Univariate and Multivariate Time Series Manifold Learning

Colin O'Reilly[a,*], Klaus Moessner[a], Michele Nati[b]

[a] *Institute for Communication Systems (ICS), Faculty of Engineering and Physical Sciences, University of Surrey, Guildford, Surrey, GU2 7XH*

[b] *Digital Catapult, 101 Euston Rd, London NW1 2RA, United Kingdom*

Abstract

Time series analysis aims to extract meaningful information from data that has been generated in sequence by a dynamic process. The modelling of the non-linear dynamics of a signal is often performed using a linear space with a similarity metric which is either linear or attempts to model the non-linearity of the data in the linear space. In this research, a different approach is taken where the non-linear dynamics of the time series are represented using a phase space. Training data is used to construct the phase space in which the data lies on or close to a lower-dimensional manifold. The basis of the non-linear manifold is derived using the kernel principal components derived using kernel principal component analysis where fewer components are retained in order to identify the lower-dimensional manifold. Data instances are projected onto the manifold, and those with a large distance between the original point and the projection are considered to be derived from a different underlying process. The proposed algorithm is able to perform time series classification on univariate and multivariate data. Evaluations on a large number of real-world data sets demonstrate the accuracy of the new algorithm and how it exceeds state-of-the-art performance.

Keywords: time series, univariate, multivariate, one-class classification, kernel principal component analysis

1. Introduction

Time series data are derived from the measurement of an underlying phenomena and are represented as sequential instances of values that may occur individually (univariate), or concurrently (multivariate). They are generated in a broad range of domains such as aviation [1, 2], financial [3, 4], meteorological [5] and industrial monitoring [6]. Extraction of knowledge from a time series using machine learning or data mining methods can enable classification of current data instances, or the prediction of future instances. Several excellent survey articles have been published in this area, for example those by Fu [7] and Längkvist *et al.* [8].

A task that is often performed is the classification of cyclostationary time series data. Given a sequence of time series data which represents one cycle, the aim is to classify the sequence into the correct category of cycle. There are two approaches that are used to perform the task. One method uses a lazy-learning approach where testing data are compared to training data to determine similarity. No model is constructed during the training phase, however, during the testing phase a window of test data is compared to windows of the training data. A similarity metric, for example Euclidean distance, is used to determine the level of similarity. An alternative approach uses the time series as a representation of a dynamical system to construct a *phase space*. A model is created of the system from the training data set, and then a similarity metric is used to determine if test data instances were generated from the same underlying process.

This research takes the second approach where the aim is to construct a model of a dynamical system using a phase space. The model is then used to identify test data instances that were generated by the same process. State of the art is extended in the following way:

- A univariate time series is represented as a dynamical system using a phase space. In the phase space, the lower-dimensional manifold on which the data lies on or close to is determined using kernel PCA. The similarity of a test data instance to the dynamical system is determined by projecting the data instance onto the manifold and determining the error in its reconstruction.
- The algorithm is extended to operate on multivariate time series using horizontal form SSA. The algorithm extracts the most important information from the multivariate streams by identifying the lower-dimensional manifold that the data lies on or close to.
- A detailed evaluation of the algorithm on both univariate and multivariate real-world time series is pro-

*Corresponding author

Email addresses: c.oreilly@surrey.ac.uk (Colin O'Reilly), k.moessner@surrey.ac.uk (Klaus Moessner), michele.nati@digicatapult.org.uk (Michele Nati)

vided and a comparison is made with many other state-of-the-art algorithms.

This paper is organized as follows. Section 2 examines the related research in the area of time series classification. The proposed algorithm is presented in detail in Section 3. Section 4 then examines the performance of the proposed algorithm on a large number of real-world datasets. The final part of this section includes a discussion on the performance of the proposed and benchmark algorithms. Finally, Section 5 provides conclusions and recommendations for future work.

2. Related work

In this section, a review of the current work related to machine learning on univariate and multivariate time series is presented. This is divided into two sections, univariate time series, which consists of one stream, and multivariate time series which consists of multiple streams which share a temporal dimension.

2.1. Univariate Time Series

A univariate, real-valued time series, defined as $\boldsymbol{x} = \{x(n) : n = 1, \cdots, N\}$, is an ordered set of N real-valued variables. This type of time series is typically generated from measurements by a sensor of a latent dynamical system, an example of which is a sensor monitoring the pressure of water in a component of a power station. Time series classification is a well-researched area. There are many proposed approaches to classify cyclostationary univariate data which can be categorized as; time domain distance, difference, dictionary, shapelets, interval, ensembles, image-related and time delay embedding.

A common approach to time series classification is to use distance in the time domain. An example of this is the Euclidean distance (ED) metric which uses each measurement in the time series as a point in Euclidean space, with the distance between points being the similarity metric using an L_p norm such as the Euclidean distance. The performance of Euclidean distance is often used as a lower bound on the achievable performance. An advantage of the approach is that there are no parameters to be determined, and this is a criticism of more complex algorithms such as Singular Spectrum Analysis (SSA) that require a number of parameters to be determined. However, there are methods to determine optimal or near optimal parameters if there is a labelled training data set, and it can be advantageous to use more complex algorithms if they can be shown to have superior performance. There is a drawback in the Euclidean distance approach; performance can decrease if the cycles in the time series are not aligned.

Elastic measures can overcome the drawback of misalignment by aligning segments before computing the distance between them. In dynamic time warping (DTW) [9] the sequences are stretched in the temporal dimension in order to align them, before the Euclidean distance is calculated. In order to prevent significant stretching to align time series, a locality constraint can be added, constrained DTW (cDTW), which limits the amount of stretching between subsequent elements in a sequence. cDTW has been shown to have better performance than DTW for certain classification problems [10]. Many extensions to DTW have been proposed. These include weighted DTW (WDTW) [11] which adds a multiplicative penalty based on the warping distance between points in the warping path and derivative DTW [12] where the series is first transformed into a series of first order differences. Another elastic measure, Time-Warp-Edit (TWE) [13], uses elements from DTW and the longest common subsequence to allow warping the temporal domain while combining a distance metric using L-norms. Move-Split-Merge (MSM) [14] uses a set of transformations to provide an alternative representation of the time series that is robust to temporal misalignment and is translation invariant.

Differential distance-based classifiers use the difference between data instances to derive an alternative set of features. Complexity-Invariant Distance (CID) [15] introduces the notion of the complexity of a time-series, where a complex series has many more peaks and troughs than a less complex one. The more complex the sequence, the greater the distance between pairs of time series. Information about the complexity differences between two time series is used as a correction factor to existing distance measures.

Dictionary-based approaches reduce the dimensionality of a time series by deriving subsets of features which are represented by words. Similarity is then measured by deriving the words from a test sample and comparing with those derived from the training samples. A sliding window is used and a word from an alphabet is assigned to the section. Sax-based Vector Space Model (SAX-VSM) [16] uses symbolic aggregate approximation (SAX) [17] to provide a high level symbolic representation of a time series and Vector Space Model [18] to transform the SAX words into class-characteristic weight vectors with the classifier constructed using cosine similarity. Bag of Patterns (BoP) [19] applies SAX to a window of data to construct a dictionary of words for the time series. Classification occurs by applying the same transform and using the nearest neighbour in the training set to determine the classification label.

Shapelets [20] aim to compare small subsequences of a cycle of a time series. Many classifiers consider the entire cycle, however. Shapelets take a different approach and aim to use local features to provide more accurate and robust classifiers when there is noise in the dataset. By using small subsequences of a cycle, shapelets can be significantly faster at classification time than other state-of-the-art approaches [20]. The first shapelets were proposed by Ye *et al.* in the form of Logical Shapelets (LS) [20]. Fast Shapelets (FS) [21] increase the speed of shapelet discovery using SAX words and random projection to reduce the dimensionality of the data. Shapelet Transform

(ST) [22] discovers the top k shapelets and then uses them to transform the dataset into a new space. The distance of a series to each shapelet is used to form a k attribute instance in the transformation space. Learned time-series shapelet LTS [23] propose a mathematical formulation of the shapelet task by using a classification objective function which is solved using a gradient descent search procedure to identify the shapelets. The approach does not search but rather learns the near-optimal shapelets.

Interval-based classifiers use the interval between contiguous data instances to derive an alternative representation of the time series. This causes a significant increase in the number of data instances in the series. Time Series Forests(TSF) [24] uses a random forest approach with summary statistics (mean, standard deviation and slope) of each interval as the features. Time Series Bag of Features (TSBF) [25] extends TSF and uses a random forest built from a bag-of-features. This has been extended to a multivariate form, MTSBF. Learned Pattern Similarity (LPS) [26] uses the intervals as attributes instead of features.

A recent proposal is to extend the ensemble technique used in machine learning to that of time series classification. Ensembles use several classifiers for the data, determining the label via a voting scheme. In order for the ensemble to exceed the accuracy of its individual members, it is necessary to have diverse and accurate classifiers [27]. In addition, weak learners can be boosted to strong learners in an ensemble [28]. Bagnall *et al.* [29] highlight the importance of an appropriate transformation in the time domain by using methods such as principal component analysis (PCA) and the autocorrelation function on the time series. Basic ensembles were constructed with the transformed data to classify time series. Results show that an appropriate transform in the time-domain can identify discriminatory features and that basic ensembles can significantly improve the performance of basic classifiers. Further research has been conducted on ensembles based on elastic distance measures (PROP) [30], shapelets (SE) and transformations (COTE) [31]. In a recent extensive evaluation of the performance of 18 time series classification algorithms on the UCR dataset [32], COTE was shown to be significantly more accurate.

Another approach is to use a transformation in order to identify salient features to discriminate between classes. Recurrence Patterns Compression Distance (RPCD) [33] uses recurrence plots as the representation domain for the time series. Recurrence plots are used in conjuction with the Campana-Keogh (CK-1) distance [34] to determine the similarity between image representations. Gramian Angular Summation/Difference Fields and Markov Transition Fields (GGM) [35] encodes a time-series using different types of images. Subsequently, tiled Convolutional Neural Networks are used to classify the images.

A model-based approach constructs a model of the training cycles with one model being constructed for each class. Testing cycles are then compared to this model in order to determine similarity. An approach is to use the time series as a representation of a dynamical system, constructing a model of the system. State space reconstruction is one such approach. It has two methods, Method of Delays (MOD) [36] and SSA [37], which are theoretically equivalent but operate differently when the time series contains noise. The MOD approach uses an m-dimensional state vector $\boldsymbol{x}_k^m = [x_k, x_{k+\rho}, \cdots, x_{(m-1)\rho}]^T$ where ρ is a multiple integer of τ_s and therefore the m entries are samples separated by a fixed period τ. SSA takes a slightly different approach where the state vector is derived from successive samples, which is equivalent to $\tau = 1$. The samples are then processed using PCA [38] to reduce dimensionality by finding a basis that maximizes variance. This ensures that the most important concepts are retained, while the noise, which is assumed to be uniform, is removed. It has been shown that for noise free and limited data, the approaches are equivalent. However, for noisy data, SSA outperforms MOD [39]. Both methods have been used to analyze univariate and multivariate time series. Broomhead and King [37, 40] state that SSA is a more robust version of MOD [36] to reconstruct the dynamics of a univariate time series.

Frank *et al.* [41] propose an algorithm, Geometric Template Matching (GeTeM), which uses MOD in order to model univariate time series. The model is then used to perform the classification of periodic and cyclostationary time series. The time series are non-linear functions modelled in a linear space with a similarity metric based on a nearest neighbour (1-NN) in order to build non-linear properties into the algorithm. GeTeM is evaluated on the UCR Time Series Classification Archive [42] and is shown to have superior performance to Euclidean distance, DTW and constrained DTW.

Non-linear spaces have also been used in order to represent the non-linear structure of the dynamical system. Ma and Perkins introduce two methods that rely on the support vector machine approach and use the *kernel trick* [43] to project the data into a non-linear space. The first approach [44] uses the phase space embedding approach of MOD and SSA to model the time series as a sequence of vectors. Anomaly detection is then performed using a one-class support vector machine that is able to identify subsequence anomalies in the form of discords. The second approach [45] uses support vector regression to identify the anomalous instances in the time series. Another kernel-based method [46] proposes a kernel which performs weighted DTW [11] in kernel space. This is used in conjunction with one [47],two and multi-class [48] support vector machines in order to classify cycles of data.

An alternative to using PCA in SSA is to use a non-linear subspace in order to model the non-linear dynamics. Teixeria *et al.* [49] extend the use of kernel principal component analysis (Kernel PCA) to operate on univariate time series by generating the kernel matrix for the trajectory matrix of SSA, termed kernel SSA. This method is used in conjunction with an algorithm that is introduced

to estimate the pre-image of a point in kernel space in order to denoise an Electroencephalogram (EEG) signal. This work is further discussed [50] where kernel SSA is used with the pre-image to remove artefacts in univariate EEG signals. Liu *et al.* [51] propose a method to perform anomaly detection on a univariate time series which uses either PCA, if the data is linear, or Kernel PCA if the data is non-linear, in order to model the dynamics of the data. From the model, trajectories of the time series are learned either using vector autoregressive or a probability density model. From this, an anomaly score is calculated which is the residual between the predicted and measure data instances.

2.2. Multivariate Time Series

A multivariate time series consists of two or more univariate time series which share the same temporal space. This is defined as $\boldsymbol{x} = \{x_d(n) : d = 1, \cdots D, n = 1, \cdots, N\}$

The application of SSA to multivariate data is performed through horizontal form multivariate singular spectrum analysis [52]. The block Hankel matrix is constructed in horizontal form in order to perform the analysis across all the streams. Applications include forecasting the economy [3, 4] and biometrics [53].

Tclass [54] aims to take advantage of the recurring substructure that exists in many time series classification problems. Metafeatures are identified for a specific application domain in order to identify recurrent substructures. They are chosen so that they have important characteristics such as being robust to noise and capturing the important elements of the time series. In addition to metafeatures that are specific to a domain, a universal set of metafeatures are detailed which include elements such as increasing and decreasing signals. Once the metafeatures have been determined, they are applied to the training data to extract observed events with the testing data then being examined for these events.

Orsenigo *et al.* [55] use a two-phase approach with a kernel method (TDVM). The first phase uses a robust similarity measure between pairs to convert the time series into sequences of the same length. The second phase uses a temporal variant of a support vector machine (SVM) which uses regularization to control the trade-off between two aspects. The first aspect is accuracy and generalization and the second aspect is the similarity of the time series.

Another approach by McGovern *et al.* [54] (Motif) aims to identify key temporal motifs in each dimension due to a common consideration that many dimensions of the data may be irrelevant or redundant. The number of passes through the data is minimized by using efficient data structures such as a trie [56]. The motifs are then ordered temporally and used to perform classification of the multivariate time series.

Symbolic Representation for Multivariate Time Series (SMTS) [26] provides a symbolic representation generated from a random forest with a bag of words then being used to classify the time series. It can be used to classify univariate and multivariate time series. Learned Pattern Similarity (LPS) [57] is a tree-based ensemble model which is shown to be fast and insensitive to parameter selection. The method operates on univariate and multivariate time series by learning a representation of segments in a time series and then uses a similarity metric based on the representation.

3. Non-linear Phase Space Modelling for univariate and multivariate time series

This research proposes an algorithm which uses a phase space to provide an alternative representation of the data derived from the underlying process. A phase space is a state space which is finite-dimensional consisting of an infinite number of points forming a smooth manifold. A phase space is able to model a deterministic dynamical system which contains the collection of possible states. Takens embedding theorum [36] proved that the time-delayed versions of one generic signal is sufficient to embed the m-dimensional manifold.

In the case of time series classification, the dynamical system that is modelled is that of the time series. Using time-delayed versions, the phase space can be constructed from the 1-dimensional times series to form a manifold in m-dimensional space, where m is the embedding dimension. A lower-dimensional manifold on which the data lies on or is close to is then identified using kernel PCA.

A phase space is constructed from the original time series using vectors of dimension m,

$$\boldsymbol{Y}(t_i) = y(t_i), y(t_{i+1}), \ldots, y(t_{i+m}) \tag{1}$$

The sequence of points form a *trajectory* of the dynamical system which stays within a bounded area. Due to the periodic and temporal nature of a time series, the phase space exhibits non-linear dynamics in the form of curves. There are two methods to model these non-linear dynamics. The first is to operate on the data in the linear space and then use a non-linear metric to determine similarity allowing the modelling of non-linear concepts. Methods derived from SSA, such as GeTeM, operate in this manner. The second approach, and the one used here, is to model the non-linear dynamics in a non-linear space.

The data in the phase space forms a multidimensional data set which will, due to noise and redundancy, lie on or close to a lower-dimensional manifold. In the phase space, a lower-dimensional manifold that the data lies on or close to is identified using a non-linear dimensionality reduction method. The aim of this is two-fold; reduce noise and redundancy in data. Finally, similarity between the manifold and a test data instance is measured using the distance between the test point and its projection onto the manifold. Data instances that lie on or close to the manifold can be considered to have been generated by the

same underlying process. In time series classification, one model is constructed for each time series, and the current test data instance is classified as the class of the model which has the highest similarity metric.

The following sections detail the approach. Firstly the univariate version of Time Series Manifold Learning (TSML) is presented, and then this is extended to a multivariate time series.

3.1. Embedding

The first step in the construction of the TSML model is to construct a phase space using the univariate time series. Techniques such as SSA use a phase space to model the time series. To obtain the phase space, the time series is embedded into a vector space of dimension m, this is known as the embedding dimension. Its value is critical to correctly model the cycle.

The first step is to project the data into a phase space using a trajectory matrix. An embedding is created where the original time series is mapped into a series of lagged vectors of size m. For example, the first lagged vector is $\boldsymbol{Y_1} = (y_1, \cdots, y_{1+m-1})^T$, the second is $\boldsymbol{Y_2} = (y_2, \cdots, y_{2+m-1})^T$ and so on. In this manner, although the time element has been removed from the data, each feature vector contains a small subset of the entire time series which allows the retaining of the dependency between data measurements in the time series (the lag). The lagged vectors form the columns of the trajectory matrix, (2). The window length is m and this results in the trajectory matrix $\boldsymbol{Y}$ with dimensions of $m \times K$ where $K = N - m + 1$. The trajectory matrix is a Hankel matrix where all the elements along the diagonal $i + j = const$ are equal. The trajectory matrix forms the features which represent the dynamics of the univariate time series in the embedded space. The formation of the trajectory matrix for one cycle is [58];

$$\boldsymbol{Y_i} = (y_i, \cdots, y_{i+m-1})^T \qquad (1 \leq i \leq K) \tag{2}$$

$$= \begin{pmatrix} y_1 & y_2 & y_3 & \cdots & y_k \\ y_2 & y_3 & y_4 & \cdots & y_{k+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_m & y_{m+1} & y_{m+2} & \cdots & y_N \end{pmatrix}$$

$$\boldsymbol{C_l} = [\boldsymbol{Y_1}, \boldsymbol{Y_2}, \cdots, \boldsymbol{Y_K}] \tag{3}$$

The Hankel matrix has been generated for one cycle, $\boldsymbol{C_l}$. This has created a matrix where the rows represent the feature vectors and the columns represent the instances. This is repeated for all the cycles of a class and these matrices are horizontally stacked in order to provide the data instances for all the cycles in a class.

$$\boldsymbol{X} = [\boldsymbol{C_1}, \boldsymbol{C_2}, \cdots, \boldsymbol{C_L}] \qquad l = 1, \ldots, L \tag{4}$$

The matrix $\boldsymbol{X}$ contains the data instances for all the cycles of a class.

3.2. Identification of the lower-dimensional manifold

The second step is to determine the lower-dimensional manifold that the data lies on or close to. In SSA this is performed using PCA where a linear lower-dimensional hyperplane is determined in the directions in the space that have the largest variance. By reducing the number of dimensions (i.e. principal components), noise and redundancy can be reduced in the data. Noise and redundancy will often have a small variance and thus will not feature in the components with a large variance. The principal components are derived by calculating the eigenvectors of the trajectory matrix and ordering them according to the eigen value.

A drawback of using PCA is that it is unable to model non-linear concepts as the basis derived is linear, i.e. the principal components are straight lines and the lower-dimensional space is a hyperplane. The data in the embedding space that the time series data is projected into will contain highly non-linear dynamics. Therefore, to identify the lower-dimensional manifold that the data lies on or close to, Kernel PCA [59] is used in the derivation of the principal components. Kernel PCA is able to perform the dimensionality reduction of manifolds, and it has been shown that other manifold learning methods such as Isomap, graph Laplacian eigenmaps and LLE can be interpreted as Kernel PCA with different kernel matrices [60]. Kernel PCA is able to represent non-linear dynamics; the kernel principal components are curves in input space, rather than straight lines as is the case for PCA. These will represent the lower-dimensional manifold on which the data lies on or close to.

The first step in the identification of the lower-dimensional manifold is to calculate the kernel matrix, $\boldsymbol{K}$, from the data matrix $\boldsymbol{X}$. There are several kernel functions in which to perform the map into feature space. In this work, the Gaussian Radial Basis Function (RBF) kernel, (5), was chosen due to its ability to map data into a high dimensional non-linear space and identify data instances that are different to the constructed model [61, 62].

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma^2}\right) \tag{5}$$

The kernel matrix (6) is calculated using (5) to determine the entries.

$$\boldsymbol{K_{ij}} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{6}$$

The next step is to identify the principal components in the projected space using Kernel PCA [59]. This is performed by identifying the principal components in feature space, where feature space $\mathcal{H}$ is related to the input domain, $\mathbb{R}^{\mathbb{N}}$, by the map $\boldsymbol{\Phi} : \mathcal{X} \to \mathcal{H}, \boldsymbol{x} \mapsto \boldsymbol{\Phi}(\boldsymbol{x})$. Using the feature space map, the mapped input vectors for a data instance are $\boldsymbol{\Phi}_x = [\boldsymbol{\phi}(\boldsymbol{x}_1)\ \boldsymbol{\phi}(\boldsymbol{x}_2)\ \ldots\ \boldsymbol{\phi}(\boldsymbol{x}_n)]$. The eigenvectors of the data in feature space can be stated in terms of

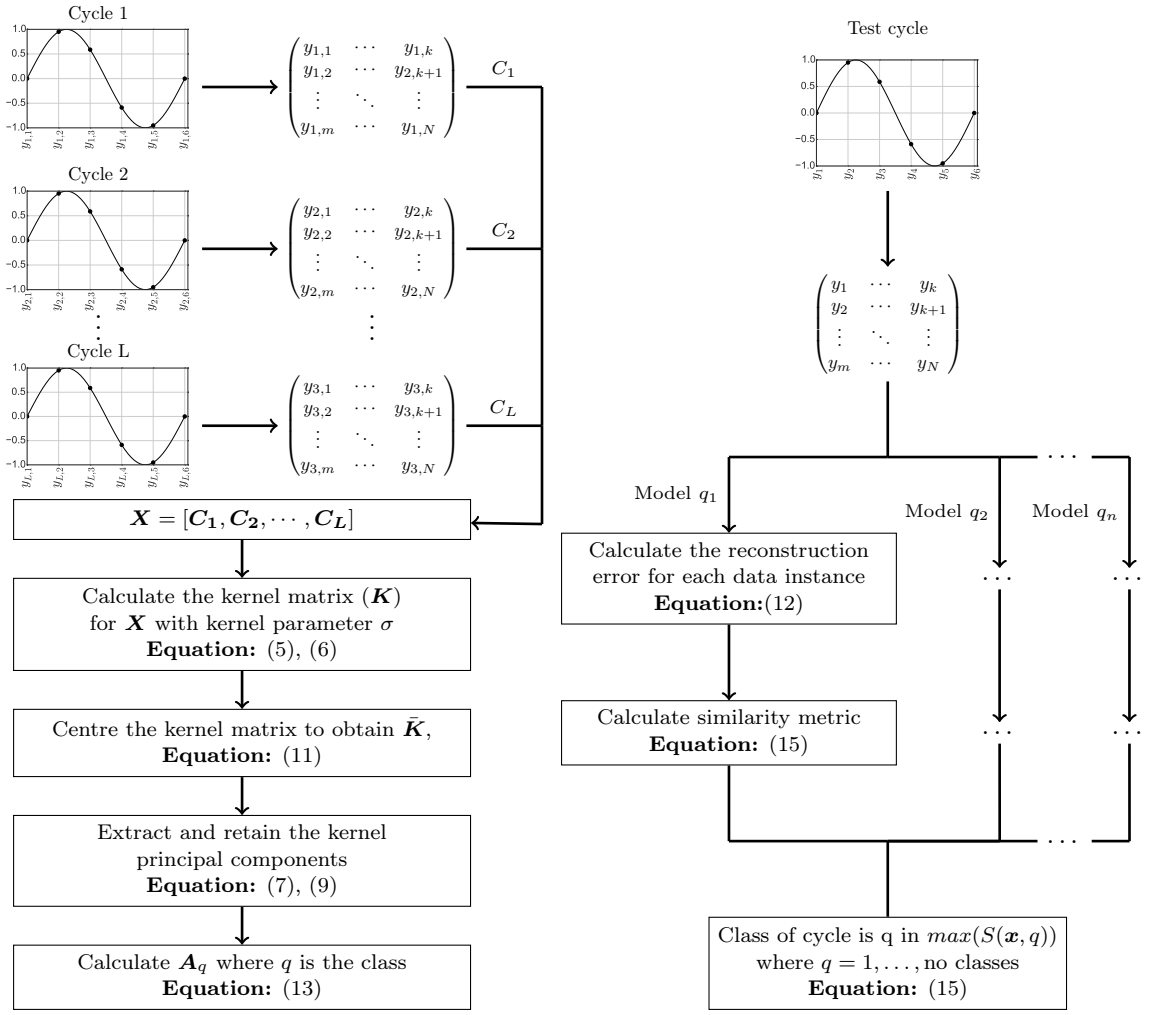

Figure 1: A pictorial representation of the algorithm. (a) Training: Construction of model q_1 using the cycles from class 1. This is repeated for all classes (b) Testing: Each testing cycle is put through each model. The testing cycle is classified as the class of the model which gives the highest similarity metric (15).

the eigenvectors of the kernel matrix. If

$$\begin{aligned} \boldsymbol{K}^x &= \boldsymbol{\Phi}_x^\top \boldsymbol{\Phi}_x \\ &= \boldsymbol{Y}\boldsymbol{\Lambda}\boldsymbol{Y}^\top \end{aligned} \tag{7}$$

and

$$\begin{aligned} \boldsymbol{\Sigma}^x &= \boldsymbol{\Phi}_x \boldsymbol{\Phi}_x^\top \\ &= \boldsymbol{U}\frac{1}{\boldsymbol{\Lambda}}\boldsymbol{U}^\top \end{aligned} \tag{8}$$

and therefore $\boldsymbol{u}^p = \frac{1}{\sqrt{\lambda^p}}\boldsymbol{\Phi}_x \boldsymbol{y}^p$ (see [59] for a detailed explanation). This is represented as

$$\boldsymbol{\alpha}^p = \frac{1}{\sqrt{\lambda^p}}\boldsymbol{y}^p \tag{9}$$

The p^{th} eigenvector of the covariance matrix can be expressed as $\boldsymbol{u}^p = \sum_{k=1}^n \boldsymbol{\alpha}_i^p \boldsymbol{\phi}(\boldsymbol{x}_i)$. Using matrix notation this is $\boldsymbol{u}^p = \boldsymbol{\Phi}_x \boldsymbol{\alpha}^p$.

The projection of a data vector $\boldsymbol{x}$ onto the p^{th} kernel principal component (KPC) is given by

$$\begin{aligned} \langle \boldsymbol{u}^p, \boldsymbol{\phi}(x) \rangle &= \sum_{k=1}^n \boldsymbol{\alpha}_k^p \langle \boldsymbol{\phi}(x_k) \\ \boldsymbol{\phi}(\boldsymbol{x}) \rangle &= \sum_{k=1}^n \boldsymbol{\alpha}_k^p \kappa(\boldsymbol{x}_k, \boldsymbol{x}) \end{aligned} \tag{10}$$

It has so far been assumed that the data are mean-centred in feature space. This might not be the case, however, the kernel eigenvectors can be obtained on data centred in feature space by performing the eigen decomposition on the centred kernel matrix [59] $\bar{\boldsymbol{K}}$, (11).

$$\begin{aligned} \bar{\boldsymbol{K}} &= \bar{\boldsymbol{\Phi}}_{\boldsymbol{x}}^\top \bar{\boldsymbol{\Phi}}_x \\ &= \boldsymbol{K} - \boldsymbol{1}_M \boldsymbol{K} - \boldsymbol{K}\boldsymbol{1}_M + \boldsymbol{1}_M \boldsymbol{K} \boldsymbol{1}_M \end{aligned} \tag{11}$$

Thus a model has been constructed which represents the lower-dimensional manifold on which the training data lies on or close to. This model is then used to perform classification of time series data.

3.3. Distance from the manifold

Once the manifold for the training data has been identified, a metric is required to determine how similar a testing data instance is to the manifold. A distance measure is performed in the manifold subspace, the distance is between the test data instance and its projection onto the manifold. This is often called the reconstruction error [61]. Thus, with one metric, the similarity of a section of the time series to the manifold is determined. The reconstruction error is a measure of how well the test sequence has dynamics that are similar to manifold constructed from the training data set. It is a good metric for measuring similarity and is used for anomaly detection [61, 62].

The reconstruction error (12) is measured by the L2 norm distance between $\bar{\boldsymbol{\phi}}(\boldsymbol{x})$ and its projection onto the KPCs. It has been shown that the L2 norm is fragile in the presence of anomalies in the training data set and the L1 norm has better performance [63]. However, in TSML the model is constructed from training data that does not contain anomalies in the training data set, and therefore the L2 norm is used.

Let $\mathbf{P}$ denote the projection of $\boldsymbol{\phi}(\boldsymbol{x})$ onto the KPCs, where $\mathbf{P}\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{i=1}^{n}(\bar{\boldsymbol{\phi}}(\boldsymbol{x}) \cdot \boldsymbol{u}_p)\boldsymbol{u}_p$.

$$\begin{aligned}\epsilon(\boldsymbol{x}) &= \|\bar{\boldsymbol{\phi}}(\boldsymbol{x}) - \mathbf{P}\bar{\boldsymbol{\phi}}(\boldsymbol{x})\|_2^2 \\ &= \bar{\boldsymbol{\phi}}(\boldsymbol{x}) \cdot \bar{\boldsymbol{\phi}}(\boldsymbol{x}) - \sum(\bar{\boldsymbol{\phi}}(\boldsymbol{x}) \cdot \boldsymbol{u}^p)^2 \\ &= \bar{\kappa}(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{r}(\boldsymbol{x})^\top \boldsymbol{A}\boldsymbol{r}(\boldsymbol{x}) \end{aligned} \tag{12}$$

where $\boldsymbol{r}(\boldsymbol{x}) = \bar{\boldsymbol{\Phi}}^\top \bar{\boldsymbol{\phi}}(\boldsymbol{x})$,

$$\boldsymbol{A} = \sum \boldsymbol{\alpha}^p \boldsymbol{\alpha}^{p\top} \tag{13}$$

and p is the p^{th} KPC. The reconstruction error is a distance metric where the higher the value, the further the test data instance is from the manifold.

During model construction, there are three parameters that need to be set;

- m - embedding dimension
- kpc - the number of dimensions to retain
- σ - width parameter for the kernel function

The parameter tuple of $[m, kpc, \sigma]$ is used to specify the parameters used.

A pictorial representation of the training phase of TSML is provided in Figure 1a and the algorithm is further detailed in Algorithm 1.

3.4. Classification

The classification of cycles is performed using the reconstruction error. For each cycle, there will be $K = n - m + 1$ feature vectors, where n is the number of data instances in a cycle and m is the embedding dimension. Each feature vector is projected onto the manifold and a reconstruction error is calculated using (12). Therefore, for each test cycle there will be K reconstruction error distances. To classify the cycle, the sum of the reconstruction errors is calculated for each model q, see (14). This is used to calculate a similarity metric (15) and the cycle is attributed to the class of the model with the highest similarity metric.

Algorithm 1: Univariate Time Series Manifold Learning

Training Phase
Determine the parameter tuple $[m, kpc, \sigma]$ using, for example, cross-validation
for *q=1,2,..., no. classes* **do**
  Put training data into phase space via (2), (3) using embedding dimension m
  Determine the manifold
  (i) Calculate the kernel matrix, (5), (6) with kernel bandwidth parameter σ
  (ii) Centre the kernel matrix, (11)
  (iii) Extract the kernel principal components, (7), (9)
  (iv) Retain the required number of KPCs
  (v) Calculate $\boldsymbol{A}$ (13)
Testing Phase
for *j=1,2,..., no. test cycles* **do**
  for *q=1,2,..., no. classes* **do**
    Calculate the similarity of the cycle, (14), (15)
  Assign cycle to the class that maximizes similarity, (15)

$$\epsilon(\boldsymbol{x}, q) = \sum_{n=1}^{K} \bar{\kappa}(\boldsymbol{x_n}, \boldsymbol{x_n}) - \boldsymbol{r}(\boldsymbol{x_n})^\top \boldsymbol{A_q}\boldsymbol{r}(\boldsymbol{x_n}) \tag{14}$$

$$S(\boldsymbol{x}, q) = exp\left(-\epsilon(\boldsymbol{x}, q)\right) \tag{15}$$

where $\boldsymbol{x}$ is a feature vector and q is the current model. A pictorial representation of the testing phase of TSML is provided in Figure 1b.

3.5. Extension to a Multivariate Time Series

Previously TSML was applied to a univariate time series. As detailed in Section 2, SSA can be applied to multivariate data using multivariate SSA, also know as multichannel SSA. In order for TSML to operate on multivariate data, it is necessary to incorporate multiple data streams into the Hankel matrix. Therefore, horizontal form SSA [52] is used.

For the analysis of a multivariate time series, the construction of the data matrix differs in order to use the multiple data streams rather than a single data stream. Once the data matrix has been generated, the identification of the manifold continues as stated previously.

Consider a multivariate time series with s channels. For each channel there is a Hankel matrix which forms the data instance matrix which is generated using (2)(3). These channels are combined to form a single matrix.

$$\begin{pmatrix} \boldsymbol{Y_{1,i}} \\ \boldsymbol{Y_{2,i}} \\ \vdots \\ \boldsymbol{Y_{s,i}} \end{pmatrix} = \begin{pmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,k} \\ y_{1,2} & y_{1,3} & \cdots & y_{1,k+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{1,m} & y_{1,m+1} & \cdots & y_{1,N} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,k} \\ y_{2,2} & y_{2,3} & \cdots & y_{2,k+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{2,m} & y_{2,m+1} & \cdots & y_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ y_{s,1} & y_{s,2} & \cdots & y_{s,k} \\ y_{s,2} & y_{s,3} & \cdots & y_{s,k+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{s,m} & y_{s,m+1} & \cdots & y_{s,N} \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \} \text{channel 1} \\ \\ \\ \\ \} \text{channel 2} \\ \\ \\ \\ \} \text{channel s} \end{matrix} \tag{16}$$

$$\boldsymbol{C_l} = [\boldsymbol{Y_1}, \boldsymbol{Y_2}, \cdots, \boldsymbol{Y_K}] \tag{17}$$

where $Y_{1,i}, Y_{2,i}, \ldots, Y_{s,i}$ are the s different channels of the multivariate time series.

This results in a matrix for one cycle where the columns represent the features from the different channels and the rows represent the data instances. As previously, this is repeated for all cycles of a class (18), and the resulting matrices are horizontally stacked in order to provide the data instances for all the cycles.

$$\boldsymbol{X} = [\boldsymbol{C_1}, \boldsymbol{C_2}, \cdots, \boldsymbol{C_L}] \qquad l = 1, \ldots, L \tag{18}$$

The matrix $\boldsymbol{X}$ contains the data instances for all the cycles of a class. The matrix X is block Hankel and is then used to construct the kernel matrix and the kernel eigenspace (KES) as detailed in Section 3

In this section the proposed algorithm, TSML was detailed in its univariate and multivariate form. The algorithm is summarized in Algorithm 1. In the next section, a detailed evaluation of TSML is performed.

4. Evaluation

In this section, the evaluation of TSML is performed on a large number of real-world univariate and multivariate data time series. TSML is evaluated on the problems of classifying periodic cyclostationary data in univariate and multivaritate time series.

4.1. Evaluation environment

The evaluations have some common elements, which will now be detailed. The first step is the preprocessing of data. It is important to preprocess the initial time series [64] and therefore the individual time series are z-normalized. All evaluations using TSML have the same preprocessing on the data in the Hankel matrix, which is mean-centred.

	Application Area	Sequential Data Type	Dataset Name
Univariate	Image Outline	Pseudo time series	Adiac, DiatomSizeReduction, FaceAll, FaceFour, FacesUCR, FiftyWords, Fish, MedicalImages,OSULeaf, SwedishLeaf, Symbols, WordSynonyms
	Spectro	Pseudo time series	Beef, Coffee, OliveOil
	Sensor Classification	Time series	Car, ItalyPowerDemand, Lightning2, Lightning7, MoteStrain, Plane,SonyAIBORobotSurface1, SonyAIBORobotSurface2
	Simulated	Pseudo time series	CBF, MALLAT, SyntheticControl
	Motion	Time series	CricketX, CricketY, CricketZ, GunPoint
Multivariate	Sensor Classification	Time series	Japanese vowels, ECG, Robot Failure LP1, Robot Failure LP2, Robot Failure LP3, Robot Failure LP4, Robot Failure LP5
	Motion	Time series	CMU MOCAP S16, Auslan, WalkvsRun, KickvsPunch, UWaveMTS, Libras, Pen digits, Character Trajectories

Table 1: The univariate and multivariate datasets used in the evaluation, their application area and category of sequential data.

TSML has three parameters which require setting, and therefore parameter optimization is performed in order to determine the parameters which yield the optimal performance. To determine the optimal parameters, a grid search is performed. As in the evaluations for the benchmark algorithms, the number of parameters to search is limited to 100 [32]. For the evaluation of TSML the parameter values chosen to be evaluated using the grid-search are;

- embedding dimension (m) - [0.1,0.3,0.5,0.7 0.9] ratio of the cycle length
- number of KPCs - [10,30,50,70,90]
- σ - [0.1, 1.0, 10.0, 100]

4.2. Univariate Time Series

In this section the performance of the algorithm is evaluated on univariate data sets where there is a single stream of data generated from different processes.

4.2.1. Evaluation Setting

In this section, TSML is evaluated on the performance of univariate classification. The aim is to determine the class of the current test cycle from a closed set of cycles. For this evaluation, time series from the UCR Time Series Classification Archive [42] are used. This contains 86 sequential data sets containing from 2 to 60 classes, with the number of instances ranging from 1400 to 1350000. The datasets in the archive can be divided into two types;

Table 2: Comparison of the performance of TSML and singular benchmark state-of-the-art time series classifiers.

	Time Domain Distance					Diff Distance		Dictionary		Shapelets				Interval			Image Based		Time Delay		
	ED	cDTW [9]	TWE [13]	WDTW [11]	MSM [14]	CID [15]	SMTS [26]	SAX-VSM [16]	BoP [19]	LS [20]	FS [21]	ST [22]	LTS [23]	TSF [24]	TSBF [25]	LPS [57]	RPCD [33]	GGM [35]	GeTeM [41]	**TSML** Proposed	Optimal Classifier
Fiftywords	0.369	0.242	0.187	0.194	0.196	0.226	0.289	0.374	0.466				0.232	0.277	0.209	0.213	0.226	0.301	0.286	**0.200**	TWE
Adiac	0.389	0.391	0.376	0.364	0.384	0.379	0.248	0.417	0.432	0.414	0.514	0.486	0.437	0.261	0.245	0.211	0.384	0.373	0.274	**0.263**	LPS
Beef	0.467	0.467	0.533	0.600	0.500	0.467	0.26	0.233	0.433	0.433	0.447	0.167	0.240	0.300	0.287	0.367	0.260	0.233	0.367	**0.133**	**TSML**
Car	0.267	0.233														0.183				**0.183**	LPS, **TSML**
CBF	0.148	0.004	0.007	0.002	0.012	0.001	0.020	0.004	0.013	0.114	0.053	0.001	0.006	0.039	0.009	0.002		0.009	0.041	**0.006**	CID, ST
CinC ECG torso	0.103	0.070				0.054		0.291		0.301	0.174	0.137	0.167	0.069	0.262	0.064	0.275		0.172	**0.053**	**TSML**
Coffee	0.250	0.179	0.214	0.133	0.236	0.179	0.029	0.000	0.036	0.036	0.068	0.000	0.000	0.071	0.004	0.071	0.000	0.000	0.143	**0.000**	SAX-VSM, ST, LTS, RPCD, GGM, **TSML**
Cricket X	0.426	0.236				0.249		0.308					0.209	0.287	0.278	0.282	0.292		0.259	**0.233**	LTS
Cricket Y	0.356	0.197				0.197		0.318					0.249	0.200	0.259	0.208	0.262		0.285	**0.244**	cDTW, CID
Cricket Z	0.380	0.180				0.205		0.297					0.201	0.239	0.263	0.305	0.292		0.236	**0.231**	cDTW
DiatomSizeRed.	0.065	0.065				0.065		0.121		0.199	0.117	0.127	0.033	0.101	0.126	0.049	0.359		0.065	**0.085**	LTS
ECGFiveDays	0.203	0.203				0.218		0.001		0.006	0.004	0.000	0.000	0.070	0.183	0.155	0.136		0.0120	**0.005**	ST, LTS
FaceAll	0.286	0.192	0.189	0.257	0.189	0.144	0.191	0.245	0.219	0.341	0.411	0.254	0.218	0.231	0.234	0.242	0.190	0.237	0.256	**0.233**	CID
FaceFour	0.216	0.114	0.024	0.136	0.057	0.125	0.165	0.114	0.023	0.511	0.090	0.102	0.048	0.034	0.051	0.040	0.057	0.068	0.034	**0.045**	BoP
FacesUCR	0.231	0.088				0.102		0.109		0.338	0.328		0.059	0.109	0.090	0.098	0.585		0.085	**0.064**	LTS
Fish	0.217	0.160	0.051	0.126	0.080	0.154	0.147	0.017	0.074	0.223	0.197	0.029	0.066	0.154	0.080	0.094	0.126	0.114	0.063	**0.029**	SAX-VSM
GunPoint	0.087	0.087	0.013	0.040	0.060	0.073	0.011	0.013	0.027	0.107	0.061	0.013	0.000	0.047	0.011	0.000	0.000	0.080	0.013	**0.020**	LTS, LPS, RPCD
ItalyPowerDemand	0.045	0.045				0.044		0.089		0.064	0.095	0.052	0.031	0.033	0.096	0.053	0.157		0.079	**0.036**	LTS
Lightning2	0.246	0.131	0.213	0.100	0.164	0.131	0.269	0.213	0.164	0.574	0.295	0.393	0.177	0.180	0.257	0.197	0.246	0.114	0.246	**0.197**	WDTW
Lightning7	0.425	0.288	0.247	0.200	0.233	0.260	0.255	0.397	0.466	0.452	0.403	0.233	0.197	0.263	0.262	0.411	0.356	0.260	0.575	**0.178**	**TSML**
MALLAT	0.086	0.086				0.075		0.035		0.344	0.033	0.074	0.046	0.072	0.037	0.093			0.074	**0.071**	FS
MedicalImages	0.316	0.253				0.258		0.516		0.413	0.433	0.393	0.271	0.232	0.269	0.297	0.290		0.267	**0.422**	TSF
MoteStrain	0.121	0.134				0.205		0.125		0.168	0.217	0.111	0.087	0.118	0.135	0.114	0.203		0.104	**0.074**	**TSML**
OliveOil	0.133	0.167	0.167	0.188	0.167	0.167	0.177	0.133	0.133				0.560	0.100	0.09	0.133	0.167	0.200	0.300	**0.133**	TSBF
OSULeaf	0.483	0.384	0.248	0.372	0.198	0.372	0.377	0.074	0.256				0.182	0.426	0.329	0.134	0.355	0.358	0.141	**0.136**	SAX-VSM
Plane	0.038	0.000														0.000				**0.000**	cDTW, LPS, **TSML**
SonyAIBORobot	0.141	0.141				0.185		0.306		0.140	0.314	0.105	0.103	0.235	0.175	0.225	0.203		0.180	**0.038**	**TSML**
SonyAIBORobot II	0.305	0.305				0.123		0.089		0.154	0.215	0.109	0.082	0.177	0.196	0.123	0.157		0.090	**0.106**	LTS
SwedishLeaf	0.213	0.157	0.102	0.138	0.104	0.117	0.080	0.278	0.198	0.187	0.269	0.118	0.087	0.109	0.075	0.072	0.098	0.065	0.138	**0.070**	GGM
Symbols	0.100	0.062				0.059		0.108		0.357	0.068	0.118	0.036	0.121	0.034	0.030	0.096		0.056	**0.048**	LPS
SyntheticControl	0.12	0.017	0.023	0.002	0.027	0.027	0.025	0.017	0.037	0.53	0.081	0.02	0.007	0.023	0.008	0.027		0.007	0.123	**0.017**	WDTW
Trace	0.240	0.010	0.050	0.000	0.070	0.010	0.000	0.000	0.000	0.000	0.002	0.020	0.000	0.000	0.020	0.020		0.000	0.010	**0.000**	WDTW, SMTS, SAX-VSM, BoP, LS, LTS, TSF, GGM, **TSML**
TwoLeadECG	0.253	0.132				0.138		0.014		0.144	0.09		0.003	0.112	0.046	0.061	0.126		0.004	**0.001**	**TSML**
WordsSynonyms	0.382	0.252				0.243		0.440					0.340	0.381	0.302	0.270	0.276		0.337	**0.303**	CID
No. data sets	34	34	16	16	16	32	16	32	16	25	25	23	32	32	32	34	28	16	32	**34**	
No. optimal	0	3	1	3	0	4	1	4	2	1	1	3	9	2	1	5	2	3	0	**10**	

The error rate is reported. Blank cells indicate the relevant publication did not report results on the data set. The results for the proposed algorithm are in bold font.

time series and pseudo time series [65]. Both a time series and a pseudo time series is an example of a sequential dataset where the data consists of a sequence of values in which the order is important. A time series is a type of sequential data where time forms the longitudinal component. A pseudo time series is a sequential dataset where the order is formed by a component other than time. For example, the image outline data sets are formed by taking a static image, determining the centroid of an object and then selecting a starting point. The sequence of data is then formed by measuring the distance from the centroid to the starting point and the subsequent points as the outline is traced. Table 1 contains further information on the datasets used in the univariate evaluation, including their application area and sequential data type.

For this evaluation, the data sets containing training data sets of less than 30000 data instances were selected. This led to a total of 34 data sets. This criteria was chosen due to the fact that TSML requires the eigen decomposition of the kernel matrix for each class. Those data sets with a large training set required a significant amount of time to perform parameter selection. The UCR Time Series Classification Archive contains data that are z-normalized, the importance of which is detailed by Rakthanmanon *et al.* [64]. Once the embedding has been constructed, the Hankel matrix for the training data set is mean-centred with the same centering being applied to the testing data. The evaluation is performed on the same single train/test split as used in previous evaluations (for example [31], [25], [14], [15], [46], [13], [23]) in order to allow a fair comparison with previous results.

The aim of the evaluation is to compare TSML with other algorithms performing the same task. A grid search across 100 parameters was performed as detailed in Section 4.1. The performance of each parameter tuple was measured using 2-fold stratified cross-validation repeated five times on the training data. The parameter tuple that obtained the optimal performance on the training set was then used on the testing set, with this performance reported.

The scenario corresponds to a known closed set of classes. This is therefore a supervised learning problem where one non-linear model is constructed for each class, and a testing cycle is classified as the class of the model which has the highest similarity metric for the cycle, (15).

The performance of 23 benchmark algorithms is also reported in order to provide comparative performance. The UCR Time Series Classification Archive contains error rates for 1-NN Euclidean Distance and 1-NN with cDTW with the locality constraint tuned by cross-validation [42]. The other benchmark algorithms are taken from current state-of-the-art research which has been detailed in the related work section. The classification error rate for the twenty-three state-of-the-art algorithms is presented in Tables 2

Table 3: Comparison of the performance of TSML and state-of-the-art ensemble time series classifiers.

	TE [29]	PROP [30]	SE [31]	COTE [31]	**TSML** Proposed	Optimal Classifier	TSML Optimal Parameters $[m, KPCs, \sigma]$
Fiftywords	0.352	0.180	0.281	0.191	**0.200**	PROP	[0.5, 30, 10]
Adiac	0.358	0.353	0.435	0.233	**0.263**	COTE	[0.1, 90, 1]
Beef	0.400	0.367	0.167	0.133	**0.133**	COTE, **TSML**	[0.7, 90, 100]
Car		0.167	0.267	0.133	**0.183**	COTE	[0.7, 90, 100]
CBF	0.171	0.002	0.003	0.001	**0.006**	COTE	[0.3, 90, 10]
CinC ECG torso		0.062	0.154	0.064	**0.053**	**TSML**	[0.9, 90, 100]
Coffee	0.214	0.000	0.000	0.000	**0.000**	PROP, SE, COTE, **TSML**	[0.1, 90, 10]
Cricket X		0.203	0.218	0.154	**0.233**	COTE	[0.5, 90, 10]
Cricket Y		0.156	0.236	0.167	**0.244**	PROP	[0.3, 90, 10]
Cricket Z		0.156	0.228	0.128	**0.231**	COTE	[0.3, 90, 10]
DiatomSizeRed.		0.059	0.124	0.082	**0.085**	PROP	[0.1, 90, 100]
ECGFiveDays		0.178	0.001	0.000	**0.005**	COTE	[0.5, 70, 10]
FaceAll	0.281	0.152	0.263	0.105	**0.233**	COTE	[0.7, 90, 10]
FaceFour	0.148	0.091	0.057	0.091	**0.045**	**TSML**	[0.5, 10, 10]
FacesUCR		0.063	0.087	0.057	**0.064**	COTE	[0.7, 30, 10]
Fish	0.194	0.034	0.023	0.029	**0.029**	SE	[0.3, 90, 10]
GunPoint	0.053	0.007	0.020	0.007	**0.02**	PROP, COTE	[0.3, 70, 100]
ItalyPowerDemand		0.039	0.048	0.036	**0.036**	COTE, **TSML**	[0.9, 10, 100]
Lightning2	0.230	0.115	0.344	0.164	**0.197**	PROP	[0.7, 90, 10]
Lightning7	0.301	0.233	0.260	0.247	**0.178**	**TSML**	[0.1, 30, 10]
MALLAT		0.050	0.060	0.036	**0.071**	COTE	[0.9, 90, 10]
MedicalImages		0.245	0.396	0.258	**0.422**	PROP	[0.3, 90, 10]
MoteStrain		0.114	0.109	0.085	**0.074**	**TSML**	[0.1, 90, 0.1]
OliveOil		0.133	0.100	0.100	**0.133**	SE, COTE	[0.1, 90, 0.1]
OSULeaf		0.194	0.285	0.145	**0.136**	**TSML**	[0.1, 90, 100]
Plane		0.000	0.000	0.000	**0.000**	PROP, SE, COTE, **TSML**	[0.1, 90, 1.0]
SonyAIBORobot		0.293	0.067	0.146	**0.038**	**TSML**	[0.1, 30, 100]
SonyAIBORobot II		0.124	0.115	0.076	**0.106**	COTE	[0.3, 10, 10]
SwedishLeaf	0.157	0.085	0.093	0.046	**0.070**	COTE	[0.7, 70, 100]
Symbols		0.049	0.114	0.046	**0.048**	COTE	[0.3, 10, 100.0]
SyntheticControl	0.083	0.010	0.017	0.000	**0.017**	COTE	[0.5, 90, 10]
Trace	0.200	0.0100	0.020	0.010	**0.000**	**TSML**	[0.1, 90, 100]
TwoLeadECG		0.067	0.004	0.015	**0.001**	**TSML**	[0.3, 90, 10]
WordsSynonyms		0.226	0.403	0.266	**0.303**	PROP	[0.7, 50, 10]
No. data sets	14	34	34	34	**34**		
No. optimal	0	9	4	19	**12**		

The error rate is reported. Blank cells indicate the relevant publication did not report results on the data set. The results for the proposed algorithm are in bold font.

Table 4: Pairwise evaluation of TSML and the benchmark classifiers.

Method	Algorithms	p-value	No	Win	Draw	Lose	Win Ratio
Time Domain	ED	0.000	34	31	1	2	0.912
	cDTW	0.004	34	24	2	8	0.706
	TWE	0.020	16	12	0	4	0.750
	WDTW	0.013	16	11	1	4	0.688
	MSM	0.006	16	13	0	3	0.812
Diff Distance	CID	0.010	32	24	0	8	0.750
	SMTS	0.009	16	12	1	3	0.750
Dictionary	SAX-VSM	0.000	32	21	4	7	0.656
	BoP	0.016	16	11	2	3	0.688
Shapelets	LS	0.000	25	23	1	1	0.920
	FS	0.000	25	23	0	2	0.920
	ST	0.002	23	17	2	4	0.739
	LTS	0.424	32	15	3	14	0.469
Interval	TSF	0.001	32	23	1	8	0.719
	TSBF	0.003	32	24	0	8	0.750
	LPS	0.034	34	20	4	10	0.588
Image-based	RPCD	0.000	28	23	1	4	0.821
	GGM	0.022	16	11	2	3	0.688
	GeTeM	0.000	32	27	0	5	0.844
Ensemble	TE	0.001	14	14	0	0	1.000
	PROP	0.852	34	16	3	15	0.471
	SE	0.001	34	21	4	9	0.618
	COTE	0.028	34	8	5	21	0.235

and 3 with the classification error reported being that stated in the relevant publication. All results are rounded to three decimal places to ensure consistency. If a cell is blank, the relevant publication did not evaluate the algorithm on the data set. At the bottom of the table the number of datasets the algorithm was evaluated on, and the number it had equal or superior performance on, are reported for the 23 benchmark algorithms.

4.2.2. Comparison of Performance

The performance of TSML is evaluated using the benchmark algorithms detailed in the Related Work section. The basis of the results are the error rates obtained on the testing data sets which are provided in Tables 2 and 3. In order to provide a robust analysis of the performance of the algorithms, statistical tests were undertaken using the error rates in order to draw conclusions on the performance of the proposed and benchmark algorithms.

For optimal performance, a classifier should be chosen which has the best performance on the dataset at hand. However, it is not known in advance which classifier will obtain the minimum error on a dataset. Therefore, it is more appropriate to choose one classifier that shows the best performance on a wide range of datasets. The pairwise similarity test, Table 4, indicates this. The purpose of this evaluation is to determine the performance of TSML when compared with the performance of another benchmark classifier. The win ratio signifies the ratio between winning (TSML obtains the minimum error) and total number of data sets both classifiers were examined on. The Wilcoxon sign rank test (WSR) is used to test for significant difference at the 1 percent significance level, as used by Bagnall *et al* [31]. The second column presents the p-value for the Wilcoxon sign rank test.

The critical difference and the corresponding critical difference diagram [66] are a Friedman test [67, 68] using the statistic derived by Iman and Davenport [69]. The performance of two classifiers is compared using the Neymenyi test [70], with the performance of two classifiers being significantly different if the corresponding average ranks differ by at least the critical difference. The value of α is set to 0.05 as in previous studies [30]. The average ranks of the classifiers are shown, with the solid horizontal lines grouping classifiers into cliques, within which there is no significant difference in rank.

The first critical difference diagram, Figure 2, compares all the classifiers which are evaluated on all datasets except car and plane. Critical difference diagrams were then created with different methods, such as shapelet, image-related etc, in order to compare performance between method types. Figure 3 shows the critical difference diagram for difference, interval and dictionary classifiers. Figure 4 shows the critical difference diagram of the shapelets, image-related and time-delay classifiers. The final critical differ-

ence diagram, Figure 5, for the UCR dataset examines the performance of the time domain distance and ensemble classifiers.

Having detailed the performance metrics that have been generated in order to examine the performance of TSML, we now proceed with descriptions of the results which occupy the tables and figures. The performance evaluation and comparisons with the benchmark algorithms show that the performance of TSML varies depending on the benchmark algorithm that it is being compared to.

The results show that TSML has superior performance to the singular classifier set [ED, cDTW, LS, FS, RPCD], This performance is shown to be statistically significant in the pairwise evaluation. In addition, the critical difference diagrams show that TSML has a higher average rank to these algorithms that is statistically significant. For example, comparing TSML and the classic time series algorithm of Euclidean distance, Table 2 shows that ED is optimal for no data sets and TSML is optimal for 10. In the pairwise evaluation, TSML has a win ratio of 0.912, with TSML winning 31 and ED winning 2. In the critical difference diagram, Figure 2, TSML has an average rank of 4.6912 compared to 11.3382 for ED. The classifiers occupy different cliques, indicating that this is statistically significant. TSML has better performance that is statistically significant than the shapelets LS and FS. In the pairwise evaluation, TSML has a win ratio of 0.920 for both LS and FS. In addition, shapelets have an average rank of 6.1 and 5.8 respectively, compared to 2.34 for TSML. The classifiers occupy different cliques illustrating that this is statistically significant.

TSML has superior performance to the singular classifier set [MSM, CID, SMTS, SAX-VSM, ST, TSF, TSBF, GeTeM] on the pairwise comparison which is statistically significant. However, for the comparison using critical difference diagrams, although TSML has a lower average rank, this result is not statistically significant as the classifiers occupy the same clique.

For the singular classifier set [TWE, WDTW, BoP, LTS, LPS, GGM], TSML has superior performance. However, the results of the pairwise evaluation and the critical difference diagrams state that this is not statistically significant. For example, the best performer of this set is LTS. In the pairwise comparison TSML has a win ratio of 0.469, with TSML winning 15, LTS winning 14, with 3 drawn. This result is close, with a p-value of 0.424. In the critical difference diagrams, LTS and TSML occupy the same clique. The conclusion we draw is that the performance of these algorithms is similar with no statistical significant difference between them.

TSML has superior performance to the ensemble classifer TE, and this is statistically significant in both the pairwise evaluation and the critical difference diagrams. For the ensemble classifer SE, TSML has superior performance and this is statistically significant in the pairwise evaluation. The win ratio for TSML is 0.618, with TSML winning 21, SE winning 9 and 4 drawn. The critical difference diagram indicates that the average rank of TSML is 2.7206 compared with 3.9706 for SE. However, the classifiers occupy the same clique indicating that there is no statistically significant difference.

For the ensemble classifier, PROP, TSML has superior performance. However, this performance is not statistically significant. TSML has a win ratio of 0.471, winning 16, losing 15 and drawing 1 with a p-value of 0.852. The algorithms occupy the same clique in the critical difference diagram indicating that this result is not statistically significant.

The ensemble classifier, COTE, has superior performance to TSML. This is expected as it is an ensemble that trains 35 classifiers and uses the best combination of classifiers in the evaluation on the testing data. However, the superior performance is not statistically significant. For the pairwise evaluation, the win ratio of TSML is 0.235, winning 8, losing 21 and drawing 5 with a p-value of 0.028. This determines that this result is not statistically significant at the 1 percent significance level. In the critical difference diagrams, although COTE has a higher average rank (2.0441 compared to 2.7206), the classifiers occupy the same clique so there is no statistically significant difference.

The performance of TSML and selected benchmark algorithms are further illustrated in Figure 6. Accuracy, $(1 - error\ rate)$ for TSML and one benchmark algorithm is plotted as a point in Euclidean space. The diagonal line indicates equal performance, with those points lying above representing higher accuracy for TSML and those below representing higher accuracy for the benchmark. Four benchmarks are chosen that are either classical benchmarks or show excellent performance in time series classification. The first comparison is with cDTW, an elastic time domain classifier. When the performance of TSML exceeds that of cDTW it is often by a large amount, the converse is not true of cDTW. The same is true of the comparison with SMTS, where if the performance of SMTS exceeds that of TSML, it is often only by a small margin. The shapelet classifier LTS is an interesting case, there are many datasets where there is similar performance, however there are a few datasets where TSML is significantly superior. This indicates that the classifier is better able to extract discriminatory features with which to classify the testing cycles for these datasets. Finally, COTE is shown to have better performance than TSML. The ensemble of classifiers is able to exploit the different classifiers to produce an ensemble classifier that is able to outperform the single classifier TSML.

4.3. Multivariate Time Series

In this section the performance of the algorithm is evaluated on multivariate data sets where there are multiple streams generated from different processes, with temporal correlation between the streams.

Table 5: Cross-validation error for the multivariate data sets.

			[57]	Oresenigo		[55]	NNDTW		[71]	[54]		TSML Optimal Parameters
	Folds	**TSML**	SMTS	TDVM	SVM	1NN	NoWin	BestWin	Tclass	Motif	Optimal Classifier	$[m, KPCs, \sigma]$
Auslan	10	**0.024**	0.027						0.210		**TSML**	[0.1,30,10]
CMU MOCAP S16	10	**0.000**	0.007							0.480	**TSML**	[0.1,10,100]
ECG	10	**0.090**	0.134				0.189	0.066			BestWin	[0.3,90,10]
Japanese Vowels	10	**0.005**	0.004	0.034	0.054	0.077					SMTS	[0.1,70,100]
Pendigits	10	**0.057**	0.100	0.037	0.066	0.055					TDVM	[0.5,50,100]
Robot Failure LP1	10	**0.034**	0.062	0.148	0.182	0.182					**TSML**	[0.3, 10, 10]
Robot Failure LP2	5	**0.128**	0.384	0.362	0.362	0.404					**TSML**	[0.5,10,10]
Robot Failure LP3	3	**0.191**	0.189	0.319	0.342	0.383					SMTS	[0.3,10,1.0]
Robot Failure LP4	5	**0.060**	0.063	0.145	0.128	0.137					**TSML**	[0.3,90,10]
Robot Failure LP5	5	**0.250**	0.261	0.329	0.379	0.348					**TSML**	[0.3,50,10]
No. data sets		**10**	10	7	7	7	1	1	1	1		
No. optimal		**6**	2	1	0	0	0	1	0	0		

The error rate is reported. Blank cells indicate the relevant publication did not report results on the data set. The results for the proposed algorithm are in bold font.

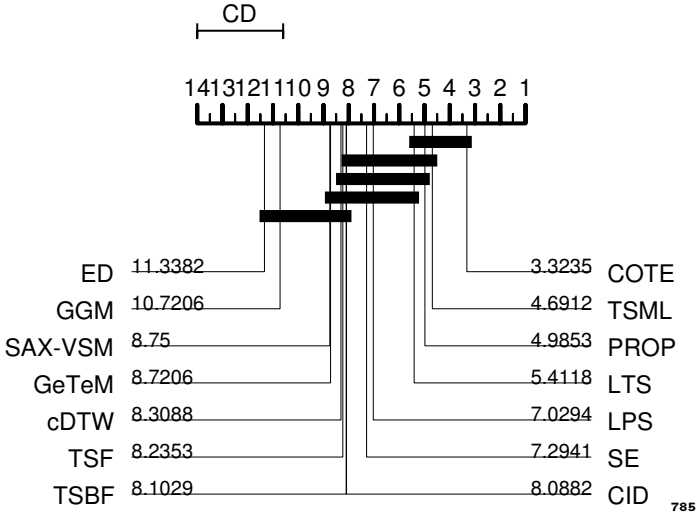

Figure 2: Critical difference diagram of classifiers evaluated on all datasets except car and plane

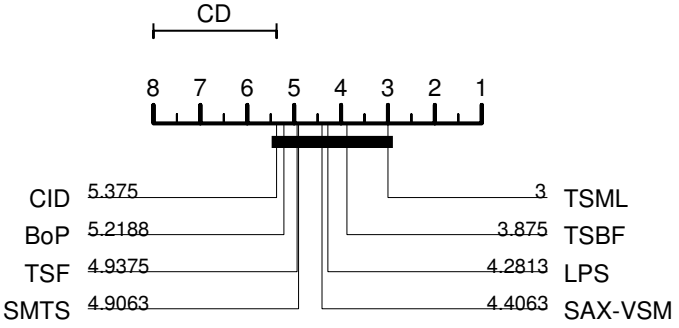

Figure 3: Critical difference diagram of difference, interval and dictionary classifiers.

4.4. Evaluation Setting

For the evaluation of TSML on multivariate time series, 15 datasets used by Baydogan and Runger for the evaluation of SMTS [26] and LPS [57] are used. Information about the datasets can be found in Table 1. In addition, TSML is compared against the performance of a multivariate extension of DTW and TSBF [26]. The DTW distance is calculated as the sum of the distance for each individual stream. For the multivariate extension to TSBF [25] (MTSBF) a representation of each feature is extracted from the time series, these are then concatenated to obtain a final representation. The performance of SMTS, LPS and TSBF on univariate datasets can be found in Table 2.

Two performance comparisons are performed, in order to match the evaluations that were performed by Baydogan and Runger [26, 57]. The first evaluation uses cross-validation and the second a train/test split. For cross-validation, either 5 or 10 folds are used as specified in the evaluation by Baydogan and Runger [26]. The Robot Failure LP3 dataset has a class with only 3 cycles, and therefore the cross-validation was reduced to 3 folds in order to perform stratified cross-validation. For the train/test split, the evaluation is conducted using a grid search across 100 parameters as detailed in Section 4.1. The performance of each parameter tuple was measured using 5-fold stratified cross-validation repeated two times on the training data. The parameter tuple that obtained the optimal performance on the training set was then used on the testing set, with this performance reported.

4.5. Comparison of Performance

The results of the cross-validation evaluation are reported in Table 5 with Table 7 reporting the pairwise eva-

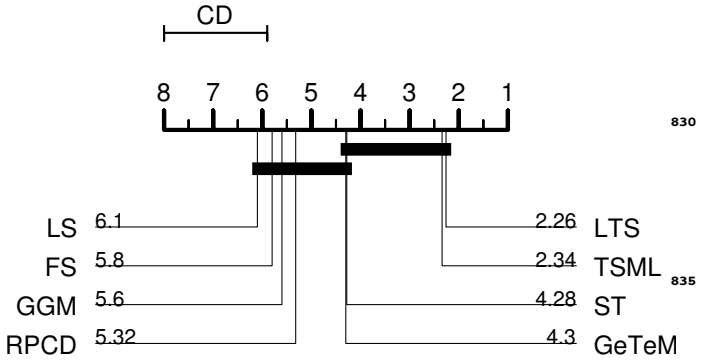

Figure 4: Critical difference diagram of shapelets, image-related and time-delay classifiers.

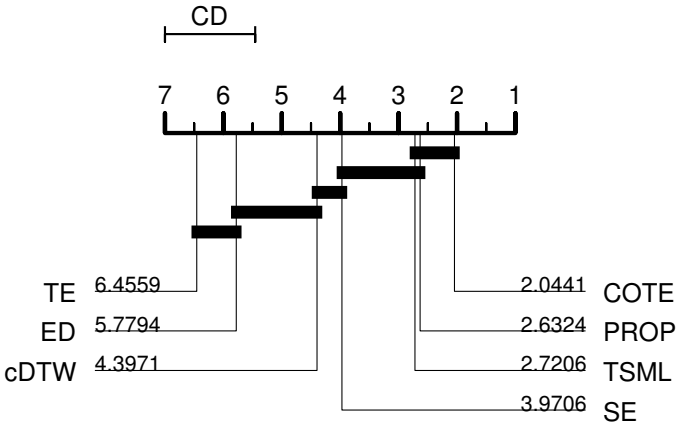

Figure 5: Critical difference diagram of time domain distance and ensemble classifiers.

lution and Figure 7 reporting the critical difference. The evaluation for the train/test split was performed on 15 multivariate datasets, with the results presented in Table 6. The pairwise evaluation is presented in Table 7 and the critical difference diagram in Figure 8.

For the cross-validation evaluation, none of the algorithms exhibit statistically significant at the 1% level for the pairwise evaluation. However, for the classifier set [SVM, 1NN], TSML has superior performance at the 5% level. In addition, the critical difference diagram, Figure 7, shows that TSML has a lower average rank and is in a different clique indicating that this is statistically significant. TSML has superior performance to the classifier set [SMTS, TDVM], at the 5% level for the pairwise evaluation. However these classifiers are in the same clique as TSML in the critical difference diagram so this result is not statistically significant. The remaining classifiers [NoWin,BestWin, Tclass, Motif] were only examined on one data set and thus the p-value is high. TSML has superior performance to all these classifiers, apart from BestWin which was optimal on the data set it was examined on.

For the train/test split of the data, TSML has superior performance to DTW at the 1% level in the pairwise evaluation. The critical difference diagram, Figure 8, shows that TSML has a higher average rank and occupies a different clique therefore this is statistically significant. For the classifier set [SMTS,LPS,MTSBF], TSML has a superior win ratio agaisnt these classifiers. However, this is not a statistically significant result. The critical difference diagram illustrates that [TSML,SMTS,LPS] share the same clique, with MTSBF being in a different clique so only this result is statistically significant.

Table 6: Error on the testing data set for the multivariate data sets.

	TSML	SMTS [57]	LPS [26]	DTW	MTSBF [25]	Optimal Classifier	TSML Optimal Parameters $[m, KPCs, \sigma]$
Auslan	**0.065**	0.053	0.246	0.238	0.000	MTSBF	[0.9,10,100]
Character Trajectories	**0.038**	0.008	0.035	0.033	0.040	SMTS	[0.5,30,10]
CMU MOCAP S16	**0.000**	0.003	0.000	0.069	0.003	**TSML**,LPS	[13,10,100]
ECG	**0.150**	0.182	0.180	0.150	0.165	TSML,DTW	[0.7,70,100]
Japanese Vowels	**0.019**	0.031	0.049	0.351		**TSML**	[0.7,10,10]
KickvsPunch	**0.100**		0.100	0.100		**TSML**,LPS,DTW	[0.5,30,100]
Libras	**0.078**	0.091	0.097	0.200	0.183	**TSML**	[0.3,90,100]
Pendigits	**0.060**	0.083		0.088		**TSML**	[0.5,10,10]
Robot Failure LP1	**0.160**	0.144		0.289		SMTS	[0.1,50,100]
Robot Failure LP2	**0.367**	0.240		0.467		SMTS	[0.5,10,100]
Robot Failure LP3	**0.433**	0.240		0.500		SMTS	[0.1,30,100]
Robot Failure LP4	**0.093**	0.105		0.187		**TSML**	[0.1,270,100]
Robot Failure LP5	**0.400**	0.349		0.480		SMTS	[0.1,10,1.0]
UWaveMTS	**0.034**	0.059	0.020	0.071	0.101	LPS	[0.5,50,100]
WalkvsRun	**0.000**		0.000	0.000		**TSML**,LPS,DTW	[0.1,10,100]
No. data sets	**15**	13	9	15	6		
No. optimal	**8**	5	4	3	1		

The error rate is reported. Blank cells indicate the relevant publication did not report results on the data set. The results for the proposed algorithm are in bold font.

Table 7: Pairwise evaluation of TSML and the benchmark classifiers.

	Algorithm	p-value	No	Win	Draw	Lose	Win Ratio
Cross-validation	SMTS	0.012	10	8	0	2	0.800
	TDVM	0.028	7	6	0	1	0.857
	SVM	0.018	7	7	0	0	1.000
	1NN	0.028	7	6	0	1	0.857
	NoWin	0.317	1	1	0	0	1.000
	BestWin	0.317	1	0	0	1	0.000
	Tclass	0.317	1	1	0	0	1.000
	Motif	0.317	1	1	0	0	1.000
Train/Test	SMTS	0.507	13	7	0	6	0.538
	LPS	0.116	9	4	3	2	0.444
	DTW	0.003	15	11	3	1	0.733
	MTSBF	0.345	6	4	0	2	0.667

4.6. Parameter Selection and Computational Complexity

Thus far, the evaluation of TSML has focused on performance in terms of accuracy. In this section, two other aspects of performance will be examined; sensitivity to parameter selection and computational complexity in terms of time.

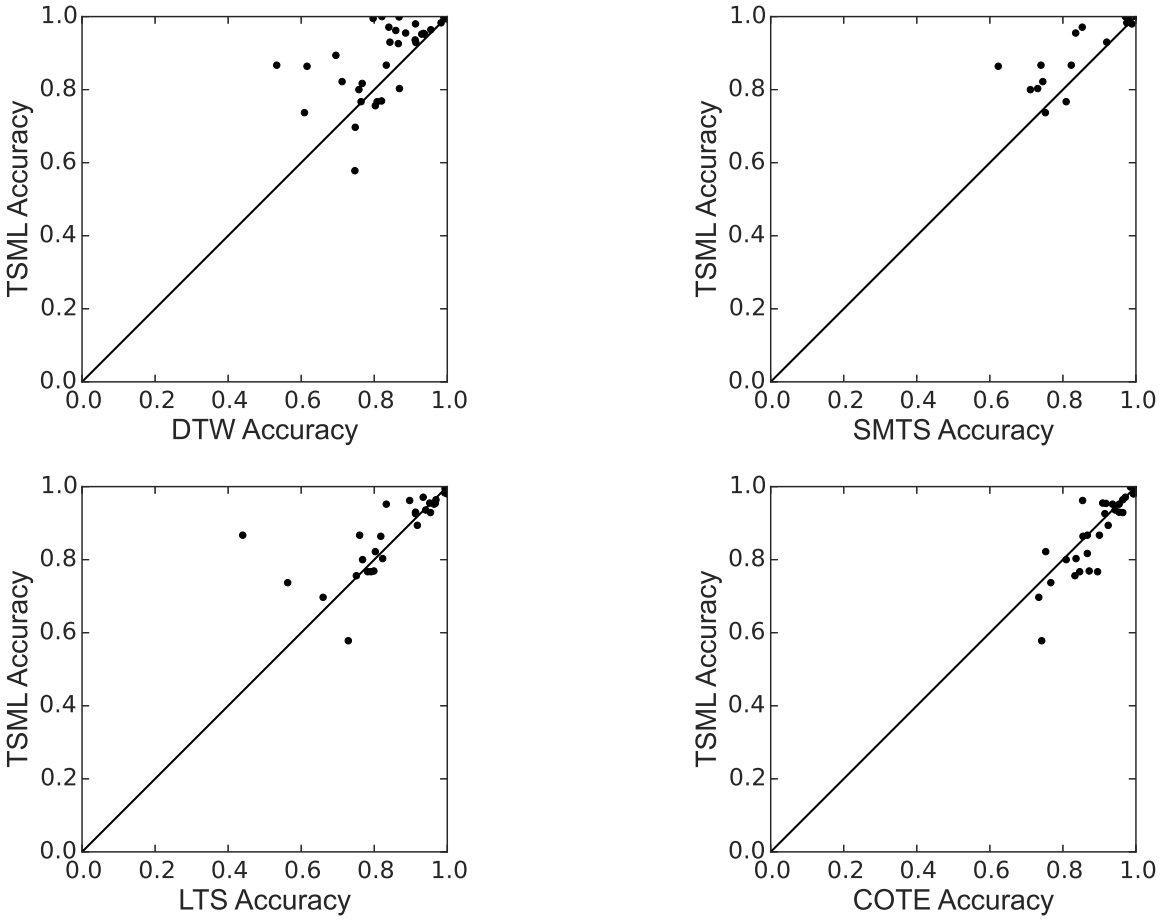

Figure 6: Performance of TSML compared to the benchmarks of Euclidean distance, constrained DTW with tuned locality constraint, SMTS, LTS and COTE. Points occupying the upper-left quadrant mean that TSML has superior performance to the benchmark.

As stated in Section 3.3, TSML requires three parameters to be set; the size of the embedding space (m), the number of kernel principal components retained ($KPCs$), and the kernel width parameter (σ). The sensitivity to these parameters was evaluated on four univariate streams from the UCR Time Series Classification Archive [42]. These datasets provide a varied and reasonable number of training and testing time series.

The sensitivity to parameter selection is examined over the 100 parameters which form part of the grid search for the optimal parameters. The results of the sensitivity analysis is displayed in Figure 9, with the embedding dimension and number of retained KPCs forming the y-axis and σ and the error rates on the train and test sets forming the x-axis. Darker areas indicate poor performance and lighter areas indicate good performance. The optimal parameters identified on the training set and the corresponding testing error are indicated with a black rectangle. A dotted grey rectangle identifies the lowest error on the testing data achieved by TSML.

For three of the data sets, there are large light areas which indicate that several parameter sets would give adequate performance on the testing set. For example, for the *Lightning 7* data set, for $\sigma = 10.0$, there are 5 parameter settings which will give performance which is better than other singular classifiers. The *Beef* dataset has a higher error on the training set, however, it is still able to identify an optimal set of parameters that achieves an error rate on the testing set that is better than other singular classifiers. Aside from this parameter set, there are other parameter settings which achieve an error rate that is equal or better than the next best singular classifier.

Another important aspect of performance is the computational complexity of an algorithm in terms of time. This is often measured in terms of Big O notation in order to give an upper bound for time complexity. TSML requires the construction of a model of each class. During the construction of the model, there are two contributions to the computational complexity. The first is the construction of the kernel matrix, (5), (6) and the second is the eigen decomposition of the kernel matrix (9). In general, the kernel matrix requires $15d^2$ flops [62] to calculate and centre, and $22d^3$ flops [72] (where d is the number of data instances) to perform the eigen decomposition and extract the eigenvalues and eigenvectors. For TSML, the number of data instances derived from the

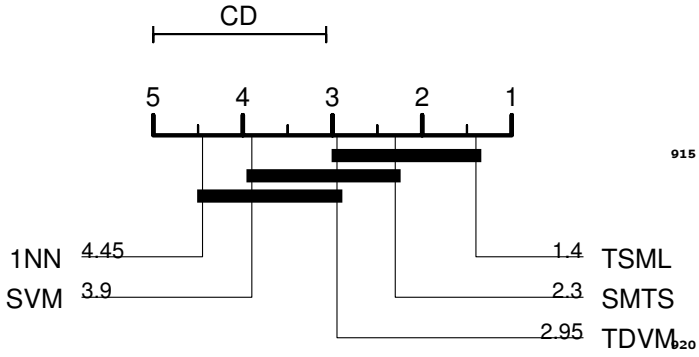

Figure 7: Critical difference diagram for TSML and four other state-of-the-art multivariate classifiers for the cross-validation error. The comparison is performed on the 7 datasets that the algorithms have in common. The other algorithms are omitted as the performance was only examined on 1 dataset.

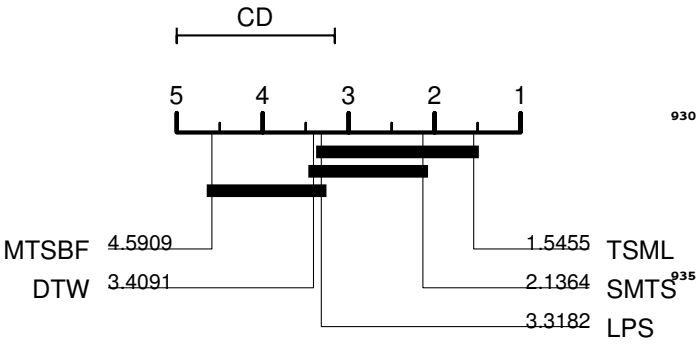

Figure 8: Critical difference diagram for TSML and four other state-of-the-art multivariate classifiers for the train-test split. The comparison is performed on the 6 datasets that the algorithms have in common.

cycles of the time series is dependent on the embedding dimension. The maximum computational complexity for TSML occurs when the minimum embedding dimension is used. The minimum possible embedding dimension is 2, and with this parameter the maximum computational complexity occurs. Therefore, complexity of the construction of the kernel matrix and the eigen decomposition for one model is $15n^2(m-1)^2+22n^3(m-1)^3$ flops. Using Big O notation the upper limit on the computational complexity is of the order $O(n^3m^3)$. TSML requires the construction of one model of each class, therefore the complexity for the algorithm is $O(cn^3m^3)$, where c is the number of classes.

Bagnall *et al.* [32] examine the computational complexity of the benchmark algorithms. Many algorithms, such as MSM and CID, have a computational complexity of $O(n^2m^2)$, which is clearly lower than that of TSML. However, TSML has been shown to be more accurate than these classifiers. Bagnall *et al.* state that ST ($O(n^2m^4)$), LTS ($O(em^2n^2r^2)$) (where e is the maximum number of iterations and r is the shapelet scale) and COTE ($O(n^2m^4)$) are among the slowest algorithms. TSML is also in this category. COTE is an ensemble classifier which is a deployment of 35 different classifiers over four representations. Bagnall *et al.* state that the computational complexity of COTE is due to the most computationally complex classifier in the ensemble. This is the ST classifier, which has complexity $O(n^2m^4)$. The computational complexity of COTE relies on the assumption that the implementation is parallelised [32].

4.7. Discussion

Through extensive analysis using a significant number of real-world data sets, and through extensive statistical analysis, TSML has been shown to have performance that extends the state of the art on univariate and multivariate time series. It has superior performance on a large number of data sets, and can equal performance on others. When compared to methods such as ensembles of time series classifiers, it is also shown to be competitive. However, due to the very nature of ensembles, they can often exceed that of the singular TSML classifier.

The evaluation included a large number of algorithms that operate on either univariate time series data, multivariate time series data, or both. It is important to have a set of criteria when selecting an algorithm. Considering all the evaluation results, both TSML and LTS are good choices for singular time series classification algorithms as both exhibit good performance. TSML and LTS have similar performance, with Table 4 and Figures 2 and 4 indicating that there is no statistically significant difference between the two algorithms. A drawback of LTS is that it does not operate on multivariate data. For classification on both univariate and multivariate data sets, TSML and SMTS are a good choice. TSML exceeds the performance of SMTS on both univariate and multivariate data sets, although the critical difference diagrams show that this is not statistically significant.

The evaluation has shown that ensemble classifiers, such as COTE, usually exceed the performance of singular classifiers, including TSML. However, TSML is still competitive and it also has the advantage that it can operate on multivariate time series.

In summary, the proposed classifier, TSML should be chosen if a high degree of accuracy is required on a large number of diverse univariate and multivariate data sets. It is the opinion of Bagnall *et al.* [32] that new classifiers proposed on the basis of accuracy alone will only be of interest to the research community if it is significantly more accurate than cDTW and not significantly less accurate than COTE. We believe that TSML is in this category and that the evaluation presented illustrates this.

A criticism of some classifiers is that there are too many parameters to tune. TSML has three parameters to determine; the embedding dimension, the kernel width parameter and the number of kernel principal components to retain. This can cause the time to perform cross-validation to take a significant amount of time if there are a large number of data instancs. However, this can also be seen as an advantage if they can be correctly determined. TSML is

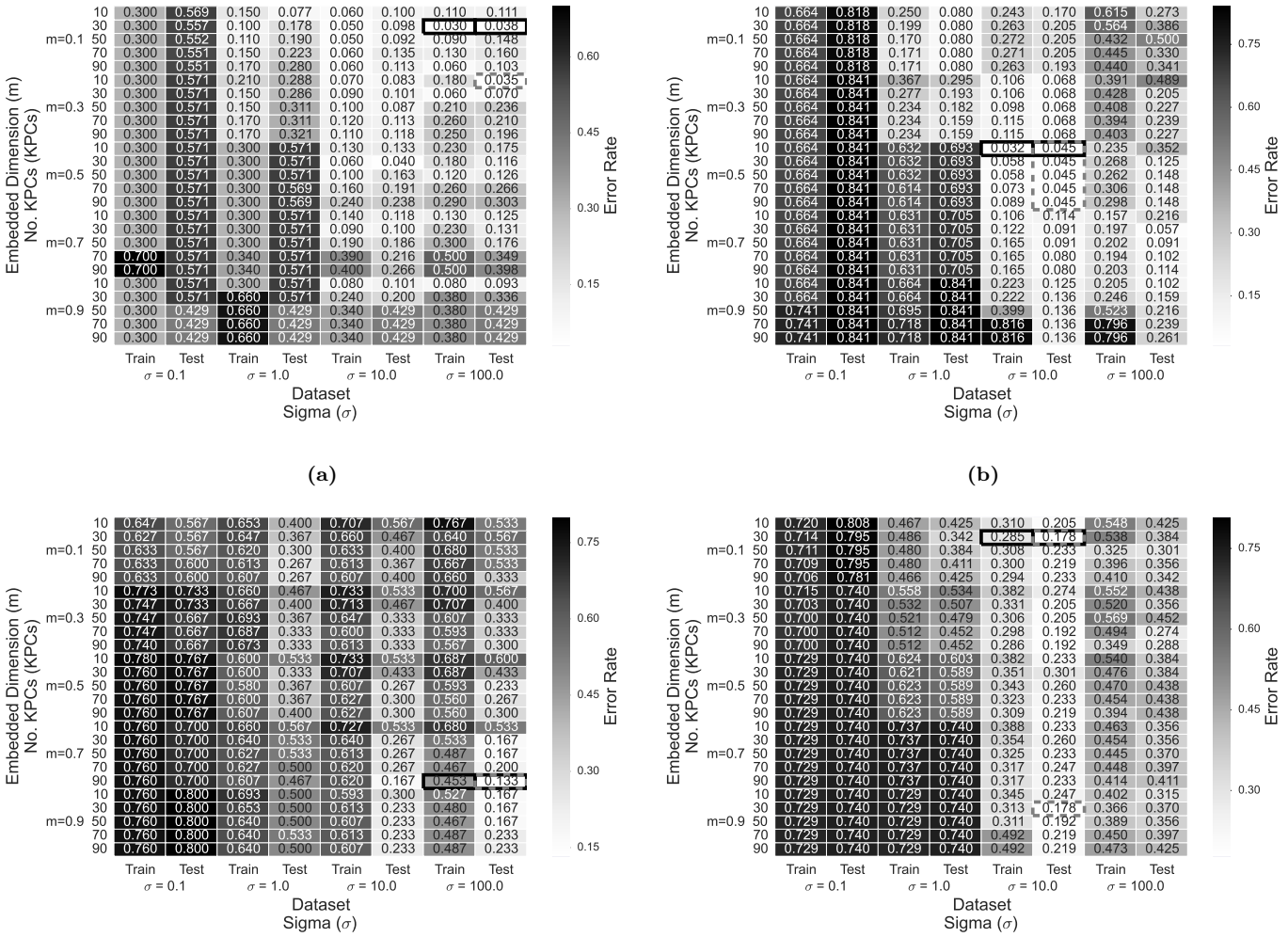

Figure 9: An analysis of the sensitivity of the algorithm to parameter selection. Four datasets which the algorithm has optimal performance on are chosen. (a) SonyAIBORobotSurface. (b) FaceFour. (c) Beef. (d) Lightning 7

able to determine a lower-dimensional manifold on which the data lies on or close to using the parameters of the classifier. In addition, by tuning the number of retained KPCs, noise and redundant features can be removed.

5. Conclusions

In this research, the problem of classifying univariate and multivariate time series was examined. The proposed algorithm, Time Series Manifold Learning (TSML), exploits Takens Embedding theorum to represent a time series as a dynamical system using a phase space. From the phase space a lower-dimensional manifold that the training data lies on or close to is identified using kernel principal component analysis. Once this manifold has been identified, it is used to determine similarity with test data instances using the reconstruction error in order to enable classification. The algorithm is extended to a multivariate time series using horizontal SSA. A block Hankel matrix is constructed using multiple data streams which is then used to obtain a lower-dimensional manifold.

Extensive evaluations were conducted on more than 40 real-world univariate and multivariate time series to determine the performance of the algorithm and compare it to other state-of-the-art methods. The performance of TSML on univariate data was examined on 34 data sets from the UCR Time Series Classification Archive [42]. The algorithm is shown to have statistically significant better performance than other state-of-the-art algorithms such as Euclidean distance and dynamic time warping. It is also shown to exceed the state-of-the-art performance when compared to other singular classifiers, and to have competitive performance when compared to ensembles. Further evaluations were performed on multivariate data where the aim was to classify multivariate time series that could only be classified by looking at a number of data streams. Also in this case, the algorithm showed superior performance to other state-of-the-art methods.

We believe that the method is successful and compet-

itive when compared to other time series classification algorithms for two reasons. The first reason is that the coupling of a phase space and kernel PCA allows the derivation of a lower-dimensional manifold on which the data lies on or close to which accurately models the underlying dynamics of the time series data while removing both noise and redundant features. The second reason is that by projecting a test data instance onto the manifold and measuring the error in its reconstruction, an accurate similarity metric can be defined.

TSML has been shown to have competitive performance on both univariate and multivariate problems, providing it with an advantage over many algorithms that only operate on univariate time series. Future work aims at reducing the computational complexity of the algorithm through selection of the most appropriate data to add to the training data set. In addition, automatic parameter selection will be investigated in order to determine the optimal parameters from the training data set so that parameter optimization is not required. Finally, ensembles have been shown to have excellent performance. Ensembles using shapelets, time domain distance and transformations have been evaluated. It will be interesting to compare them with the performance of ensembles using a time-delay embedding.

Acknowledgement

This paper describes work undertaken in the context of the TagItSmart! project (www.tagitsmart.eu). TagItSmart! is a collaborative project supported by the European Horizon 2020 programme, contract number: 688061.

References

[1] B. Matthews, S. Das, K. Bhaduri, K. Das, R. Martin, N. Oza, Discovering anomalous aviation safety events using scalable data mining algorithms, Journal of Aerospace Information Systems 10 (10) (2013) 467–475.

[2] S. W. Wegerich, Similarity based modeling of time synchronous averaged vibration signals for machinery health monitoring, in: Proceedings of the 2004 IEEE Aerospace Conference, Vol. 6, 2004, pp. 3654–3662.

[3] H. Hassani, S. Heravi, A. Zhigljavsky, Forecasting UK industrial production with multivariate singular spectrum analysis, Journal of Forecasting 32 (5) (2013) 395–408.

[4] H. Hassani, A. S. Soofi, A. Zhigljavsky, Predicting inflation dynamics with singular spectrum analysis, Journal of the Royal Statistical Society: Series A (Statistics in Society) 176 (3) (2013) 743–760.

[5] A. Mellit, A. M. Pavan, M. Benghanem, Least squares support vector machine for short-term prediction of meteorological time series, Theoretical and Applied Climatology 111 (1-2) (2013) 297–307.

[6] M. Jones, D. Nikovski, M. Imamura, T. Hirata, Anomaly detection in real-valued multidimensional time series, in: Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, Academy of Science and Engineering (ASE), USA, 2014.

[7] T.-C. Fu, A review on time series data mining, Engineering Applications of Artificial Intelligence 24 (1) (2011) 164–181.

[8] M. Längkvist, L. Karlsson, A. Loutfi, A review of unsupervised feature learning and deep learning for time-series modeling, Pattern Recognition Letters 42 (2014) 11–24.

[9] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Transactions on Acoustics, Speech and Signal Processing 26 (1) (1978) 43–49.

[10] C. A. Ratanamahatana, E. Keogh, Making time-series classification more accurate using learned constraints, in: Proceedings of the SIAM Conference on Data Mining (SDM), 2004.

[11] Y.-S. Jeong, M. K. Jeong, O. A. Omitaomu, Weighted dynamic time warping for time series classification, Pattern Recognition 44 (9) (2011) 2231–2240.

[12] E. J. Keogh, M. J. Pazzani, Derivative dynamic time warping., in: Proceedings of the SIAM International Conference on Data Mining (SDM), Vol. 1, 2001, pp. 5–7.

[13] P.-F. Marteau, Time warp edit distance with stiffness adjustment for time series matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2) (2009) 306–318.

[14] A. Stefan, V. Athitsos, G. Das, The move-split-merge metric for time series, IEEE Transactions on Knowledge and Data Engineering 25 (6) (2013) 1425–1438.

[15] G. E. Batista, E. J. Keogh, O. M. Tataw, V. M. de Souza, Cid: an efficient complexity-invariant distance for time series, Data Mining and Knowledge Discovery 28 (3) (2014) 634–669.

[16] P. Senin, S. Malinchik, SAX-VSM: Interpretable time series classification using SAX and vector space model, in: Proceedings of the 13th IEEE International Conference on Data Mining (ICDM), 2013, pp. 1175–1180.

[17] P. Patel, E. Keogh, J. Lin, S. Lonardi, Mining motifs in massive time series databases, in: Proceedings of the 2002 IEEE International Conference on Data Mining, 2002, pp. 370–377.

[18] G. Salton, A. Wong, C.-S. Yang, A vector space model for automatic indexing, Communications of the ACM 18 (11) (1975) 613–620.

[19] J. Lin, R. Khade, Y. Li, Rotation-invariant similarity in time series using bag-of-patterns representation, Journal of Intelligent Information Systems 39 (2) (2012) 287–315.

[20] L. Ye, E. Keogh, Time series shapelets: A new primitive for data mining, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 947–956.

[21] T. Rakthanmanon, E. Keogh, Fast shapelets: A scalable algorithm for discovering time series shapelets, in: Proceedings of the 13th SIAM International Conference on Data Mining (SDM), 2013, pp. 668–676.

[22] J. Hills, J. Lines, E. Baranauskas, J. Mapp, A. Bagnall, Classification of time series by shapelet transformation, Data Mining and Knowledge Discovery 28 (4) (2014) 851–881.

[23] J. Grabocka, N. Schilling, M. Wistuba, L. Schmidt-Thieme, Learning time-series shapelets, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 392–401.

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

[25] M. G. Baydogan, G. Runger, E. Tuv, A bag-of-features framework to classify time series, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (11) (2013) 2796–2802.

[26] M. G. Baydogan, G. Runger, Learning a symbolic representation for multivariate time series classification, Data Mining and Knowledge Discovery 29 (2) (2015) 400–422.

[27] L. K. Hansen, P. Salamon, Neural network ensembles, IEEE Transactions on Pattern Analysis & Machine Intelligence (10) (1990) 993–1001.

[28] R. E. Schapire, The strength of weak learnability, Machine learning 5 (2) (1990) 197–227.

[29] A. Bagnall, L. M. Davis, J. Hills, J. Lines, Transformation based ensembles for time series classification, in: Proceedings of the 2012 SIAM International Conference on Data Mining (SDM), Vol. 12, 2012, pp. 307–318.

[30] J. Lines, A. Bagnall, Time series classification with ensembles of elastic distance measures, Data Mining and Knowledge Discovery 29 (3) (2015) 565–592.

[31] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classifica-

tion with COTE: the collective of transformation-based ensembles, IEEE Transactions on Knowledge and Data Engineering 27 (9) (2015) 2522–2535.
[32] A. Bagnall, J. Lines, A. Bostrom, J. Large, E. Keogh, The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances, Data Mining and Knowledge Discovery (2016) 1–55.
[33] D. F. Silva, V. De Souza, G. E. Batista, Time series classification using compression distance of recurrence plots, in: Proceedings of the 13th IEEE International Conference on Data Mining (ICDM), 2013, pp. 687–696.
[34] B. J. Campana, E. J. Keogh, A compression-based distance measure for texture, Statistical Analysis and Data Mining 3 (6) (2010) 381–398.
[35] Z. Wang, T. Oates, Imaging time-series to improve classification and imputation, CoRR abs/1506.00327.
URL http://arxiv.org/abs/1506.00327
[36] F. Takens, Detecting strange attractors in turbulence, in: Dynamical systems and turbulence, Warwick 1980, Springer, 1981, pp. 366–381.
[37] D. S. Broomhead, G. P. King, Extracting qualitative dynamics from experimental data, Physica D: Nonlinear Phenomena 20 (2) (1986) 217–236.
[38] H. Hotelling, Analysis of a complex of statistical variables into principal components, Journal of educational psychology 24.
[39] D. Kugiumtzis, N. D. Christophersen, State space reconstruction: Method of delays vs singular spectrum approach, Research report http://urn. nb. no/URN: NBN: no-35645.
[40] D. Broomhead, G. P. King, On the qualitative analysis of experimental dynamical systems, Nonlinear Phenomena and Chaos 113 (1986) 114.
[41] J. Frank, S. Mannor, J. Pineau, D. Precup, Time series analysis using geometric template matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (3) (2013) 740–754.
[42] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, The UCR time series classification archive, www.cs.ucr.edu/~eamonn/time_series_data/ (July 2015).
[43] A. Aizerman, E. M. Braverman, L. I. Rozoner, Theoretical foundations of the potential function method in pattern recognition learning, Automation and Remote Control 25 (1964) 821–837.
[44] J. Ma, S. Perkins, Time-series novelty detection using one-class support vector machines, in: Proceedings of the IEEE International Joint Conference on Neural Networks, Vol. 3, 2003, pp. 1741–1745.
[45] J. Ma, S. Perkins, Online novelty detection on temporal sequences, in: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 613–618.
[46] Y.-S. Jeong, R. Jayaraman, Support vector-based algorithms with weighted dynamic time warping kernel function for time series classification, Knowledge-Based Systems 75 (2015) 184–191.
[47] D. M. Tax, R. P. Duin, Support vector data description, Machine Learning 54 (1) (2004) 45–66.
[48] Y. Ji, S. Sun, Multitask multiclass support vector machines: model and experiments, Pattern Recognition 46 (3) (2013) 914–924.
[49] A. Teixeira, A. Tome, M. Bohm, C. Puntonet, E. Lang, How to apply nonlinear subspace techniques to univariate biomedical time series, IEEE Transactions on Instrumentation and Measurement 58 (8) (2009) 2433–2443.
[50] A. R. Teixeira, A. M. Tomé, K. Stadlthanner, E. W. Lang, KPCA denoising and the pre-image problem revisited, Digital Signal Processing 18 (4) (2008) 568–580.
[51] B. Liu, H. Chen, A. Sharma, G. Jiang, H. Xiong, Modeling heterogeneous time series dynamics to profile big sensor data in complex physical systems, in: Proceedings of the IEEE International Conference on Big Data, 2013, pp. 631–638.
[52] H. Hassani, R. Mahmoudvand, Multivariate singular spectrum analysis: A general view and new vector forecasting approach, International Journal of Energy and Statistics 1 (01) (2013) 55–83.
[53] T. K. Lee, S. S. Gan, J. Lim, S. Sanei, A multivariate singular spectrum analysis approach to clinically-motivated movement biometrics, in: Proceedings of the 22nd IEEE European Signal Processing Conference (EUSIPCO), IEEE, 2014, pp. 1397–1401.
[54] A. McGovern, D. H. Rosendahl, R. A. Brown, K. K. Droegemeier, Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction, Data Mining and Knowledge Discovery 22 (1-2) (2011) 232–258.
[55] C. Orsenigo, C. Vercellis, Combining discrete SVM and fixed cardinality warping distances for multivariate time series classification, Pattern Recognition 43 (11) (2010) 3787–3794.
[56] E. Keogh, J. Lin, A. Fu, Hot SAX: Efficiently finding the most unusual time series subsequence, in: Proceedings of the 5th IEEE International Conference on Data Mining, 2005, pp. 8–pp.
[57] M. G. Baydogan, G. Runger, Time series representation and similarity based on local autopatterns, Data Mining and Knowledge Discovery (2015) 1–34.
[58] N. Golyandina, A. Zhigljavsky, Singular Spectrum Analysis for time series, Springer Science & Business Media, 2013.
[59] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (5) (1998) 1299–1319.
[60] J. Ham, D. D. Lee, S. Mika, B. Schölkopf, A kernel view of the dimensionality reduction of manifolds, in: Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, p. 47.
[61] H. Hoffmann, Kernel PCA for novelty detection, Pattern Recognition 40 (3) (2007) 863–874.
[62] C. O'Reilly, A. Gluhak, M. Imran, Adaptive anomaly detection with kernel eigenspace splitting and merging, IEEE Transactions on Knowledge and Data Engineering 27 (1) (2015) 3–16.
[63] Y. Xiao, H. Wang, W. Xu, J. Zhou, L1 norm based KPCA for novelty detection, Pattern Recognition 46 (1) (2013) 389–396.
[64] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, E. Keogh, Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping, ACM Transactions on Knowledge Discovery from Data (TKDD) 7 (3) (2013) 10.
[65] X. Xi, E. Keogh, C. Shelton, L. Wei, C. A. Ratanamahatana, Fast time series classification using numerosity reduction, in: Proceedings of the 23rd ACM International Conference on Machine Learning, 2006, pp. 1033–1040.
[66] J. Demšar, Statistical comparisons of classifiers over multiple data sets, The Journal of Machine Learning Research 7 (2006) 1–30.
[67] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (200) (1937) 675–701.
[68] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics 11 (1) (1940) 86–92.
[69] R. L. Iman, J. M. Davenport, Approximations of the critical region of the fbietkan statistic, Communications in Statistics-Theory and Methods 9 (6) (1980) 571–595.
[70] P. Nemenyi, Distribution-free multiple comparisons, Ph.D. thesis, Princeton University (1963).
[71] M. W. Kadous, C. Sammut, Classification of multivariate time series and structured data using constructive induction, Machine learning 58 (2) (2005) 179–216.
[72] X. Liu, U. Kruger, T. Littler, L. Xie, S. Wang, Moving window kernel PCA for adaptive monitoring of nonlinear processes, Chemometrics and intelligent laboratory systems 96 (2) (2009) 132–143.