5-23-2016

Time Dependent Kernel Density Estimation: A New Parameter Estimation Algorithm, Applications in Time Series Classification and Clustering

Xing Wang
University of South Florida, wx6807@gmail.com

Time Dependent Kernel Density Estimation: A New Parameter Estimation Algorithm, Applications in Time Series Classification and Clustering

by

Xing Wang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Mathematics and Statistics
College of Arts and Sciences
University of South Florida

Date of Approval:
May 20, 2016

Keywords: Feature Selection, Artificial Neural Networks, Random Forests,
Self-organizing Maps

Dedication

This doctoral dissertation is dedicated to my mother, my father and my husband.

Acknowledgments

I am really grateful that God could give me this chance to study at USF. I would like to thank everyone who helped and inspired me during the past three years. I would like to express my deepest gratitude to my advisor, the Distinguished University Professor Chris P. Tsokos. His diligence, encouragement, positive thinking and concerns for students have greatly inspired me. His constant assistance and advices throughout the entire process perpetuated my interest in research.

I am also truly grateful for Prof. Kandethody Ramachandran, Prof. Dan Shen and Prof. Lu Lu for their time and kind assistance during my dissertation research. Furthermore, I would also like to thank the Department of Mathematics and Statistics and Walt Disney Company- Visitation Research group for providing financial assistance to me for pursuing my doctoral degree.

Many thanks for all my friends in the ifound fellowship, the professors who have offered great classes in the Department of Mathematics & Statistics and Industrial & Management Systems Engineering, and my friends including Shuang Na, Pubudu, Pulahinge Hansa S. Rodrigo, Bhikhari Tharu and all the other fellow graduate students for their support during the past three years.

Table of Contents

List of Tables

List of Figures

Abstract

The Time Dependent Kernel Density Estimation (TDKDE) developed by Harvey & Oryshchenko (2012) is a kernel density estimation adjusted by the Exponentially Weighted Moving Average (EWMA) weighting scheme. The Maximum Likelihood Estimation (MLE) procedure for estimating the parameters proposed by Harvey & Oryshchenko (2012) is easy to apply but has two inherent problems. In this study, we evaluate the performances of the probability density estimation in terms of the uniformity of Probability Integral Transforms (PITs) on various kernel functions combined with different preset numbers. Furthermore, we develop a new estimation algorithm which can be conducted using Artificial Neural Networks to eliminate the inherent problems with the MLE method and to improve the estimation performance as well.

Based on the new estimation algorithm, we develop the TDKDE-based Random Forests time series classification algorithm which is significantly superior to the commonly used statistical feature-based Random Forests method as well as the Kernel Density Estimation (KDE)-based Random Forests approach.

Furthermore, the proposed TDKDE-based Self-organizing Map (SOM) clustering algorithm is demonstrated to be superior to the widely used Discrete-Wavelet-Transform (DWT)-based SOM method in terms of the Adjusted Rand Index (ARI).

1 Introduction

The subject study consists of four chapters related to the Time Dependent Kernel Density Estimation and an overview of the dissertation is given below.

1.1 Comments on Maximum Likelihood Estimation Methods for Time Dependent Kernel Density Estimation

Kernel Density Estimation (KDE) method is an important nonparametric procedure to estimate the probability density function which needs only mild assumptions [1]. The exponentially weighted moving average (EWMA) filter is a widely used scheme for weighting current and past observations by discounting older observations in an exponential manner. It works by giving more weight to the recent observations and less weight to the past observations. [2]

If the probability density estimation is thought to vary with time, it would be reasonable to apply the EWMA weighting scheme to adjust the traditional kernel density estimation. The Time Dependent Kernel Density Estimation (TDKDE) developed by Harvey and Oryshchenko (2012) [22] is such an estimation adjusted by the EWMA weighting scheme, and it is an appealing methodology to estimate the time dependent probability density function (PDF), or the corresponding cumulative distribution function (CDF).

However, as discussed by Perez (2012) [5], the estimates of the bandwidth and the discount parameter vary with the preset number of observations in the Maximum Likelihood Estimation procedure. In our study, we would like to extend the research of Perez by evaluating the performance of the density estimation on various kernel functions combined with different preset numbers.

1.2 Parameter Estimation of Time Dependent Kernel Density Using Artificial Neural Networks

Harvey and Oryshchenko (2012) proposed the Time Dependent Kernel Density Estimation which is an appealing methodology to estimate the time dependent probability density function. In order to obtain the PDF or CDF of the Time Dependent Kernel Density, the estimates of the bandwidth and the discount parameter need to be obtained. However, the Maximum Likelihood Estimation procedure proposed by Harvey and Oryshchenko (2012) has two inherent problems: the estimates of the two parameters vary with the preset number of observations and for the bounded support kernel functions, the likelihood function might need to be adjusted before using this method.

Consequently, we would like to develop a new approach, a supervised learning method which can be conducted using Artificial Neural Networks, to eliminate the above mentioned problems. Moreover, this study confirms that our new approach improves the performance of the estimates in terms of the uniformity of Probability Integral Transforms (PITs) as well.

1.3 Time-Dependent-Kernel-Density-Based Time Series Classification

Time series classification, which maps time series data into predefined classes [14], is one of the most appealing domains of data mining due to the abundance of its application areas.

Selecting an appropriate representation of the time series is critical to the quality of time series classification algorithms. If a time series is represented with a set of derived properties, such as mean, variance or quantiles, and the classification algorithm is on the basis of these derived properties, then this approach is defined as the feature-based classification.

In this study, a new time series classification algorithm with the Time Dependent Kernel Density Estimates (TDKDE) as the feature is developed to improve the performance of the existing statistical feature-based classification approach proposed by Nanopoulos et al. (2001)[18]. The performance evaluation is going to be illustrated using twenty datasets from the UCR Time Series Classification Archive, and the out-of-bag (OOB) error is used as the criterion to demonstrate the excellent performance of our proposed method.

1.4 Time-Dependent-Kernel-Density-Based Time Series Clustering

Time series clustering works by mining the underlying structure in an unlabeled time series dataset to organize data into similar groups [33]. Clustering the time series is particularly advantageous because labels or targets are not needed in this technique, which means that it does not rely on time-consuming annotation of the data [51]. Time series clustering has extracted significant attention in the last few decades, including the area of anomaly detection, intrusion detection, process control, and character recognition [52],[19],[53].

In the feature-based approach, the raw time-series is represented by a set of derived properties [17], namely features. Then, a clustering algorithm is applied to the extracted feature vectors [32]. Feature vectors usually have lower dimensions compared to the length of the raw data, and features can make distance calculations to be more meaningful and feasible [37].

In this study, we present a new feature-based time series clustering algorithm using the Time Dependent Kernel Density Estimation (TDKDE) as the time varying feature, and compare its performance to that of the widely used feature-based method: the Discrete Wavelet Transform (DWT) approach. We are going to demonstrated that our new feature-based Self-organizing Map approach is superior to the DWT-based approach evaluated on datasets from the UCR Time Series Data Mining Archive[28].

2 Comments on Maximum Likelihood Estimation for Time Dependent Kernel Density Estimation

2.1 Introduction

Density estimation provides vital foundations of data modeling, supervised and unsupervised learning. In time series analysis, density estimates can also be applied to address a wide variety of questions. For example, in the fields of economics and finance, the density estimation can provide information about the likelihood of recession, or the probability of stock returns exceeding a certain value.

Kernel Density Estimation (KDE) method is an important nonparametric procedure to estimate the probability density function which needs only mild assumptions [1]. The exponentially weighted moving average (EWMA) filter is a widely used scheme for weighting current and past observations by discounting older observations in an exponential manner. It works by giving more weight to the recent observations and less weight to the past observations. [2]

If the probability density estimation is thought to vary with time, it would be reasonable to apply the EWMA weighting scheme to adjust the traditional kernel density estimation. The Time Dependent Kernel Density Estimation (TDKDE) developed by Harvey and Oryshchenko (2012) [22] is such an estimation adjusted by the EWMA weighting scheme, and it is an appealing methodology to estimate the time dependent probability density function (PDF), or the corresponding cumulative distribution function (CDF).

However, as discussed by Perez (2012) [5], the estimates of the bandwidth and the discount parameter vary with the preset number of observations in the maximum likelihood estimation procedure. In our study, we would like to extend the research of Perez by evaluating the performance of the density estimation on various kernel functions combined with different preset numbers.

The rest of this study is organized as follows: section 2 illustrates the background and related methodologies. In section 3, the inherent problems with the Maximum Likelihood Estimation (MLE) procedure are presented in detail. The performance evaluations based on different kernel functions are included in section 4. Finally, section 5 concludes the study by summarizing the main contributions.

2.2 Background

The background of the Time Dependent Kernel Density Estimation (TDKDE), the specification and diagnostic checking tool, i.e. the Probability Integral Transform (PIT), will be presented in detail in the following subsections.

2.2.1 Kernel Density Estimation

Kernel Density Estimation (KDE) is a non-parametric method to estimate the probability density function of a random variable Y. Let $(y_1, y_2, ..., y_T)$ be an independent and identically distributed sample drawn from a distribution with an unknown probability density f. The traditional kernel estimator of $f(y)$ at point y can be expressed as:

$$\hat{f}_T(y) = \frac{1}{Th}\sum_{i=1}^{T} K(\frac{y - y_i}{h}), \tag{2.2.1}$$

where T is the number of observations, h is the bandwidth which determines the smoothness of the density estimate, and $K(\cdot)$ is the kernel, which is a non-negative function that integrates to one with mean zero. [6]

In general, any function satisfying the following conditions can be used as a kernel: $K(x) \geq 0$, $\int K(x)dx = 1$, $\int xK(x)dx = 0$, and $\int x^2K(x)dx < \infty$. [4] A number of classical kernel functions are listed in Table 2.1.

2.2.2 Time Dependent Kernel Density Estimation

The basic KDE can be modified to satisfy some particular needs for a specific research. When the density estimation is applied to a time series data, including both stationary and non-stationary time series, and is thought to vary with time, it would be reasonable to introduce a weighting scheme to adjust the traditional kernel density estimation. One of the widely used schemes is the exponentially weighted moving

Table 2.1: The Classical Kernel Functions

Kernel	Kernel Function $K(u)$
Gaussian	$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$
Epanechnikov	$K(u) = \frac{3}{4}(1-u^2)\,\mathbf{I}_{\{\lvert u\rvert \le 1\}}$
Uniform	$K(u) = \frac{1}{2}\,\mathbf{I}_{\{\lvert u\rvert \le 1\}}$
Triangular	$K(u) = (1-\lvert u\rvert)\,\mathbf{I}_{\{\lvert u\rvert \le 1\}}$
Triweight	$K(u) = \frac{35}{32}(1-u^2)^3\,\mathbf{I}_{\{\lvert u\rvert \le 1\}}$
Tricube	$K(u) = \frac{70}{81}(1-\lvert u\rvert^3)^3\,\mathbf{I}_{\{\lvert u\rvert \le 1\}}$
Biweight	$K(u) = \frac{15}{16}(1-u^2)^2\,\mathbf{I}_{\{\lvert u\rvert \le 1\}}$
Cosine	$K(u) = \frac{\pi}{4}\cos\left(\frac{\pi}{2}u\right)\mathbf{I}_{\{\lvert u\rvert \le 1\}}$
Silverman	$K(u) = \frac{1}{2}e^{-\frac{\lvert u\rvert}{\sqrt{2}}} \cdot \sin\left(\frac{\lvert u\rvert}{\sqrt{2}} + \frac{\pi}{4}\right)$

average (EWMA) filter, which works by discounting older observations in an exponentially decaying manner. The time dependent kernel density estimation developed by Harvey and Oryshchenko (2012) [22] is such an estimation adjusted by the EWMA weighting scheme which is given by:

$$\hat{f}_t(y) = \frac{1}{h}\sum_{i=1}^{t} K(\frac{y-y_i}{h})w_{t,i}, \;\; t = 1, ..., T. \tag{2.2.2}$$

In the general case, $\sum_{i=1}^{t} w_{t,i} = 1$. Here, $w_{t,i}$ is chosen to be:

$$w_{t,i} = \frac{1-\omega}{1-\omega^t}\omega^{t-i}, \;\; i = 1, ..., t, \tag{2.2.3}$$

which is the exponentially weighted moving average (EWMA) filter with the discount parameter ω. Consequently, the time dependent kernel density estimator for the corresponding CDF could be expressed as:

$$\hat{F}_t(y) = \sum_{i=1}^{t} H(\frac{y - y_i}{h}) \frac{1-\omega}{1-\omega^t} \omega^{t-i}, \tag{2.2.4}$$

where $H(\cdot)$ is the CDF form of the corresponding kernel $K(\cdot)$.

2.2.3 Probability Integral Transforms

An important tool of evaluating the adequacy of a density forecast is the Probability Integral Transform (PIT), which is the cumulative probability evaluated at the realized value of the target variable. If a sequence of density forecasts is correctly specified, then the corresponding PITs should be independently, uniformly distributed with the range [0,1] [8].

According to Harvey and Oryshchenko (2009) [7], the PIT of y_{t+1} , denoted by u_{t+1}, is expressed as:

$$u_{t+1} = \sum_{i=1}^{t} H(\frac{y_{t+1} - y_i}{h}) w_{t,i} \tag{2.2.5}$$

where $H(\cdot)$ is the CDF form of the corresponding kernel $K(\cdot)$ and $w_{t,i}$ was defined in the section 2.2.2.

2.3 Inherent Problems with the Maximum Likelihood Estimation Procedure

In order to obtain the time dependent PDF or CDF, feasible estimates of the bandwidth h and the discount parameter ω need to be obtained. As proposed by Harvey & Oryshchenko (2012), h and ω can be estimated by the Maximum Likelihood Method subject to $\omega \in (0, 1]$ and $h > 0$, with the log-likelihood function given in the equation below:

$$L(\omega, h) = \frac{1}{T-m} \sum_{t=m}^{T-1} ln \hat{f}_{t+1|t}(y_{t+1}) \tag{2.3.6}$$

$$= \frac{1}{T-m} \sum_{t=m}^{T-1} ln[\frac{1}{h} \sum_{i=1}^{t} K(\frac{y_{t+1} - y_i}{h}) w_{t,i}]. \tag{2.3.7}$$

Here m is a preset parameter to initialize the estimation procedure. The guideline to select an appropriate m was not given in the research of Harvey and Oryshchenko (2012). The general suggestion, as mentioned by Harvey and Oryshchenko (2012), was setting $m = 50$ or 100 if the sample size was big.

However, as illustrated by (Perez, 2012), the estimates of the bandwidth h and the discount parameter ω may vary with different selections of m. The results based on simulated data showed that the estimates for the discount parameter ω were almost stable, but estimates for the bandwidth h varied with different values of m. The detailed implementation and further discussion will be presented in the next section.

Another problem within the maximum likelihood function is that $\hat{f}_{t+1|t}(y_{t+1}) >$

0 should be satisfied for all $t = m, m+1, ..., T-1$ to make $ln\hat{f}_{t+1|t}(y_{t+1})$ meaningful. For unbounded support kernel functions such as Gaussian Kernel and Silverman Kernel, this condition can be guaranteed theoretically [22]. However, for the bounded kernel functions including Epanechnikov Kernel and Uniform Kernel, this condition cannot always be satisfied. The possible solution provided by Harvey and Oryshchenko (2012) was setting $\hat{f}_{t+1|t}(\cdot)$ equal to a very small positive number, but it may severely affect the estimates when large numbers of data points are adjusted in this way.

2.4 Performance Evaluation with Different Combinations of Kernel Functions and m

As illustrated by Perez (2012), the maximum likelihood estimates of ω and h vary with the preset parameter m. However, the performance in terms of the adequacy of a density forecast was not discussed in her study. In our study, we choose the Probability Integral Transform (PIT) as a criterion to evaluate the performance based on the data described below.

2.4.1 Dataset

To illustrate the problems with the Maximum Likelihood Estimation (MLE) method, and to make the results comparable to those in the research of Harvey and Oryshchenko (2012), we generate a series of $T = 1000$ observations (shown in Figure 2.1) from the same distribution as the filtered stock data in the research of Harvey and Oryshchenko

(2012) , which is an $MA(1) - GARCH(1,1)$ model with 7 degrees of freedom of the $t-$distribution for errors. To be specific, the $MA(1) - GARCH(1,1)$ model [9] is expressed as:

$$\begin{aligned} y_t &= \mu + \theta\varepsilon_{t-1} + \varepsilon_t \\ \varepsilon_t &= z_t\sigma_t \\ \sqrt{\frac{\nu}{\nu-2}}z_t &\sim t_\nu \\ \sigma_t &= \sqrt{\alpha_0 + \alpha_1\varepsilon_{t-1}^2 + \beta_1\sigma_{t-1}^2}. \end{aligned}$$

Here, the parameters are chosen as $\theta = 0.2102, \alpha_1 = 0.0979, \beta_1 = 0.9010, \nu = 7$, in order to be consistent with the model of the filtered stock data.

2.4.2 Performance Comparisons with Different Combinations of Kernel Functions and m

As introduced in section 2.2.3, if a sequence of density forecasts is correctly specified, then the corresponding PITs should be uniformly distributed with a range [0,1]. A commonly used method to compare a sample with a reference probability distribution is the Kolmogorov-Smirnov test (KS test). In this study, we employed the p-value of KS test to evaluate uniformity of the PITs. For the bounded support kernel functions, $\hat{f}_{t+1|t}(\cdot)$ might need to be adjusted to make this condition $\hat{f}_{t+1|t}(y_{t+1}) > 0$ satisfied, so that the estimation performance might not be comparable to that of the unbounded

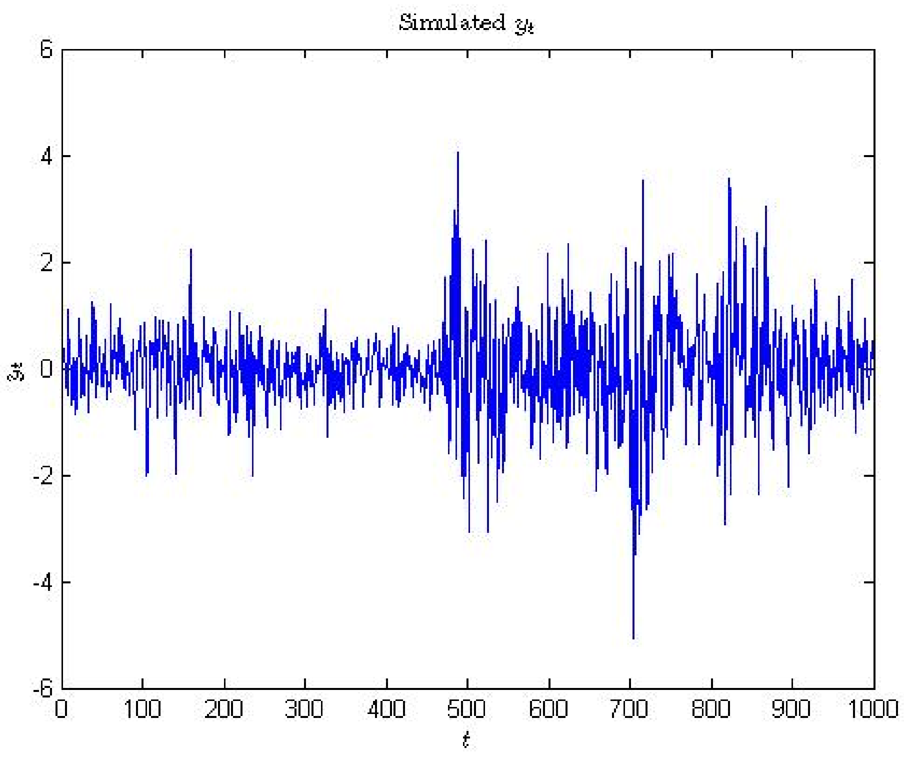


Figure 2.1: 1000 Observations Generated from $MA(1)-GARCH(1,1)$ Model with Student-7 Errors and Parameters: $\theta = 0.2102, \alpha_1 = 0.0979, \beta_1 = 0.9010, \nu = 7$.

support kernel functions. In this study, the performance of the two unbounded support kernel function, i.e. Gaussian Kernel and Silverman Kernel are compared, and the results are presented in Table 2.2 and Figure 2.2.

As indicated from Figure 2.2, for both Gaussian Kernel and Silverman Kernel, as m increases from 1 to 100, $\hat{h}$ shows a slightly increasing trend and $\hat{\omega}$ has a trend of decreasing. When Gaussian Kernel is employed, the p-value is decreasing while m is increasing, which indicates that the PITs have a trend to deviate from the uniform distribution. There is no obvious trend for the p-value when Silverman Kernel is used, and since the p-value varies within the range of $(0.3, 0.6)$, which means that there is no significant evidence to say that the PITs deviate from the uniform distribution. In terms of the criterion of uniformity of PITs, the Silverman Kernel has a better performance with regard to this dataset.

Table 2.2: Performance Comparisons in Terms of the Uniformity of the PITs Measured by the P-value of KS Test

	Gaussian Kernel			**Silverman Kernel**		
m	$\hat{h}$	$\hat{\omega}$	KS Test P-value	$\hat{h}$	$\hat{\omega}$	KS Test P-value
1	0.4232	0.9580	0.0865	0.3149	0.9508	0.5646
10	0.4195	0.9585	0.0865	0.3100	0.9508	0.5807
20	0.4219	0.9581	0.0865	0.3633	0.9055	0.3994
30	0.4233	0.9576	0.0865	0.3636	0.9052	0.3994
40	0.4246	0.9572	0.0865	0.3634	0.9052	0.3994
50	0.4266	0.9569	0.0868	0.3636	0.9051	0.3994
60	0.4299	0.9567	0.0814	0.3640	0.9052	0.3860
70	0.4318	0.9563	0.0762	0.3642	0.9047	0.3994
80	0.4349	0.9561	0.0713	0.3646	0.9046	0.3994
90	0.4365	0.9560	0.0713	0.3644	0.9047	0.3994
100	0.4778	0.9307	0.0330	0.3647	0.9045	0.4797

2.5 Contributions

In this study, we discussed two inherent problems with the existing Maximum Likelihood Estimation procedure of estimating the parameters in the Time Dependent

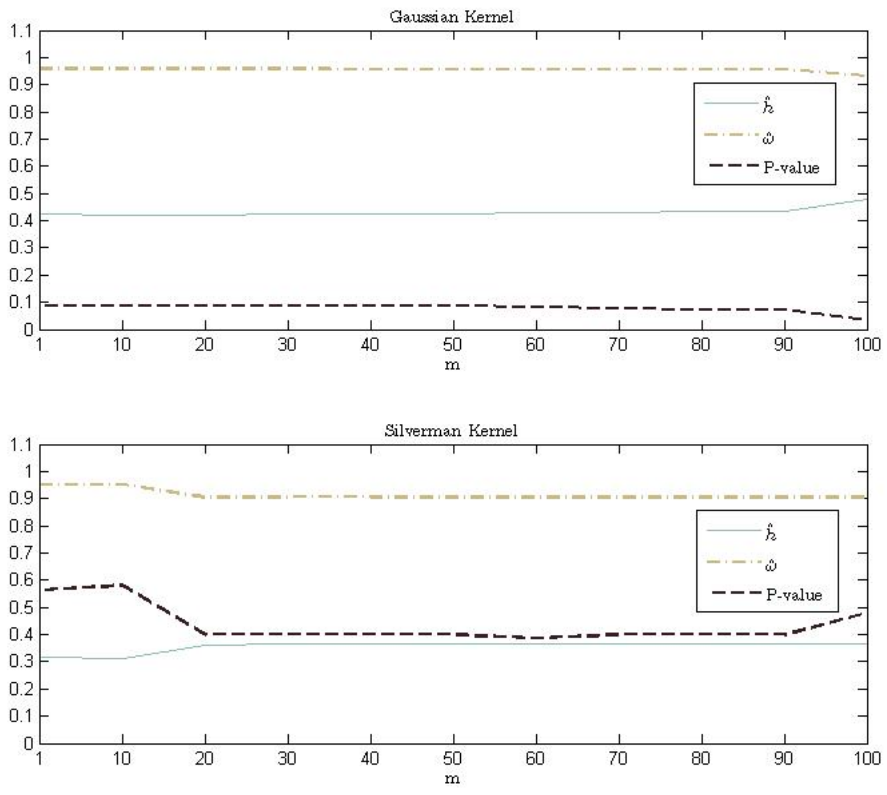


Figure 2.2: Variations of $\hat{h}$, $\hat{\omega}$, and P-value for KS Test with Different m

Kernel Density Estimation, i.e., the estimates vary with the preset parameter m, and the likelihood functions may need to be adjusted with the bounded-support-kernel functions. We extended the research of Perez (2012) by evaluating the performance of the probability density estimation for various kernel functions combined with different preset numbers. In terms of the criterion of uniformity of Probability Integral Transforms (PITs), we have found that the Silverman Kernel has a better performance than the commonly used Gaussian Kernel with regard to the simulated filtered stock dataset.

3 Parameter Estimation of Time Dependent Kernel Density Using Artificial Neural Networks

3.1 Introduction

The time dependent kernel density developed by Harvey and Oryshchenko (2012) is a kernel density estimation adjusted by the exponentially weighted moving average (EWMA) weighting scheme. It is an appealing methodology that can be easily applied to estimate the time dependent probability density function (PDF), or the corresponding cumulative distribution function (CDF).

As proposed by Harvey and Oryshchenko (2012), the bandwidth and the discount parameter can be estimated using the Maximum Likelihood Estimation procedure. However, as illustrated by Perez (2012), the estimates of the two parameters vary with the preset number of observations.[5].

Consequently, we would like to develop a new approach, a supervised learning method which can be conducted using Artificial Neural Networks, to eliminate this problem caused by the preset number. Moreover, this study confirms that our new approach improves the performance of the estimates as well.

The rest of this study is organized as follows: section 2 illustrates the background and related methodologies. In section 3, our proposed approach that could eliminate the problem caused by the preset number will be illustrated. Section 4 will carry out the diagnostic checking to compare the performance of our new approach with that of the MLE method. Finally, section 5 concludes the study by summarizing the main contributions.

3.2 Background and Related Methodologies

The background of the inherent problems with the Maximum Likelihood Estimation method and the Artificial Neural Networks that we are going to use in our new algorithm are discussed in the following subsections.

3.2.1 Inherent Problems with the Maximum Likelihood Estimation Procedure

As proposed by Harvey and Oryshchenko (2012), h and ω can be estimated by the Maximum Likelihood Method subject to $\omega \in (0, 1]$ and $h > 0$, with the log-likelihood function given by:

$$L(\omega, h) = \frac{1}{T-m} \sum_{t=m}^{T-1} ln \hat{f}_{t+1|t}(y_{t+1}) \tag{3.2.1}$$

$$= \frac{1}{T-m} \sum_{t=m}^{T-1} ln[\frac{1}{h} \sum_{i=1}^{t} K(\frac{y_{t+1} - y_i}{h}) w_{t,i}]. \tag{3.2.2}$$

Here m is a preset parameter to initialize the estimation procedure. The guideline to select an appropriate m was not given in the research of Harvey and Oryshchenko (2012). The general suggestion, as mentioned by Harvey and Oryshchenko (2012), was setting $m = 50$ or 100 if the sample size was big.

However, as illustrated by Perez (2012), the estimates of the bandwidth h and the discount parameter ω may vary with different selections of m. The results based on simulated data showed that the estimates for the discount parameter ω were almost stable, but estimates for the bandwidth h varied with different values of m. The detailed implementation and further discussion will be presented in the next section.

Another problem within the maximum likelihood function is that $\hat{f}_{t+1|t}(y_{t+1}) > 0$ should be satisfied for all $t = m, m+1, ..., T-1$ to make $ln\hat{f}_{t+1|t}(y_{t+1})$ meaningful. For unbounded support kernel functions such as Gaussian Kernel and Silverman Kernel, this condition can be guaranteed theoretically [22]. However, for the bounded Kernel functions including Epanechnikov Kernel and Uniform Kernel, this condition cannot always be satisfied. The possible solution provided by Harvey and Oryshchenko (2012) was setting $\hat{f}_{t+1|t}(\cdot)$ equal to a very small positive number, but it may severely affect the estimates when large numbers of data points are adjusted in this way.

3.2.2 Artificial Neural Networks

In machine learning, Artificial Neural Networks (ANNs) are nonlinear models motivated by the physiological architecture of the nervous system which can be trained to learn to approximate functions of complex non-linear systems that usually depend on a large number of inputs [10]. ANNs are typically organized in layers: the input layer, hidden layer and output layer, which consist of several neurons. Each neuron in a layer is connected to adjacent layers by the weights. Data are presented to the network via the input layer, which are multiplied by weights, and then go through the activation function of the neuron in one or more hidden layers [12]. The function in the output layer computes the output of the artificial neuron [13]. The structure of the typical ANNs and an artificial neuron are shown in Figure 3.1 and Figure 3.2 below.

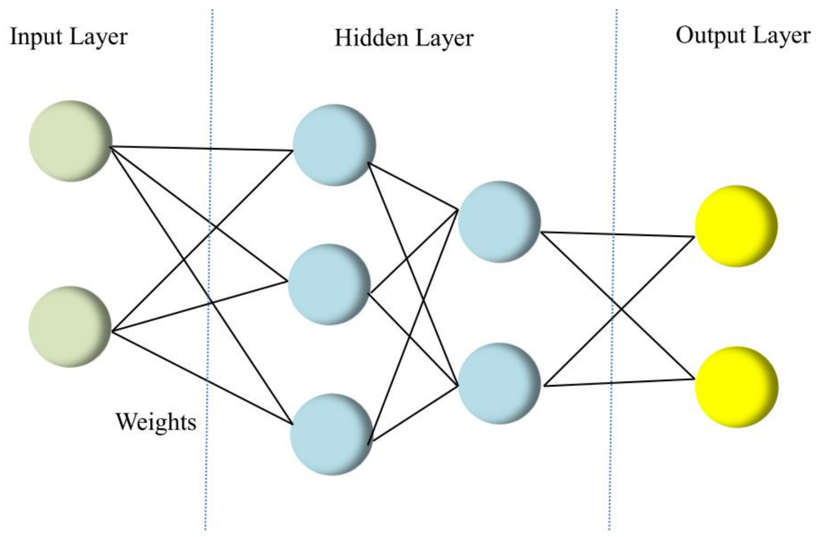


Figure 3.1: A Typical Structure of an ANN

Back-propagation (BP) algorithm is the most frequently used, effective, and easy to learn model for multilayered networks. It is a supervised learning technique which is based on the gradient descent method that attempts to minimize the error of the network by moving down the gradient of the error curve. Levenberg-Marquardt

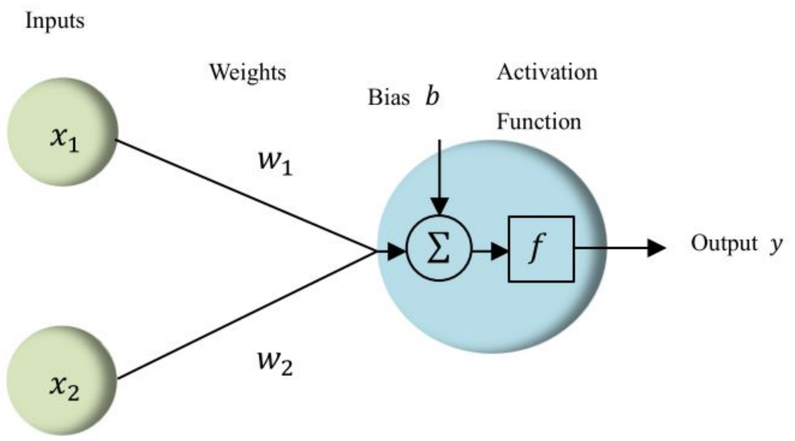


Figure 3.2: The Typical Artificial Neuron in ANNs

algorithm is one of the fastest back-propagation algorithm which works well for training small and medium sized networks and patterns [11]. This algorithm is applied in our study, because while providing a numerical solution to the problem of minimizing a nonlinear function, it has a speed advantage in the computation as well.

3.3 A Proposed Method for Kernel Density Estimation Using Neural Networks

In order to eliminate the volatility of the estimates, and to improve performance of the density estimation evaluated by Probability Integral Transforms, we develop a new approach–a supervised learning method which can be conducted using Artificial Neural Networks.

3.3.1 The New Estimation Method Using Artificial Neural Networks

The idea of this new estimation procedure came from the fact that the CDF of a random variable Y should follow the uniform distribution $U(0,1)$. Thus, when the estimated $CDF(\hat{F}_t(y))$ is as close as possible to the empirical CDF of the standard uniform distribution, the corresponding estimates of the parameters (ω, h) should be the optimal estimates.

Consequently, the estimation procedure could be turned into a supervised learning task which can be conducted using Artificial Neural Networks to reduce the computation time significantly.

We will train the network to map the estimated CDF to the CDF of the standard uniform distribution. Ideally, we would like to see $\hat{F}_{kt}(y) = U_t$, $(t = 1, ..., T)$, where $\hat{F}_{kt}(y)$ is the time dependent kernel estimator for the CDF evaluated at y_t corresponding to the pair (ω_k, h_k), and U_t is the t^{th} observation of the CDF for the discrete standard uniform distribution. One measurement of the deviation of $\hat{F}_{kt}(y)$ from U_t is the squared difference between $\hat{F}_{kt}(y)$ and U_t, *i.e.* $E_k = \sum_{t=1}^{T}[\hat{F}_{kt}(y) - U_t]^2$. Consequently, the optimal pair of parameters, (ω^*, h^*), is the pair corresponding to the minimal E_k with constraints $0 < \omega < 1$ and $h > 0$. The key steps of our algorithm are developed as follows, and a schematic diagram of the flow of our algorithm for estimating h and ω is given by Figure 3.3.

Algorithm 1 Parameter Estimation Algorithm

```
1: procedure Parameter Estimation Algorithm
2:     Obtain the initial bandwidth $h_0$ and $\omega_0$ .
3:     Generate $K$ pairs of $h$ and $\omega : (\omega_k, h_k), k = 1, 2, ..., K$, where $h \in [0.5h_0, 1.5h_0]$, and
       $\omega \in [0.5\omega_0, 1.5\omega_0]$.
4:     Generate $T$ points: $U_t = t/T, t = 1, 2, ..., T$, to represent the CDF of the discrete
       standard uniform distribution.
5:     Calculate $\hat{F}_{kt}(y)$ according to each pair of $(\omega_k, h_k)$ where
           $$\hat{F}_{kt}(y) = \sum_{i=1}^{t} H(\frac{y_t - y_i}{h_k}) \frac{1 - \omega_k}{1 - \omega_k^t} \omega_k^{t-i}$$
       and $k = 1, 2, ..., K$. Sort $\hat{F}_{kt}(y)$ such that: $\hat{F}_{k1}(y) \leq \hat{F}_{k2}(y) \leq ... \leq \hat{F}_{kT}(y)$.
6:     Calculate the squared difference between $\hat{F}_{kt}(y)$ and $U_t$:
           $$E_k = \sum_{t=1}^{T} [\hat{F}_{kt}(y) - U_t]^2$$
       where $k = 1, 2, ..., K, t = 1, ..., T$.
7:     Train the Neural Network with $(\omega_k, h_k)$ as the input, and $E_k$ as the output to sub-
       stitute step 4 to step 6.
8:     Select the pair of $(\omega_k, h_k)$ corresponding to the minimal $E_k$ with constraints $0 < \omega <$
       $1$ and $h > 0$ to be the optimal pair of parameters: $(\omega^*, h^*)$.
9: end procedure
```

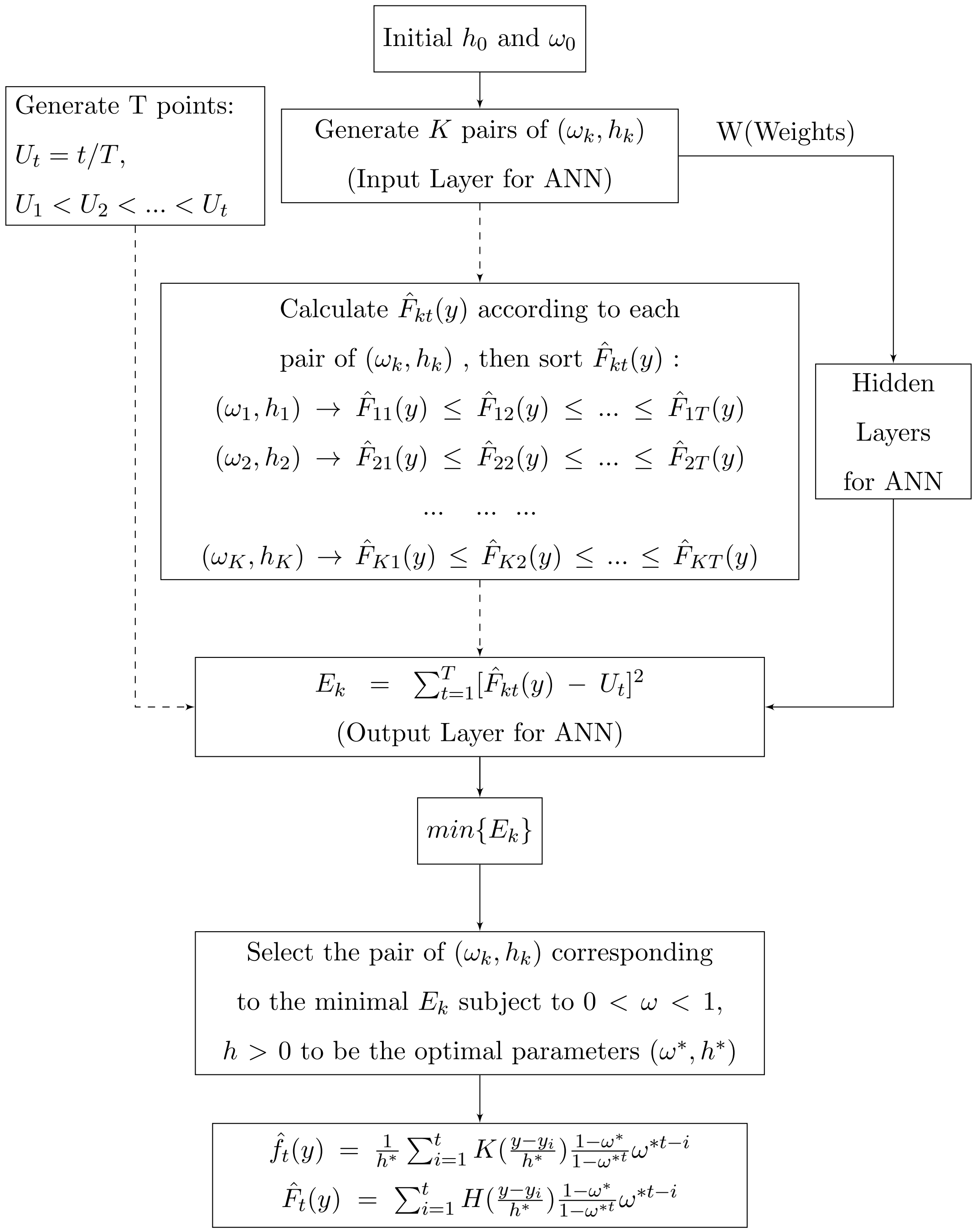

Initial h_0 and ω_0
Generate T points: $U_t = t/T$, $U_1 < U_2 < ... < U_t$
Generate K pairs of (ω_k, h_k) (Input Layer for ANN)
W(Weights)
Calculate $\hat{F}_{kt}(y)$ according to each pair of (ω_k, h_k) , then sort $\hat{F}_{kt}(y)$:
$(\omega_1, h_1) \rightarrow \hat{F}_{11}(y) \leq \hat{F}_{12}(y) \leq ... \leq \hat{F}_{1T}(y)$
$(\omega_2, h_2) \rightarrow \hat{F}_{21}(y) \leq \hat{F}_{22}(y) \leq ... \leq \hat{F}_{2T}(y)$
... ...
$(\omega_K, h_K) \rightarrow \hat{F}_{K1}(y) \leq \hat{F}_{K2}(y) \leq ... \leq \hat{F}_{KT}(y)$
Hidden Layers for ANN
$E_k = \sum_{t=1}^{T}[\hat{F}_{kt}(y) - U_t]^2$ (Output Layer for ANN)
$min\{E_k\}$
Select the pair of (ω_k, h_k) corresponding to the minimal E_k subject to $0 < \omega < 1$, $h > 0$ to be the optimal parameters (ω^*, h^*)
$\hat{f}_t(y) = \frac{1}{h^*}\sum_{i=1}^{t} K(\frac{y-y_i}{h^*})\frac{1-\omega^*}{1-\omega^{*t}}\omega^{*t-i}$
$\hat{F}_t(y) = \sum_{i=1}^{t} H(\frac{y-y_i}{h^*})\frac{1-\omega^*}{1-\omega^{*t}}\omega^{*t-i}$

Figure 3.3: Key Steps of the New Algorithm of Parameters Estimation.

3.3.2 Dataset

In order to make the results comparable to those in the research of Harvey and Oryshchenko (2012), we generate a series of $T = 1000$ observations from the same distribution as the filtered stock data in the research of Harvey and Oryshchenko (2012), which is an $MA(1) - GARCH(1,1)$ model with 7 degrees of freedom of the $t-$distribution for errors. To be specific, the $MA(1) - GARCH(1,1)$ model [9] is expressed as:

$$\begin{aligned} y_t &= \mu + \theta\varepsilon_{t-1} + \varepsilon_t \\ \varepsilon_t &= z_t\sigma_t \\ \sqrt{\frac{\nu}{\nu-2}}z_t &\sim t_\nu \\ \sigma_t = \sqrt{\alpha_0 + \alpha_1\varepsilon_{t-1}^2 + \beta_1\sigma_{t-1}^2}. \end{aligned}$$

Here, the parameters are chosen as $\theta = 0.2102, \alpha_1 = 0.0979, \beta_1 = 0.9010, \nu = 7$, in order to be consistent with the model of the filtered stock data.

3.3.3 The Selection of Initial Values h_0 and ω_0

The initial values, i.e. h_0 and ω_0, could have essential influence on the estimates, so the proper selection of h_0 and ω_0 is necessary in our estimation process. In order to obtain a guidance for the proper choice of initial values, we may visualize the relationship between (ω_k, h_k) and the corresponding E_k as our first step. In order to obtain relevant stable estimates, K is chosen to be 500 in this case. For the purpose of comparison with the MLE method, $\hat{F}_{kt}(y)$ is calculated using Gaussian Kernel and Silverman Kernel respectively. The visualization is based on the Gaussian Kernel for the purpose of illustration. The 3-D plot (Figure 3.4) gives us a clear trend for the

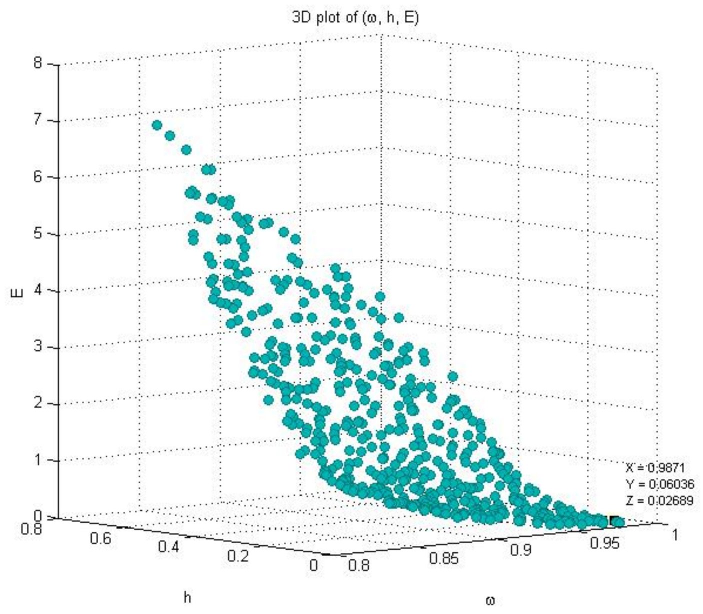


Figure 3.4: 3-D Plot of (ω, h, E) to Visualize the Relationship between Parameters ω, h and the Squared Error E.

relationship between parameters ω, h and the squared error E. In general, E decreases as ω increases and h decreases. The trend is more obvious in a 2-D view which is shown in Figure 3.5 and Figure 3.6. With constrains $0 < \omega < 1$ and $h > 0$, the global minimal E_k could be achieved in the region defined by our initial values (i.e. $h \in [0.5h_0, 1.5h_0]$ and $\omega \in [0.5\omega_0, 1.5\omega_0]$), which indicates that our choice of the pair of initial values $(\omega_0, h_0) = (0.9, 0.3)$ is appropriate. In the case of our simulated data, the optimal pair of estimates for ω and h are 0.9871 and 0.0604 respectively, with a

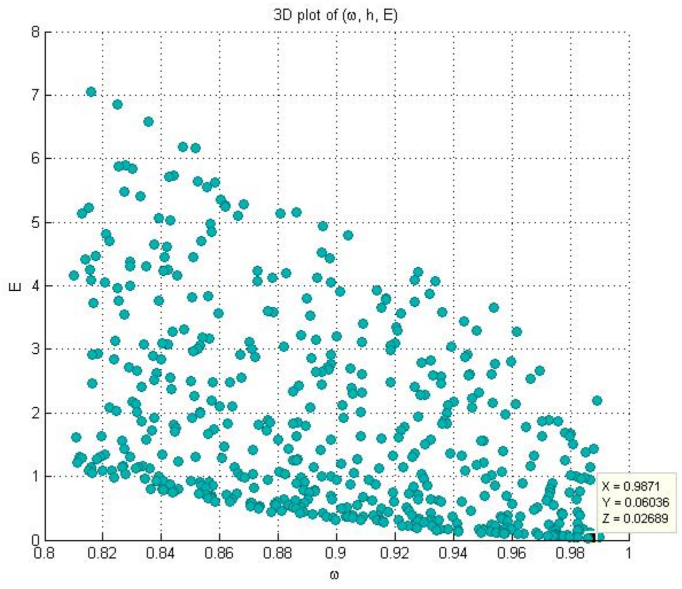


Figure 3.5: In General, E Decreases with Increasing ω, and E Reaches its Minimum in $\omega \in [0.5\omega_0, 1.5\omega_0](\omega_0 = 0.9)$.

squared error $E = 0.0269$, which is marked by the black dot in Figure 3.4, Figure 3.5 and Figure 3.6.

3.3.4 Analysis of the Network Response

The inputs of the neural network are the K pairs of (ω_k, h_k) simulated following the guidance in the previous section, and the outputs are K squared errors E_k $(k = 1, 2, ..., K, K = 500$ in this case). The Levenberg - Marquardt algorithm, which works well for training small and medium sized networks and patterns (Yu and Wilamowski, 2011), is applied for the Artificial Neural Networks (designed with two hidden layers) in this study. This algorithm provides a numerical solution to the problem of minimizing a nonlinear function, and it has a speed advantage in the computation as

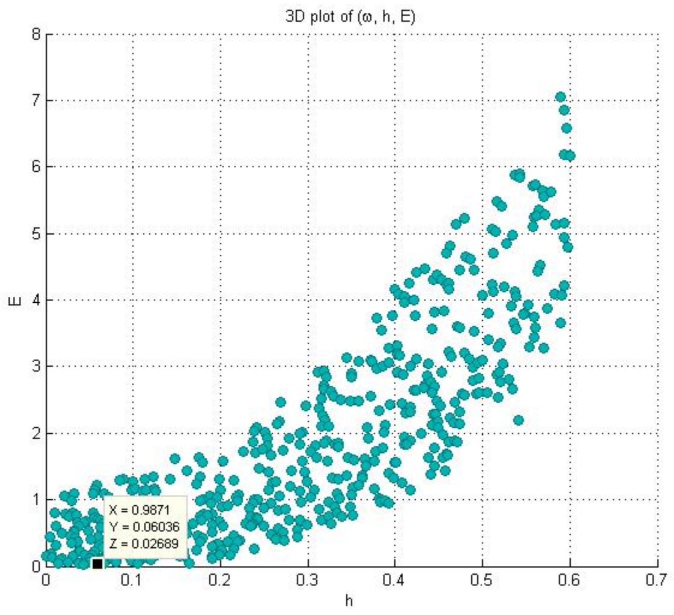


Figure 3.6: In General, E Decreases with Decreasing h , and E Reaches its Minimum in $h \in [0.5h_0, 1.5h_0](h_0 = 0.3)$.

well.

The network performance can be evaluated by errors, which are the differences between the network outputs and the corresponding targets. One way to perform this analysis is to conduct a regression analysis between the network outputs and the targets on the training, validation, and test sets, respectively.

Figure 3.7 illustrates the network performance by regression plots. The network outputs are plotted versus the targets in each plot. The 45-degree dashed line represents the perfect fit, which indicates that outputs are equal to targets. The best linear fit is shown by the solid line. In this case, the best linear fit line almost overlaps with the perfect fit line in each plot, which indicates a very good fit. The R-value, which is a correlation coefficient between the outputs and targets, is close to 1 for the

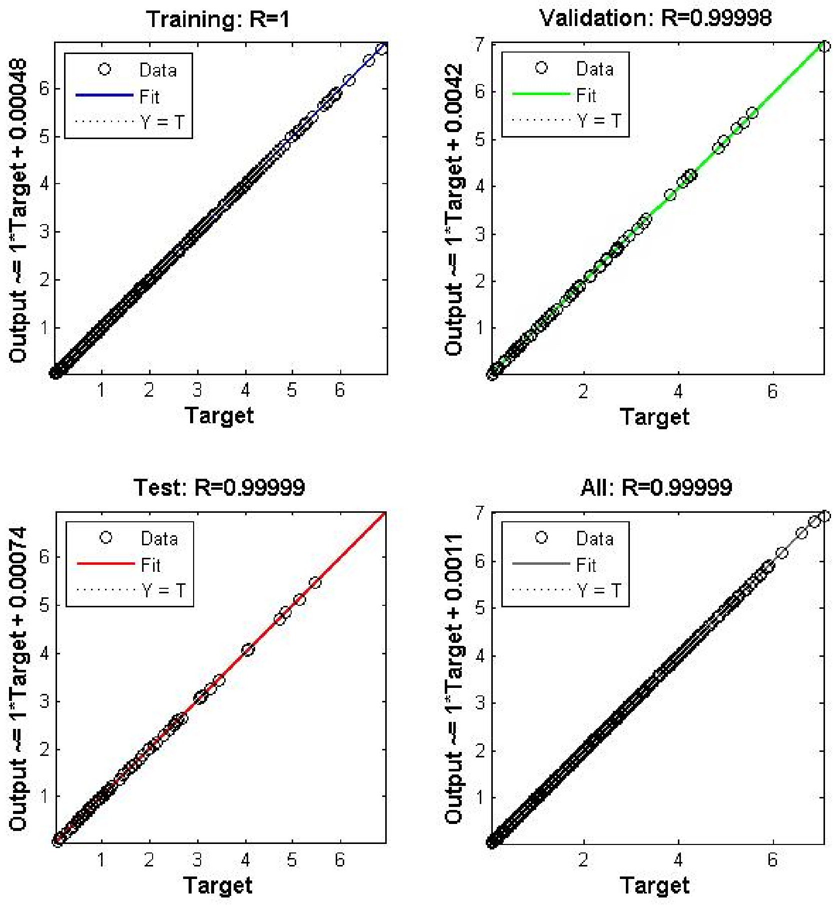


Figure 3.7: Regression Plots of the Network Outputs with Respect to Targets for the Training Set, Validation Set, and Testing Set.

training, validation, and test sets respectively, which yields the conclusion of a perfect fit as well.

3.4 Diagnostic Checking

An important tool of evaluating the adequacy of a density forecast is the probability integral transform (PIT), which is the cumulative probability evaluated at the realized value of the target variable. If the density forecast is correctly specified, then the corresponding PITs should be uniformly distributed.

According to Harvey and Oryshchenko, the PIT of y_{t+1} , denoted by u_{t+1}, is expressed as:

$$u_{t+1} = \sum_{i=1}^{t} H(\frac{y_{t+1} - y_i}{h})w_{t,i}, \tag{3.4.3}$$

where $H(\cdot)$ is the CDF form of the corresponding kernel $K(\cdot)$.

To compare the estimates of our new approach with the MLE method, we need to select some values for the preset parameter m to initialize the estimation procedure. Here, the values of m are chosen from 1 to 100, and the kernel $K(\cdot)$ is calculated using the Gaussian Kernel in all cases for the purpose of comparison. As can be seen from Table 3.1, the estimates of h and ω vary with the value of m for the MLE method.

The Kolmogorov-Smirnov test (KS test) could be employed to test if the PITs follow the standard uniform distribution or not. The p-values for KS test reported in Table 3.1 indicate that, the distribution of PITs generated from our new technique is much closer to the standard uniform distribution compared to that of the MLE method.

The same conclusion can be obtained from the graph as well. For the purpose of illustration, we only visualize the PITs in the cases of our new method and $m = 50$, $m = 100$ in the MLE method. If PITs are uniformly distributed, their empirical CDF should be close to a 45-degree line. In Figure 3.8, 3.9 and 3.10, the green line represents the empirical CDF of PITs, and the black line stands for the CDF of the standard uniform distribution, which is a 45-degree line. We can see the

Table 3.1: Performance Comparisons: The MLE Approach v.s. the Proposed ANN Method (with Gaussian Kernel)

Maximum Likelihood Method	$\hat{h}$	$\hat{\omega}$	KS Test P-value
$m = 1$	0.4232	0.9580	0.0865
$m = 10$	0.4195	0.9585	0.0865
$m = 20$	0.4219	0.9581	0.0865
$m = 30$	0.4233	0.9576	0.0865
$m = 40$	0.4246	0.9572	0.0865
$m = 50$	0.4266	0.9569	0.0868
$m = 60$	0.4299	0.9567	0.0814
$m = 70$	0.4318	0.9563	0.0762
$m = 80$	0.4349	0.9561	0.0713
$m = 90$	0.4365	0.9560	0.0713
$m = 100$	0.4778	0.9307	0.0330
The Proposed Method	0.0604	0.9871	**0.9995**

obvious deviation between the empirical CDF of PITs and the CDF of the standard uniform distribution in Figure 3.8 and Figure 3.9, where the estimates are derived by MLE method. In contrast, the empirical CDF of PITs and the CDF of the standard uniform distribution are hardly distinguishable in Figure 3.10, in which the estimates are obtained from our Artificial Neural Networks approach.

The similar conclusions can be obtained using Silverman Kernel. Table 3.2 indicate that, the distribution of PITs generated from our new technique is also much closer to the standard uniform distribution compared to that of the MLE method

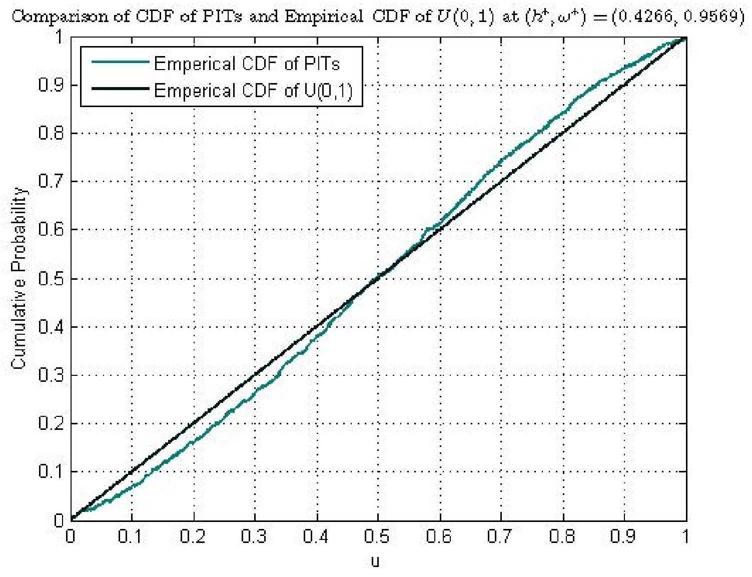


Figure 3.8: A Comparison of the Empirical CDF of PITs and CDF of $U(0,1)$ by $MLE(m = 50) : (h^*, \omega^*) = (0.4266, 0.9569)$. (KS test p-value = 0.0868)

using Silverman Kernel.

3.5 Financial Applications: NASDAQ Stock Returns

NASDAQ stock returns dataset can be obtained from Yahoo-Finance. The sample starts on June 15th 2004, and ends on May 13th 2016, which contains 3001 observations. We follow the same rule to calculate stock returns as mentioned in the research of Harvey and Oryshchenko (2012). If we denote the daily adjusted close price as y_t, then the returns can be computed as $\Delta ln(y_t)$, where adjusted close price is the close price adjusted for dividends and splits. We finally obtain 3000 data points of NASDAQ stock returns, and the y_t together with $\Delta ln(y_t)$ are presented in the Figure 3.11.

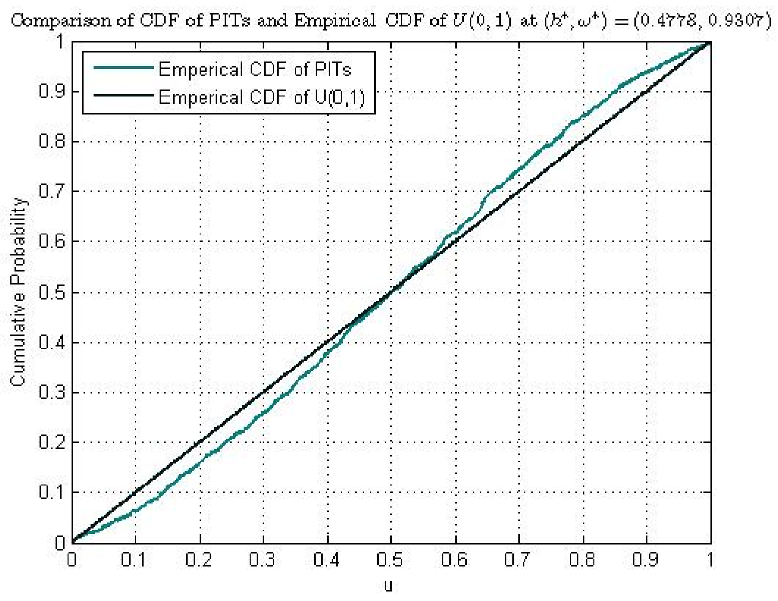


Figure 3.9: A Comparison of the Empirical CDF of PITs and CDF of $U(0,1)$ by $MLE(m = 100) : (h^*,\omega^*) = (0.4778, 0.9307)$. (KS test p-value =0.0330)

When Gaussian Kernel is applied, the estimates obtained by the new algorithm are $(h^*,\omega^*) = (0.0028, 0.9628)$, and the corresponding performance can be evaluated by the uniformity of PITs. In Figure 3.12, the empirical CDF is pretty close to a 45-degree line. Furthermore, the p-value for KS test is 0.5816, which indicate that the PITs are uniformly distributed.

3.6 Contributions

In this study, we have developed a new method to estimate the parameters in the Time Dependent Kernel Density Estimation (TDKDE). The new estimation procedure is able to solve the two problems with the existing Maximum Likelihood Estimation method, which means that it eliminates the problem caused by the preset parameter

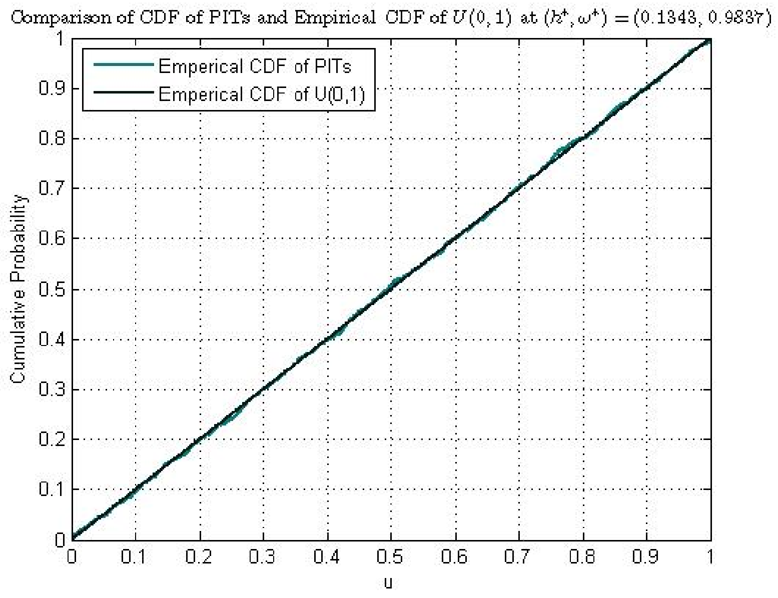


Figure 3.10: A Comparison of the Empirical CDF of PITs and CDF of $U(0,1)$ by the Proposed ANN Method: $(h^*, \omega^*) = (0.0604, 0.9871)$. (KS Test P-value = 0.9995)

m, and makes it possible to combine with the bounded-support-kernel functions without adjusting the likelihood functions. More importantly, our new method improves the performance of the estimates evaluated by the uniformity of Probability Integral Transforms (PITs), and it can be applied to the both stationary and non-stationary time series including the real data-NASDAQ stock returns-with excellent results.

Table 3.2: Performance Comparisons: The MLE Approach v.s. the Proposed ANN Method (with Silverman Kernel)

Maximum Likelihood Method	$\hat{h}$	$\hat{\omega}$	KS Test P-value
$m = 1$	0.3149	0.9508	0.5646
$m = 10$	0.3100	0.9508	0.5807
$m = 20$	0.3633	0.9055	0.3994
$m = 30$	0.3636	0.9052	0.3994
$m = 40$	0.3634	0.9052	0.3994
$m = 50$	0.3636	0.9051	0.3994
$m = 60$	0.3640	0.9052	0.3860
$m = 70$	0.3642	0.9047	0.3994
$m = 80$	0.3646	0.9046	.3994
$m = 90$	0.3644	0.9047	0.3994
$m = 100$	0.3647	0.9045	0.4797
The Proposed Method	0.0762	0.9567	**0.9824**

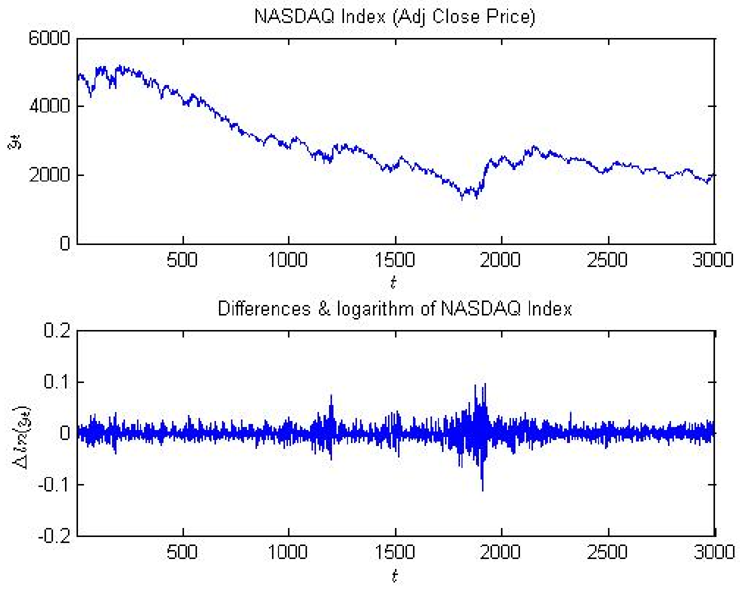


Figure 3.11: The NASDAQ Stock Returns

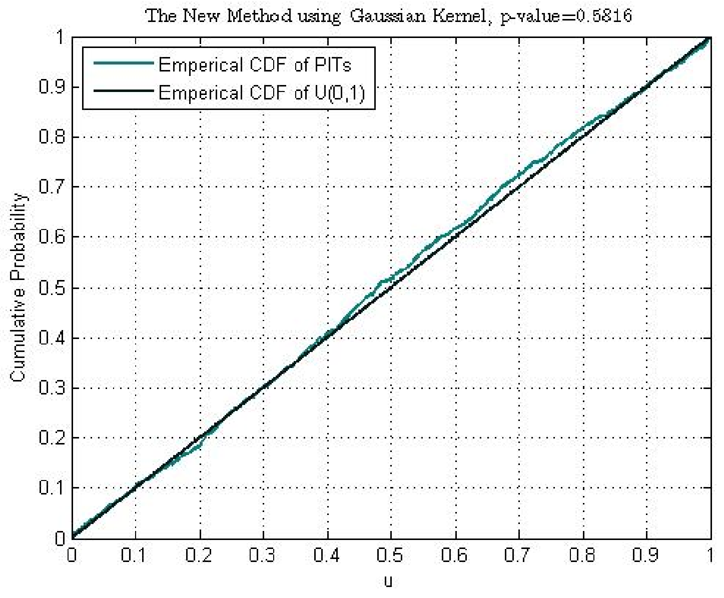


Figure 3.12: A Comparison of the Empirical CDF of PITs and CDF of $U(0,1)$ by the Proposed ANN Method: $(h^*, \omega^*) = (0.0028, 0.9628)$. (KS test P-value = 0.5816)

4 Time-Dependent-Kernel-Density-Based Time Series Classification

4.1 Introduction

Time series data, which represents a sequence of values collected at different time points, is common in a wide range of fields. The classification of time series is particularly beneficial for the areas including healthcare, finance, economics, signal processing and video retrieval.

4.1.1 Time Series Classification

Time series classification, which maps time series data into predefined classes [14], is one of the most appealing domains of data mining due to the abundance of its application areas. For example, in the domain of healthcare, the electrocardiogram (ECG) signals, which represent the cardiac function[15], can be classified as normal and abnormal signals. In this way, the hidden information conveyed by the ECG signal plays a significant role in the investigation of cardiac disorders. As another example, images can be converted to "pseudo time series" data to ease classification tasks. In the leaf classification problem, the image of a leaf can be converted into a time series by measuring the local angle at each point of the image contour[16]. Figure 4.1, a video screen shot created by Eamonn Keogh and Chotirat Ann Ratanamahatana, shows the conversion from a shape to a "time series".

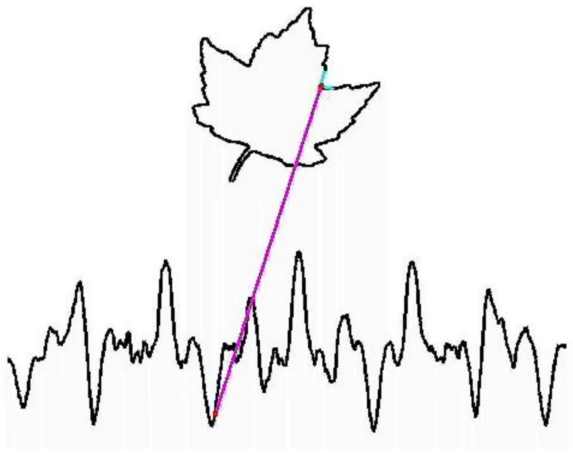


Figure 4.1: A Shape of a Leaf can be Converted into a One Dimensional "Pseudo Time Series".

4.1.2 Representation of Time Series

Selecting an appropriate representation of the time series is critical to the quality of time series classification algorithms. As illustrated by Fulcher (2014)[17], the existing time series classification methods can be categorized as the instance-based classification and the feature-based classification. For the time series in the time-domain form, the distance between any two time series is a function of the difference between the time-ordered observations in these two sequences. In this case, a new time series can be classified by matching it to the similar instance of time series with a known class. This method of classification is defined as the instance-based classification. Alternatively, if a time series is represented with a set of derived properties, such as mean, variance or quantiles, and the classification algorithm is on the basis of these derived properties, then this approach is defined as the feature-based classification.

Compared to the instance-based classification method, the feature-based classification approach has the advantages of reducing the dimensionality, keeping the most important information while removing noises.

To choose appropriate features is one of the hardest problems of the feature-based classification approach. Some feature-based representations of time series have been explored in previous studies. For example, Nanopoulos et al. (2001) [18] developed a method using the mean, standard deviation, skewness, and kurtosis of the first and second order of the data to classify the time series. Wang et al.(2006) [19] proposed a set of features with a variety of measures which included periodicity, serial correlation, measures of trend, seasonality, self-similarity, chaos, nonlinearity, etc. Deng et al.(2013)[20] used measures of mean, spread, along with the trend in local time-series intervals as features to classify time series.

Since in the feature-based classification methods mentioned above, most features are directly related to properties of the distribution, we would like to use the adjusted probability density itself, Time Dependent Kernel Density Estimation, as a feature, and compare the classification performance based on this new feature with some of the existing algorithms.

4.1.3 Structure of the Study

In this study, a new time series classification algorithm with the Time Dependent Kernel Density Estimates (TDKDE) as the feature is developed to improve the performance of the existing statistical feature-based classification proposed by Nanopoulos et al. (2001)[18].

The rest of this study is organized as follows: section 2 illustrates the background and related methodologies. In section 3, the proposed algorithm on the basis of TDKDE is presented in detail. The dataset description and the performance comparison are included in section 4 and section 5. Finally, section 6 concludes the study by summarizing the main contributions.

4.2 Background and Related Methodologies

In this study, we introduce the Time Dependent Kernel Density Estimates as a new feature with the Random Forest classification algorithm built on it. The background and related methodologies are explained in detail in the following subsections.

4.2.1 Time Dependent Kernel Density Estimates (TDKDE)

Kernel Density Estimation (KDE) is a non-parametric method to estimate the probability density function of a random variable Y. Let $(y_1, y_2, ..., y_T)$ be an independent and identically distributed sample drawn from a distribution with an unknown probability density f [38]. The traditional kernel estimator of $f(y)$ at point y can be expressed as:

$$\tilde{f}_T(y) = \frac{1}{Th}\sum_{j=1}^{T} K(\frac{y-y_j}{h}). \tag{4.2.1}$$

Here T is the number of observations, h is the bandwidth, and $K(\cdot)$ is the kernel, which is a non-negative function that integrates to one with mean zero.

The Time Dependent Kernel Density Estimates (TDKDE) can be seen as a combination of the Kernel Density Estimation(KDE) and the time factor. When the density estimation is thought to vary with time, it could be reasonable to introduce a weighting scheme to adjust the traditional kernel density estimation. One of the widely used schemes is the exponentially weighted moving average (EWMA) filter, which works by discounting older observations in an exponentially decaying manner. The time dependent kernel density estimation developed by Harvey and Oryshchenko (2012) [22]is such an estimation adjusted by the EWMA weighting scheme which is given by:

$$\tilde{f}_t(y) = \frac{1}{h}\sum_{j=1}^{t} K(\frac{y-y_j}{h})w_{t,j},\ t = 1, ..., T. \tag{4.2.2}$$

In the general case, $\sum_{j=1}^{t} w_{t,j} = 1$. In the study of Harvey and Oryshchenko (2012), $w_{t,j}$ is chosen to be:

$$w_{t,j} = \frac{1-\omega}{1-\omega^t}\omega^{t-j},\ j = 1, ..., t, \tag{4.2.3}$$

which is the EWMA filter with the discount parameter ω. Consequently, the TDKDE for the corresponding CDF can be expressed as:

$$\tilde{F}_t(y) = \sum_{j=1}^{t} H(\frac{y-y_j}{h})\frac{1-\omega}{1-\omega^t}\omega^{t-j}, \tag{4.2.4}$$

where $H(\cdot)$ is the CDF form of the corresponding kernel $K(\cdot)$.

In order to obtain the TDKDE, two parameters, the bandwidth h and the discount parameter ω, need to be estimated. The new estimation procedure has been developed and discussed in Chapter 3.

4.2.2 Feature Extraction

The selection of an appropriate representation of the time series is one of the most critical steps for time series classification. Two common ways are the instance-based representation and the feature-based representation.

The feature-based representation is commonly applied to reduce the dimensionality, as well as discard noises. The feature-based time series classification technique works by transforming the data in the time domain into feature set before handing it to the classification algorithms [23]. Two existing feature extraction methods and the new method will be illustrated as follows.

Nanopoulos et al.(2001)[18] used two types of statistical features in his study: the first order and the second order features. The first order features are directly generated from the raw data y_t $(t = 1, 2, ..., T)$, whereas the second order features are extracted from the differences of nearby values Δy_t $(t = 1, 2, ..., T)$. Here $\Delta y_t = y_{t+D} - y_t$ $(1 \leq t \leq T - D)$, where D is the time distance between the two points. Four statistics are computed for each order: the mean μ, standard deviation σ, skewness S and kurtosis K. For the i^{th} sample in the dataset with length T: $(y_{i1}, y_{i2}, ..., y_{iT})$, the corresponding statistical feature vector $\boldsymbol{f}_{1i}$ can be expressed by

$$\boldsymbol{f}_{1i} = (\mu_i^{(1)}, \sigma_i^{(1)}, S_i^{(1)}, K_i^{(1)}, \mu_i^{(2)}, \sigma_i^{(2)}, S_i^{(2)}, K_i^{(2)}),$$

where $\mu_i^{(1)}, \sigma_i^{(1)}, S_i^{(1)}$ and $K_i^{(1)}$ are the first order statistics, and $\mu_i^{(2)}, \sigma_i^{(2)}, S_i^{(2)}$ and $K_i^{(2)}$ are the second order statistics.

The Kernel Density Estimates(KDE) were also used as features for time series clustering based on forecast densities (Alonso et al.(2006)[24]). We would like to modify the mentioned method and combine KDE with a classification algorithm for the purpose of comparison. For the i^{th} sample in the dataset with length T: $(y_{i1}, y_{i2}, ..., y_{iT})$, the corresponding statistical feature vector $\boldsymbol{f}_{2i}$ can be expressed by KDE at point $y_i^{(q)}, (q = 1, 2, ..., Q)$, i.e.

$$\boldsymbol{f_{2i}} = (\hat{f}_T(y_i^{(1)}), \hat{f}_T(y_i^{(2)}), ..., \hat{f}_T(y_i^{(Q)})),$$

where $\hat{f}_T(y_i^{(q)})$ is the KDE at point $y_i^{(q)}$.

To enhance the probability density-based feature extraction method, we can further replace the traditional KDE with Time Dependent Kernel Density Estimates (TDKDE), which was developed by Harvey and Oryshchenko (2012)[22], as illustrated in section 3.2.1. For the i^{th} sample in the dataset with length T: $(y_{i1}, y_{i2}, ..., y_{iT})$, the corresponding statistical feature vector $\boldsymbol{f}_{3i}$ can be expressed by TDKDE at point $y_i^{(q)}$, $(q = 1, 2, ..., Q)$, i.e.

$$\boldsymbol{f_{3i}} = (\tilde{f}_T(y_i^{(1)}), \tilde{f}_T(y_i^{(2)}), ..., \tilde{f}_T(y_i^{(Q)})),$$

where $\tilde{f}_T(y_i^{(q)})$ is the TDKDE at point $y_i^{(q)}$. We will focus on the new algorithm based on the TDKDE feature. Table 4.1 gives a brief summary of the three methods of feature extractions.

Table 4.1: Comparison of the Three Feature Extraction Methods

Feature extraction	**Feature Vector**
A) Statistical-based feature	$\boldsymbol{f}_1 = (\mu^{(1)}, \sigma^{(1)}, S^{(1)}, K^{(1)}, \mu^{(2)}, \sigma^{(2)}, S^{(2)}, K^{(2)})$
B) KDE-based feature	$\boldsymbol{f_2} = (\hat{f}_T(y^{(1)}), \hat{f}_T(y^{(2)}), ..., \hat{f}_T(y^{(Q)}))$
C) TDKDE-based feature	$\boldsymbol{f_3} = (\tilde{f}_T(y^{(1)}), \tilde{f}_T(y^{(2)}), ..., \tilde{f}_T(y^{(Q)}))$

4.2.3 Classification: Random Forests Algorithm

The common classification algorithms include Decision Trees, Random Forests, Support Vector Machines (SVMs), Neural Networks, and so on. The Random forests (RF) algorithm is a multi-class classifier method which has a high classification capability and enables high-speed learning and classification. Random forests are a group of unpruned classification or regression trees made from the bootstrap samples of the data. The final classification of an individual is determined by voting over all trees in the forest [25].

The key steps of RF algorithm are as follows. Firstly, randomly draw the same number of cases with the original data with replacement to form a subset, and repeat this process several times. Secondly, for each subset, grow an unpruned classification or regression tree. At each node of the tree, randomly sample a small group of attributable variables and the best split is calculated based on those selected attributable variables. Lastly, decide a final predicted outcome by combining the results over all trees (an average for the regression tree, a majority vote for the classification tree)[26]. The procedure of RF classification algorithm is illustrated in Figure 4.2.

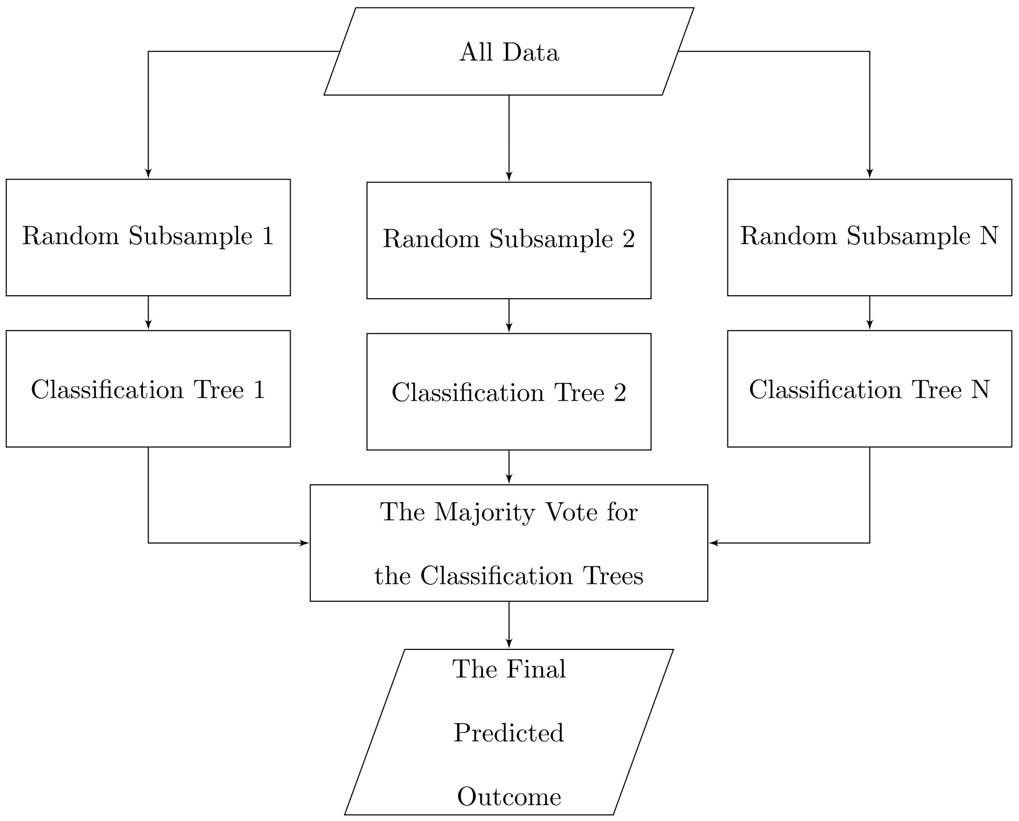


Figure 4.2: Key Steps of RF Classification Algorithm.

RF is chosen as the classifier in our study, because while having about the same accuracy as other algorithms, it is efficient for large datasets, and it does not overfit the data. In addition, the cross validation is not necessary for RF, because it generates an internal unbiased estimate of the true prediction error, which is called the out-of-bag (OOB) error. Even though the OOB error overestimates the true error in some cases ([27]), however, because the bias is low, one may simply report the OOB error as an expected upper bound to the actual prediction error.

4.3 Time Dependent Kernel Density-Based Classification Algorithm

The representation of our new feature in the time series classification and the comparative studies are presented in detail in the following subsections.

4.3.1 Feature Vector Representation

Since the method proposed by Nanopoulos et al.(2001) was based on the basic statistics of a distribution, a possible way to improve this method is to replace those statistics by the probability density, which usually contains more information about the distribution. Here the probability density can be estimated by the Time Dependent Kernel Density Estimates (TDKDE).

As illustrated in section 3.2, the Time Dependent Kernel Density Estimates

(TDKDE) at point y is expressed by

$$\tilde{f}_T(y) = \frac{1}{h}\sum_{j=1}^{T} K(\frac{y-y_j}{h})\frac{1-\omega}{1-\omega^T}\omega^{T-j},$$

where $K(\cdot)$ is the kernel, h is the bandwidth, ω is the discount parameter for the exponentially weighted moving average (EWMA) filter.

For the i^{th} sample in the dataset with length T: $(y_{i1}, y_{i2}, ..., y_{iT})$, the corresponding statistical feature vector $\boldsymbol{f}_{3i}$ can be expressed by TDKDE at point $y_i^{(q)}$, $(q = 1, 2, ..., Q)$, i.e.

$$\boldsymbol{f_{3i}} = (\tilde{f}_T(y_i^{(1)}), \tilde{f}_T(y_i^{(2)}), ..., \tilde{f}_T(y_i^{(Q)})).$$

Since the length of the TDKDE-based feature vector is Q, when $Q < T$, this feature vector can be a lower-dimensional representation of the raw data, which means it also works as a meaningful dimensionality reduction technique.

In order to obtain the appropriate TDKDE, one needs to estimate the parameters h and ω. In this study, the optimal pair of parameters (h^*, ω^*) can be estimated using the new estimation procedure developed in Chapter 3.

4.3.2 Classification

Once the TDKDE-based feature vectors are obtained, they are fed to a supervised learning classifier: the Random Forests(RF) algorithm. To compare the classification performances, the out-of-bag(OOB) error is used as a criterion of measuring the true prediction error. In the Breiman's (1996b) study of error estimates, the empirical evidence was given to show that the out-of-bag estimate is as accurate as using a testing set of the same size as the training set. Therefore, we combine the training set and the testing set of each original dataset in this study, because there is no need to separate a testing set to get an unbiased estimate of the prediction error (Breiman's (1996b)[25]).

4.3.3 New Algorithm and Comparative Studies

The key steps of the new algorithm that we have developed combined with the feature extraction and classification are shown in Algorithm 2 as stated below. The flowchart for the parameter estimation was presented in Figure 3.3 in Chapter 3.

Figure 4.3 illustrates the process of the proposed algorithm and the comparative study with other two existing feature-based time series classification algorithms. In Method A, the first and second order statistical features proposed by Nanopoulos et al.(2001) were illustrated in the section 4.2.2. The idea of Method B came from Alonso et al.(2006), where the KDE was used to provide estimates for the forecast densities in a time series clustering problem. In Method C, the feature vectors are obtained by the Time Dependent Kernel Density Estimation (TDKDE). The optimal parameters (ω^*, h^*) are estimated using the method developed in Chapter 3.

For the purpose of comparison, the Gaussian kernel is selected as the kernel function $K(\cdot)$ for the KDE-based algorithm (Method B in Figure 4.3) and TDKDE-based algorithm (Method C in Figure 4.3), where

$$K(\cdot) = \frac{1}{\sqrt{2\pi}} exp(-\frac{y^2}{2}).$$

The optimal bandwidth (Silverman (1986)[30]) is chosen as the bandwidth h in Method B as well as the initial bandwidth h_0 in the Method C. The practical estimation of the bandwidth is given by

$$h = \left(\frac{4\hat{\sigma}^5}{3n}\right)^{\frac{1}{5}} \approx 1.06\hat{\sigma}n^{-1/5},$$

where $\hat{\sigma}$ is the standard deviation of the sample. K can be any reasonable integer, and we have $K = 500$ in this study in order to provide relatively robust estimators.

4.4 Datasets

In order to compare the performance of these three feature-based algorithms, we would like to verify the results using multiple datasets, because in general, the results evaluated on multiple datasets should be more reliable than those evaluated on only one special dataset.

The UCR Time Series Data Mining Archive (Keogh et al. 2011[28]) has been widely used as a benchmark for evaluating the performance of time series classification or clustering algorithms. It contains multiple datasets that were gathered from diverse resources. The application domains vary broadly across the archive, including

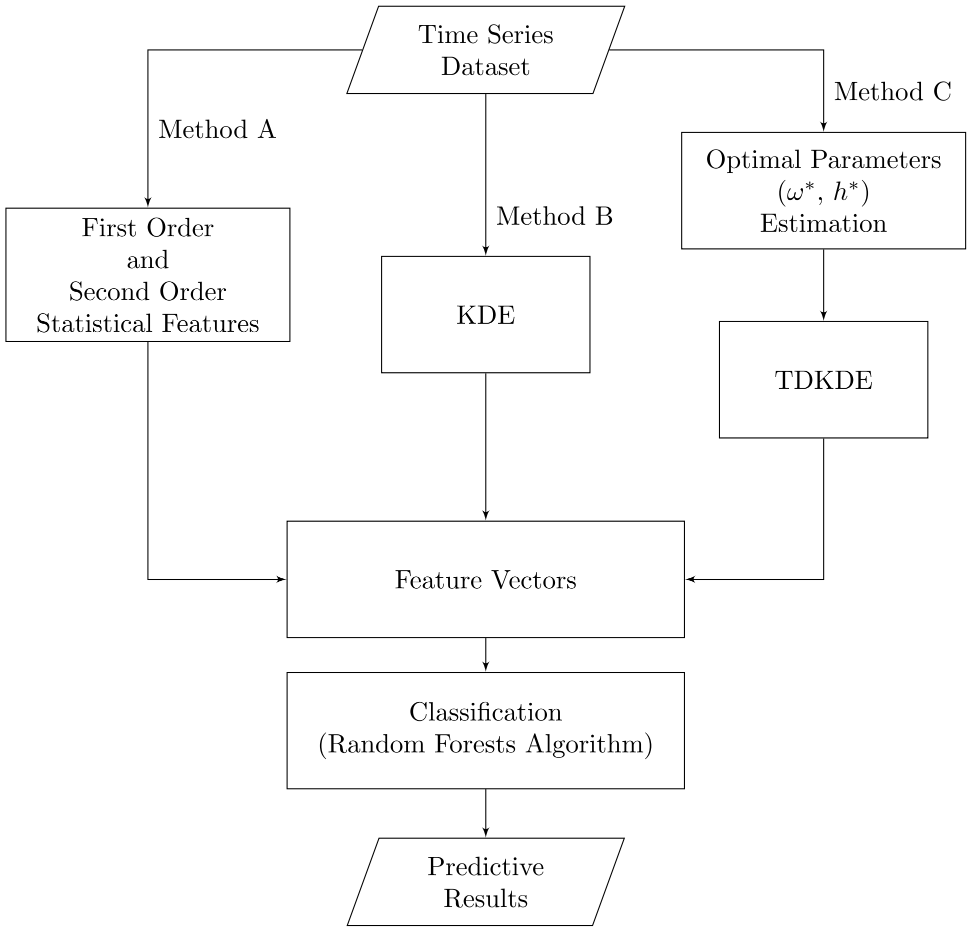

Time Series Dataset
Method A
Method B
Method C
First Order and Second Order Statistical Features
KDE
Optimal Parameters (ω^*, h^*) Estimation
TDKDE
Feature Vectors
Classification (Random Forests Algorithm)
Predictive Results

Figure 4.3: The Network of the Proposed Algorithm (Method C) and the Comparative Studies.

natural science, health science, social science, and so on. The number of classes, the length of time series data and the sample size also vary from one dataset to another. All the samples within the same dataset have equal length, and this property can ease the data prepossessing procedure for classification purposes. All the datasets in this data mining archive are labeled, univariate time series, and each dataset is split into a training set and a testing set, which is convenient for the performance comparison of supervised learning algorithms.

In our study, twenty datasets from the UCR Time Series Data Mining Archive are analyzed. Even though this archive has kept updating since 2011 to include more datasets, the original twenty datasets in this archive are able to provide sufficient information for comparative studies[17].

4.4.1 Classical Examples in the Archive

To have a better understanding of the datasets in the UCR Time Series Data Mining Archive, we select three classical datasets and illustrate each of them in detail.

(a) Gun Point Dataset

This dataset is transformed from the video surveillance. The dataset contains two classes: Gun-Draw and Point, each class has 100 samples. In the Gun-Draw class, the following hands motions have been recorded: drawing a replicate gun from a hip-mounted holster, pointing it at a target for a second, and returning the gun to the holster. In the Point class, the hands motions have been recorded when actors pretended drawing a gun. The time series are converted by tracking the centroid of the actors right hands in both the horizontal axis (X-axis) and the vertical axis (Y-axis)[16]. In this dataset, only the motion in the X-axis is used. The time series data from two classes are visualized in Figure 4.4.

(b) Face (Four)Dataset

The images might be converted into “pseudo time series” data to make the classification task easier. The Face (Four) dataset comes from a face retrieval problem where all the head profiles are converted into the “pseudo time series”. The side view photos

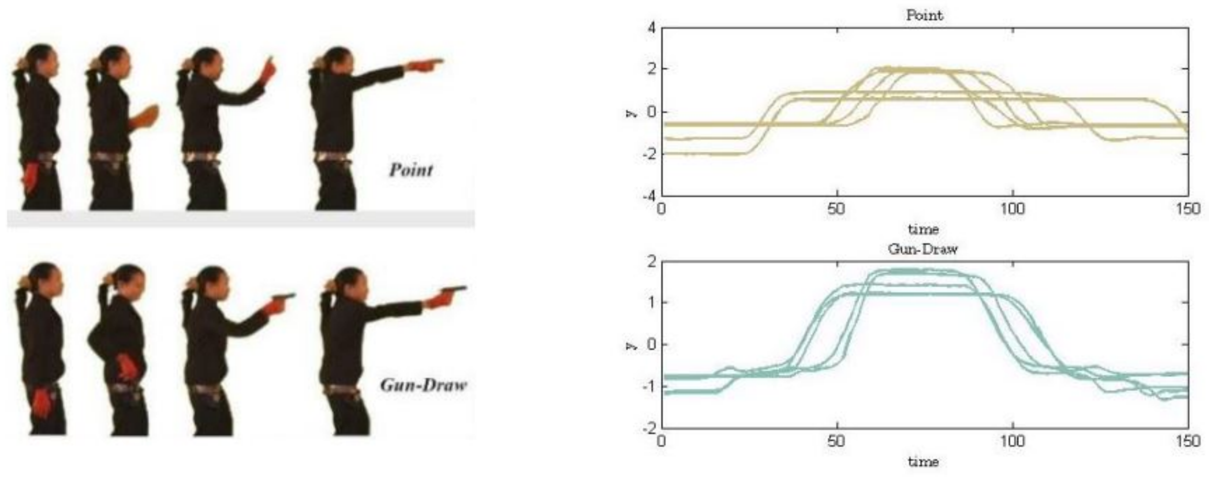


Figure 4.4: The Gun/Point Dataset with Two Classes (The Gun-Draw Class and the Point Class)

of four different individuals are taken with different facial expressions: talking, smiling, frowning, laughing, etc. The time series is converted by measuring a local angle at each point of the image contour, starting from the neck area of a head profile, as shown in Figure 4.5. The dataset contains 112 samples in total with 4 classes, i.e. 4 different individuals. Each time series has been normalized by subtracting its mean, and then divide by its standard deviation to have the mean of zero and the standard deviation of one. [16]

(c) ECG Dataset

The electrocardiogram (ECG) signals, which represent the cardiac function, are recordings of the electric waves being generated during cardiac activities. The ECG dataset contains 200 samples with 133 normal samples and 67 abnormal samples.[49] The standardized signals are visualized in Figure 4.6.

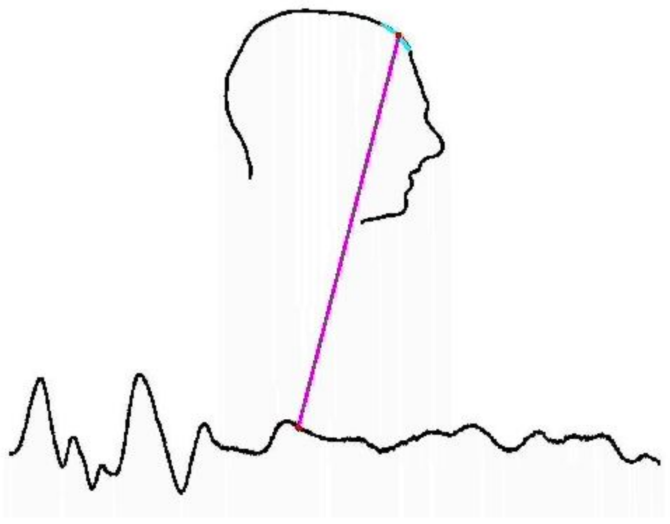


Figure 4.5: Face (Four) Dataset. The Side View of a Head Profile Might be Converted into a “Pseudo Time Series”

4.5 Classification Results

Table 4.2 shows the classification results for all the twenty datasets from the UCR Time Series Classification Archive. The OOB error rate from the Random Forests is used as a criterion for the performance comparison. The number of samples refers to the sample size of each dataset. In this study, the training set and the testing set in each original dataset are combined.

In order to make the algorithm comparable to each other, we choose the same number of points Q and the same kernel function (Gaussian kernel in this case) for the probability density estimates in both Method B and Method C. Here Q is proportional to the time series length T, which is set to be $Q = \frac{7}{8}T$ in our study. Since $Q < T$, both Method B and C can be taken as dimensionality reduction techniques. In addition, the optimal bandwidth is chosen as the bandwidth h in Method B and the initial

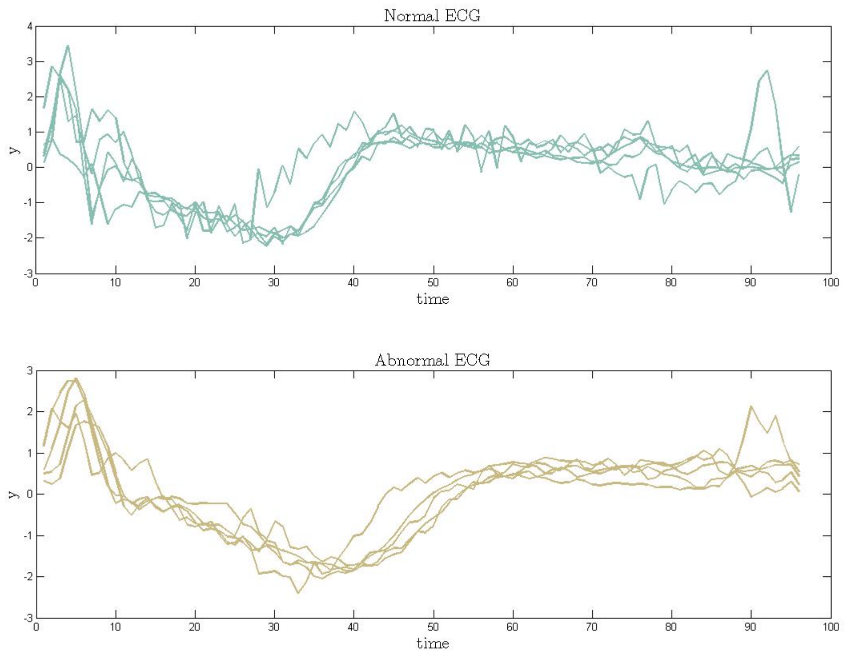


Figure 4.6: Standardized ECG Signals with 5 Samples from Each Class

bandwidth h_0 in Method C. For the parameters in the Random Forests, the number of trees is fixed to be 500, and the number of eligible splitters is chosen by the rule of thumb as $\sqrt{Q}$.

For each dataset, the best performance, i.e. the lowest OOB error rate, is denoted by the boldface number. As indicated from Table 4.2, the KDE-based RF approach (Method B) has the best performance on 1 out of 20 datasets and TDKDE-based RF approach (Method C) has the best performance on 19 out of 20 datasets.

A better illustration of performance comparisons can be shown in Figure 4.7, Figure 4.8 and Figure 4.9. These figures clearly visualize the pairwise error rates for any two algorithms. Take Figure 4.7 as an example, each dot in the graph represents a pair of OOB error rates of the KDE-based RF approach and the statistical feature-

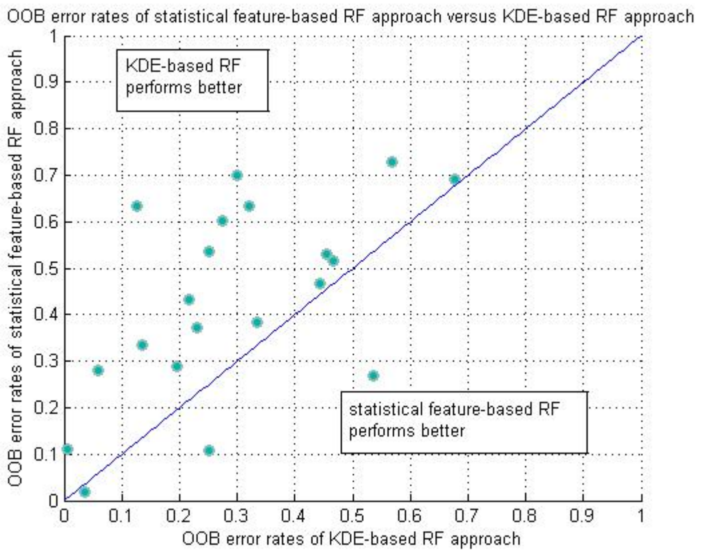


Figure 4.7: OOB Error Rates of Statistical-Feature-Based RF Approach Versus KDE-Based RF Approach.

based RF approach for a specific dataset. The dots on the 45 degree line indicate that these two algorithms have equal error rates on the corresponding datasets. The dots on the upper half of the graph indicate that the KDE-based RF approach outperforms the statistical feature-based RF approach, and vice versa.

As can be concluded from the figures, the KDE-based RF approach and the TDKDE-based RF approach significantly outperform the statistical feature-based RF approach; the TDKDE-based RF approach also significantly outperforms the KDE-based RF approach on these twenty datasets, with the odds 19:1.

4.6 Contributions

In the present study, we have developed a new feature-based time series classification algorithm, the Time Dependent Kernel Density Estimation (TDKDE) based Random

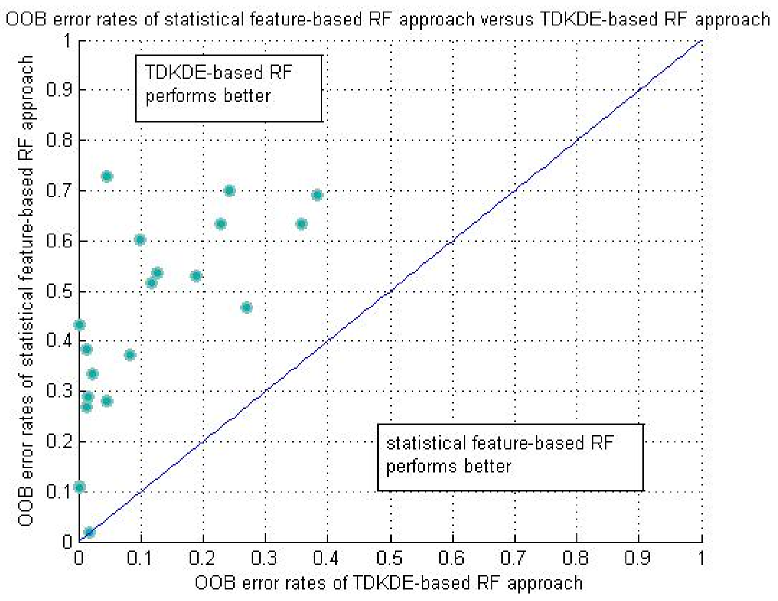


Figure 4.8: OOB Error Rates of Statistical-Feature-Based RF Approach Versus TDKDE-Based RF Approach.

Forests classification algorithm, that gives excellent results in classifying both stationary and non-stationary time dependent information (signals). The evaluation performance is illustrated using twenty datasets from the UCR Time Series Classification Archive. The analysis of the classification results verifies that our proposed method, the TDKDE-based Random Forests approach, is significantly superior to the commonly used statistical feature-based Random Forests method as well as KDE-based Random Forests approach in terms of the out-of-bag (OOB) errors. Furthermore, our new method is able to address a number of open real world problems.

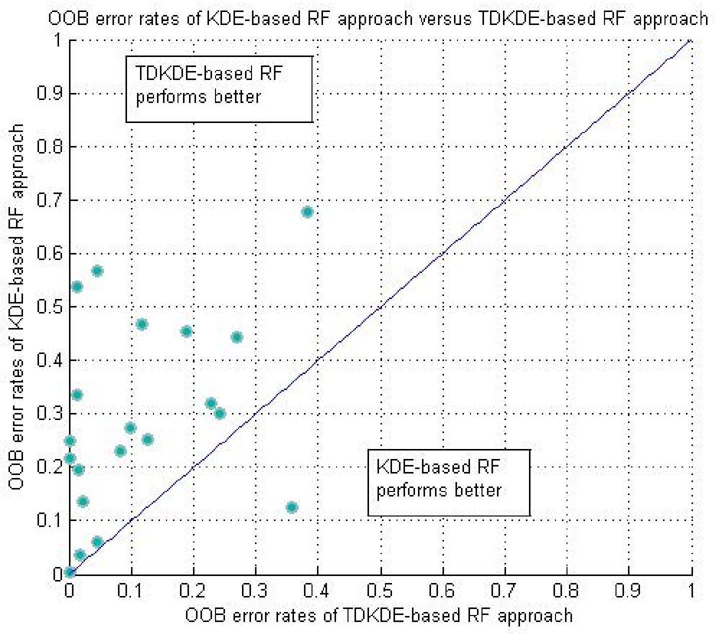


Figure 4.9: OOB Error Rate of KDE-Based RF Approach Versus TDKDE-Based RF Approach.

Algorithm 2 Time-Dependent-Kernel-Density-Based Time Series Classification

```
1: procedure TIME-DEPENDENT-KERNEL-DENSITY-BASED TIME SERIES CLASSIFICATION
2:     Obtain the initial bandwidth $h_0$ and $\omega_0$ .
3:     Generate $K$ pairs of $h$ and $\omega$ : $(\omega_k, h_k), k = 1, 2, ..., K$, where $h \in [0.5h_0, 1.5h_0]$, and $\omega \in [0.5\omega_0, 1.5\omega_0]$.
4:     Generate $T$ points: $U_t = t/T, t = 1, 2, ..., T$, to represent the CDF of the discrete standard uniform distribution.
5:     Calculate $\tilde{F}_{kt}(y)$ according to each pair of $(\omega_k, h_k)$ where
           $$\tilde{F}_{kt}(y) = \sum_{j=1}^{t} H(\frac{y_t - y_j}{h_k}) \frac{1 - \omega_k}{1 - \omega_k^t} \omega_k^{t-j}$$
       and $k = 1, 2, ..., K$. Sort $\tilde{F}_{kt}(y)$ such that: $\tilde{F}_{k1}(y) \leq \tilde{F}_{k2}(y) \leq ... \leq \tilde{F}_{kT}(y)$.
6:     Calculate the squared difference between $\tilde{F}_{kt}(y)$ and $U_t$:
           $$E_k = \sum_{t=1}^{T} [\tilde{F}_{kt}(y) - U_t]^2$$
       where $k = 1, 2, ..., K, t = 1, ..., T$.
7:     Select the pair of $(\omega_k, h_k)$ corresponding to the minimal $E_k$ with constraints $0 < \omega < 1$ and $h > 0$ to be the optimal pair of parameters: $(\omega^*, h^*)$.
8:     Obtain the Time Dependent Kernel Density Estimation (TDKDE) by
           $$\tilde{f}_T(y) = \frac{1}{h^*} \sum_{j=1}^{T} K(\frac{y - y_j}{h^*}) \frac{1 - \omega^*}{1 - \omega^{*T}} \omega^{*T-j}$$
       for each sample, and record the density estimate at a point $y^{(q)}, (q = 1, 2, ..., Q)$ as the density-based feature vector
           $$\boldsymbol{f_3} = (\tilde{f}_T(y^{(1)}), \tilde{f}_T(y^{(2)}), ..., \tilde{f}_T(y^{(Q)}))$$
       for each sample in the dataset.
9:     Classify the time series based on the TDKDE-based feature vectors using Random Forests
10: end procedure
```

Table 4.2: Classification OOB Error Rates of Statistical Feature-Based RF Approach (Method A), KDE-Based RF Approach (Method B) and the New Method: TDKDE-Based RF Approach (Method C).

Dataset	Number of Samples	Time Series Length	Number of Classes	Method A: OOB Error Rate	Method B: OOB Error Rate	New Method C: OOB Error Rate
synthetic_control	600	60	6	0.2700	0.5367	**0.0133**
Gun_Point	200	150	2	0.2800	0.0600	**0.0450**
CBF	930	128	3	0.3828	0.3355	**0.0118**
FaceAll	2250	131	14	0.6027	0.2747	**0.0982**
OSULeaf	442	427	6	0.4683	0.4434	**0.2692**
SwedishLeaf	1125	128	15	0.5360	0.2507	**0.1271**
50words	905	270	50	0.6906	0.6773	**0.3834**
Trace	200	275	4	0.1100	0.0050	**0.0000**
Two_Patterns	5000	128	4	0.7286	0.5676	**0.0460**
wafer	7164	152	2	0.0188	0.0354	**0.0174**
FaceFour	112	350	4	0.6339	**0.1250**	0.3571
Lightning-2	121	637	2	0.3719	0.2314	**0.0826**
Lightning-7	143	319	7	0.5315	0.4545	**0.1888**
ECG	200	96	2	0.2900	0.1950	**0.0150**
Adiac	781	176	37	0.7004	0.2996	**0.2420**
Yoga	3300	426	2	0.3336	0.1361	**0.0218**
Fish	350	463	7	0.6343	0.3200	**0.2286**
Beef	60	470	5	0.5167	0.4667	**0.1167**
Coffee	56	286	2	0.1071	0.2500	**0.0000**
OliveOil	60	570	4	0.4333	0.2167	**0.0000**

5 Time-Dependent-Kernel-Density-Based Time Series Clustering

5.1 Introduction

Unsupervised learning is becoming increasingly popular, as it is used for detecting hidden patterns or clusters in data without labeled responses. As one of the most important branches of unsupervised learning, time series clustering is being studied and a new feature for clustering is developed in this study.

5.1.1 Time Series Clustering

Time series clustering is a branch of the unsupervised learning to separate time series data into different clusters. It works by mining the underlying structure in an unlabeled time series dataset to organize data into similar groups, so that the within-group dissimilarity is minimized and the between-group dissimilarity is maximized [33]. Clustering the time series is particularly advantageous because labels or targets are not needed in this technique, which means that it does not rely on time-consuming annotation of the data [51]. Leading to the discovery of dynamic changes in the sequences, time series clustering has extracted significant attention in the last few decades, including the area of anomaly detection, intrusion detection, process control, and character recognition [52],[19],[53].

5.1.2 Representations of Time Series

Shape-based, model-based and feature-based methods are three major approaches to cluster time-series [32]. The shape-based approach, also named raw-data-based or instance-based approach [17], works directly with the raw time series data by matching them to the similar instances of the time series.

The model-based methods convert a raw time-series into model parameters, and then the distances between the extracted model parameters are measured to be applied to the clustering algorithm [33]. The model-based method is able to handle the time series of different lengths [34]. However, it has scalability problems [35], and its performance reduces when the clusters are close to each other [36].

In the feature-based approach, the raw time-series is represented by a set of derived properties [17], namely features. Then, a clustering algorithm is applied to the extracted feature vectors [32]. Feature vectors usually have lower dimensions compared to the length of the raw data, and features can make distance calculations to be more meaningful and feasible [37]. The commonly used features for time series clustering include the autocorrelation, skewness, Discrete Wavelet Transform (DWT) and Discrete Fourier Transform (DFT).

5.1.3 Structure of the Study

The commonly used features, such as the autocorrelation and skewness, might not take the time as a varying factor. In this study, we present a new feature-based time series clustering algorithm using the Time Dependent Kernel Density Estimation (TDKDE) as the time varying feature, and compare its performance to that of the widely used feature-based method: Discrete Wavelet Transform (DWT) approach. Our new method is demonstrated to be superior to the DWT-based approach evaluated on datasets from the UCR Time Series Data Mining Archive[28].

The rest of this study is organized as follows: section 2 illustrates the background and related methodologies. In section 3, the proposed algorithm on the basis of TDKDE is presented in detail. The dataset description and the performance comparison are included in section 4 and section 5. Finally, we conclude the study by summarizing the main contributions in Section 6.

5.2 Background and Related Methodologies

In this study, we introduce the Time Dependent Kernel Density Estimates as a new feature with the Self-organizing Map clustering algorithm built on it. The background and related methodologies are explained in detail in the following subsections.

5.2.1 Time Dependent Kernel Density Estimates (TDKDE)

Kernel Density Estimation (KDE) is a non-parametric method to estimate the probability density function of a random variable Y. Let $(y_1, y_2, ..., y_T)$ be an independent and identically distributed sample drawn from a distribution with an unknown probability density f [38]. The traditional kernel estimator of $f(y)$ at point y can be expressed as:

$$\tilde{f}_T(y) = \frac{1}{Th}\sum_{j=1}^{T} K(\frac{y-y_j}{h}). \tag{5.2.1}$$

Here T is the number of observations, h is the bandwidth, and $K(\cdot)$ is the kernel, which is a non-negative function that integrates to one with mean zero.

The Time Dependent Kernel Density Estimates (TDKDE) can be seen as a combination of the Kernel Density Estimation(KDE) and the time factor. When the density estimation is thought to vary with time, it could be reasonable to introduce a weighting scheme to adjust the traditional kernel density estimation. One of the widely used schemes is the exponentially weighted moving average (EWMA) filter, which works by discounting older observations in an exponentially decaying manner. The time dependent kernel density estimation developed by Harvey and Oryshchenko (2012) [22]is such an estimation adjusted by the EWMA weighting scheme which is given by:

$$\tilde{f}_t(y) = \frac{1}{h}\sum_{j=1}^{t} K(\frac{y-y_j}{h})w_{t,j},\ t = 1, ..., T. \tag{5.2.2}$$

In the general case, $\sum_{j=1}^{t} w_{t,j} = 1$. In the study of Harvey and Oryshchenko (2012), $w_{t,j}$ is chosen to be:

$$w_{t,j} = \frac{1-\omega}{1-\omega^t}\omega^{t-j},\ j = 1, ..., t, \tag{5.2.3}$$

which is the EWMA filter with the discount parameter ω. Consequently, the TDKDE for the corresponding CDF can be expressed as:

$$\tilde{F}_t(y) = \sum_{j=1}^{t} H(\frac{y-y_j}{h})\frac{1-\omega}{1-\omega^t}\omega^{t-j}, \tag{5.2.4}$$

where $H(\cdot)$ is the CDF form of the corresponding kernel $K(\cdot)$.

In order to obtain the TDKDE, two parameters, the bandwidth h and the discount parameter ω, need to be estimated. The new estimation procedure has been developed and discussed in Chapter 3.

5.2.2 Discrete Wavelet Transformations

The wavelet transformation is one of the most popular time-frequency transform techniques for hierarchically decomposing sequences [39]. The discrete wavelet transform (DWT) is a linear transformation of a sequence in the time domain into wavelet coefficients in the frequency domain [40]. DWT is very appropriate for the dimensionality reduction, noise filtering, as well as singularity detections.[41].

An original time series is divided into two components by DWT. The first component is a sequence of coefficients denoted as the scaling coefficients (or smooth coefficients [42]), which are proportional to the averages over the original time series.

The second component is a sequence of the wavelet coefficients, or denoted as the detail coefficients, which are proportional to the differences of averages[43].

For every pair j, k of integers, the basis of the DWT is a set of functions which are defined by:

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k), \quad t \in \mathbf{R},$$

where ψ is the mother wavelet function, and 2^j is the scaling of t [37]. There are a number of basis functions that can be used as the mother wavelet. A simple and commonly used wavelet is the Haar wavelet. The Haar wavelet is memory efficient, and it preserves Euclidean distance as well [44]. The mother wavelet function of the Haar wavelet $\psi(t)$ can be described as

$$\psi(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2}, \\ -1 & \frac{1}{2} \leq t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Its scaling function $\phi(t)$ can be expressed by

$$\phi(t) = \begin{cases} 1 & 0 \leq t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

The procedure to find the Haar wavelet transform of a discrete sequence can be illustrated using this one-dimensional sequence $(5, 1, 9, 5)$ (shown in Table 5.1).

Table 5.1: Procedure to Find the Haar Transform: A Simple Example

Level	**Averages**	**Detailed Coefficient**
1	5, 1, 9, 5	
2	3, 7	2, 2
3	5	-2

Level 1 keeps the full resolution (four dimensions) of the discrete sequence. For the Level 2, the averages of $(5, 1)$ and $(9, 5)$ are taken to make a reduced sequence $(3, 7)$. The values in the sequence $(2, 2)$ are the differences of $(5, 1)$ and $(9, 5)$ divided by two respectively. The sequences in the third level are obtained following the same procedure, and we continue this process until a resolution of 1 is reached.

5.2.3 Clustering: Self-Organizing Maps

Kohonen's Self-organizing Maps (SOM) are one of the most popular neural network models [45] which map high-dimensional data into the lower-dimensional topological structures [46]. These mappings are able to preserve the important topological relationships, i.e. the relative distances between the original data. [54] The SOM is one of the most distinguished unsupervised learning algorithms which learns to cluster inputs in such a way that the adjacent neurons (or nodes) on the map respond to similar input vectors [47] .

The SOM consists of the input layer and the output layer (competitive layer), connected with each other by weights. Figure 5.1 shows the topological structure with a simple SOM as an example. This is a network with 3×3 nodes in the output layer connected to each vector in the input layer. Each node has a specific topological position and contains a vector of weights with the same dimension as the input vectors.

The learning process of the SOM is shown in the flowchart in Figure 5.2,

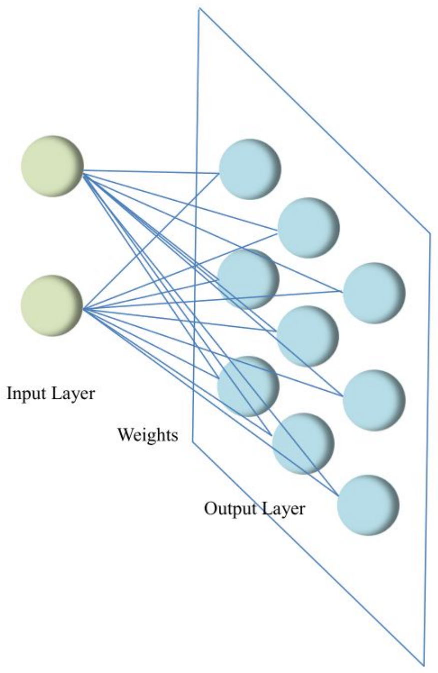


Figure 5.1: Topological Structure of the SOM

where $X = (x_1, x_2, ..., x_n)$ is a set of training samples; W_{ij} is a $p \times q$ grid of unites where i and j are the coordinates on that grid; α is the learning rate with the bounded range $[0, 1]$; r is the radius of the neighborhood function $h(W_{ij}, W_{mn}, r)$.

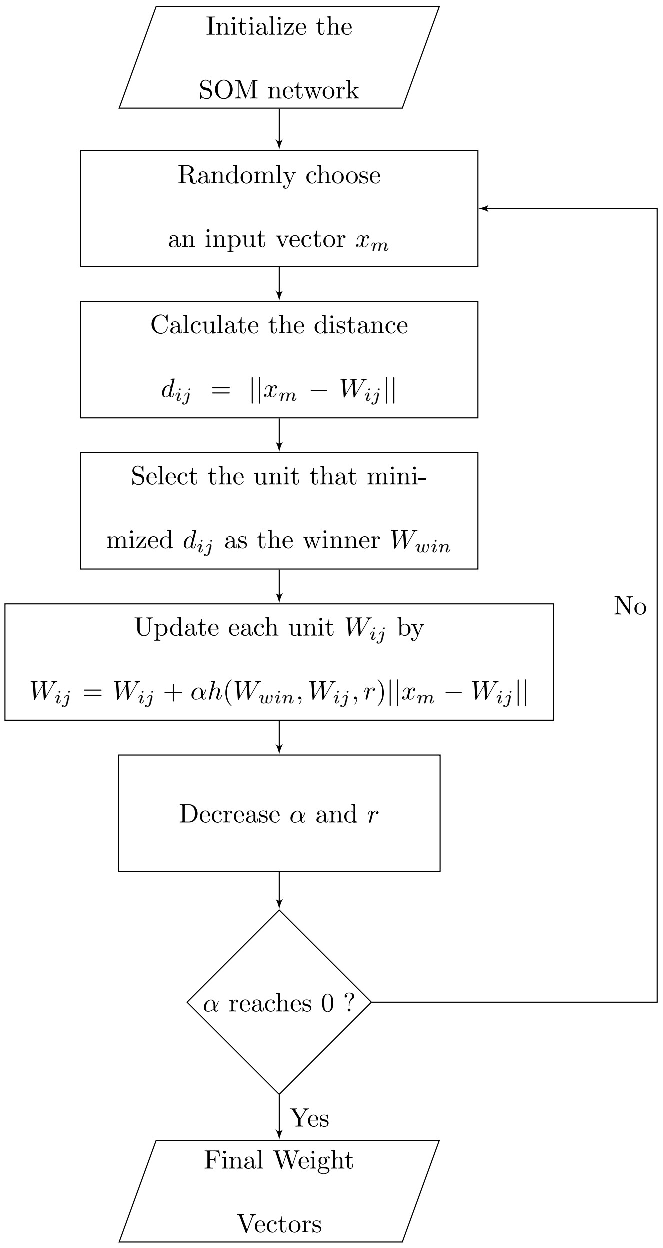

Initialize the SOM network
Randomly choose an input vector x_m
Calculate the distance $d_{ij} = ||x_m - W_{ij}||$
Select the unit that minimized d_{ij} as the winner W_{win}
Update each unit W_{ij} by $W_{ij} = W_{ij} + \alpha h(W_{win}, W_{ij}, r)||x_m - W_{ij}||$
Decrease α and r
α reaches 0 ?
No
Yes
Final Weight Vectors

Figure 5.2: The SOM Learning Process

5.3 Time Dependent Kernel Density-Based Clustering Algorithm

In this study, we would like to use the Time Dependent Kernel Density Estimates (TDKDE) as a new time varying feature in the clustering of time series, and compare the clustering performances of the TDKDE-based approach to the widely used DWT-based method.

5.3.1 Feature Vector Representations

In the TDKDE-based approach, the feature vector $\boldsymbol{f}_{TDKDE_i}$ for the i^{th} sample in the dataset with length T: $(y_{i1}, y_{i2}, ..., y_{iT})$, can be expressed by TDKDE at point $y_i^{(q)}, (q = 1, 2, ..., Q)$, i.e.

$$\boldsymbol{f_{TDKDE_i}} = (\tilde{f}_T(y_i^{(1)}), \tilde{f}_T(y_i^{(2)}), ..., \tilde{f}_T(y_i^{(Q)})).$$

In order to obtain the appropriate TDKDE, one needs to estimate the parameters h and ω. In this study, the optimal pair of parameters (h^*, ω^*) can be estimated using the new estimation procedure developed in Chapter 3.

In the DWT-based approach, the feature vector $\boldsymbol{f}_{DWT_i}$ for the i^{th} sample in the dataset with length T: $(y_{i1}, y_{i2}, ..., y_{iT})$, can be expressed by the wavelet coefficients W_i, i.e.

$$\boldsymbol{f_{DWT_i}} = (W_i^{(1)}, W_i^{(2)}, ..., W_i^{(Q)}).$$

For the purposed of comparison, these two feature vectors are designed to have the equal length $Q = \frac{7}{8}T$, which means that the first three levels of the wavelet coefficients are used in our study. Since $Q < T$, both feature vectors can be adopted as meaningful dimensionality reduction techniques.

5.3.2 New Algorithm and Comparative Studies

Once the TDKDE-based feature vectors are obtained, they are fed to an unsupervised learning algorithm, the Self-organizing Map(SOM) algorithm, as the input vectors. The key steps of the TDKDE-based clustering algorithm and the existing DWT-based method are shown in Figure 5.3. For the purpose of comparison, the feature vectors extracted from both methods are designed to have equal length.

5.3.3 Clustering Validation

To compare the clustering performances, the Adjusted Rand Index (ARI) is used as a criterion. The ARI is frequently used for measuring the similarity of a cluster to a reference. It is a function that measures the agreement between two partitions: one given by the prediction results from the clustering process and the other defined by the target data.

The ARI aims to analyze how similar a cluster to a reference is by counting the correctly classified pairs of elements. As demonstrated by Santos (2009) [48], it is especially good for multi-class classification. The ARI may yield a value with the bounded range $[-1, 1]$. A higher ARI indicates a better match between the predicted clusters and the target clusters, and 1 is the perfect match score.

5.4 Datasets

5.4.1 The UCR Time Series Data Mining Archive

In order to compare the performance of the DWT-based clustering and the TDKDE-based clustering algorithms, we would like to verify the results using multiple datasets, because in general, the results evaluated on multiple datasets should be more reliable than those evaluated on only one special dataset.

The UCR Time Series Data Mining Archive (Keogh et al. 2011[28]) has been widely used as a benchmark for evaluating the performance of time series classification or clustering algorithms. It contains multiple datasets that were gathered from diverse resources. The application domains vary broadly across the archive, including natural science, health science, social science, and so on. The applications are not limited to the real time series, but also contain the "pseudo time series" retrieved from videos, images, handwritten materials and texts.

All the datasets in this data mining archive are labeled, univariate time series, and each dataset is split into a training set and a testing set, which is convenient for the performance comparison of supervised and unsupervised learning algorithms. Since we are focusing on the unsupervised learning algorithms in this study, we train each of the dataset without training labels, and then evaluate the performance by comparing the predicted labels with the true labels using the testing set.

In our study, twenty datasets from the UCR Time Series Data Mining Archive are analyzed. Even though this archive has kept updating since 2011 to include more datasets, the original twenty datasets in this archive are able to provide sufficient information for comparative studies[17].

5.4.2 An example: ECG Dataset

To have a better understanding of the clustering algorithm, we choose the ECG dataset as an example to illustrate the clustering results in detail.

The electrocardiogram (ECG) signals, which represent the cardiac function, are recordings of the electric waves being generated during cardiac activities. The ECG dataset contains 200 samples with 133 normal samples and 67 abnormal samples [49]. The 200 samples are equally separated into two sets: 100 samples in the training set and 100 samples in the testing set. The standardized signals are visualized in Figure 5.4.

The SOM is usually presented by a two-dimensional regular grid of nodes. The weight vector is associated with each node in the output layer. The weight vector of each node optimally describes the inputs mapped to that node. They are automatically organized into a meaningful two-dimensional structure in which similar weight vectors are closer to each other in the map than the dissimilar ones [50]. By visualizing the weight vectors on this two-dimensional map, we can see patterns and clusters in the distribution of inputs.

In this exemplary application with the ECG dataset, we visualize weight vectors of the DWT-based SOM and the proposed TDKDE-based SOM in Figure 5.5 and Figure 5.6 respectively. As indicated from the two figures, the neighboring nodes are mutually similar in the weight vectors, and thus the samples can be isolated into different clusters. The nodes with distinct colors represent different clusters and the boundaries are denoted by the bold lines. Compared to the DWT-based approach, the weight vectors from different clusters in the TDKDE-based approach seem to be more dissimilar, which indicate that using the TDKDE instead of DWT as a feature may lead to a better clustering performance on this dataset. Furthermore, as evaluated by the ARI, the ARI score on the TDKDE-based approach is higher than that on the DWT-based approach, which demonstrates that the TDKDE-based approach outperforms the DWT-based approach on this ECG dataset.

5.5 Clustering Results

Table 5.2 shows the clustering results for all the twenty datasets from the UCR Time Series Data Mining Archive. The Adjusted Rand Index (ARI) is used as a criterion for the performance comparison. In this study, the training set and the testing set are separated in the same way as those in the original database.

In order to make the algorithm comparable to each other, we choose the same number of points Q for the DWT and the TDKDE feature vectors. For a time series with length T, Q is set to be $Q = \frac{7}{8}T$ in our study, which means that only the first, second and third level of the wavelet coefficients are used. Since $Q < T$, both DWT-based approach and TDKDE-based approach can be taken as dimensionality reduction techniques. The Haar wavelet is chosen as the wavelet function in the DWT-based method. In the TDKDE-based approach, the optimal bandwidth is adopted as the initial bandwidth h_0. In addition, the Gaussian kernel is selected as the kernel function $K(\cdot)$, i.e.,

$$K(\cdot) = \frac{1}{\sqrt{2\pi}} exp(-\frac{y^2}{2}).$$

For each dataset, the relatively better performance, i.e. the higher ARI, is denoted by the boldface number. As indicated from Table 5.2, the TDKDE-based SOM approach is superior to the DWT-based SOM approach on 16 out of 20 datasets.

The performance comparison results in terms of the ARI can be visualized in Figure 5.7. Each dot in the graph represents a pair of ARI of the TDKDE-based SOM approach and the DWT-based SOM approach for a specific dataset. The dots on the 45 degree line indicate that these two algorithms have the same ARI on the corresponding datasets. The dots on the upper half of the graph indicate that the DWT-based SOM approach is superior to the TDKDE-based SOM approach, and vice versa. As can be concluded from the figure, the TDKDE-based SOM approach generally outperforms the DWT-based SOM approach on these twenty datasets, with the odds 16:4.

5.6 Contributions

In this chapter, we have introduced a new time varying feature–the Time Dependent Kernel Density Estimation (TDKDE) as a feature for both stationary and non-stationary time series clustering. Our new feature combined with Self-organizing Maps (SOM) clustering algorithm provides excellent performance on clustering. The merits of the proposed method have been validated by the clustering performance of twenty datasets from the UCR Time Series Data Mining Archive. The analysis of the results verifies that, the proposed TDKDE-based Self-organizing Map (SOM) approach is superior to the commonly used DWT-based SOM method in terms of the Adjusted Rand Index (ARI). Finally, our new method can be used to evaluate very important time dependent signals in health science, business, environmental science, among others.

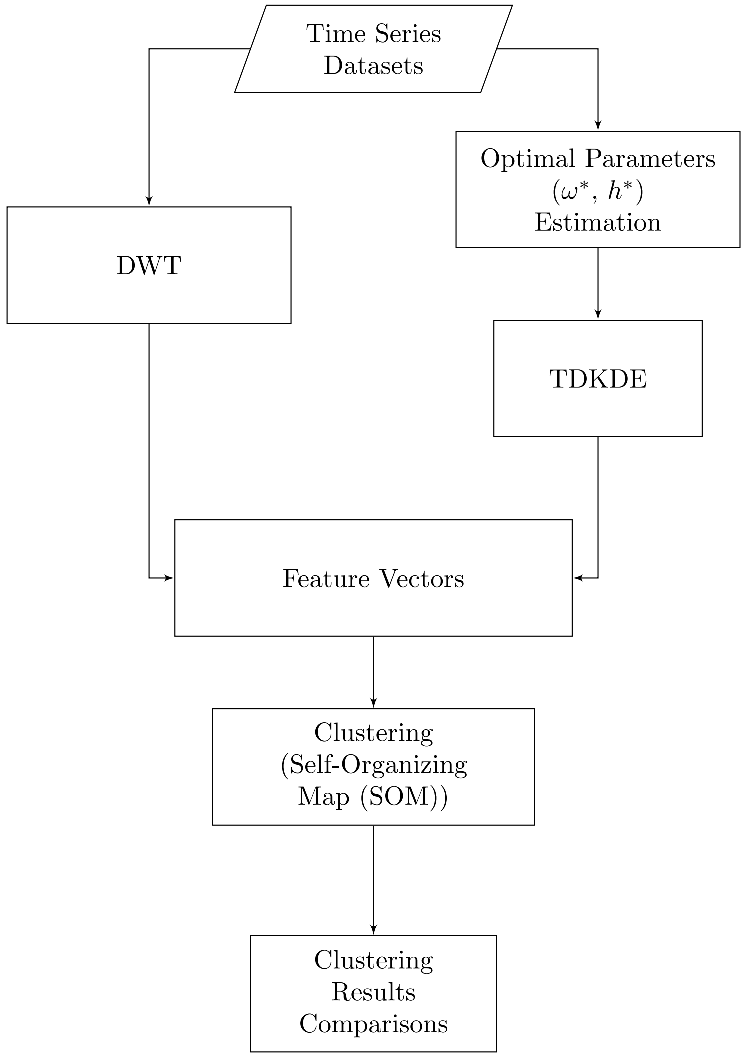
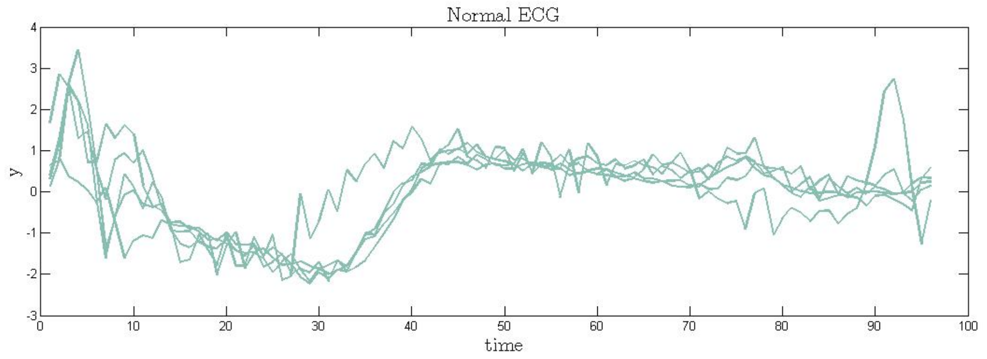

Time Series Datasets
Optimal Parameters (ω^*, h^*) Estimation
DWT
TDKDE
Feature Vectors
Clustering (Self-Organizing Map (SOM))
Clustering Results Comparisons

Figure 5.3: The Comparison of the Two Feature-Based Time Series Clustering Algorithms: the DWT-Based SOM (the Existing Method) and the TDKDE-Based SOM (the Proposed Method).



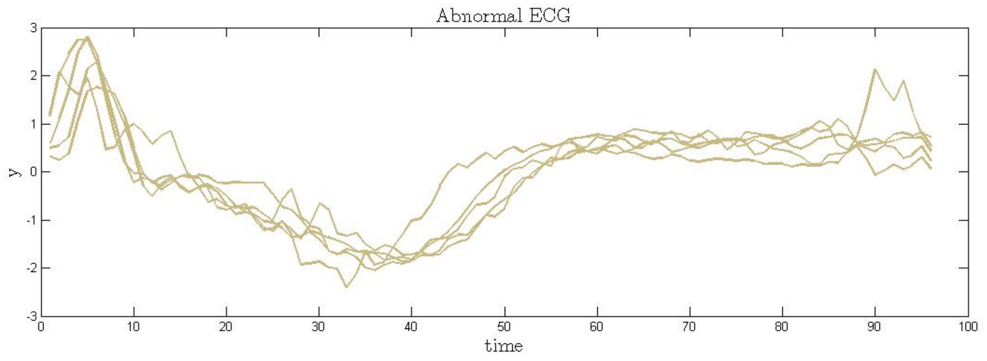


Figure 5.4: Standardized ECG Signals with 5 Samples from Each Class

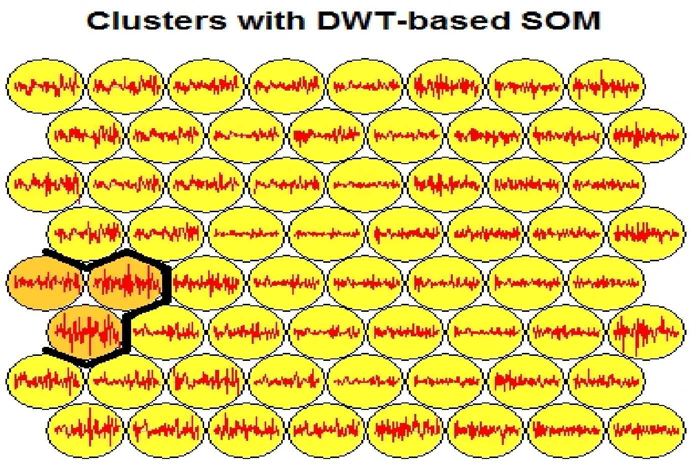


Figure 5.5: The Weight Vectors of the DWT-Based SOM on 8-by-8 Grid of Nodes

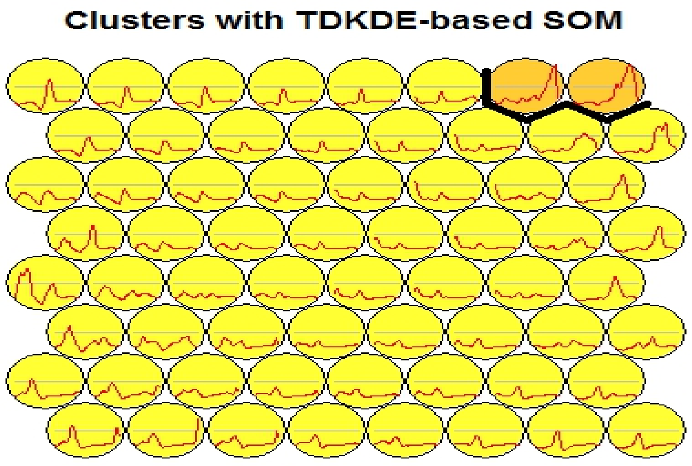


Figure 5.6: The Weight Vectors of the TDKDE-Based SOM on 8-by-8 Grid of Nodes

Table 5.2: Clustering Performance Comparison: Adjusted Rand Index (ARI) of the Existing DWT-based SOM Approach Versus the Proposed TDKDE-Based SOM Approach for Twenty Datasets.

Dataset	Size of Training Set	Size of Testing Set	Number of Classes	Time Series Length	ARI of DWT-based SOM	ARI of TDKDE-based SOM
synthetic_control	300	300	6	60	0.1906	**0.4443**
Gun_Point	50	150	2	150	0.1451	**0.1765**
CBF	30	900	3	128	0.0011	**0.1774**
FaceAll	560	1690	14	131	0.1254	**0.2999**
OSULeaf	200	242	6	427	0.1211	**0.1369**
SwedishLeaf	500	625	15	128	0.3295	**0.3398**
50words	450	455	50	270	**0.3652**	0.1240
Trace	100	100	4	275	0.3445	**0.4764**
Two_Patterns	1000	4000	4	128	0.0537	**0.1395**
Wafer	1000	6174	2	152	**0.7160**	0.0265
FaceFour	24	88	4	350	**0.2194**	0.0863
Lightning-2	60	61	2	637	-0.0132	**0.0000**
Lightning-7	70	73	7	319	0.0653	**0.1618**
ECG200	100	100	2	96	0.3772	**0.8816**
Adiac	390	391	37	176	**0.4604**	0.2124
Yoga	300	3000	2	426	0.1194	**0.2268**
Fish	175	175	7	463	0.3217	**0.3350**
Beef	30	30	5	470	0.1613	**0.3894**
Coffee	28	28	2	286	0.4922	**1.0000**
OliveOil	30	30	4	570	0.4077	**0.6991**

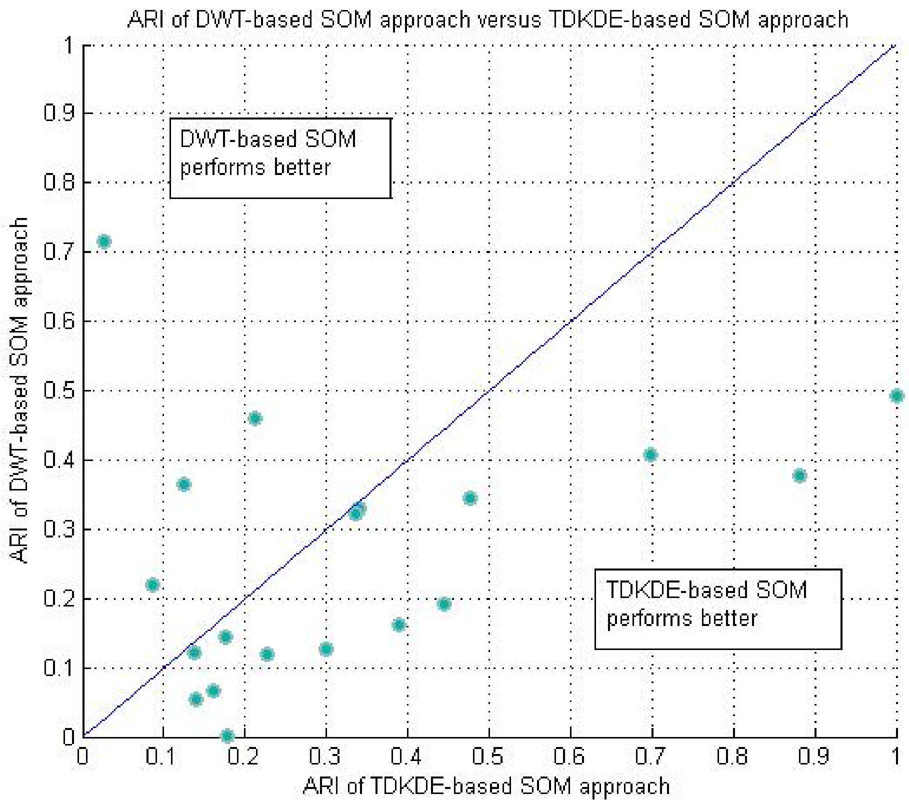


Figure 5.7: The ARI of the TDKDE-Based SOM Approach Versus the DWT-Based SOM Approach on the Twenty Datasets

6 Future Research

Our future works consist of two main directions, the first is to involve various kernel functions in the new estimation procedure of the Time Dependent Kernel Density Estimation (TDKDE) in order to compare performances based on different kernels. The second is for the time series classification and clustering problems, where the length of the feature vectors can be modified and the performances based on various lengths can be compared.

6.1 Future Research for the ANN Parameter Estimation Algorithm

For the purpose of comparison, Gaussian Kernel was employed for the new parameter estimation algorithm in Chapter 3. It would be valuable if other kernel functions can be involved in the new algorithm, and the performance comparisons can be conducted with different combinations of kernel functions and the estimates of the discount parameter ω and the bandwidth h. Furthermore, the computing time can be added as a criterion to measure the performance of estimation procedure.

6.2 Future Research for the TDKDE-Based Time Series Classification and Clustering

In both supervised and unsupervised learning, the length of the feature vector was set to be $Q = \frac{7}{8}T$, which means that the dimension of the feature vector was 87.5% of the raw data. In the future research, it would be worth trying different lengths, and study the relationship between the classification/clustering performances and the lengths of feature vectors.

In addition, the Gaussian Kernel function can be replaced with other kernel functions, and the Random Forests classification algorithm can be changed to other supervised learning algorithms (such as Support Vector Machines, Naive Bayes and Neural Networks) to see if there is any improvement for the classification performance.

References

[1] Miladinovic, B. (2008). Kernel density estimation of reliability with applications to extreme value distribution.

[2] Giraitis, L., Kapetanios, G., & Price, S. (2013). Adaptive forecasting in the presence of recent and ongoing structural change. Journal of Econometrics, 177(2), 153-170.

[3] Harvey, A., & Oryshchenko, V. (2012). Kernel density estimation for time series data. International Journal of Forecasting, 28(1), 3-14.

[4] Guidoum, A. C. (2013). Kernel Estimator and Bandwidth Selection for Density and its Derivatives.

[5] Perez, A. (2012). Comments on Kernel density estimation for time series data. International Journal of Forecasting, 28(1), 15-19.

[6] Izenman, A. (2008). Modern multivariate statistical techniques (Vol. 1). New York: Springer.

[7] Harvey, A. C., & Oryshchenko, V. (2009). Local kernel density estimation from time series data (Preliminary and incomplete), (1964), 120.

[8] Diebold, F. X., Gunther, T. A., & Tay, A. S. (1997). Evaluating density forecasts. International Economic Review, 39(1), 863883.

[9] Wuertz, D., Chalabi, Y., & Luksan, L. (2006). Parameter Estimation of ARMA Models with Garch/Aparch errors. Journal of Statistical Software, VV(II).

[10] Hagan, M. T., Demuth, H. B., Beale, M. H., & De Jess, O. (1996). Neural network design (Vol. 20). Boston: PWS publishing company.

[11] Yu, H., & Wilamowski, B. M. (2011). Levenbergmarquardt training. Industrial electronics handbook, 5(12), 1-16.

[12] Baliyan, A., Gaurav, K., & Mishra, S. K. (2015). A Review of Short Term Load Forecasting using Artificial Neural Network Models. Procedia Computer Science, 48, 121-125.

[13] Gershenson, C. (2003). Artificial neural networks for beginners. arXiv preprint cs/0308031.

[14] Wei, L., & Keogh, E. (2006, August). *Semi-supervised time series classification.* In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 748-753). ACM.

[15] Karpagachelvi, S., Arthanari, M., & Sivakumar, M. (2010). ECG feature extraction techniques-a survey approach. arXiv preprint arXiv:1005.0957.

[16] Ratanamahatana, C. A., & Keogh, E. (2004, August). Everything you know about dynamic time warping is wrong. In Third Workshop on Mining Temporal and Sequential Data.

[17] Fulcher, B. D., & Jones, N. S. (2014). Highly comparative feature-based time-series classification. Knowledge and Data Engineering, IEEE Transactions on, 26(12), 3026-3037.

[18] Nanopoulos, A., Alcock, R., & Manolopoulos, Y. (2001). Feature-based classification of time-series data. International Journal of Computer Research, 10(3), 49-61.

[19] Wang, X., Smith, K., & Hyndman, R. (2006).Characteristic-based clustering for time series data. Data mining and knowledge Discovery, 13(3), 335-364.

[20] Deng, H., Runger, G., Tuv, E., & Vladimir, M. (2013). A time series forest for classification and feature extraction. Information Sciences, 239, 142-153.

[21] Hansen, B. E. (2009). Lecture notes on nonparametrics. Lecture notes.

[22] Harvey, A., & Oryshchenko, V. (2012). Kernel density estimation for time series data. International Journal of Forecasting, 28(1), 3-14.

[23] Xing, Z., Pei, J., Philip, S. Y., & Wang, K. (2011, April). Extracting Interpretable Features for Early Classification on Time Series. In SDM (Vol. 11, pp. 247-258).

[24] Alonso, A. M., Berrendero, J. R., Hernandez, A., & Justel, A. (2006). Time series clustering based on forecast densities. Computational Statistics & Data Analysis, 51(2), 762-776.

[25] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[26] Taylor, S. L., & Kim, K. (2011). A jackknife and voting classifier approach to feature selection and classification. Cancer informatics, 10, 133.

[27] Mitchell, M. W. (2011). Bias of the Random Forest out-of-bag (OOB) error for certain input parameters. Open Journal of Statistics, 1(03), 205.

[28] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen and Gustavo Batista (2015). The UCR Time Series Classification Archive.

[29] Olszewski, R. T. (2001). Generalized feature extraction for structural pattern recognition in time-series data (No. CMU-CS-01-108). CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE.

[30] Silverman, B. W. (1986). Density estimation for statistics and data analysis (Vol. 26). CRC press.

[31] Santos, J. M., & Embrechts, M. (2009). On the use of the adjusted rand index as a metric for evaluating supervised classification. In Artificial neural networksICANN 2009 (pp. 175-184). Springer Berlin Heidelberg.

[32] Aghabozorgi, S., Shirkhorshidi, A. S., & Wah, T. Y. (2015). Time-series clusteringA decade review. Information Systems, 53, 16-38.

[33] Liao, T. W. (2005). Clustering of time series dataa survey. Pattern recognition, 38(11), 1857-1874.

[34] Xiong, Y., & Yeung, D. Y. (2003). Model-based clustering of sequential data using ARMA mixtures. Proceedings of the 4th ACM Postgraduate Research Day, 203-210.

[35] Vlachos, M., Gunopulos, D., & Das, G. (2004). Indexing time-series under conditions of noise. Data mining in time series databases, 67.

[36] Mitsa, T. (2010). Temporal data mining. CRC Press.

[37] Mrchen, F. (2003). Time series feature extraction for data mining using DWT and DFT.

[38] Hansen, B. E. (2009). Lecture notes on nonparametrics. Lecture notes.

[39] Zhang, H., Ho, T. B., Zhang, Y., & Lin, M. S. (2006). Unsupervised feature extraction for time series clustering using orthogonal wavelet transform. Informatica, 30(3).

[40] Huhtala, Y., Karkkainen, J., & Toivonen, H. T. (1999, February). Mining for similarities in aligned time series using wavelets. In AeroSense'99 (pp. 150-160). International Society for Optics and Photonics.

[41] Chaovalit, P., Gangopadhyay, A., Karabatis, G., & Chen, Z. (2011). Discrete wavelet transform-based time series analysis and mining. ACM Computing Surveys (CSUR), 43(2), 6.

[42] Kim, B. R., McMurry, T., Zhao, W., Wu, R., & Berg, A. (2010). Wavelet-based functional clustering for patterns of high-dimensional dynamic gene expression. Journal of Computational Biology, 17(8), 1067-1080.

[43] Kim, B. R., McMurry, T., Zhao, W., Wu, R., & Berg, A. (2010). Wavelet-based functional clustering for patterns of high-dimensional dynamic gene expression. Journal of Computational Biology, 17(8), 1067-1080.

[44] Chan, K. P., & Fu, A. W. C. (1999, March). Efficient time series matching by wavelets. In Data Engineering, 1999. Proceedings., 15th International Conference on (pp. 126-133). IEEE.

[45] Yin, H. (2008). The self-organizing maps: background, theories, extensions and applications. In Computational intelligence: A compendium (pp. 715-762). Springer Berlin Heidelberg.

[46] Fritzke, B. (1994). Growing cell structuresa self-organizing network for unsupervised and supervised learning. Neural networks, 7(9), 1441-1460.

[47] Wang, L., Jiang, M., Lu, Y., Noe, F., & Smith, J. C. (2006). Self-organizing map clustering analysis for molecular data. In Advances in Neural Networks-ISNN 2006 (pp. 1250-1255). Springer Berlin Heidelberg.

[48] Santos, J. M., & Embrechts, M. (2009). On the use of the adjusted rand index as a metric for evaluating supervised classification. In Artificial neural network-sICANN 2009 (pp. 175-184). Springer Berlin Heidelberg.

[49] Olszewski, R. T. (2001). Generalized feature extraction for structural pattern recognition in time-series data (No. CMU-CS-01-108). CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE.

[50] Kohonen, T. (1990). The self-organizing map. Proceedings of the IEEE, 78(9), 1464-1480.

[51] Paparrizos, J., & Gravano, L. (2015, May). k-Shape: Efficient and Accurate Clustering of Time Series. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (pp. 1855-1870). ACM.

[52] Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases (Vol. 23, No. 2, pp. 419-429). ACM.

[53] Chi, M., Banerjee, S., & Hassanien, A. E. (2009). Clustering time series data: an evolutionary approach. In Foundations of Computational, IntelligenceVolume 6 (pp. 193-207). Springer Berlin Heidelberg.

[54] Kohonen, T., Oja, E., Simula, O., Visa, A., & Kangas, J. (1996). Engineering applications of the self-organizing map. Proceedings of the IEEE, 84(10), 1358-1384.