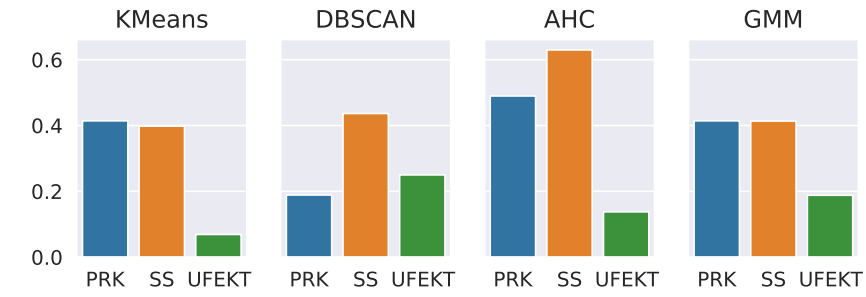


(c) DodgerLoopDay

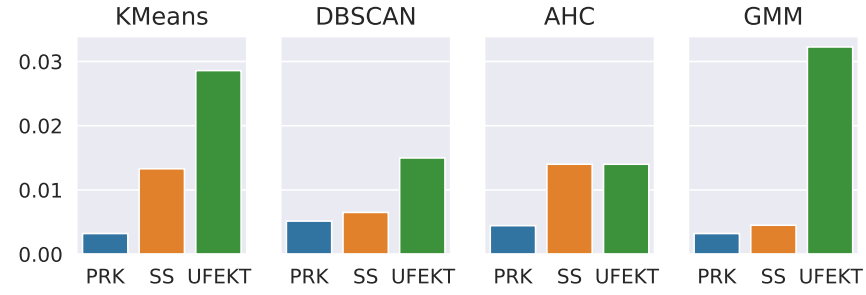


(d) DodgerLoopGame

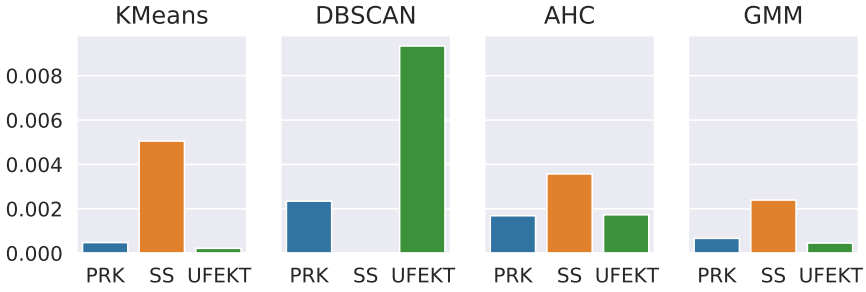
Figure A.6: NMIs for UCR (#6).



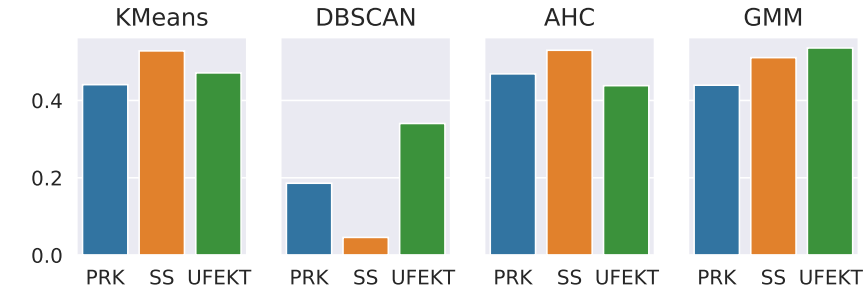
(a) DodgerLoopWeekend



(b) ECG200

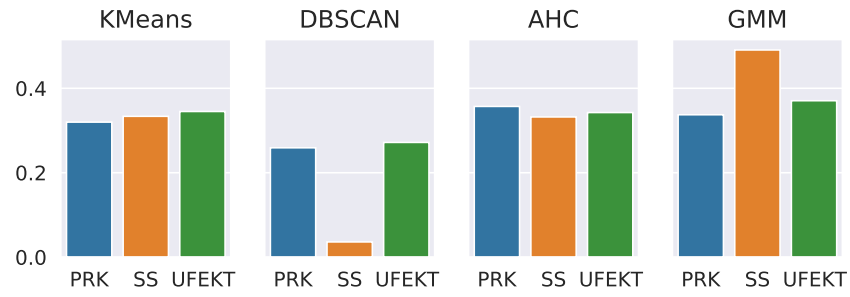


(c) ECGFiveDays

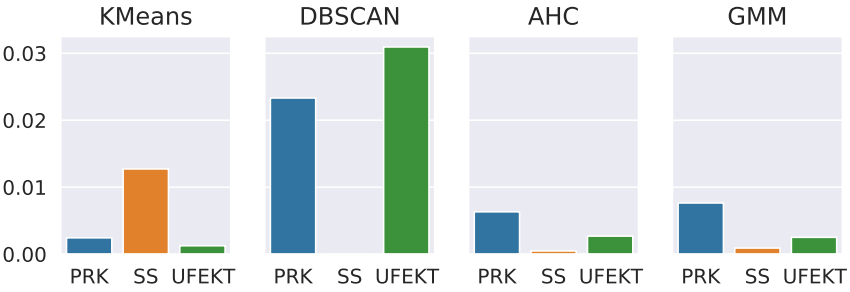


(d) EOGHorizontalSignal

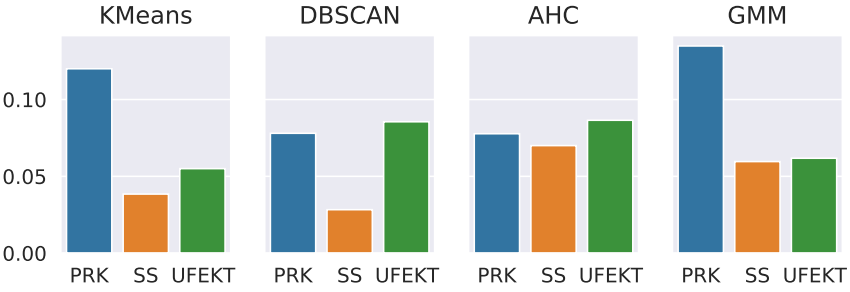
Figure A.7: NMIs for UCR (#7).



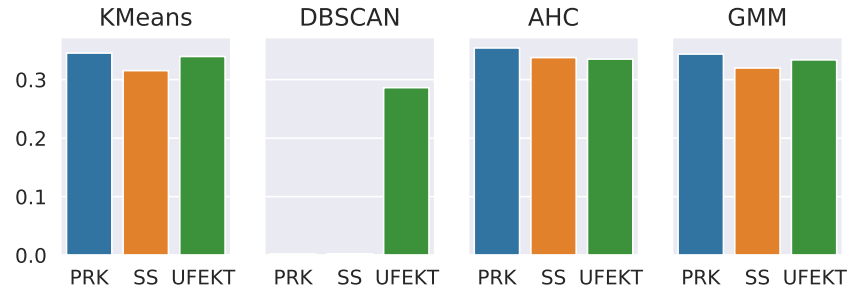
(a) EOGVerticalSignal



(b) Earthquakes

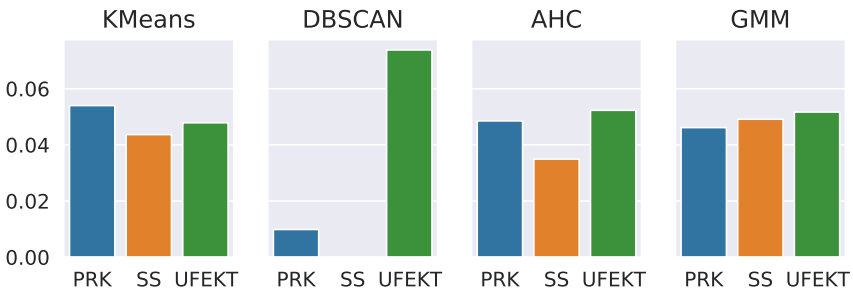


(c) FaceFour

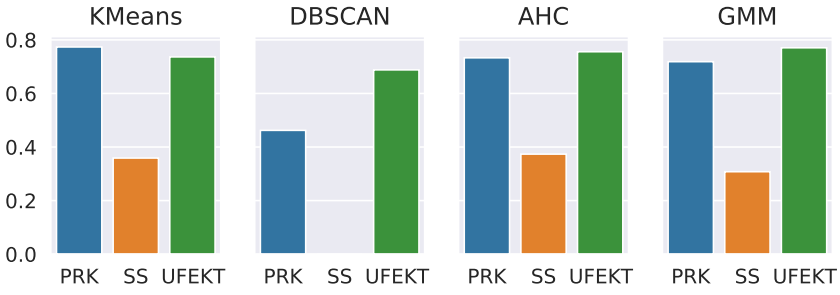


(d) FiftyWords

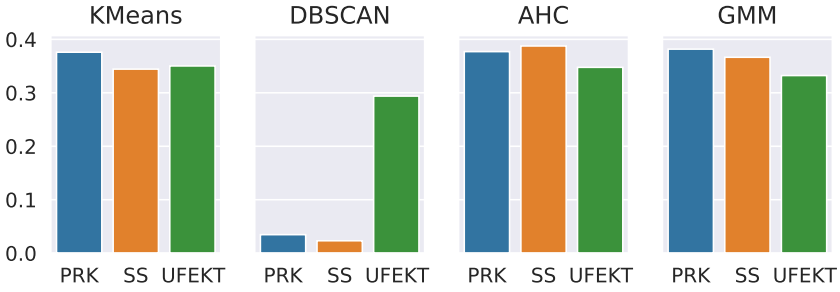
Figure A.8: NMIs for UCR (#8).



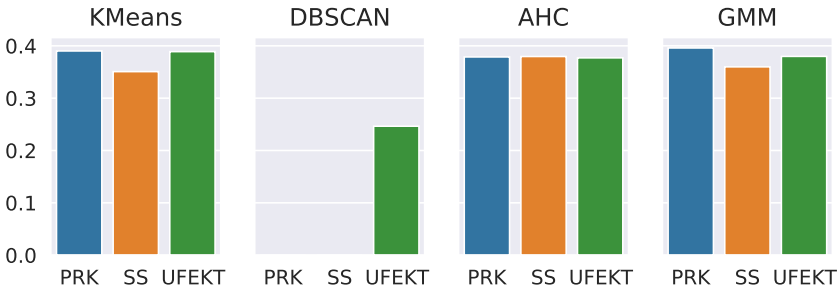
(a) Fish



(b) Fungi

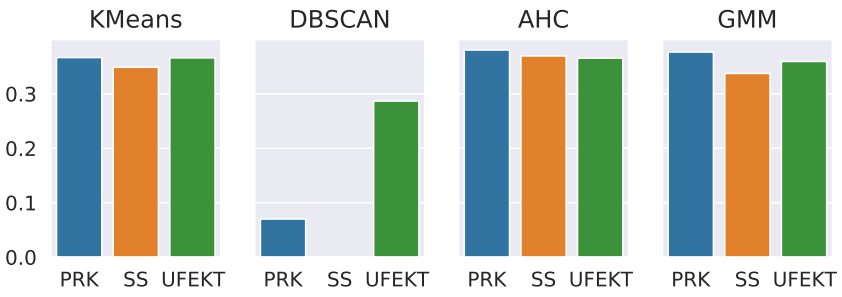


(c) GestureMidAirD1

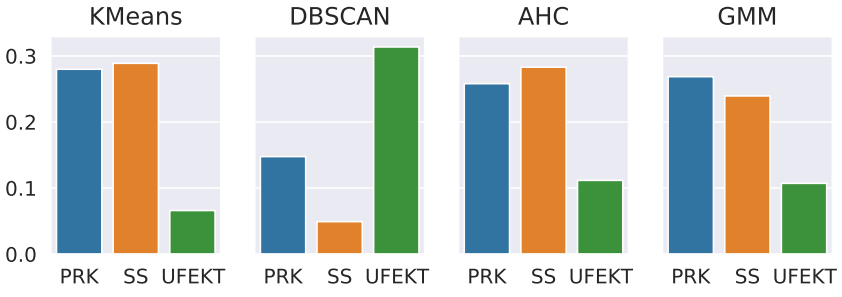


(d) GestureMidAirD2

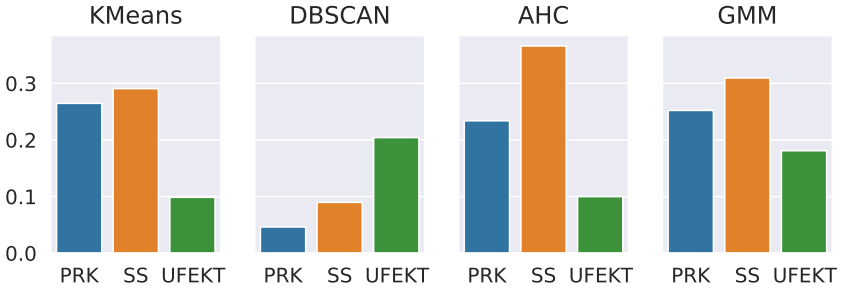
Figure A.9: NMIs for UCR (#9).



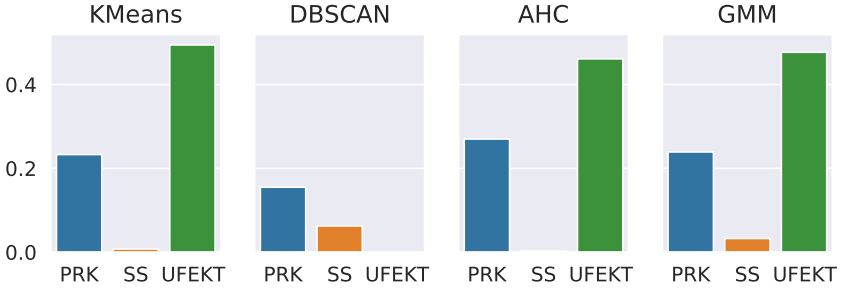
(a) GestureMidAirD3



(b) GesturePebbleZ1

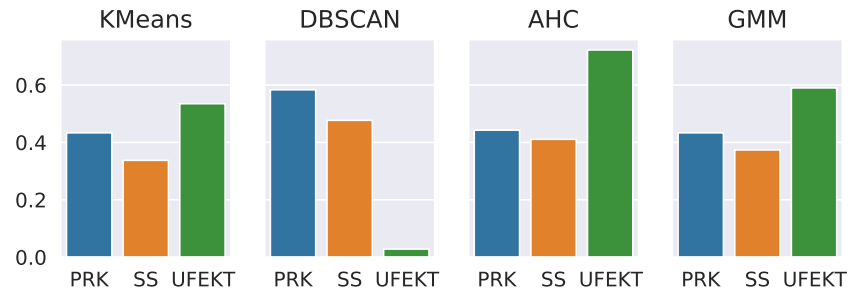


(c) GesturePebbleZ2

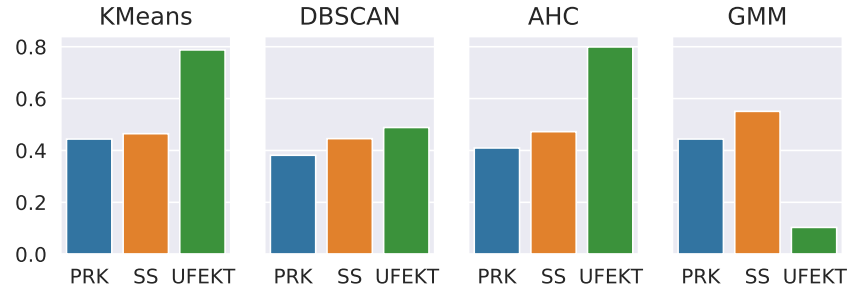


(d) GunPointAgeSpan

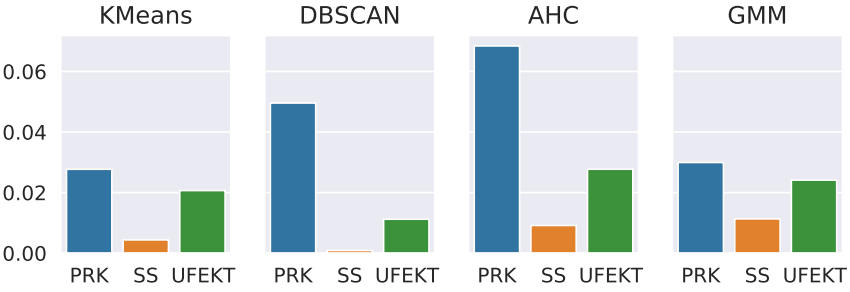
Figure A.10: NMIs for UCR (#10).



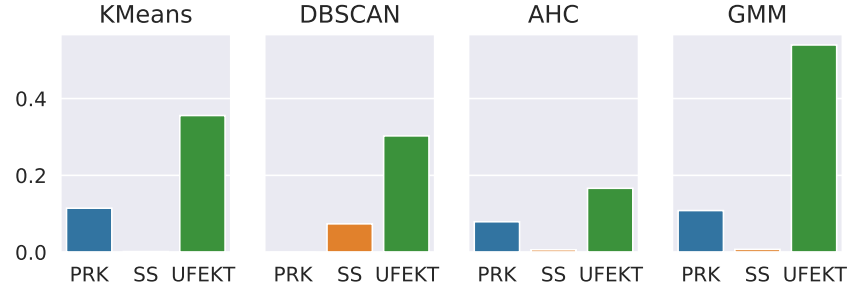
(a) GunPointMaleVersusFemale



(b) GunPointOldVersusYoung

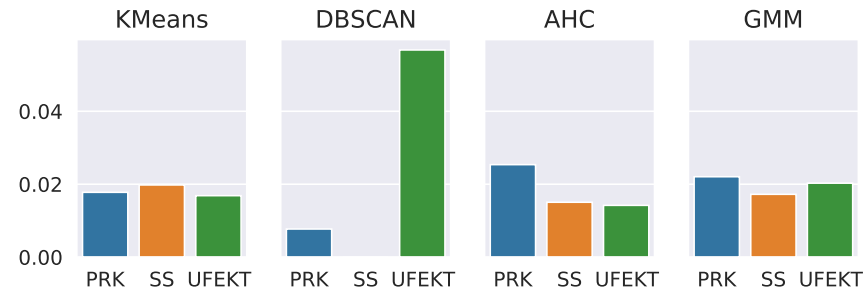


(c) GunPoint

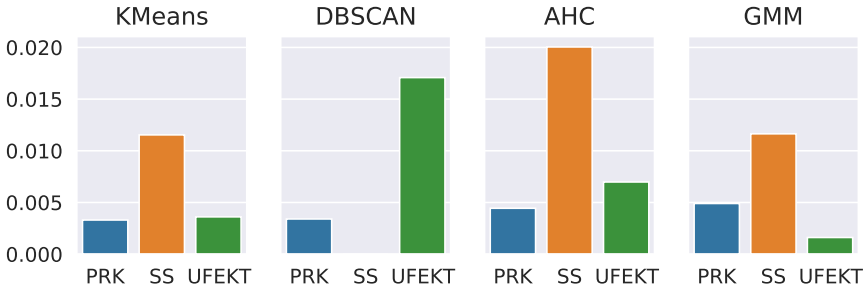


(d) Ham

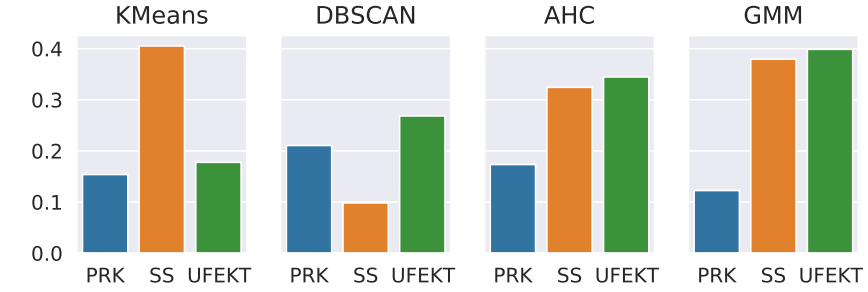
Figure A.11: NMIs for UCR (#11).



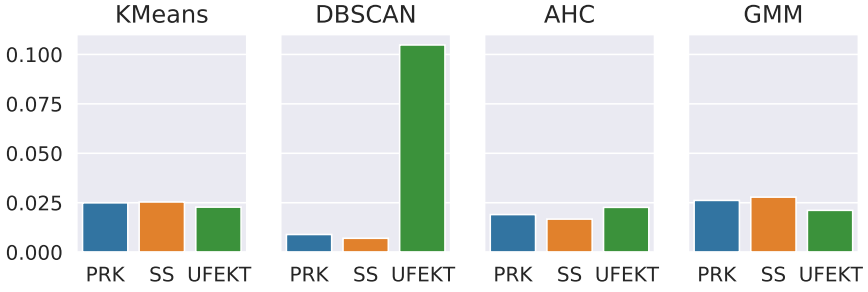
(a) Haptics



(b) Herring

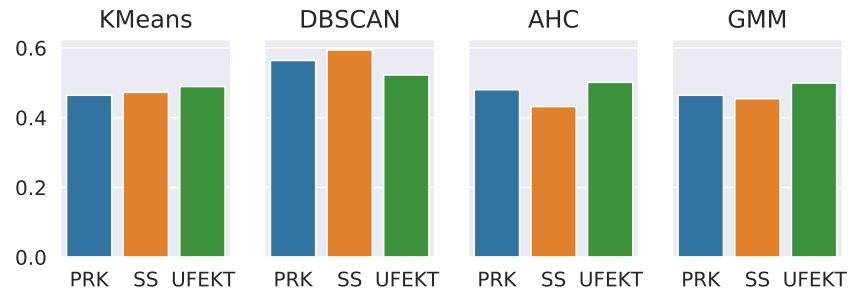


(c) HouseTwenty

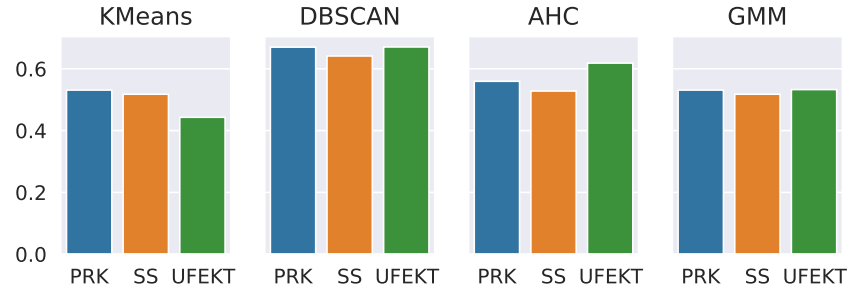


(d) InlineSkate

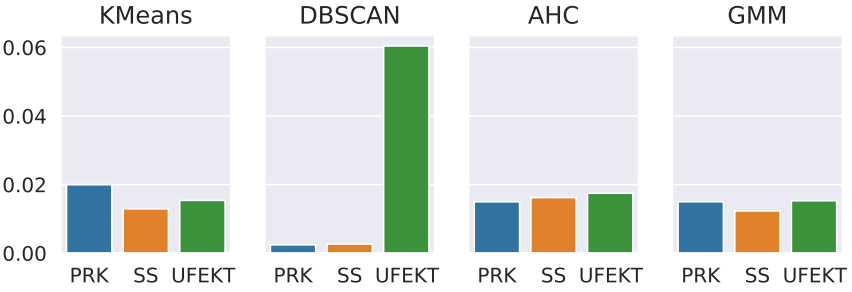
Figure A.12: NMIs for UCR (#12).



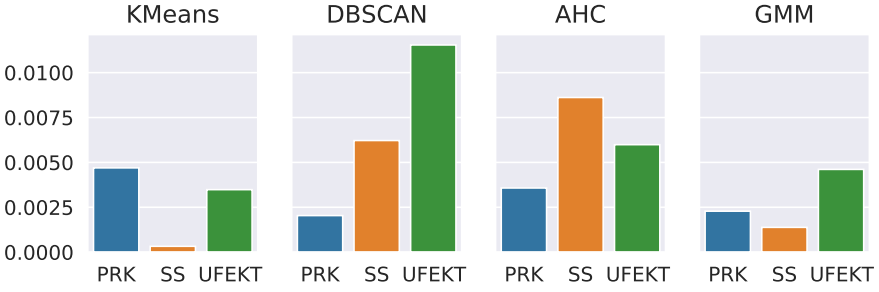
(a) InsectEPGRegularTrain



(b) InsectEPGSmallTrain

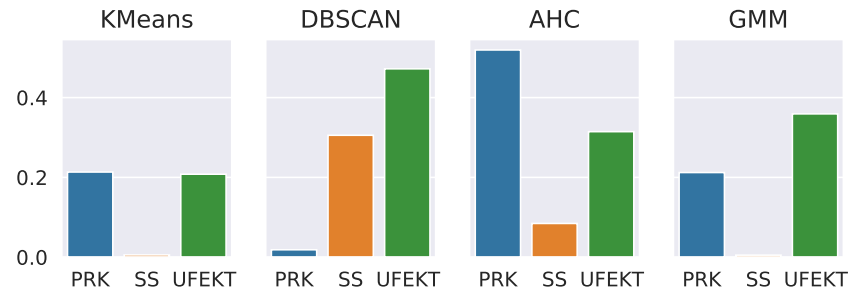


(c) InsectWingbeatSound

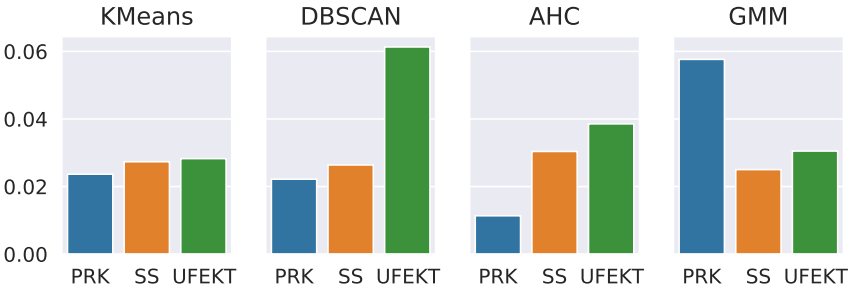


(d) ItalyPowerDemand

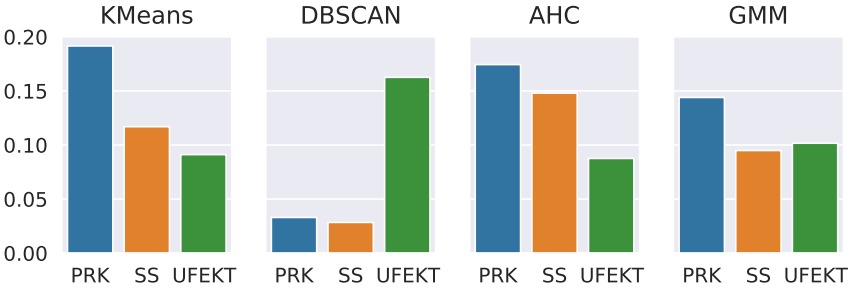
Figure A.13: NMIs for UCR (#13).



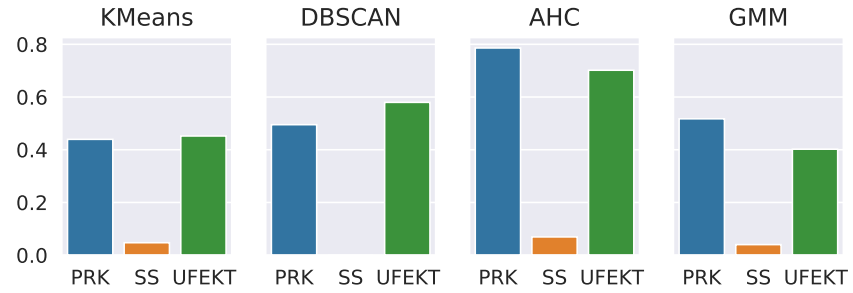
(a) LargeKitchenAppliances



(b) Lightning2

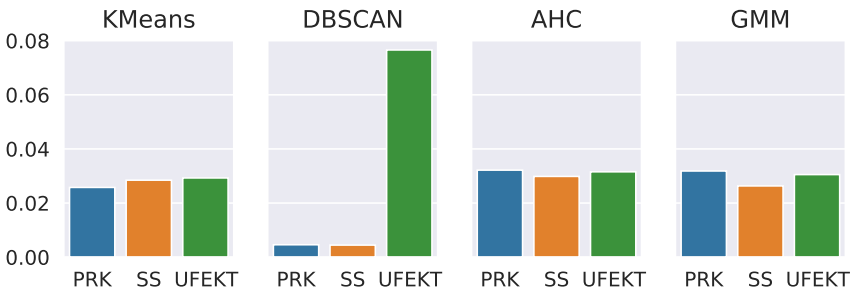


(c) Lightning7

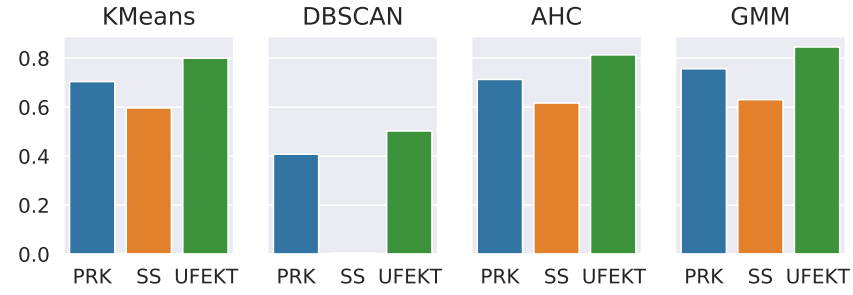


(d) Meat

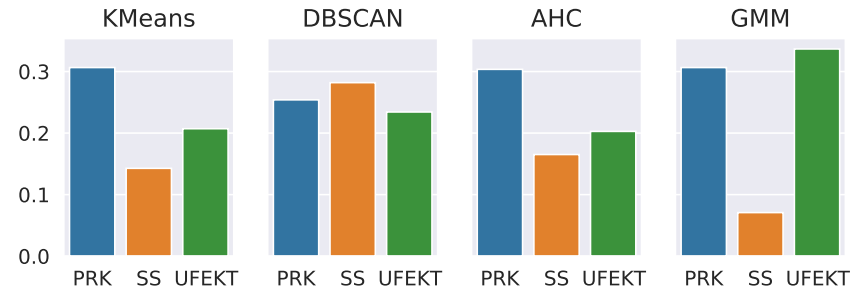
Figure A.14: NMIs for UCR (#14).



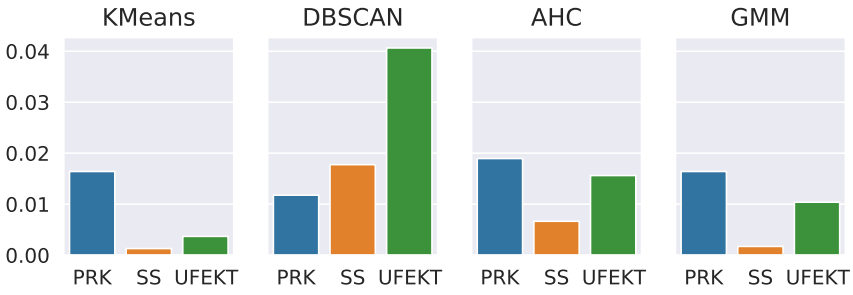
(a) MedicalImages



(b) MelbournePedestrian

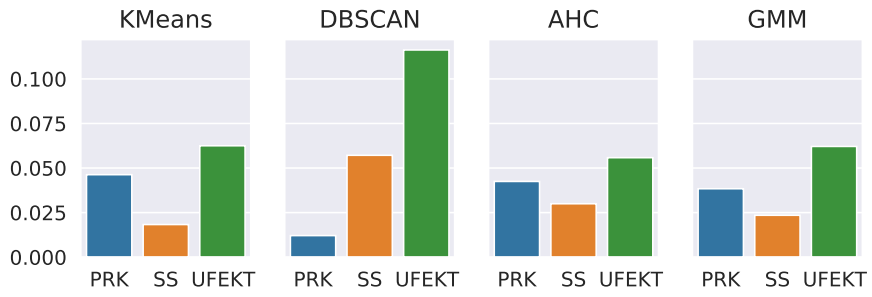


(c) MiddlePhalanxOutlineAgeGroup

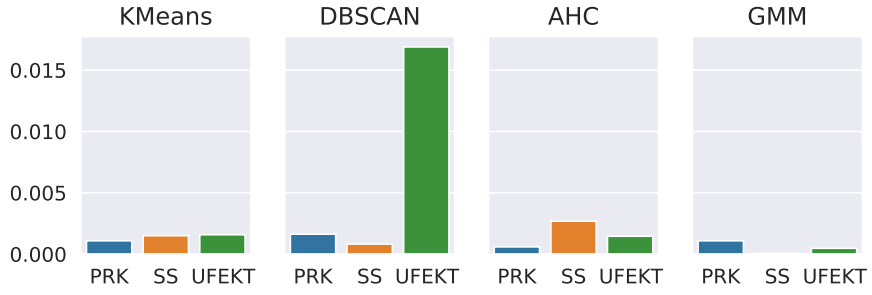


(d) MiddlePhalanxOutlineCorrect

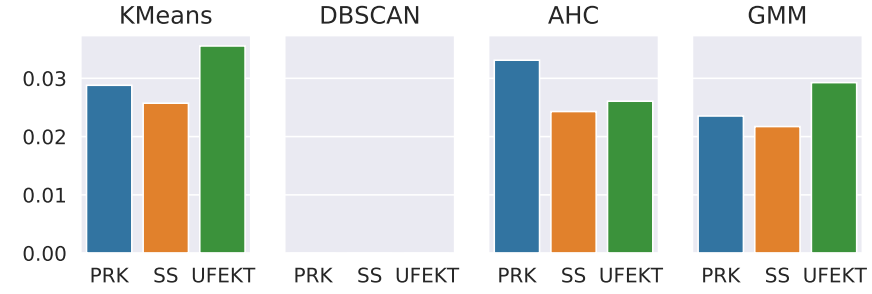
Figure A.15: NMIs for UCR (#15).



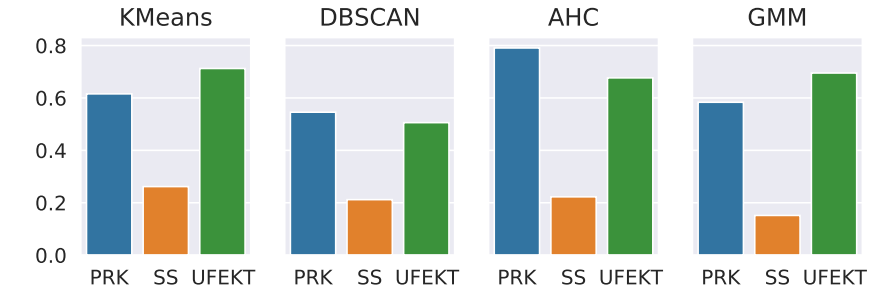
(a) MiddlePhalanxTW



(b) MoteStrain

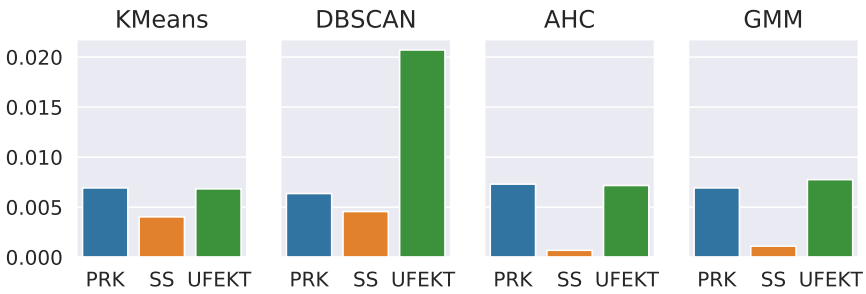


(c) OSULeaf

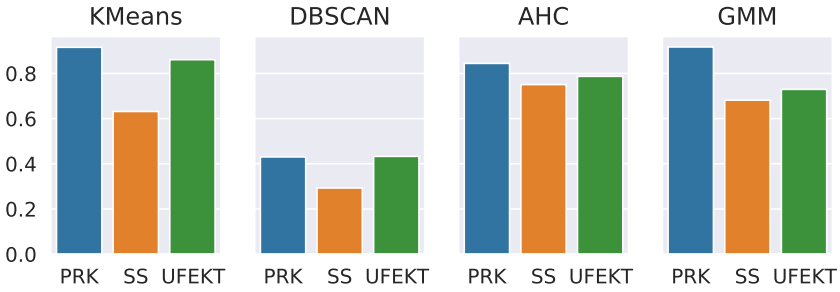


(d) OliveOil

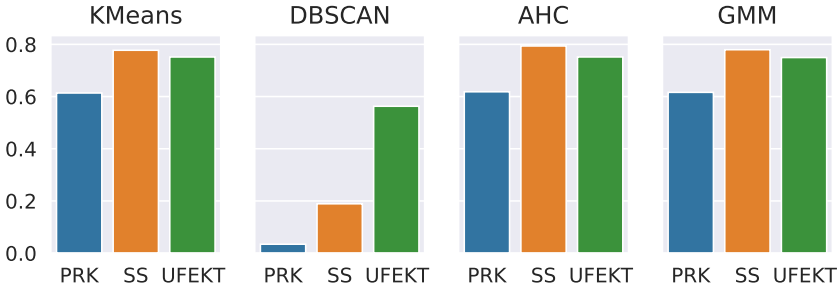
Figure A.16: NMIs for UCR (#16).



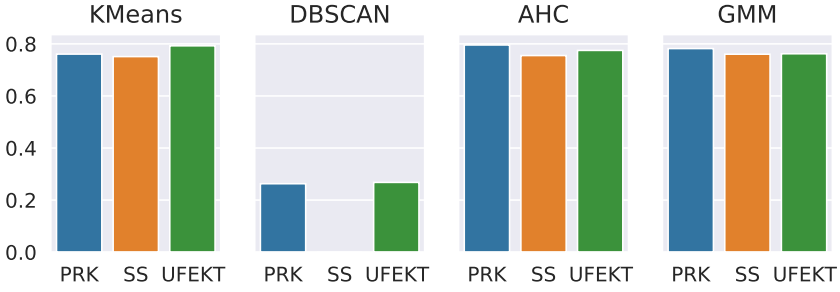
(a) PhalangesOutlinesCorrect



(b) PickupGestureWiimoteZ

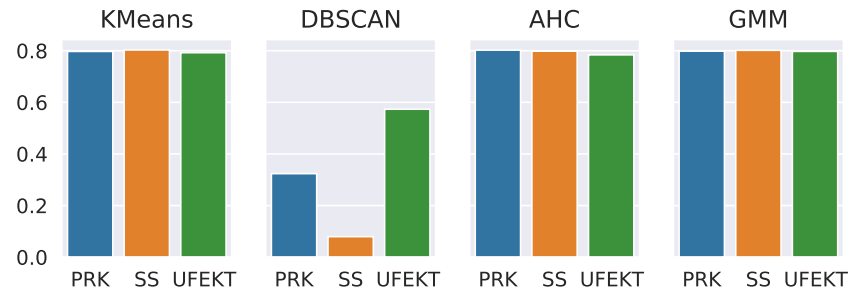


(c) PigAirwayPressure

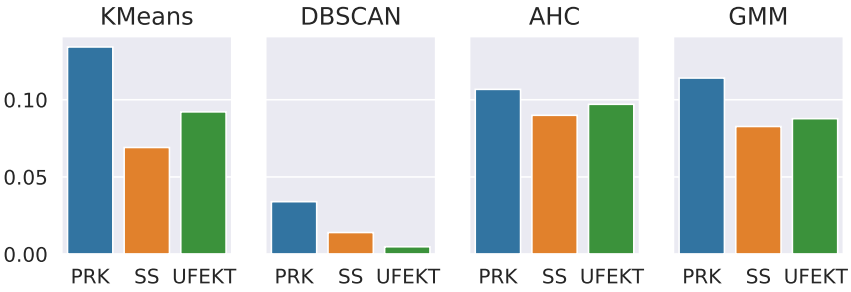


(d) PigArtPressure

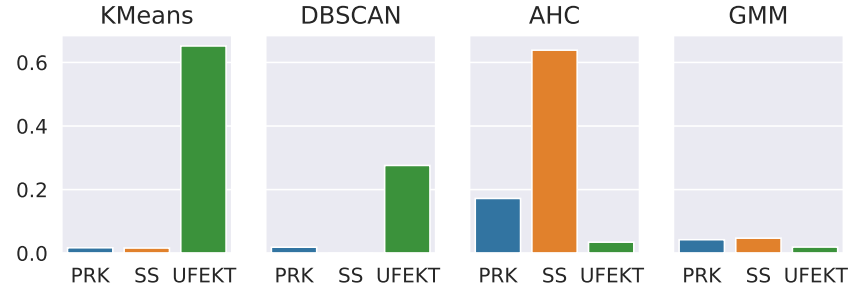
Figure A.17: NMIs for UCR (#17).



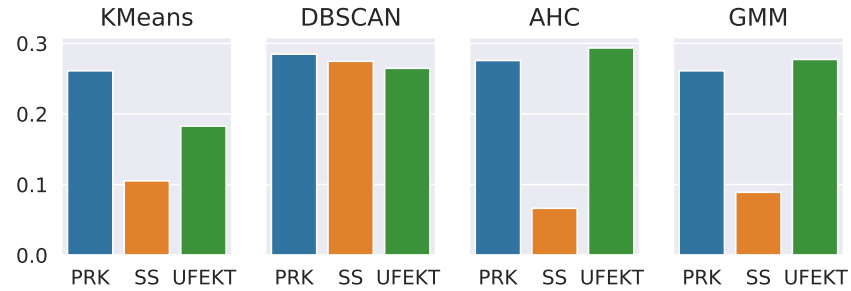
(a) PigCVP



(b) Plane

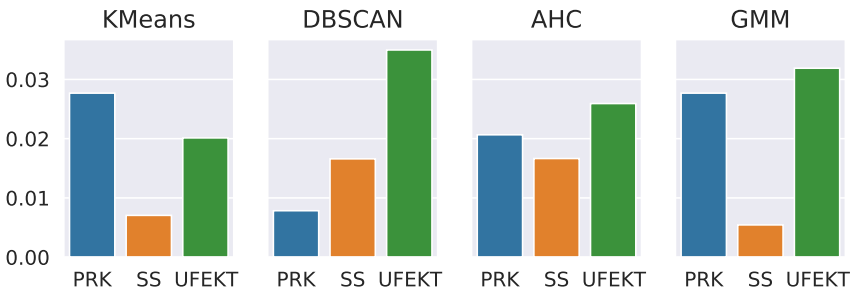


(c) PowerCons

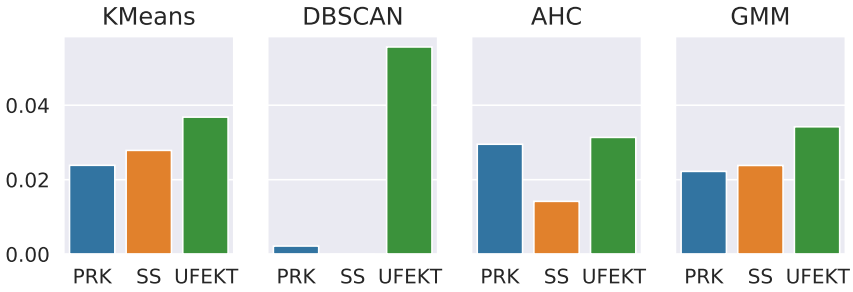


(d) ProximalPhalanxOutlineAgeGroup

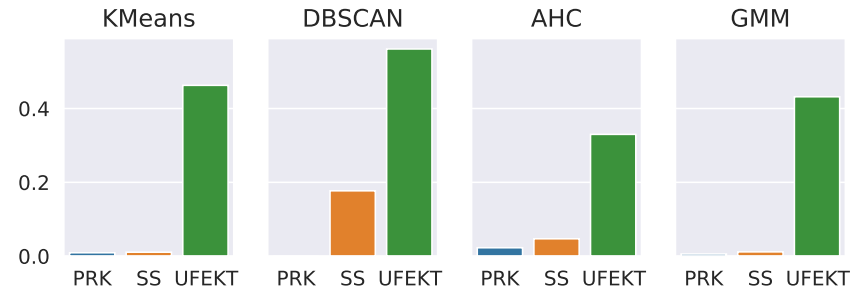
Figure A.18: NMIs for UCR (#18).



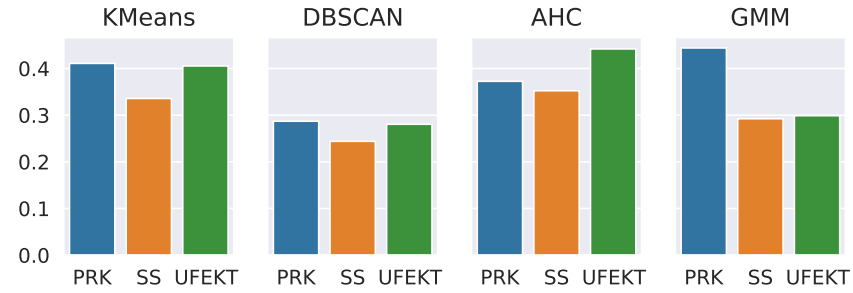
(a) ProximalPhalanxOutlineCorrect



(b) ProximalPhalanxTW

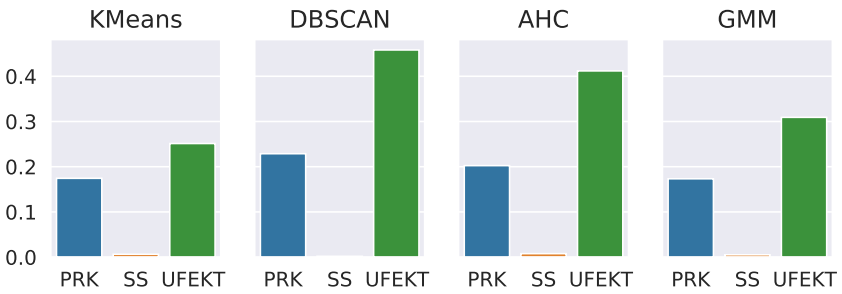


(c) RefrigerationDevices

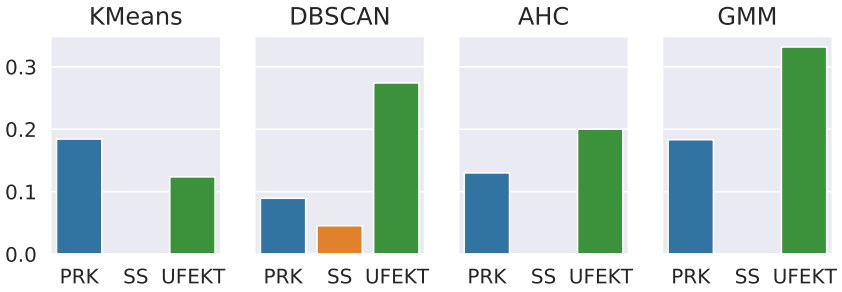


(d) Rock

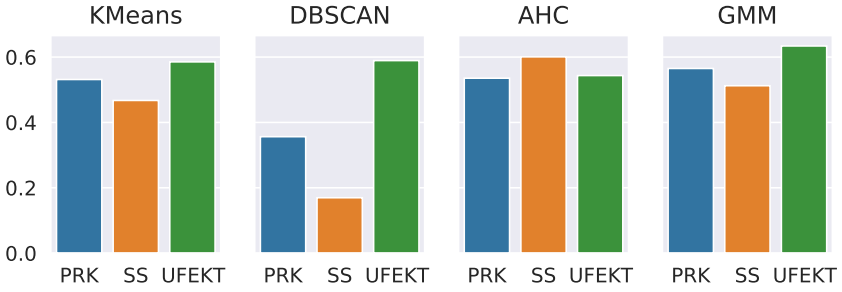
Figure A.19: NMIs for UCR (#19).



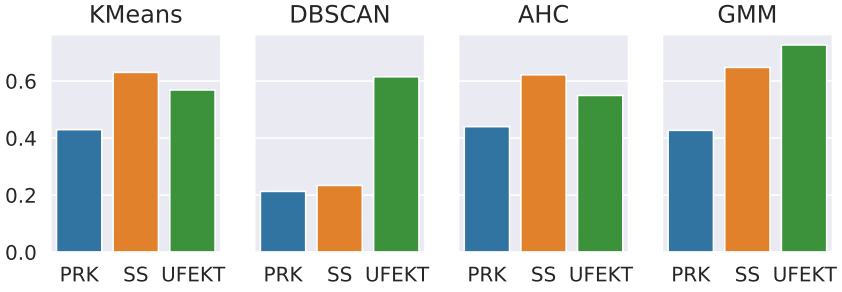
(a) ScreenType



(b) SemgHandGenderCh2

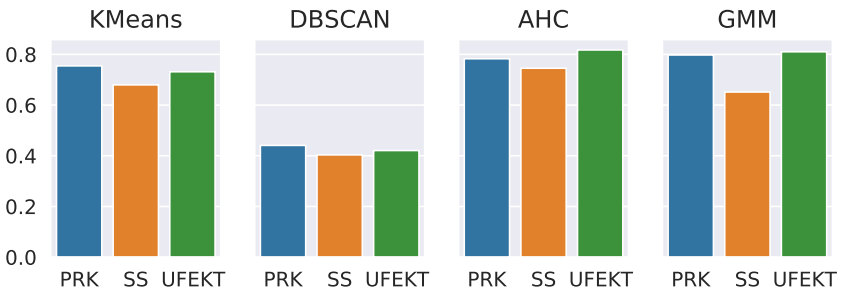


(c) SemgHandMovementCh2

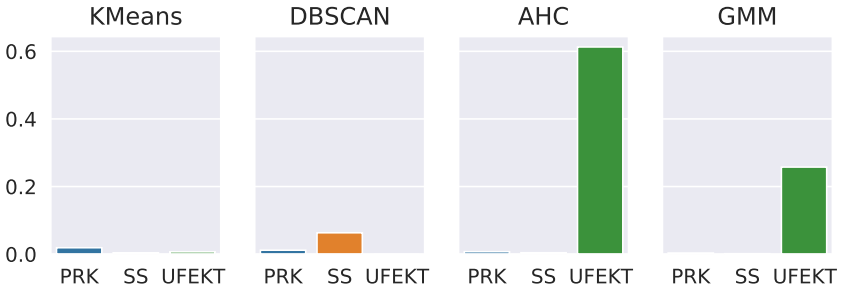


(d) SemgHandSubjectCh2

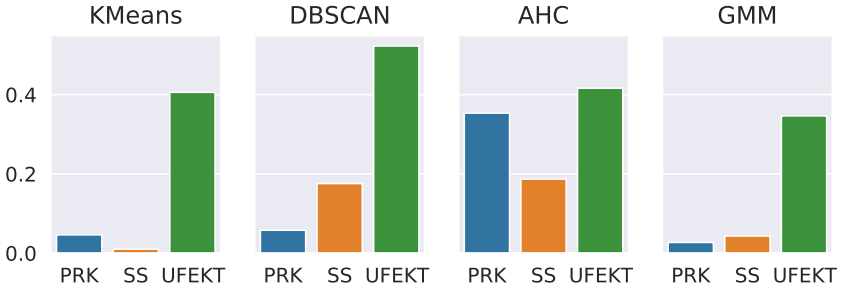
Figure A.20: NMIs for UCR (#20).



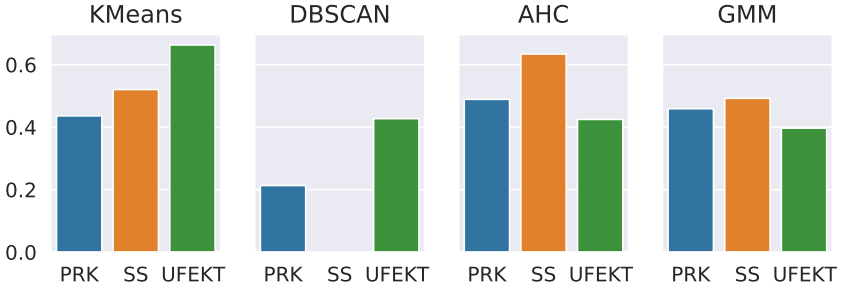
(a) ShakeGestureWiimoteZ



(b) ShapeletSim

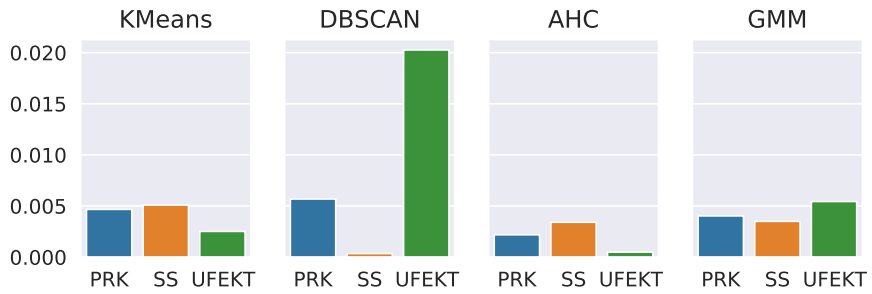


(c) SmallKitchenAppliances

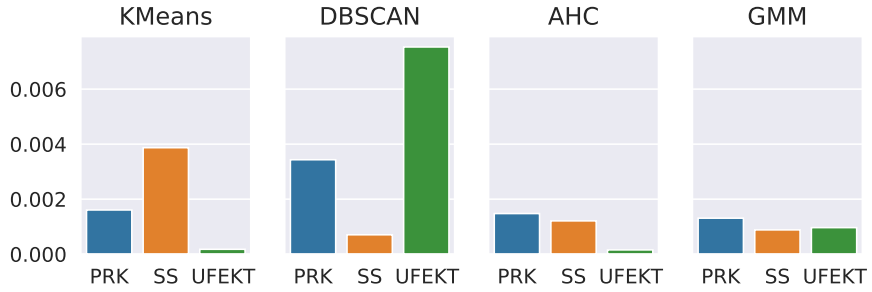


(d) SmoothSubspace

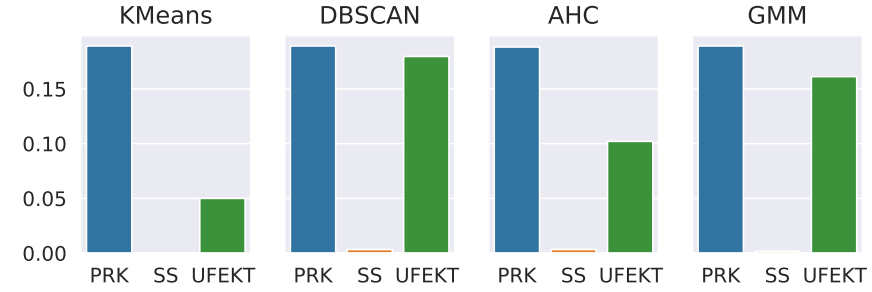
Figure A.21: NMIs for UCR (#21).



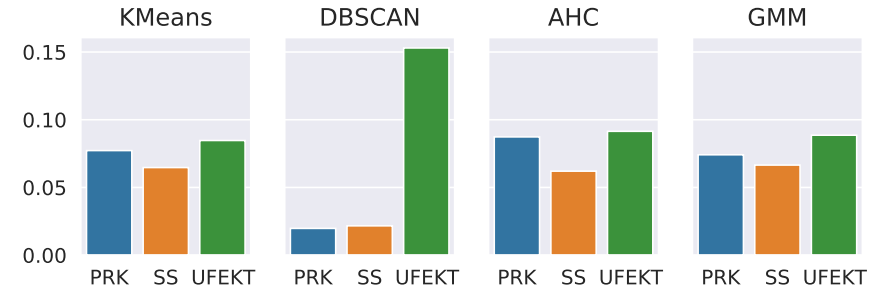
(a) SonyAIBORobotSurface1



(b) SonyAIBORobotSurface2

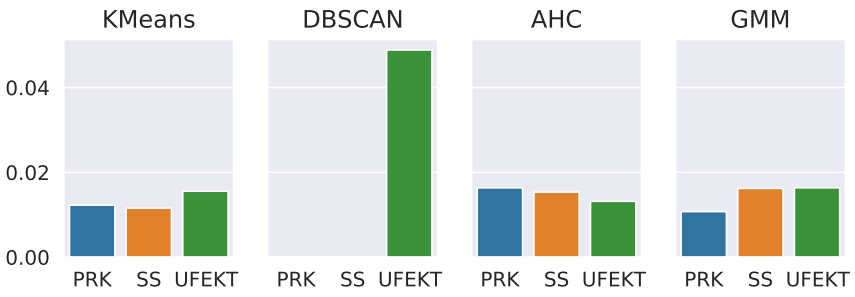


(c) Strawberry

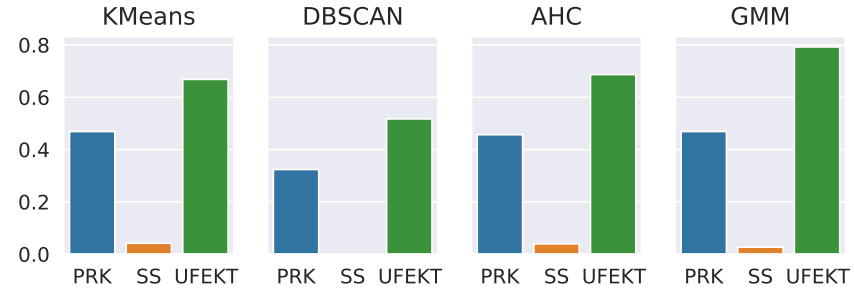


(d) SwedishLeaf

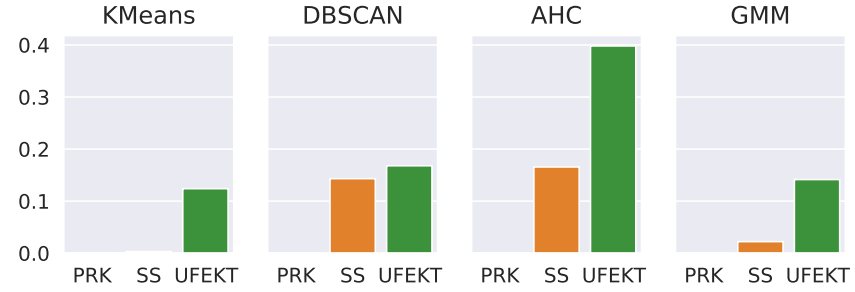
Figure A.22: NMIs for UCR (#22).



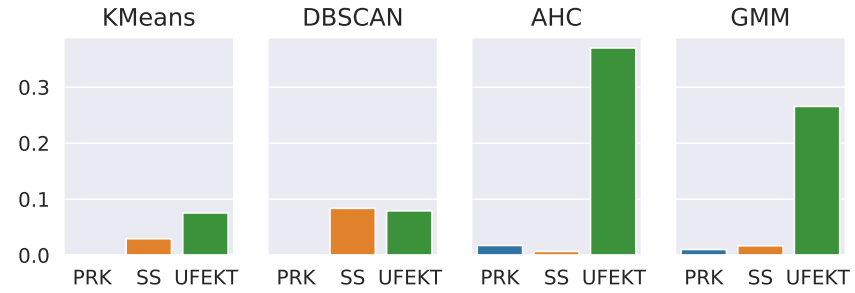
(a) Symbols



(b) SyntheticControl

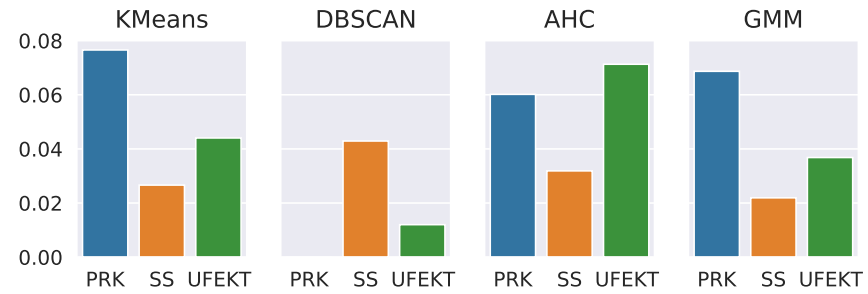


(c) ToeSegmentation1

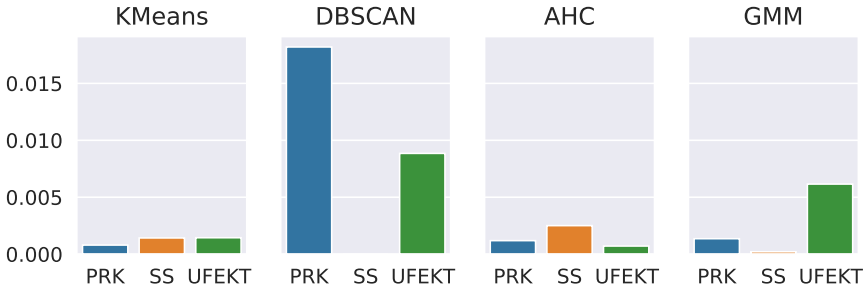


(d) ToeSegmentation2

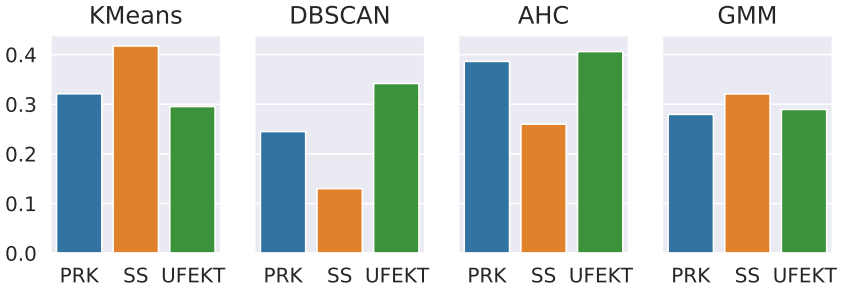
Figure A.23: NMIs for UCR (#23).



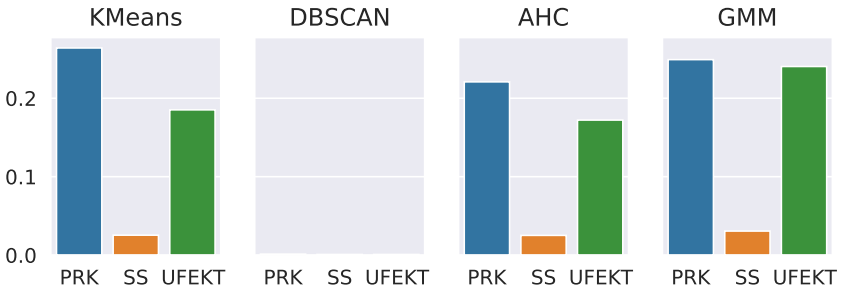
(a) Trace



(b) TwoLeadECG

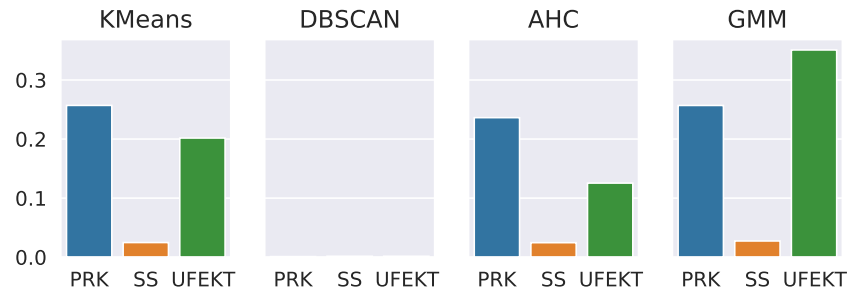


(c) UMD

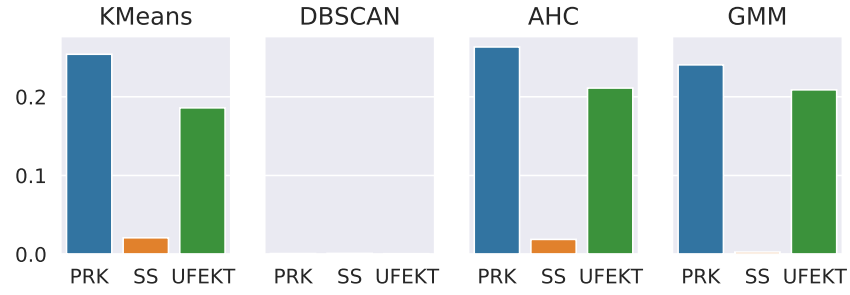


(d) UWaveGestureLibraryX

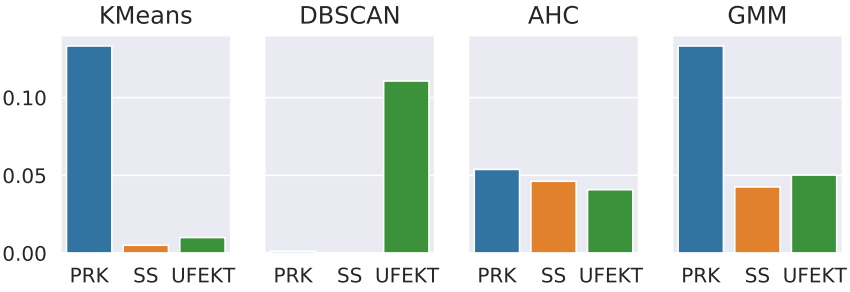
Figure A.24: NMIs for UCR (#24).



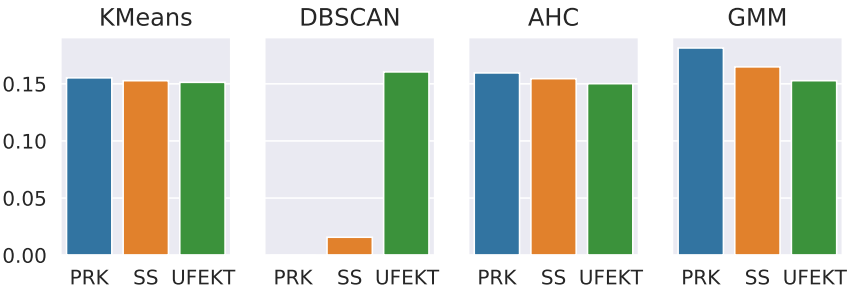
(a) UWaveGestureLibraryY



(b) UWaveGestureLibraryZ

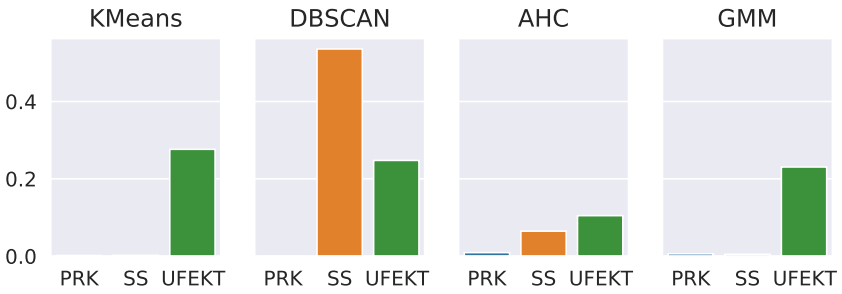


(c) Wine

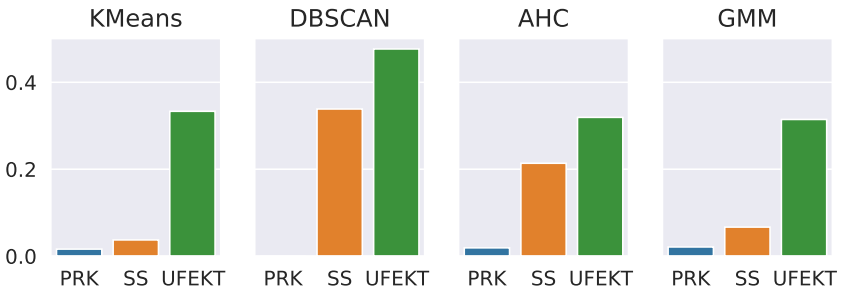


(d) WordSynonyms

Figure A.25: NMIs for UCR (#25).



(a) WormsTwoClass



(b) Worms

Figure A.26: NMIs for UCR (#26).

Bibliography

[1] Charu C. Aggarwal. *Outlier Analysis*. Springer, 2017.

[2] Charu C. Aggarwal and Saket Sathe. *Outlier Ensembles*. Springer, 2017.

[3] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering – A decade review. *Information Systems*, 53:16–38, oct 2015.

[4] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, nov 2017.

[5] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 3395–3404. ACM, 2020.

[6] Stephen D Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, 2003.

[7] Donald Berndt and James Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. *Workshop on Knowledge Knowledge Discovery in Databases*, 398:359–370, 1994.

[8] Kanishka Bhaduri, Bryan L. Matthews, and Chris R. Giannella. Algorithms for speeding up distance-based outlier detection. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 859–867, 2011.

[9] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 93–104, 2000.

[10] J. Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, sep 1970.

[11] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection. *ACM Computing Surveys*, 41(3):1–58, jul 2009.

[12] Haibin Cheng, Pang-Ning Tan, Christopher Potter, and Steven Klooster. Detection and Characterization of Anomalies in Multivariate Time Series. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, volume 1, pages 413–424, apr 2009.

[13] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer New York, 2006.

[14] William H E Day and Herbert Edelsbrunner. Efficient Algorithms for Agglomerative Hierarchical Clustering Methods. *Journal of Classification*, 1:7–24, 1984.

[15] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[16] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. Differential-geometric Newton method for the best rank-(R_1, R_2, R_3) approximation of tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, jan 2000.

[17] Martin Ester, Hans-peter Kriegel, Jorg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings 1996 International Conference Knowledge Discovery and Data Mining (KDD'96)*, pages 226–231, 1996.

[18] Hadi Fanaee-T and João Gama. Tensor-based anomaly detection: An interdisciplinary survey. *Knowledge-Based Systems*, 98:130–147, apr 2016.

[19] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. Outlier Detection for Temporal Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9), sep 2014.

[20] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining Concepts and Techniques*. IEEE, third edition, dec 2012.

[21] Richard a Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16(10):1– 84, 1970.

[22] Mariya Ishteva, P.-A. Absil, Sabine Van Huffel, and Lieven De Lathauwer. Best Low Multilinear Rank Approximation of Higher-Order Tensors, Based on the Riemannian Trust-Region Scheme. *SIAM Journal on Matrix Analysis and Applications*, 32(1):115–135, jan 2011.

[23] Mariya Ishteva, Lieven De Lathauwer, P.-A. Absil, and Sabine Van Huffel. Differential-geometric Newton method for the best rank-(R_1, R_2, R_3) approximation of tensors. *Numerical Algorithms*, 51(2):179–194, jun 2009.

[24] Andrew K.S. Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483–1510, 2006.

[25] Eamonn Keogh, Jessica Lin, and Ada Fu. HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, 2005.

[26] Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. Distance-based outliers: Algorithms and applications. *The VLDB Journal*, 8(3-4):237–253, 2000.

[27] Tamara G. Kolda and Brett W. Bader. Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500, aug 2009.

[28] Jongsun Lee, Hyun Soo Choi, Yongkweon Jeon, Yongsik Kwon, Donghun Lee, and Sungroh Yoon. Detecting System Anomalies in Multivariate Time Series with Information Transfer and Random Walk. In *Proceedings of 5th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, pages 71–80, 2018.

[29] Zeyan Li, Wenxiao Chen, and Dan Pei. Robust and Unsupervised KPI Anomaly Detection Based on Conditional Variational Autoencoder. In *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, nov 2018.

[30] Chaoguang Lin, Qiuhan Zhu, Shunan Guo, Zhuochen Jin, Yu-Ru Lin, and Nan Cao. Anomaly detection in spatiotemporal data via regularized non-negative tensor analysis. *Data Mining and Knowledge Discovery*, 32(4):1056–1073, jul 2018.

[31] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *Proceedings of 2008 IEEE International Conference on Data Mining*, pages 413–422. IEEE, dec 2008.

[32] Stuart P Lloyd. Least Squares Quantization in PCM. In *IEEE Transactions Information Theory*, volume 28, pages 128–137, 1982.

[33] Junshui Ma and Simon Perkins. Time-series Novelty Detection Using One-class Support Vector Machines. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 1741–1745, 2003.

[34] J. MACQUEEN. SOME METHODS FOR CLASSIFICATION AND ANALYSIS OF MULTIVARIATE OBSERVATIONS. *Proceedings of 5th Berkeley Symp. Math. Stat.Prob.*, pages 281–297, 1967.

[35] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. In *ICML 2016 Anomaly Detection Workshop*, 2016.

[36] Kiyotaka Matsue and Mahito Sugiyama. Unsupervised Tensor based Feature Extraction and Outlier Detection for Multivariate Time Series. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, oct 2021.

[37] Jiangyuan Mei, Meizhu Liu, Yuan-Fang Wang, and Huijun Gao. Learning a Mahalanobis Distance-Based Dynamic Time Warping Measure for Multivariate Time Series Classification. *IEEE Transactions on Cybernetics*, 46(6):1363–1374, jun 2016.

[38] Mingyan Teng. Anomaly detection on time series. In *2010 IEEE International Conference on Progress in Informatics and Computing*, pages 603–608, dec 2010.

[39] Huida Qiu, Yan Liu, Niranjan A. Subrahmanya, and Weichang Li. Granger Causality for Time-Series Anomaly Detection. In *Proceedings of 12th International Conference on Data Mining*, pages 1074–1079, 2012.

[40] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 262–270, 2012.

[41] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. *ACM SIGMOD International Conference on Management of Data*, 29(2):427–438, jun 2000.

[42] N. N. R. Ranga Suri, Narasimha Murty M, and G. Athithan. *Outlier Detection: Techniques and Applications*, volume 155 of *Intelligent Systems Reference Library*. Springer, 2019.

[43] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, 13(7):1443–1471, jul 2001.

[44] Lynne Seymour, Peter J. Brockwell, and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer, 2016.

[45] Jong-Ho Shin and Hong-Bae Jun. On condition based maintenance policy. *Journal of Computational Design and Engineering*, 2(2):119–127, 2015.

[46] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network.

In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2828–2837. ACM, jul 2019.

[47] Mahito Sugiyama and Karsten M. Borgwardt. Rapid distance-based outlier detection via sampling. *Advances in Neural Information Processing Systems*, pages 1–9, 2013.

[48] Naoya Takeishi and Takehisa Yairi. Anomaly detection from multivariate time-series with sparse representation. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pages 2651–2656, 2014.

[49] Sean J Taylor and Benjamin Letham. Forecasting at Scale. *PeerJ*, 2017.

[50] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, sep 1966.

[51] Hongzhi Wang, Mohamed Jaward Bah, and Mohamed Hammad. Progress in Outlier Detection Techniques: A Survey. *IEEE Access*, 7:107964–108000, 2019.

[52] Mohammed J. Zaki and Wagner Meira Jr. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, 2013.

[53] Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Zhang. Deep Structured Energy Based Models for Anomaly Detection. *Proceedings of 33rd International Conference on Machine Learning*, 3:1742–1751, may 2016.

[54] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V. Chawla. A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:1409–1416, jul 2019.

[55] Xing Zhang, Gongjian Wen, and Wei Dai. A Tensor Decomposition-Based Anomaly Detection Algorithm for Hyperspectral Image. *IEEE Transactions on Geoscience and Remote Sensing*, 54(10):5801–5820, oct 2016.

[56] Chong Zhou and Randy C. Paffenroth. Anomaly Detection with Robust Deep Autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674, aug 2017.

[57] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. In *Proceedings of 6th International Conference on Learning Representations*, pages 1–19, 2018.