SoftwareX 25 (2024) 101623

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

LLT: An R package for linear law-based feature space transformation

Marcell T. Kurbucz [a,b,*], Péter Pósfay [a], Antal Jakovác [a]

[a] Department of Computational Sciences, Institute for Particle and Nuclear Physics, HUN-REN Wigner Research Centre for Physics, 29-33 Konkoly-Thege Miklós Street, H-1121 Budapest, Hungary

[b] Institute of Data Analytics and Information Systems, Corvinus University of Budapest, 8 Fővám Square, H-1093 Budapest, Hungary

ARTICLE INFO

Dataset link: http://www.timeseriesclassification.com/description.php?Dataset=PowerCons

Keywords:
R package
Time series classification
Linear law
Feature space transformation
Artificial intelligence

ABSTRACT

The goal of the linear law-based feature space transformation (LLT) algorithm is to assist with the classification of univariate and multivariate time series. The presented R package, called LLT, implements this algorithm in a flexible yet user-friendly way. This package first splits the instances into training and test sets. It then utilizes time-delay embedding and spectral decomposition techniques to identify the governing patterns (called linear laws) of each input sequence (initial feature) within the training set. Finally, it applies the linear laws of the training set to transform the initial features of the test set. These steps are performed by three separate functions called `trainTest`, `trainLaw`, and `testTrans`. Their application requires a predefined data structure; however, for fast calculation, they use only built-in functions. The LLT R package and a sample dataset with the appropriate data structure are publicly available on GitHub.

Code metadata

Current code version	v0.1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-23-00590
Code Ocean compute capsule	Not applicable
Legal Code License	GNU General Public License v3.0
Code versioning system used	Git
Software code languages, tools, and services used	R
Compilation and installation requirements, operating environments & dependencies	R 4.2.2 or later. OS agnostic (Linux, OS X, MS Windows).
Link to developer documentation and user manual	https://github.com/mtkurbucz/LLT/blob/master/README.md
Support email for questions	kurbucz.marcell@wigner.hun-ren.hu

Software metadata

Current code version	v0.1.0
Permanent link to code/repository used for this code version	https://github.com/mtkurbucz/LLT
Legal Code License	GNU General Public License v3.0
Compilation and installation requirements, operating environments & dependencies	R 4.2.2 or later. OS agnostic (Linux, OS X, MS Windows).
Link to developer documentation and user manual	https://github.com/mtkurbucz/LLT/blob/master/README.md
Support email for questions	kurbucz.marcell@wigner.hun-ren.hu

1. Motivation and significance

Over the past decade, time series classification (TSC) has become a crucial task of machine learning and data mining. While its growing popularity is primarily due to the rapidly increasing amount of temporal data collected by widespread sensors [1], TSC is extensively studied across a wide variety of fields, including finance [2–6], activity recognition [7–12], and biology [13–17]. Despite the large effort dedicated

* Corresponding author at: Department of Computational Sciences, Institute for Particle and Nuclear Physics, HUN-REN Wigner Research Centre for Physics, 29-33 Konkoly-Thege Miklós Street, H-1121 Budapest, Hungary.
E-mail addresses: kurbucz.marcell@wigner.hun-ren.hu (Marcell T. Kurbucz), posfay.peter@wigner.hun-ren.hu (Péter Pósfay), jakovac.antal@wigner.hun-ren.hu (Antal Jakovác).

https://doi.org/10.1016/j.softx.2023.101623
Received 4 September 2023; Received in revised form 11 December 2023; Accepted 18 December 2023
Available online 28 December 2023

to this topic, it remains a challenging task due to the nature of time series data, which have large data sizes and high dimensionality and are continuously updated [18–21].

Depending on whether one or more values (features) are observed at a given time, the TSC problem can be defined as a univariate [22–24] or multivariate [25–27] task. In the related literature, a number of approaches have been proposed to solve both tasks, and these approaches can be divided into feature-based and distance-based methods (see, e.g., [27,28]). The most commonly used feature-based methods are the discrete wavelet transform (DWT) [29], wavelet packet transform (WPT) [30], and discrete Fourier transform (DFT) [31], which are used in conjunction with a classification algorithm, where dynamic time warping with the one-nearest neighbor (DTW-1NN) [32] is a typical distance-based approach.

To address TSC problems, various R packages offer diverse functionalities. For instance, the `dtw` package [33] includes a wide variety of algorithms and constraints for the computation and visualization of DTW alignments. Package `pdc` [34] implements a clustering method for time series based on measuring divergence between permutation distributions of the series. The `wavelets` package [35] computes and plots DWTs and the recently published `mlmts` package [36] attempts to provide a set of widespread data mining techniques for multivariate time series.

The recently published linear law-based feature space transformation (LLT) [11] aims to facilitate univariate and multivariate time series classification tasks by transforming the structure of the feature set (or the original time series) to make the data easier to classify. As a first step, this algorithm splits the instances into training and test sets. Then, it applies time-delay embedding and spectral decomposition techniques to identify the governing patterns (called linear laws) of each input sequence (initial feature) within the training set. Finally, it utilizes the linear laws of the training set to transform the initial features of the test set. This transformation procedure has low computational complexity and provides the opportunity to develop a learning algorithm.

This paper presents an R package called LLT, which is the first implementation of the LLT algorithm. This package implements LLT in a flexible yet user-friendly way while using separate functions for each computational step, which facilitates the further development of the algorithm. In addition, it does not rely on functions written by the community, which results in low computational demand. The LLT R package and a sample dataset with the appropriate data structure are publicly available on GitHub [37].

The rest of this paper is organized as follows. Section 2 presents the concept of linear laws and briefly introduces the LLT algorithm. Sections 3 and 4 describe the structure and use of the software in detail. In Section 5, the application of the software is presented on an electric power consumption dataset. Finally, Section 6 discusses the impacts of the software and provides conclusions.

2. LLT algorithm

This section briefly overviews the definition of linear laws and how this concept can be applied to feature space transformation. Note that the LLT algorithm is described in detail by [11], while derivations and proofs related to the linear laws can be found in [38].

2.1. Linear laws of time series

First, consider a generic time series $\boldsymbol{z}_t$ where $t \in \{1,2,\dots,k\}$ represents the time. The lth order ($l \in \mathbb{Z}^+$ and $l < k$) time-delay embedding [39] of this series is defined by:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{z}_1 & \boldsymbol{z}_2 & \cdots & \boldsymbol{z}_l \\ \boldsymbol{z}_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{z}_{k-l} & \cdots & \cdots & \boldsymbol{z}_k \end{pmatrix}. \tag{1}$$

Then, a symmetric $l \times l$ matrix $\boldsymbol{S}$ is generated from $\boldsymbol{A}$ as follows:

$$\boldsymbol{S} = \boldsymbol{A}^\intercal \boldsymbol{A}. \tag{2}$$

The term law in our case implies that we are seeking those weights that transform the values of the $\boldsymbol{S}$ matrix so that they are close to zero; that is, we seek the coefficients ($\boldsymbol{v}$) that satisfy the following equation:

$$\boldsymbol{S}\boldsymbol{v} \approx \boldsymbol{0}, \tag{3}$$

where $\boldsymbol{0}$ is a column vector containing l elements of null value, $\boldsymbol{v}$ is a column vector with l elements and $\boldsymbol{v} \neq \boldsymbol{0}$. To find the $\boldsymbol{v}$ coefficients of Eq. (3), we first perform eigendecomposition on the $\boldsymbol{S}$ matrix. Then, we select the eigenvector that is related to the smallest eigenvalue. Finally, we apply this eigenvector as $\boldsymbol{v}$ coefficients, and hereinafter, we refer to it as the linear law of $\boldsymbol{z}_t$. Note that this logic is related to principal component analysis (PCA) [40,41]; however, in contrast to PCA, we look for components that minimize the variance of the projected data (see [11,38,42]).

2.2. Feature space transformation

Let us consider input data as $\boldsymbol{X} = \{\boldsymbol{X}_t \mid t \in \{1,2,\dots,k\}\}$ sets (time series), where t represents the observation times. The composition of this input data can be expressed as $\boldsymbol{X}_t = \{\boldsymbol{x}_t^{i,j};\mid i \in \{1,2,\dots,n\},\ j \in \{1,2,\dots,m\}\}$, where i denotes the instances and j identifies the different input series (initial features) belonging to a given instance. The output $\boldsymbol{y} \in \{1,2,\dots,c\}$ is a vector that records the classes (c) of instances ($\boldsymbol{y} = \{y^i \in \mathbb{R};\mid i \in \{1,2,\dots,n\}\}$). The aim of the LLT is to provide a new feature space that improves the accuracy of applied classifier algorithms.[1]

During the first step of the LLT algorithm, instances (i) are separated into training ($tr \in \{1,2,\dots,\tau\}$) and test ($te \in \{\tau+1,\tau+2,\dots,n\}$) sets in such a way that ensures that each class is represented in the training set.[2] (For transparency, we assume that the arrangement of the instances within the dataset meets this condition for the tr and te sets.) We then identify the linear law (see $\boldsymbol{v}$ in Eq. (3)) of each input series of the training set ($\boldsymbol{x}_t^{1,1}, \boldsymbol{x}_t^{2,1}, \dots, \boldsymbol{x}_t^{\tau,m}$), thus obtaining a total of $\tau \times m$ laws (eigenvectors). These laws are grouped by input series and classes as follows: $\boldsymbol{V}^j = \{\boldsymbol{V}_1^j, \boldsymbol{V}_2^j, \dots, \boldsymbol{V}_c^j\}$, where $\boldsymbol{V}_c^j$ refers to the laws of the training set associated with input series j and class c.

In the next step, $\boldsymbol{S}^{te,j}$ matrices (see Eq. (2)) are calculated from the input series of the test instance, which results in m matrices per instance (one for each initial feature). We then left-multiply the $\boldsymbol{V}^j$ matrices obtained from the training set by the $\boldsymbol{S}^{te,j}$ matrices of the test set related to the same initial feature ($\boldsymbol{S}^{\tau+1,1}\boldsymbol{V}^1, \boldsymbol{S}^{\tau+1,2}\boldsymbol{V}^2, \dots, \boldsymbol{S}^{n,m}\boldsymbol{V}^m$). The laws of the $\boldsymbol{V}^j$ matrices provide an estimate of whether the $\boldsymbol{S}^{te,j}$ matrices of the test set belong to the same class as them. That is, only those columns of the $\boldsymbol{S}^{te,j}\boldsymbol{V}^j$ matrices that are in proximity to the null vector with relatively small variance, for which the classes of the corresponding training and testing data match.

Finally, the dimension of the resulting matrices is reduced by an f function that selects the column vectors with the smallest variance and/or absolute mean from the $\boldsymbol{S}^{te,j}\boldsymbol{V}^j = \{\boldsymbol{S}^{te,j}\boldsymbol{V}_1^j, \boldsymbol{S}^{te,j}\boldsymbol{V}_2^j, \dots, \boldsymbol{S}^{te,j}\boldsymbol{V}_c^j\}$ matrices for each class, as follows:

$$f(\boldsymbol{S}^{te,j}\boldsymbol{V}^j) = \{\boldsymbol{o}_1^{te,j}, \boldsymbol{o}_2^{te,j}, \dots, \boldsymbol{o}_c^{te,j}\}. \tag{4}$$

For example, the output series $\mathbf{o}_c^{te,j}$ is a column vector of the $\boldsymbol{S}^{te,j}\boldsymbol{V}_c^j$ matrix that has the smallest variance and/or absolute mean. After these calculation steps, the transformed feature space of the test set has $((n-\tau)l)\times(mc)$ dimensions. The calculation steps are illustrated in Fig. 1.

[1] Note that in Section 5, we used LLT without additional classifiers to demonstrate the effectiveness of the performed transformation.

[2] It is not necessary for binary classification problems. In the example, presented in Section 5, classes across both sets were balanced.

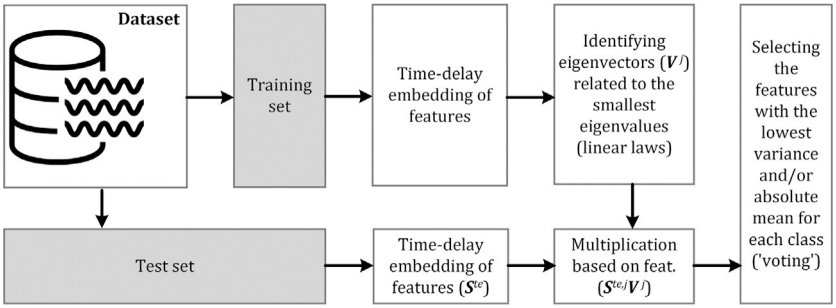


Fig. 1. Steps of the LLT algorithm.

3. Software description

The LLT R package is the first to implement the LLT algorithm. This package contains three main functions (`trainTest`, `trainLaw`, and `testTrans`) and two auxiliary functions (`tdembed` and `linlaw`). The auxiliary functions are called by the main functions, so the user does not need to use them to perform the LLT algorithm.[3]

Description of the main functions[4]*:*

- **`trainTest(path,test_ratio,seed)`** (*trainTest.R*): This function generates a two-level *list* that splits the instances into training and test sets. The first level separates the training and test sets, and the second level groups the instances by class (see Fig. A.1).
- **`trainLaw(path,train_test,dim,lag)`** (*trainLaw.R*): This function creates a *data.frame* containing the set of laws generated from the instances of the training set.
- **`testTrans(path,train_test,train_law,lag, select)`** (*testTrans.R*): This function transforms the instances of the test set by using the LLT algorithm. It generates a *data.frame* object in which columns are new features and rows are the `dim`-length time series created from the test instances and placed one below the other.

Description of the auxiliary functions[4]*:*

- **`tdembed(series,dim,lag)`** (*tdembed.R*): This function generates the $\boldsymbol{S}$ matrix from a time series (see Eq. (2)).
- **`linlaw(series,dim,lag)`** (*linlaw.R*): By applying the `tdembed` function, it generates the law ($\boldsymbol{v}$) of a time series (see Eq. (3)).

Description of the arguments:

- `path` (*character*): The path to the directory that contains the instances grouped by class.
- `test_ratio` (*double* $\in [0, 1]$): Test set ratio.
- `seed` (*integer*): The initial value of the random number seed. By default, it is not fixed.
- `train_test` (*list*): A two-level list that splits the instances into training and test sets. It can be generated by the `trainTest` function or defined by the user manually. Fig. A.1 presents an example of the appropriate structure of this object.
- `dim` (*integer* $\in [2, k]$): It defines the row and column dimension (l) of the symmetric matrix $\boldsymbol{S}$. (The value k is the length of the input series.)
- `lag` (*integer* $\in [1, l]$): It defines the successive row lag of the $\boldsymbol{A}$ matrix. By default, it is 1 (see Eq. (1)). (The value l is the order of the time-delayed embedding.)
- `train_law` (*data.frame*): The set of laws generated from the training instances. It can be generated by the `trainLaw` function. (For development purposes, e.g., for the creation of a learning algorithm, the user can easily modify this *data.frame*.)
- `select` (*character* $\in$ {"rank", "var", "mean"}): New features are defined based on this (f) function (see Section 2.2). The "var" option selects a column vector per class and input series with the smallest variance, while the "mean" option performs this selection based on the minimum absolute mean value. The "rank" minimizes both at the same time by ranking the columns by variance and absolute mean and selecting the column with the smallest sum of ranks. All three selection criteria result in as many new features as the number of classes multiplied by the number of input series. The default value is "rank".

4. Usage

4.1. Installation

The LLT can be installed by using the `devtools` R package as follows.

```
# install.packages("devtools")
# library(devtools)
devtools::install_github("mtkurbucz/LLT")
```

4.2. Data preparation

After installation, the dataset to be transformed must be converted into a data structure in which instances are grouped by classes. Furthermore, time series features must be tab-separated column vectors with the name of the feature in the header. The appropriate data structure is presented in Fig. 2.

4.3. Data transformation

A dataset with the appropriate structure can be transformed in the following way using the LLT package.

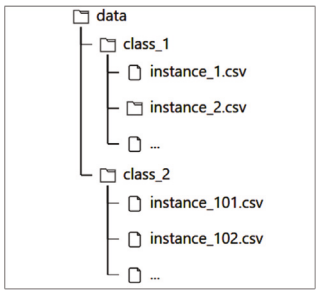
```
# Loading package
library(LLT)

# Setting parameters
path <- "./data"
test_ratio <- 0.30
dim <- 9
seed <- 12345
lag <- 9
select <- "var"

```

[3] The LLT R package and a sample dataset with the appropriate data structure are publicly available on GitHub [37].

[4] Mandatory arguments are indicated by underlining. Each argument is described at the end of this section.



(a) Structure of the 'data' directory

	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6
1	39.67	0.47	8.75	0.43	14.67	0.47
2	39.67	0.47	8.75	0.43	14.67	0.47
3	39.33	0.47	8.00	0.82	14.25	0.43
4	39.67	0.47	7.50	0.50	13.67	1.25
5	39.33	0.47	9.00	0.00	13.50	0.50
6	39.80	0.75	8.40	0.49	13.25	0.83
7	39.33	0.47	7.67	0.47	13.75	0.83
8	39.50	0.50	8.50	0.50	13.75	1.30
9	39.50	0.50	6.75	1.48	13.75	1.09
10	39.50	0.50	1.50	0.50	14.25	0.43

(b) Structure of 'instance_1.csv' file

Fig. 2. Appropriate data structure for 2 classes and 6 features.

	ID	Var1_1	Var1_2	class
1	62.txt	0.079733281	0.042304636	1
2	62.txt	-0.206828213	-0.218476525	1
3	62.txt	0.241362206	0.244597711	1
4	62.txt	-0.206363682	-0.221350028	1
5	62.txt	0.080019230	0.030281274	1
6	34.txt	-0.033704115	0.037350172	1

Fig. 3. Output of the transformation.

```
# Calculation
train_test <- LLT::trainTest(path,seed,test_ratio)
train_law <- LLT::trainLaw(path,train_test,dim,lag)
result <- LLT::testTrans(path,train_test,train_law,lag
    ,select)
```

4.4. Transformed data

The structure of the transformed dataset is presented in Fig. 3. In the visualized case, we had two classes (1 and 2) and one input series (`Var1`). The variable `ID` identifies the input file.

5. Illustrative examples

This section presents a simple example of using the LLT package. In this example, we employ the PowerCons dataset collected by the Research and Development branch of Electricité de France (EDF) in Clamart (France), which is publicly available in the UCR Time Series Classification Archive [43]. It contains the individual household electric power consumption over the course of one year, categorized into two seasonal classes: "Warm" and "Cold", based on whether the power consumption was recorded during the warm seasons (from April to September) or the cold seasons (from October to March). Each instance in the dataset represents a day, with electric power consumption recorded at a sampling rate of ten minutes. Instances are associated with a class and comprise 144 consecutive values. Fig. 4 displays examples of daily power consumption from each class.

Before the transformation, we merged the training and test sets of instances that were previously separated by the authors. Then, we repeated the transformation 300 times based on the `dim = 5` and `test_ratio = 0.1` parameter setting. After each transformation, we calculated the mean absolute value of the resulting features for both classes and as a predicted class, we chose the class whose law resulted in a smaller absolute mean value. Based on the result of the repeated calculation procedure, we obtained an average accuracy of 87.204% with a standard deviation of 5.536%. The histogram of accuracies achieved after each transformation is shown in Fig. 5.[5]

Note that in the case of more difficult classification tasks, it may be worthwhile to compute additional statistics (such as variance) from the new features and then apply a classification algorithm to the obtained feature space. Based on our preliminary calculations (see, e.g., [11]), we achieve the most accurate result with the least computational demand by combining the LLT and the k-nearest neighbor (KNN) [44,45] algorithms.

6. Impact and conclusion

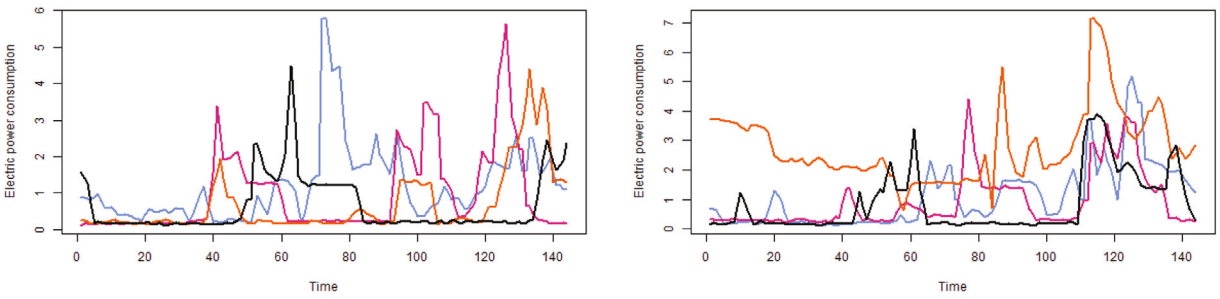
The goal of the linear law-based feature space transformation (LLT) algorithm is to assist with the classification of univariate and multivariate time series. The presented R package, referred to as LLT, offers a versatile and user-friendly implementation of the algorithm. In this paper, we demonstrated the application of the package through an example using the PowerCons dataset [43], where we employed LLT without additional classifiers. Both the package and a sample dataset with the appropriate data structure are publicly available on GitHub [37].

An additional application example is provided by [11], in which the effectiveness of LLT combined with various classifiers is evaluated on a real-world dataset for human activity recognition (HAR), named the Activity Recognition system based on Multisensor data fusion (AReM) [46]. According to the results, LLT vastly increased the accuracy of traditional classifiers, which outperformed state-of-the-art methods after the proposed feature space transformation. A rudimentary version of the LLT R package was also employed and benchmarked in price movement prediction for cryptocurrencies in [47].

In conclusion, the value of the LLT R package can be summarized as follows:

- The LLT package implements the linear law-based feature space transformation (LLT) algorithm in the R programming language.
- The calculation steps are performed by separate functions, which facilitate the further development of the algorithm.
- Despite the flexibility of the package, its functions have been designed in a user-friendly way and require only the most important parameters.
- To maintain low computational requirements, the LLT package only uses built-in functions.

[5] The impact of the dim parameter on the outcome is demonstrated by two additional simulations, whose results are presented in Fig. A.2.



(a) "Warm" (b) "Cold"

Fig. 4. Examples of the time series belonging to each class.

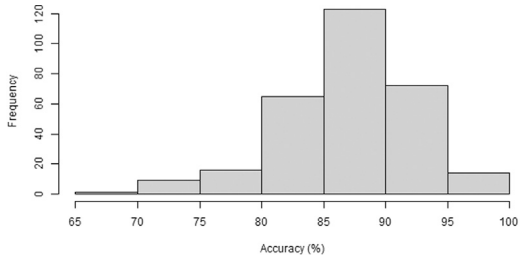


Fig. 5. Histogram of accuracies.

train_test	List of length 2
train	List of length 3
[[1]]	'11.csv' '12.csv' '13.csv' '14.csv' '15.csv' '18.csv' ...
[[2]]	'101.csv' '102.csv' '103.csv' '104.csv' '106.csv' '107.csv' ...
[[3]]	'201.csv' '202.csv' '203.csv' '206.csv' '208.csv' '209.csv' ...
test	List of length 3
[[1]]	'20.csv' '54.csv' '80.csv' '9.csv' '91.csv' '3.csv' ...
[[2]]	'120.csv' '116.csv' '180.csv' '162.csv' '196.csv' '160.csv' ...
[[3]]	'223.csv' '295.csv' '226.csv' '255.csv' '207.csv' '215.csv' ...

Fig. A.1. Example of the structure of `train_test` with 3 classes.

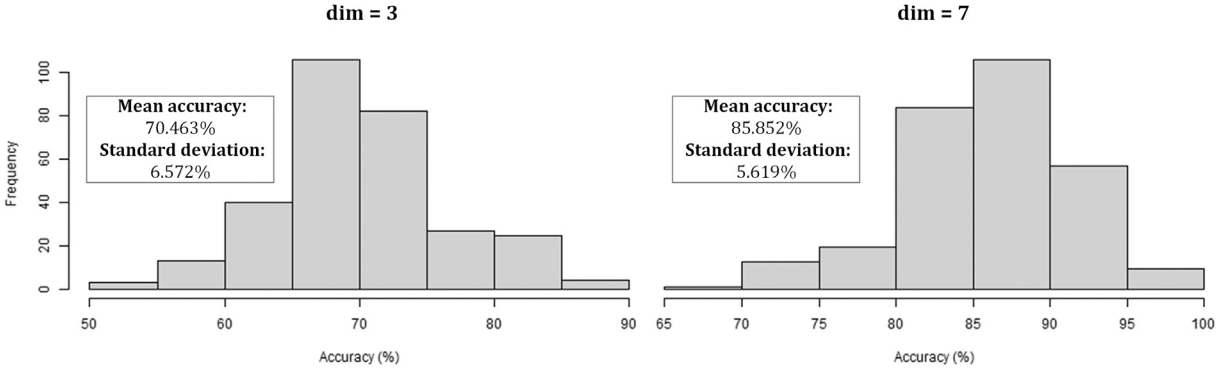


Fig. A.2. Histogram of accuracies for various `dim` parameters (all other settings identical to those in Fig. 5).

CRediT authorship contribution statement

Marcell T. Kurbucz: Conceptualization, Data curation, Formal analysis, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Péter Pósfay:** Conceptualization, Methodology, Supervision, Validation, Writing – original draft, Writing – review & editing. **Antal Jakovác:** Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Marcell T. Kurbucz reports financial support was provided by National Research, Development, and Innovation Office. Antal Jakovac reports financial support was provided by National Research, Development, and Innovation Office.

Data availability

The PowerCons dataset was collected by the Research and Development branch of Electricité de France (EDF) in Clamart (France). It is publicly available in the UCR Time Series Classification Archive (Dau et al., 2018) at http://www.timeseriesclassification.com/description.php?Dataset=PowerCons, retrieved: 22 November 2023.

Acknowledgments

Project no. PD142593 was implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development, and Innovation Fund, Hungary, financed under the PD_22 "OTKA" funding scheme. The research was supported by the Ministry of Innovation and Technology NRDI Office within the framework of the MILAB Artificial Intelligence National Laboratory Program. A.J. received support from the Hungarian Scientific Research Fund (OTKA/NRDI Office) under contract number K123815.

Appendix

See Figs. A.1 and A.2.

References

[1] Marussy K, Buza K. Success: A new approach for semi-supervised classification of time-series. In: International conference on artificial intelligence and soft computing. Springer; 2013, p. 437–47.
[2] Chao L, Zhipeng J, Yuanjie Z. A novel reconstructed training-set SVM with roulette cooperative coevolution for financial time series classification. Expert Syst Appl 2019;123:283–98.
[3] Kwon D-H, Kim J-B, Heo J-S, Kim C-M, Han Y-H. Time series classification of cryptocurrency price trend based on a recurrent LSTM neural network. J Inf Process Syst 2019;15(3):694–706.
[4] Fons E, Dawson P, Zeng X-j, Keane J, Iosifidis A. Evaluating data augmentation for financial time series classification. 2020, arXiv preprint arXiv:2010.15111.
[5] Feo G, Giordano F, Niglio M, Parrella ML. Financial time series classification by nonparametric trend estimation. In: Methods and applications in fluorescence. Springer; 2022, p. 241–6.
[6] Assis CA, Machado EJ, Pereira AC, Carrano EG. Hybrid deep learning approach for financial time series classification. Revista Brasileira de Computação Aplicada 2018;10(2):54–63.
[7] Mocanu DC, Ammar HB, Lowet D, Driessens K, Liotta A, Weiss G, et al. Factored four way conditional restricted Boltzmann machines for activity recognition. Pattern Recognit Lett 2015;66:100–8.
[8] Karim F, Majumdar S, Darabi H, Harford S. Multivariate LSTM-FCNs for time series classification. Neural Netw 2019;116:237–45.
[9] Wang J, Chen Y, Hao S, Peng X, Hu L. Deep learning for sensor-based activity recognition: A survey. Pattern Recognit Lett 2019;119:3–11.
[10] Yang C, Jiang W, Guo Z. Time series data classification based on dual path CNN-RNN cascade network. IEEE Access 2019;7:155304–12.
[11] Kurbucz MT, Pósfay P, Jakovác A. Facilitating time series classification by linear law-based feature space transformation. Sci Rep 2022;12(1):18026.
[12] Vidya B, Sasikumar P. Wearable multi-sensor data fusion approach for human activity recognition using machine learning algorithms. Sensors Actuators A 2022;341:113557.
[13] Schäfer P, Leser U. Multivariate time series classification with WEASEL+MUSE. 2017, arXiv preprint arXiv:1711.11343.
[14] Rajan D, Thiagarajan JJ. A generative modeling approach to limited channel ECG classification. In: 2018 40th annual international conference of the IEEE engineering in medicine and biology society. IEEE; 2018, p. 2571–4.
[15] Elsayed N, Maida AS, Bayoumi M. An analysis of univariate and multivariate electrocardiography signal classification. In: 2019 18th IEEE international conference on machine learning and applications. IEEE; 2019, p. 396–9.
[16] Tripto NI, Kabir M, Bayzid MS, Rahman A. Evaluation of classification and forecasting methods on time series gene expression data. PLoS One 2020;15(11):e0241686.
[17] Bock C, Moor M, Jutzeler CR, Borgwardt K. Machine learning for biomedical time series classification: From shapelets to deep learning. Artif Neural Netw 2021;33–71.
[18] Fu T-c, Chung F-l, Luk R, Ng C-m. Representing financial time series based on data point importance. Eng Appl Artif Intell 2008;21(2):277–300.
[19] Fu T-c. A review on time series data mining. Eng Appl Artif Intell 2011;24(1):164–81.
[20] Zhao B, Lu H, Chen S, Liu J, Wu D. Convolutional neural networks for time series classification. J Syst Eng Electron 2017;28(1):162–9.
[21] Gao J, Murphey YL, Zhu H. Multivariate time series prediction of lane changing behavior using deep neural network. Appl Intell 2018;48:3523–37.
[22] Sun J, Yang Y, Liu Y, Chen C, Rao W, Bai Y. Univariate time series classification using information geometry. Pattern Recognit 2019;95:24–35.
[23] del Campo FA, Neri MCG, Villegas OOV, Sánchez VGC, Domínguez HdJO, Jiménez VG. Auto-adaptive multilayer perceptron for univariate time series classification. Expert Syst Appl 2021;181:115147.
[24] Khan M, Wang H, Riaz A, Elfatyany A, Karim S. Bidirectional LSTM-RNN-based hybrid deep learning frameworks for univariate time series classification. J Supercomput 2021;77:7021–45.
[25] Baydogan MG, Runger G. Learning a symbolic representation for multivariate time series classification. Data Min Knowl Discov 2015;29(2):400–22.
[26] Ruiz AP, Flynn M, Large J, Middlehurst M, Bagnall A. The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances. Data Min Knowl Discov 2021;35(2):401–49.
[27] Hao S, Wang Z, Alexander AD, Yuan J, Zhang W. MICOS: Mixed supervised contrastive learning for multivariate time series classification. Knowl-Based Syst 2023;260:110158.
[28] Susto GA, Cenedese A, Terzi M. Time-series classification methods: Review and applications to power systems data. In: Big data application in power systems. Elsevier; 2018, p. 179–220.
[29] Gupta N, Seethalekshmi K, Datta SS. Wavelet based real-time monitoring of electrical signals in distributed generation (DG) integrated system. Eng Sci Technol Int J 2021;24(1):218–28.
[30] Ray P, Mishra DP. Support vector machine based fault classification and location of a long transmission line. Eng Sci Technol Int J 2016;19(3):1368–80.
[31] Kriegel FL, Köhler R, Bayat-Sarmadi J, Bayerl S, Hauser AE, Niesner R, et al. Cell shape characterization and classification with discrete Fourier transforms and self-organizing maps. Cytometry A 2018;93(3):323–33.
[32] Berndt DJ, Clifford J. Using dynamic time warping to find patterns in time series. In: KDD workshop, vol. 10, no. 16. Seattle, WA, USA:; 1994, p. 359–70.
[33] Giorgino T. Computing and visualizing dynamic time warping alignments in R: The dtw package. J Stat Softw 2009;31:1–24.
[34] Brandmaier AM. pdc: An R package for complexity-based clustering of time series. J Stat Softw 2015;67:1–23.
[35] Aldrich E. wavelets: Functions for computing wavelet filters, wavelet transforms and multiresolution analyses. 2020, Repository: https://cran.r-project.org/web/packages/wavelets/index.html.
[36] López-Oriona Á, Vilar JA. Machine learning for multivariate time series with the R package mlmts. Neurocomputing 2023;537:210–35.
[37] Kurbucz MT, Pósfay P, Jakovác A. LLT R package for linear law-based feature space transformation. 2023, GitHub, URL https://github.com/mtkurbucz/LLT.
[38] Jakovác A. Time series analysis with dynamic law exploration. 2021, http://dx.doi.org/10.48550/ARXIV.2104.10970, arXiv, URL https://arxiv.org/abs/2104.10970.
[39] Takens F. Dynamical systems and turbulence. In: Lecture notes in mathematics, vol. 898. 1981, p. 366.
[40] Pearson K. LIII. On lines and planes of closest fit to systems of points in space. Lond Edinb Dublin Philos Mag J Sci 1901;2(11):559–72.
[41] Hotelling H. Analysis of a complex of statistical variables into principal components. J Educ Psychol 1933;24(6):417.
[42] Jakovác A, Kurbucz MT, Pósfay P. Reconstruction of observed mechanical motions with artificial intelligence tools. New J Phys 2022.
[43] Dau HA, Keogh E, Kamgar K, Yeh C-CM, Zhu Y, Gharghabi S, et al. The UCR time series classification archive. 2018, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.
[44] Fix E. Discriminatory analysis: nonparametric discrimination, consistency properties, vol. 1. USAF School of Aviation Medicine; 1985.
[45] Cover T, Hart P. Nearest neighbor pattern classification. IEEE Trans Inform Theory 1967;13(1):21–7.
[46] Palumbo F, Gallicchio C, Pucci R, Micheli A. Human activity recognition using multisensor data fusion based on reservoir computing. J Ambient Intell Smart Environ 2016;8(2):87–107.
[47] Kurbucz MT, Pósfay P, Jakovác A. Predicting the price movement of cryptocurrencies using linear law-based transformation. 2023, arXiv preprint arXiv:2305.04884.