Data Min Knowl Disc
DOI 10.1007/s10618-014-0364-z

# Invariant time-series factorization

**Josif Grabocka · Lars Schmidt-Thieme**

Received: 22 December 2013 / Accepted: 12 June 2014


**Abstract** Time-series analysis is an important domain of machine learning and a plethora of methods have been developed for the task. This paper proposes a new representation of time series, which in contrast to existing approaches, decomposes a time-series dataset into latent patterns and membership weights of local segments to those patterns. The process is formalized as a constrained objective function and a tailored stochastic coordinate descent optimization is applied. The time-series are projected to a new feature representation consisting of the sums of the membership weights, which captures frequencies of local patterns. Features from various sliding window sizes are concatenated in order to encapsulate the interaction of patterns from different sizes. The derived representation offers a set of features that boosts classification accuracy. Finally, a large-scale experimental comparison against 11 baselines over 43 real life datasets, indicates that the proposed method achieves state-of-the-art prediction accuracy results.

**Keywords** Time-series classification · Time-series factorization · Data mining

## 1 Introduction

The analysis of time-series, including representation and learning, is a challenging branch of machine learning and its existence spans over decades of research. Series

Responsible editors: Toon Calders, Floriana Esposito, Eyke Hüllermeier, Rosa Meo.

J. Grabocka (✉) · L. Schmidt-Thieme
Information Systems and Machine Learning Lab, Samelsonplatz 22, 31141 Hildesheim, Germany
e-mail: josif@ismll.uni-hildesheim.de

L. Schmidt-Thieme
e-mail: schmidt-thieme@ismll.uni-hildesheim.de

Published online: 04 July 2014

data emerge in a myriad of application domains, from health-care and astronomy up to economic and botanic. In comparison to other types of data, time series exhibit a high degree of intra-class variability, where patterns occur shifted in time, distorted and scaled. Therefore traditionally strong classifiers, such as support vector machines (SVM), fail to excel in terms of prediction accuracy (Gudmundsson et al. 2008).

A series of algorithms have been proposed to address the intra-class variations of time-series patterns. An early pioneer method called dynamic time warping (DTW), [still considered competitive (Ding et al. 2008; Rakthanmanon et al. 2012)], computes the similarity among series by re-aligning the time indexes. The algorithm explores all the possible relative alignments of time indexes of two series and picks the one yielding the minimum overall distance (Keogh et al. 2000).

The research of time-series classification can be approximately categorized into distance metrics, invariant classifiers, feature extraction and bag-of-patterns streams. Distance metrics focus on defining measurements on the similarity of two series instances (Keogh et al. 2000; Chen and Ng 2004; Chen et al. 2007; Batista et al. 2014). Invariant classifiers, on the other hand, aim at embedding similarities into classifiers. For instance, the invariant kernel functions have been applied to measure instance similarities in the projected space of a non-linear SVM (Zhang et al. 2010; Gudmundsson et al. 2008). Another paper proposes to generate all pattern variations as new instances and inflate the training set (Grabocka et al. 2012b). The bag-of-patterns approach splits the time-series into local segments and collects statistics over the segments. Those local segments are converted into symbolic words and a histogram of the words' occurrences is built (Lin et al. 2012; Lin and Li 2009). Another study constructs a supervised codebook generated from local patterns, which is used to create features for a random forest classifiers (Baydogan et al. 2013).

This paper introduces a new representation of time series, which can capture patterns that are invariant to shifts and scales. We assume that time series are generated by a set of latent (hidden) patterns which occur at different time stamps and different frequencies across instances. In addition those patterns might be convoluted and/or distorted to produce derived local patterns.

We would like to introduce the concept through the illustration of Fig. 1. A synthetic dataset consists of two classes A (green) and B (black), each having two instances. All the time series are composed of three segments of 20 points, while each segment is a convolutional derivative of two latent patterns depicted in red and blue. In other words, each segment is a weighted sum of a single-peaked and double-peaked pattern. The shown coefficients of the convolution are degrees of membership that each local segment has to one of those two latent patterns.

Both Euclidean and DTW based nearest neighbor classifiers have 100 % error on a leave-one-out experiment on the dataset of Fig. 1. As can be observed, instance A1 is closer to B1 than A2, and the same applies for all other series. In fact the rationale behind this dataset is that A has a higher frequency of the red single-peaked pattern, while B has a higher domination of the blue double-peaked pattern. The method presented in this paper detects the latent patterns, measures the degrees of membership and sums them up into a bag-of-pattern approach. Our approach converts the series of Fig. 1 into a new representation F, concretely: $F_{A1} = [1.9, 1.1]$,

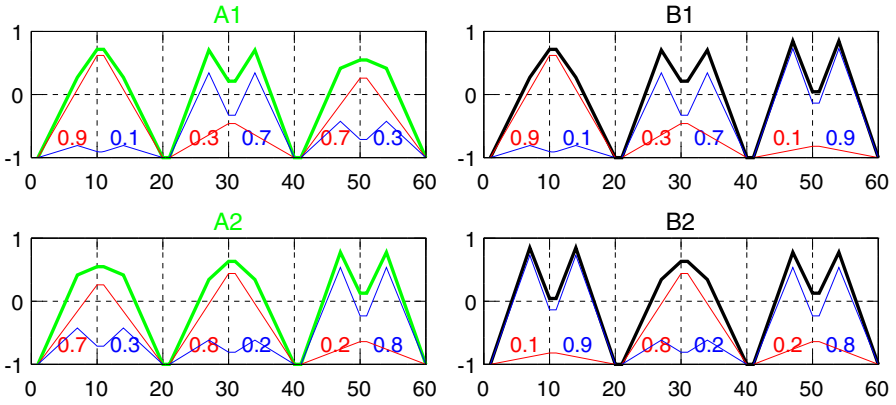


**Fig. 1** Four series of two classes A={A1,A2} and B={B1,B2}, each generated as a convolution of latent patterns

$F_{A2} = [1.7, 1.3]$, $F_{B1} = [1.3, 1.7]$, $F_{B2} = [1.1, 1.9]$. A nearest neighbor classifier over the new representation F yields 0 % error.

In this paper, we will propose a method which detects a set of latent patterns for a time series dataset together with a convolutional degree of membership weights. Such a decomposition is a tailored dimensionality reduction for time-series. The product of the membership weights with the patterns approximates the original segments. In contrast to the aforementioned synthetic example, real datasets have segments occurring at arbitrary locations and being of different sizes. Our method employs a sliding window approach to split the series into overlapping local segments and utilizes a factorization model to decompose the segments into latent patterns and weights. We formalize the objective function of the factorization and propose a stochastic coordinate descent technique in order to optimize the objective. The sum of the learned membership degrees is used to project the time series into a new representation. Ultimately, in order to resolve the scale invariance of the patterns, sums of memberships from different sliding window sizes are concatenated.

Patterns in time-series often occur at different time indices, which is a behavior known as "shifts". The factorization method we propose in this paper, captures weights of patterns in a sliding window segmentation and is invariant to the position of a pattern. In addition, time-series patterns often appear in different scales/sizes. Our method is invariant to the scale of the pattern because we factorize patterns corresponding to various sliding window sizes. Therefore, we name the proposed factorization as "invariant" to shifts and scale variations of patterns.

A thorough experimental comparison is conducted on 43 datasets of the UCR time-series collection against six state of the art baselines. Our method achieves state-of-the-art results in terms of prediction accuracy.

## 2 Related work

Time-series representation and classification has been elaborated on a vast number of occasions, therefore a complete survey of all the published papers is out of our scope.

Instead, we will structure the related work into a set of categories and mention relevant prominent studies.

### 2.1 Distance metrics and invariant classifiers

A significant portion of time-series research has centralized around the definition of accurate similarity metrics. The most popular of those approaches is the DTW measure (Keogh et al. 2000), which overcomes deficiencies of the $L_2$ norm distance by aligning the time indexes of two series instances. The similarity measure is typically plugged into a nearest neighbor classifier. DTW produces competitive prediction accuracies (Ding et al. 2008; Wang et al. 2013) and has been speed up using lower boundary heuristics (Rakthanmanon et al. 2012).

Other similarity based distance metrics have extended the edit distance of strings into the time-series domain (Chen and Ng 2004; Chen et al. 2007). Furthermore, the longest common subsequence of time series has also been used as an indication of similarity (Vlachos et al. 2002). Moreover, similarities of sequential data have been measured using sparse spatial sample kernels (Kuksa and Pavlovic 2010). A state of the art method called complexity-invariant distance metric (CID) introduces the total variation regularization for time-series. CID significantly improves the accuracy of the nearest neighbor classifier with Euclidean distance (ED) (Batista et al. 2011, 2014).

Efforts have been dedicated to incorporating time-series variations into popular classifiers. For instance DTW has been used as a SVM kernel (Gudmundsson et al. 2008), even though the resulting kernel is not positive semi definite. Consecutively, another study has proposed a Gaussian elastic kernel (Zhang et al. 2010). A method which produces a semi-definite kernel is called global alignment kernels and builds an average statistics from all possible warping paths of time indexes (Cuturi 2011). In addition, another study has inflated the training set by adding new instances that represent variations of original training data (Grabocka et al. 2012b).

### 2.2 Feature extraction and bag-of-patterns

Other researchers have emphasized the extraction of series features for boosting classification. Dimensionality reduction has been used to project the time series into a low-rank data space (Keogh et al. 2001), while a recent method incorporates class segregation into the projection (Grabocka et al. 2012a).

However, the most prominent state of the art technique for extracting time-series features is called shapelets mining. Shapelets represent the most discriminative series segment (or set of segments), which yields the maximal prediction accuracy (Rakthanmanon and Keogh 2013; Mueen et al. 2011). A related study detects a set of shapelets and transforms the series data into a new representation, defined by the distance to those shapelets (Hills et al. 2013).

A recent direction of research has drawn attention on the need to segment the time series into local patterns and measure the frequencies of patterns as classification features. For instance frequencies of time-series motifs have been fed into standard classifiers (Buza and Schmidt-Thieme 2010). Another attempt has focused on building

histograms of local patterns represented as symbolic words (Lin and Li 2009). Those symbolic words are produced by a piecewise constant approximation technique called SAX (Lin et al. 2007), while the frequencies of the SAX words are used ultimately for classification (Lin et al. 2012; Lin and Li 2009). One similar bag-of-words approach has also been applied to long bio-medical data (Wang et al. 2013). Moreover, a bag-of-patterns study proposes to extract series segments of various lengths and positions and generate a supervised codebook of those patterns (Baydogan et al. 2013). A random forest classifier has been trained over the extracted features. That study demonstrates considerable improvements over baselines in terms of prediction accuracy (Baydogan et al. 2013).

### 2.3 Factorization of time series

There have been a few attempts in generating invariant time-series features through factorization. A shift-invariant sparse coding of signals has been proposed for reconstructing noisy or missing series segments (Lewicki and Sejnowski 1999). In similar domains, sparse coding factorization has been applied for deriving shift and 2D rotation invariant features of hand writing data (Barthelemy et al. 2012), and also invariant features of audio data (Huang et al. 2012). Moreover, a temporal decomposition of multivariate streams has been used to discover patterns in patients' clinical events (Wang et al. 2012).

Our method is fundamentally different from distance metrics. Instead of measuring the similarity of series, we project the data into a new representation, where similar instances are positioned close to each other. Furthermore, the proposed method is different to from existing bag-of-patterns methods because we learn a latent decomposition of patterns, instead of counting the occurrence of segments on the original time-series. Finally, our contributions over the existing factorization methods rely on (i) a novel approach in detecting both shift and scale invariant features for time series, and (ii) building a bag-of-patterns representation of the learned invariant features for a classification scenario.

## 3 Definitions and notations

1. **Time-series** A time-series is an ordered sequence of point values. In a dataset of $N$ series instances, where each series has $Q$ points, we denote the series dataset as $T \in \mathbb{R}^{N \times Q}$.
2. **Sliding Window Segment** A sliding window content of size $L \in \mathbb{N}$, is a series subsequence starting at a position $j \in \{1, \ldots, Q - L\}$ of a series $i$ of dataset T, and is denoted as $S_{i,j} \in \mathbb{R}^L$, $S_{i,j} := \left(T_{i,j}, T_{i,j+1}, \ldots, T_{i,j+L-1}\right)$.
3. **All Dataset Segments** The starting position of each sliding window segment is incremented by an offset $\delta = \{1, \ldots, L\}$, therefore the maximum number of segments per series is defined as $M := \frac{Q-L}{\delta}$. All the segments of a time-series datasets are denoted as $S \in \mathbb{R}^{N \times M \times L}$.

4. **Latent Patterns** Our method mines for $K$-many latent patterns, each having the same size as one segment, i.e $L$. So, the latent patterns are denoted as $P \in \mathbb{R}^{K \times L}$.
5. **Degrees of Membership** Each instance of a dataset will be approximated via the product of latent patterns and the set of membership degrees to those patterns. Each segment of a series will have one membership weight to each of the $K$ latent patterns. Consequently, the degrees of membership of all time-series are defined as $D \in \mathbb{R}^{N \times M \times K}$.

## 4 Invariant factorization of time series

The method presented in this paper is a new feature representation for time series data. The representation reduces the dimensionality of the original series by factorizing the series data into a set of patterns and weights of segments to those patterns. The sum of weights over all the sliding window segments of a time-series is the new feature vector that our paper constructs. The factorization process is fully unsupervised and doesn't take into account the label information. However, the derived representation provides a set of features that boost the classification accuracy of standard classifiers, because the new representation is invariant to shifts and scales of patterns. This section will walk the reader through the details of our method, including the learning algorithms.

### 4.1 Segmentation of time series

As a first step, the series of the dataset are segmented in a sliding window approach having size $L$ and increment $\delta$. The segmentation of each series is described in Algorithm 1. Once derived, the segments are normalized to mean 0 and deviation 1.

**Algorithm 1** SegmentSeries

**Require:** $T \in \mathbb{R}^{N \times Q}, L \in \mathbb{N}, \delta \in \mathbb{N}$
**Ensure:** $S \in \mathbb{R}^{N \times M \times L}$
$M \leftarrow \frac{Q-L}{\delta}$
**for** $i = 1, \ldots, N,\ j = 1, \ldots, M$ **do**
  **for** $l = 1, \ldots, L$ **do**
    $S_{i,j,l} \leftarrow T_{i,\delta(j-1)+l}$
  **end for**
  $S_{i,j} \leftarrow \text{normalize}(S_{i,j})$
**end for**
**return** $S$

### 4.2 From the problem definition to an objective function

The definition of our problem is to learn degrees of memberships $D \in \mathbb{R}^{N \times M \times K}$ and the patterns $P \in \mathbb{R}^{K \times L}$ that reconstruct/approximate the segments $S \in \mathbb{R}^{N \times M \times L}$. A linear convolution $DP$ is used as the model that reconstructs the segments. The loss

of reconstruction is measured using the squared-error (L2) error, while two different types of regularization are applied on $D$ and $P$.

Therefore, the objective function is defined as a regularized loss and is described in Eq. 1. The solution of the objective function returns the optimal $D$, $P$ matrices that cause the loss achieve its minimal value.

$$\underset{D,P}{\operatorname{argmin}} \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{L}\left(S_{i,j,l} - \sum_{k=1}^{K} D_{i,j,k} P_{k,l}\right)^2 + \lambda_P \sum_{k=1}^{K}\sum_{l=1}^{L} P_{k,l}^2 \tag{1}$$

Subject To:

$$\sum_{k=1}^{K} D_{i,j,k} = 1, \quad D_{i,j,k} \geq 0, \quad \forall i, j, k$$

The objective function is composed of two loss terms and one constraint. Firstly, the latent patterns $P$ and the memberships $D$ should approximate the normalized segments of the series dataset. Therefore, minimizing the L2 norm of the reconstruction error achieves the goal. In addition, a second regularization loss term is added in order to prohibit the patterns $P$ from over-fitting. A hyper-parameter $\lambda_P$ controls the degree of regularization. Finally, we impose equality and positivity constraints on the membership degrees. The membership degrees of every segment $D_{i,j}$ sum-up to one, because each segment needs to have the same impact factor. Otherwise, in a bag-of-patterns representation of series, different segments would have different scales of memberships. The positivity constraint, on the other hand, prohibits non-interpretable negative memberships.

We would like to illustrate the invariant factorization objective with a concrete illustration, shown in Fig. 2. A learned decomposition, as in Eq. 1, is depicted for the Gun Point dataset. On the left top, a series instance is presented, while the dataset's latent patterns and the membership degrees of the instance are found below. The product of the patterns and memberships yield the series approximation shown in the right top chart. The series is split into eight overlapping segments of size 45, each starting at an offset of 13 points. For instance, the 7-th segment starts at 79 and has a high membership value to the 6-th pattern, which matches the descending structure. However, please note that other patterns also contribute with smaller membership degrees (patterns 4 and 5) in order to fit exactly the original segment content.

### 4.3 Learning the patterns and memberships

In order to learn the latent patterns and the memberships we are going to optimize the objective function of Eq. 1 via **stochastic coordinate descent**, which operates by updating each cell of $D$, $P$ in the direction of the first derivative of the objective.

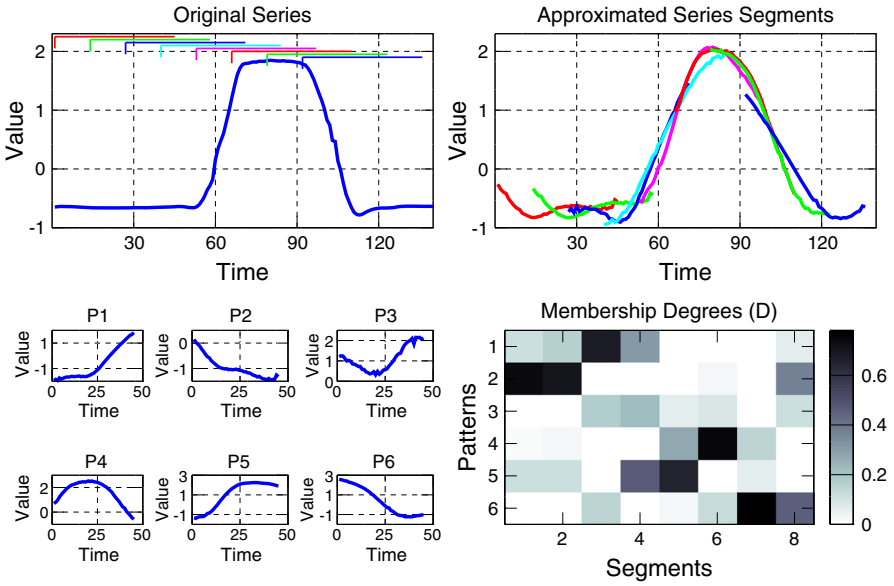


**Fig. 2** A factorized instance of the gun point dataset with parameters $K = 6$, $L = 45$, $\delta = 13$, $\lambda_P = 1$

### *4.3.1 Update rules for latent patterns*

In order to compute the update rules for the patterns, we first define in Eq. 2 the error in approximating a point $l$ of the segment $j$, in time-series $i$, as $\xi_{i,j,l}$. A stochastic coordinate descent optimization fixes the error of approximating single points $S_{ijl}$ and is different to gradient approaches that consider the full error.

$$\text{Let } \xi_{i,j,l} := S_{i,j,l} - \sum_{k=1}^{K} D_{i,j,k} P_{k,l} \tag{2}$$

The optimization technique learns the optimal values of the patterns and membership weights, which eliminate the error of each point $\xi_{i,j,l}$. In the case of a pattern point $P_{k,l}$, the optimal value can be found by isolating the residual that the point contributes from the error. Such an optimization technique is named as "Coordinate Descent" and is popular for the factorization community (Yu et al. 2012). In our context, we try to isolate the residual error of $P_{k,l}$ by introducing a placeholder variable $z$ as is shown in Eq. 3. The optimal value of $z$ that minimizes the error $\xi_{i,j,l}$, subject to the regularization, is denoted as $P^*_{k,l}$

$$P^*_{k,l} := \underset{z}{\text{argmin}} \left( \lambda_P z^2 + \sum_{i,j} \left( \xi_{i,j,l} + P_{k,l} D_{i,j,k} - z D_{i,j,k} \right)^2 \right) \tag{3}$$

Subsequently the optimal value of every point $l$ of a latent pattern $k$ (denoted as $P^*_{k,l}$) is found by solving the first derivative as presented in Eq. 4.

$$2\lambda_P P^*_{k,l} - 2\sum_{i,j}\left(\xi_{i,j,l} + D_{i,j,k}(P_{k,l} - P^*_{k,l})\right) D_{i,j,k} = 0 \tag{4}$$

Therefore the optimal value $P^*_{k,l}$ is defined in Eq. 5 as a derivation of Eq. 4. Please note that our learning algorithm will iterate through the error of all the segment points $\xi_{i,j,l}$ and then update all $P_{k,l}$ cells to the optimal value with respect to the $\xi_{i,j,l}$.

$$P^*_{k,l} := \frac{\sum_{i,j}\left(\xi_{i,j,l} + D_{i,j,k}P_{k,l}\right) D_{i,j,k}}{\lambda_P + \sum_{i,j} D^2_{i,j,k}} \tag{5}$$

Please note that the error values don't have to be recomputed for each point over all latent patterns, instead we can incrementally update the error terms (Yu et al. 2012). Equation 6 refreshes the error terms after the change of the pattern value.

$$\xi_{i,j,l} \leftarrow \xi_{i,j,l} - (P^*_{k,l} - P_{k,l})D_{i,j,k} \tag{6}$$

#### *4.3.2 Update rules for membership degrees*

The update rules for the membership degrees needs to preserve an equality constraint, which enforce the memberships of a segment to sum to one. Therefore any direct update of a membership $D_{i,j,k}$ will violate the constraint. In order to avoid this bottleneck, we propose to update the memberships in pairs, inspired by a similar strategy known as the Sequential Minimal Optimization algorithm (Platt 1999). The idea is to draw two random membership weights $D_{i,j,k}$, $D_{i,j,w}$ and update them such that their sum, denoted $Q = D_{i,j,k} + D_{i,j,w}$, remains equal before and after the updates. In that way, if we increase one membership, then the other would have to decrease and vice versa, while the aim is to find the combination which yield the smallest approximation error.

Since the value of $D_{i,j,k}$ depends on the other value of the pair $D_{i,j,w}$, then we can no longer find the minimum value of the points separately. The optimization should find the combination of pair values that both minimize the residual error $\xi_{i,j,l}$ of approximating a segment point $S_{i,j,l}$. Therefore, the optimal value of $D_{i,j,k}$, will be denoted by $D^*_{i,j,k}$ and can be solved by isolating the residual error of both values by introducing a variable $z$ and creating an optimization sub-problem (Yu et al. 2012). Since the sum of the pair of membership values is bound to $Q$, then we can replace the optimal value of $D_{i,j,w}$ as $Q - z$. The resulting optimal value is expressed in Eq. 7.

$$D^*_{i,j,k} = \underset{z}{\operatorname{argmin}} \sum_{l}\left(\xi_{i,j,l} + D_{i,j,k}P_{k,l} + D_{i,j,w}P_{w,l} - QP_{w,l} + z(P_{w,l} - P_{k,l})\right)^2 \tag{7}$$

The solution of Eq. 7 can be algebraically derived as the solution of the first derivative and is presented in Eq. 8. Our forthcoming learning algorithm will update pairs

of membership weights for all the points of series segments, in a series of iterations.

$$D^*_{i,j,k} = \frac{-\sum_l \left(\xi_{i,j,l} - D_{i,j,k}\left(P_{w,l} - P_{k,l}\right)\right)\left(P_{w,l} - P_{k,l}\right)}{\sum_l \left(P_{w,l} - P_{k,l}\right)^2} \tag{8}$$

Once the optimal value $D^*_{i,j,k}$ is defined then we have to ensure the constraints. Equation 9 crops the optimal value to be nonnegative and not exceed the sum $Q$ of the membership pairs. As mentioned during the description of the objective function, we constraint the membership values to be non-negative and sum to one.

After updating the pair of membership weights we can also update the reconstruction error term $\xi_{i,j,l}$. The error terms are refreshed in order to avoid recomputing the error of Eq. 2 before every update. Therefore the computation of the error terms can be reduced from $\mathcal{O}(K)$ to $\mathcal{O}(1)$. Equations 10 and 11 define the steps of updating the errors as a result of changing $D_{i,j,k}$, $D_{i,j,w}$.

As a last step we can commit the optimal values, by preserving their sum before the updates. As Eq. 12 represents, the best value of $D_{i,j,w}$ can be deduced from the optimal value of $D_{i,j,k}$.

- Crop value of $D_{i,j,k}$, keep it between $[0, Q]$:

$$D^*_{i,j,k} \leftarrow \max(0, \min(Q, D^*_{i,j,k})) \tag{9}$$

- Update error $\xi_{i,j,l}$ pursuant to changing $D_{i,j,k}$:

$$\xi_{i,j,l} \leftarrow \xi_{i,j,l} - (D^*_{i,j,k} - D_{i,j,k})P_{k,l} \tag{10}$$

- Update error $\xi_{i,j,l}$ pursuant to changing $D_{i,j,w}$:

$$\xi_{i,j,l} \leftarrow \xi_{i,j,l} - (Q - D^*_{i,j,k} - D_{i,j,w})P_{w,l} \tag{11}$$

- Set $D_{i,j,k}$, $D_{i,j,2}$ to their optimal values:

$$D_{i,j,k} \leftarrow D^*_{i,j,k}, \quad D_{i,j,w} \leftarrow Q - D^*_{i,j,k} \tag{12}$$

### 4.4 Efficient initialization

Since the objective function of Eq. 1 is nonlinear in terms of $P$ and $D$ together, then a coordinate descent optimization is not guaranteed to avoid local optima. Therefore, good initial values of the patterns and the memberships are crucial for the learning process. The intuition leads into assigning some of the segments as initial patterns, however it is not obvious which of them provide the best initialization.

The answer is addressed via a technique utilized to find the initial centroids in a clustering setup (Arthur and Vassilvitskii 2007). The patterns (analogy to centroids) are initialized to segments with a probability proportional to the distance to all the other segments (Arthur and Vassilvitskii 2007). Therefore, we are assured to pick centroid

segments which are evenly distributed across the space of all series segments. The initialization steps are detailed in Algorithm 2. Please note that the first pattern has to be drawn randomly in a uniform distribution, while the other patterns are chosen randomly from the dataset segments based on the probability of their distance to the existing patterns. The function $\mathcal{C}$ measures the distance of a segment to the closest existing pattern.

**Algorithm 2** Initialize

**Require:** $S \in \mathbb{R}^{N \times M \times L}$, $L \in \mathbb{N}$, $K \in \mathbb{N}$
**Ensure:** $D \in \mathbb{R}^{N \times M \times K}$, $P \in \mathbb{R}^{K \times L}$
1: $P_1 \leftarrow S_{i',j'}$, drawn $i'$, $j' \sim \mathcal{U}(N, M)$
2: **for** $k = 2, \ldots, K$ **do**
3: $\quad P_k \leftarrow S_{i',j'}$, with probability weights $\dfrac{\mathcal{C}\left(S_{i',j'}\right)^2}{\sum_{i,j} \mathcal{C}\left(S_{i,j}\right)^2}$
4: **end for**
5: **for** $i = 1, \ldots, N$; $j = 1, \ldots, M$ **do**
6: $\quad k' = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmin}} \; ||S_{i,j} - P_k||^2$
7: $\quad D_{i,j,k} \leftarrow \begin{cases} 1 & k = k' \\ 0 & k \neq k' \end{cases}, \quad k = 1, \ldots, K$
8: **end for**
9: **return** $D$, $P$

The initialization of the membership degrees is more trivial than patterns. The degree index $k'$ denotes that pattern $P_{k'}$ is the closest to segment $S_{i,j}$ and its membership $D_{i,j,k'}$ is set to 1, while all the other membership degrees are initialized to zero.

### 4.5 Learning algorithm

Algorithm 3 finally combines all the steps of the factorization process. In the beginning, the memberships and the patterns are initialized using Algorithm 2. Next the errors are initialized, then the coordinate descent technique updates all the parameters in a number of iterations, denoted as a hyper-parameter $\mathcal{I}$. Subsequently, the degrees of membership and the patterns are learned by setting the aforementioned optimal values. The membership and pattern indexes are visited in random order to speed up the convergence.

For the sake of clarity, we are going to describe the steps of the algorithm in a line by line fashion. In line 1 the segmentation of series using Algorithm 1 is run, while in line 2 we initialize the membership weights $D$ and the patterns $P$ via Algorithm 2. The lines 4–6 initialize the errors $\xi$ with the initial reconstruction error between segments $S$ and their convolutional reconstruction $DP$. Once the initializations are completed, the invariant factorization learns the matrices $D$, $P$ in a series of iterations that are located in lines 8–37. For all the series $i$ of the dataset and the sliding window segments $j$, our method learns all the $K$-many pairs ofdegrees of memberships and patterns. The

**Algorithm 3** InvariantFactorization

**Require:** $T \in \mathbb{R}^{N \times Q}$, $L \in \mathbb{N}$, $\delta \in \mathbb{N}$, $K \in \mathbb{N}$, $\lambda_P \in \mathbb{R}$, $\mathcal{I} \in \mathbb{N}$
**Ensure:** $D \in \mathbb{R}^{N \times M \times K}$, $P \in \mathbb{R}^{K \times L}$
1: $S \leftarrow \text{SegmentSeries}(T, L, \delta)$
2: $(D, P) \leftarrow \text{Initialize}(S, L, K)$
3: {Initialize the errors}
4: **for** $\forall i \in \mathbb{N}_1^N, \forall j \in \mathbb{N}_1^M, \forall l \in \mathbb{N}_1^L$ **do**
5: $\quad \xi_{i,j,l} := S_{i,j,l} - \sum_{k=1}^{K} D_{i,j,k} P_{k,l}$
6: **end for**
7: {Update the patterns & memberships iteratively}
8: **for** iteration = $1, \ldots, \mathcal{I}$ **do**
9: $\quad$ {Update all degrees of membership}
10: $\quad$ **for** $\forall i \in \mathbb{N}_1^N, \forall j \in \mathbb{N}_1^M$ randomly **do**
11: $\quad\quad$ **for** $1, \ldots, K$, {Draw K-many pairs} **do**
12: $\quad\quad\quad k, w \sim \mathcal{U}(K, K)$, s.t. $D_{i,j,k} + D_{i,j,w} \neq 0$
13: $\quad\quad\quad Q \leftarrow D_{i,j,k} + D_{i,j,w}$
14: $\quad\quad\quad$ {Solve and crop the optimal memberships}
15: $\quad\quad\quad D^*_{i,j,k} = \frac{-\sum_l (\xi_{i,j,l} - D_{i,j,k}(P_{w,l} - P_{k,l}))(P_{w,l} - P_{k,l})}{\sum_l (P_{w,l} - P_{k,l})^2}$
16: $\quad\quad\quad D^*_{i,j,k} \leftarrow \max\left(0, \min(Q, D^*_{i,j,k})\right)$
17: $\quad\quad\quad$ {Update the error terms}
18: $\quad\quad\quad$ **for** $l = 1, \ldots, L$ **do**
19: $\quad\quad\quad\quad \xi_{i,j,l} \leftarrow \xi_{i,j,l} - (D^*_{i,j,k} - D_{i,j,k}) P_{k,l}$
20: $\quad\quad\quad\quad \xi_{i,j,l} \leftarrow \xi_{i,j,l} - (Q - D^*_{i,j,k} - D_{i,j,w}) P_{w,l}$
21: $\quad\quad\quad$ **end for**
22: $\quad\quad\quad$ {Commit the values of the pair}
23: $\quad\quad\quad D_{i,j,k} \leftarrow D^*_{i,j,k}$
24: $\quad\quad\quad D_{i,j,w} \leftarrow Q - D^*_{i,j,k}$
25: $\quad\quad$ **end for**
26: $\quad$ **end for**
27: $\quad$ {Update all patterns}
28: $\quad$ **for** $\forall k \in \mathbb{N}_1^K$; $\forall l \in \mathbb{N}_1^L$, randomly **do**
29: $\quad\quad P^*_{k,l} = \frac{\sum_{i,j} (\xi_{i,j,l} + D_{i,j,k} P_{k,l}) D_{i,j,k}}{\lambda_P + \sum_{i,j} D^2_{i,j,k}}$
30: $\quad\quad$ {Update the error terms}
31: $\quad\quad$ **for** $i = 1, \ldots, N$; $j = 1, \ldots, M$ **do**
32: $\quad\quad\quad \xi_{i,j,l} \leftarrow \xi_{i,j,l} - (P^*_{k,l} - P_{k,l}) D_{i,j,k}$
33: $\quad\quad$ **end for**
34: $\quad\quad$ {Commit the pattern's point value}
35: $\quad\quad P_{k,l} \leftarrow P^*_{k,l}$
36: $\quad$ **end for**
37: **end for**
38: **return** $D, P$

learning of membership degrees and the refreshing of errors as a result of the update of degrees, located in lines 8–26, is a direct mirroring of Eqs. 7–12. The update of the latent patterns is conducted through lines 28–37 of the algorithm and uses the defined update rules of Eqs. 5 and 6.

### 4.6 A new invariant representation

The final representation will sum the membership degrees in a bag-of-patterns strategy. It enables a quantification of which local patterns appear in a series and how often. The shift invariance is achieved by segmenting the series in a sliding window approach and the scale invariance is addressed using different sliding window sizes. Algorithm 4 describes the algorithmic steps. The algorithm iterates over $\Phi$ many different scales of an initial sliding windows size $L$ and solves an invariant factorization from Algorithm 3 per each size. The frequencies of the learned memberships are summed up for all $K$ patterns and the procedure is repeated for every sliding window size. Finally each time series contains $K\Phi$ many features, which denote the frequencies of patterns at different sizes and positions.

More concretely, in line 3 we apply a factorization with an initial sliding window size $L'$ and receive in return the factorized membership weights $D$ from Algorithm 3. Then a set of $K$ features, denoted $F$, can be constructed as the sum of the $K$-many dimensions of $D$ by summing up the weights of every sliding window of a series. The summation step is located in line 6. Then the sliding window size $L'$ is incremented in line 8 and the algorithm continues with a bigger sliding window size. In each iteration we extract $K$ factorization features that are appended to $F$.

The new representation will be used for classification, instead of the original time series. We deployed a polynomial kernel SVM, because we need to capture the interaction among features, i.e. the interaction among patterns of various sizes.

**Algorithm 4** InvariantRepresentation

**Require:** $T \in \mathbb{R}^{N \times Q}, L \in \mathbb{N}, \delta \in \mathbb{N}, K \in \mathbb{N}, \lambda_P \in \mathbb{R}, \mathcal{I} \in \mathbb{N}, \Phi \in N$
**Ensure:** $F \in \mathbb{R}^{N \times (K\Phi)}$
1: $L' \leftarrow L$
2: **for** $s = 1, \ldots, \Phi$ **do**
3: $D \leftarrow \text{InvariantFactorization}(T, L', \delta, K, \lambda_P, I)$
4: **for** $i = 1, \ldots, N;\ k = 1, \ldots, K$ **do**
5: $M \leftarrow \frac{Q - L'}{\delta}$
6: $F_{i,k+(s-1)K} \leftarrow \sum_{j=1}^{M} D_{i,j,k}$
7: **end for**
8: $L' \leftarrow L' + L$
9: **end for**
10: **return** $F$

We would like to explain the resulting representation ($F$) that is produced by the proposed method, with the aid of Fig. 3. Eight time series from the Trace dataset (shown in the left side) are factorized into a set of weights ($D$) and a set of eight patterns ($P$) (shown on the right side). We assign an ordinal number to the patterns from 1 to 8 in a top-down order. The new representation (middle plots) is the sum over all the sliding window weights of each pattern (line 6 in Algorithm 4). The reduced dimensionality not only is more compact, but also can help understand the differences among classes. For instance, classes B and C have a very subtle difference that is

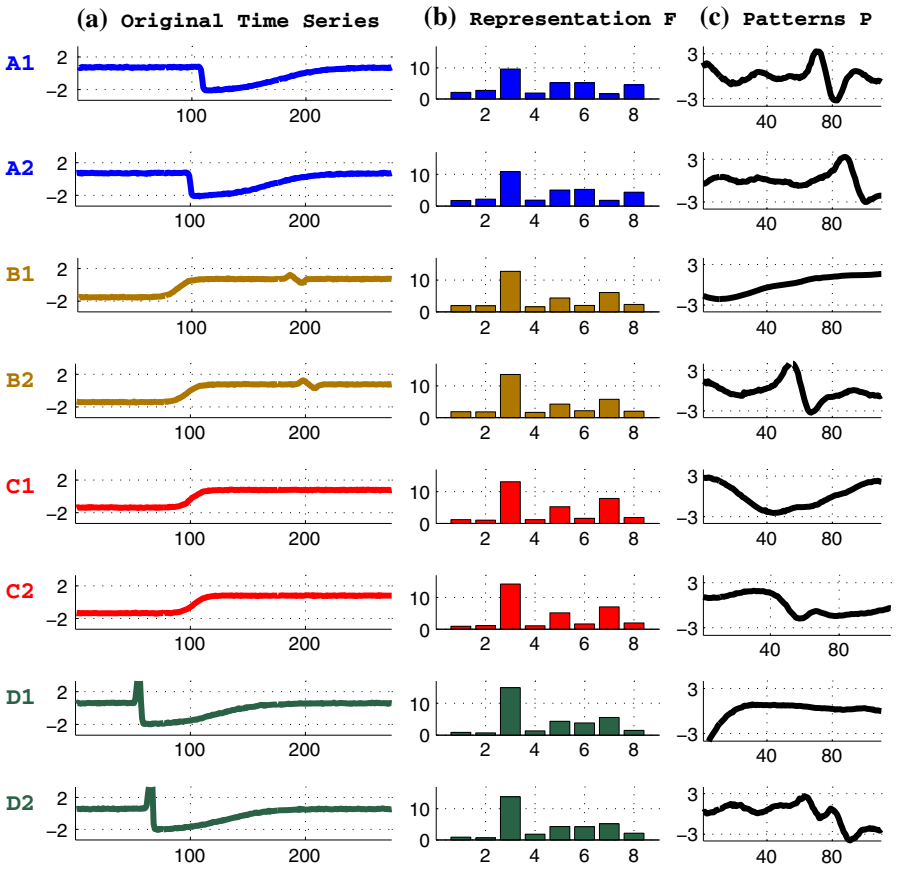


**Fig. 3** Illustration of the original time series (*left*) and the factorized representation $F$ (*center*) of eight series from the trace dataset. The series belong to four different classes (A,B,C,D) shown in *colors*. On the right, the patterns of the factorization are displayed. Parameters: $K = 8$, $L = 110$, $\Phi = 1$, $\delta = 1$, $\lambda_P = 0.01$, $I = 15$

reflected as a perturbation around point 200. In our latent representation, the B class instances have higher weights of patterns 1 and 2 that reflect the subtle perturbation, while the C class instances have lower weights for the patterns 1 and 2. It is possible to realize with mere human inspection that the factorized representation could detect the difference between classes by inspecting the derived representation $F$, which is the sum of $D$ membership degrees. In fact, a SVM applied over the derived representation $F$ achieves 0 % error on the Trace dataset.

### 4.7 Algorithmic complexity

The run-time complexity of the method is dominated by the updates of memberships and has an order $O(NMKL\mathcal{I})$. Concretely our method needs 48.4 h to compute on the StarLightCurves (the largest) dataset, while for instance DTW needs 87 h. The

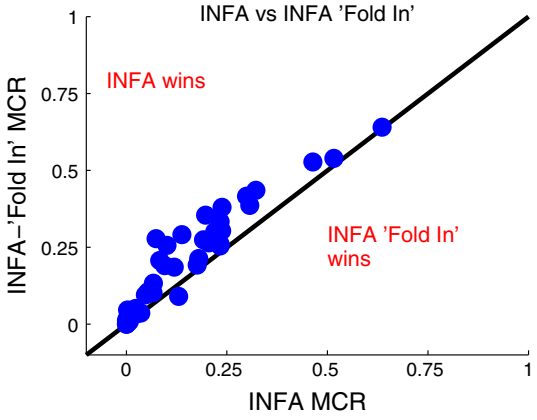


**Fig. 4** Error rate comparison of the transductive INFA against the inductive 'Fold In' variant of INFA

space complexity of our method depends on the storage of the segments $S$ and the memberships $D$, which is $O(NM\max(K, L))$.

### 4.8 Inductive and transductive factorizations

Our factorization method is a fully unsupervised dimensionality reduction that projects all the time series of a dataset to a new, reduced representation. As Algorithm 3 explained, the learning process needs to have access to a batch of time-series instances, in order to jointly minimize the reconstruction error of Eq. 1. However, the method proposed in this paper can operate in two modes: *Inductive* and *Transductive*. The transductive case factorizes all the instances jointly, using both the training and the testing predictors (not labels). On the other hand, the inductive mode factorizes the training instances first and then *folds in* the testing predictors *one at a time* to the latent representation, without modifying the patterns.

It is worth clarifying that in contrast to modern semi-supervised methods that use the test predictors to **directly** alter the decision boundary of their classifiers during the learning of the discriminative model, our transductive dimensionality reduction is unsupervised. Once the data is factorized to a reduced representation, then we classify the reduced instances using a standard SVM that does not utilize the predictors of the test instances.

The strategy of enabling an inductive operation of our method relies on a so-called 'Fold In' factorization. The 'Fold In' learns a factorization from a batch of available training time-series data and stores the $D$, $P$ representation. Then, the 'folding in' refers to the process of converting a test time series $T_t \in \mathbb{R}^Q$ to $D_t \in \mathbb{R}^{M\times L}$ by learning only one instance $D_t$ from Eq. 1, keeping all the other $D$, $P$ constant. For triviality, the update rule of learning $D_t$ is avoided because it is the same as lines 10–25 of Algorithm 3. In Fig. 4 the inductive 'Fold In' (denoted as INFA 'Fold In') mode is compared to the transductive factorization (denoted as INFA) in terms of error rates. Results show that the accuracy of the inductive 'Fold In' mode is inferior to the transductive INFA, which is due to the fact that folding in does not fully minimize the reconstruction objective of Eq. 1, but simply the weights $D$ of one test instance

at a time. As a result, the total reconstruction error will be higher, meaning that the factorized data will not be anymore a close representation of the original data, leading to a deteriorated prediction accuracy in case the inductive representation is fed to a SVM.

## 5 Experimental results

We will keep refering to our Invariant Factorization via acronyms, INFA for the transductive case and shortly INFAI for the inductive case, throughout the remaining parts of this document. This paper proposes a new representation of time series that is computed using an invariant factorization, not at all a classification method. Nevertheless, we apply a standard SVM upon the reduced dimensionality representation, in order to classify time series. For making the narration smoother, we are often going to refer to INFA as a joint method that includes a factorized representation plus an SVM over the factorized instances.

### 5.1 Baselines

We compared the prediction accuracy of our method, denoted **INFA** for the transductive mode and **INFAI** for the inductive mode, against the following seven state of the art baselines:

- **TSBF** The bag-of-features framework for time series (TSBF) uses a supervised codebook to extract features for a random forest classifier (Baydogan et al. 2013).
- **SSSK** Sparse spatial similarity kernel (SSSK) measures sequence similarity through sampling sequence features at different resolutions (Kuksa and Pavlovic 2010).
- **BOW** The Bag of words (BOW) method decomposes the series into local SAX words and uses a histogram representation of words as the new feature representation (Lin et al. 2012; Lin and Li 2009).
- **ENN** The nearest neighbor classifier with ED based similarity of series is a classical, yet competitive, approach in the time-series domain.
- **DTW** Dynamic time warping computes the best alignment of time indexes resulting in the mininal distance (Keogh et al. 2000; Rakthanmanon et al. 2012).
- **CID** The complexity invariant distance (CID) adds a L2-based total variation regularization term into the ED (Batista et al. 2014).
- **FSH** Fast shapelet (FSH) extracts the most discriminative segment of the series dataset, such that the distance from the dataset instances to the optimal shapelet can be used as a feature for classification (Rakthanmanon and Keogh 2013).

### 5.2 Setup and reproducibility

We conducted a large-scale experimentation in 43 time-series dataset from the UCR collection.[1] Our protocol complied to the default train/test split of the data, which is

[1] www.cs.ucr.edu/~eamonn/time_series_data.

**Table 1** Error rates–comparison of prediction accuracies on the UCR collection of datasets

| Dataset | Cls. | Train | Test | Len. | INFA | INFAI | TSBF | SSSK | BOW | ENN | DTW | CID | FSH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 50words | 50 | 450 | 455 | 270 | 0.220 | 0.301 | **0.209** | 0.488 | 0.316 | 0.369 | 0.310 | 0.336 | 0.557 |
| Adiac | 37 | 390 | 391 | 176 | 0.322 | 0.435 | **0.245** | 0.575 | 0.325 | 0.389 | 0.396 | 0.373 | 0.514 |
| Beef | 5 | 30 | 30 | 470 | **0.233** | 0.333 | 0.287 | 0.633 | 0.267 | 0.333 | 0.500 | 0.367 | 0.447 |
| CBF | 3 | 30 | 900 | 128 | **0.001** | 0.001 | 0.009 | 0.090 | 0.048 | 0.148 | 0.003 | 0.016 | 0.053 |
| Chlorine | 3 | 467 | 3840 | 166 | 0.464 | 0.526 | **0.336** | 0.428 | 0.405 | 0.350 | 0.352 | 0.351 | 0.417 |
| CinCECG | 4 | 40 | 1380 | 1639 | 0.138 | 0.291 | 0.262 | 0.438 | 0.164 | **0.103** | 0.349 | 0.084 | 0.174 |
| Coffee | 2 | 28 | 28 | 286 | **0.000** | 0.000 | 0.004 | 0.071 | 0.036 | **0.000** | 0.179 | **0.000** | 0.068 |
| CricketX | 12 | 390 | 390 | 300 | **0.205** | 0.264 | 0.278 | 0.585 | 0.305 | 0.423 | 0.223 | 0.372 | 0.527 |
| CricketY | 12 | 390 | 390 | 300 | **0.197** | 0.354 | 0.259 | 0.654 | 0.313 | 0.433 | 0.208 | 0.421 | 0.505 |
| CricketZ | 12 | 390 | 390 | 300 | **0.192** | 0.274 | 0.263 | 0.574 | 0.295 | 0.413 | 0.208 | 0.405 | 0.547 |
| Diatom | 4 | 16 | 306 | 345 | **0.003** | 0.046 | 0.126 | 0.173 | 0.111 | 0.065 | 0.033 | 0.065 | 0.117 |
| ECG200 | 2 | 100 | 100 | 96 | 0.130 | 0.090 | 0.145 | 0.220 | **0.110** | 0.120 | 0.230 | **0.110** | 0.227 |
| ECGF. | 2 | 23 | 861 | 136 | **0.001** | 0.001 | 0.183 | 0.360 | 0.164 | 0.203 | 0.232 | 0.218 | 0.004 |
| FaceAll | 14 | 560 | 1690 | 131 | 0.238 | 0.380 | 0.234 | 0.369 | 0.238 | 0.286 | **0.192** | 0.269 | 0.411 |
| FaceFour | 4 | 24 | 88 | 350 | **0.000** | 0.011 | 0.051 | 0.102 | 0.102 | 0.216 | 0.170 | 0.193 | 0.090 |
| FacesUCR | 14 | 200 | 2050 | 131 | **0.083** | 0.207 | 0.090 | 0.356 | 0.137 | 0.231 | 0.095 | 0.235 | 0.328 |
| Fish | 7 | 175 | 175 | 463 | **0.023** | 0.051 | 0.080 | 0.177 | 0.029 | 0.217 | 0.167 | 0.217 | 0.197 |
| GunPoint | 2 | 50 | 150 | 150 | **0.007** | 0.007 | 0.011 | 0.133 | 0.407 | 0.087 | 0.093 | 0.073 | 0.061 |
| Haptics | 5 | 155 | 308 | 1092 | 0.516 | 0.539 | **0.488** | 0.591 | 0.630 | 0.630 | 0.623 | 0.584 | 0.616 |
| InlineSkate | 7 | 100 | 550 | 1882 | 0.636 | 0.640 | **0.603** | 0.729 | 0.629 | 0.658 | 0.616 | 0.629 | 0.741 |
| ItalyPower | 2 | 67 | 1029 | 24 | **0.036** | 0.036 | 0.096 | 0.101 | 0.044 | 0.045 | 0.050 | 0.044 | 0.095 |
| Lighting2 | 2 | 60 | 61 | 637 | 0.180 | 0.213 | 0.257 | 0.393 | 0.328 | 0.246 | **0.131** | 0.246 | 0.295 |
| Lighting7 | 7 | 70 | 73 | 319 | **0.233** | 0.260 | 0.262 | 0.438 | 0.370 | 0.425 | 0.274 | 0.397 | 0.403 |
| MALLAT | 8 | 55 | 2345 | 1024 | 0.047 | 0.095 | 0.037 | 0.153 | 0.098 | 0.086 | 0.066 | 0.075 | **0.033** |
| Medical. | 10 | 381 | 760 | 99 | 0.299 | 0.102 | 0.269 | 0.463 | 0.401 | 0.316 | **0.263** | 0.309 | 0.433 |
| MoteStrain | 2 | 20 | 1252 | 84 | **0.066** | 0.102 | 0.135 | 0.166 | 0.177 | 0.121 | 0.165 | 0.212 | 0.217 |
| OliveOil | 4 | 30 | 30 | 570 | **0.067** | 0.133 | 0.090 | 0.300 | 0.233 | 0.133 | 0.133 | 0.133 | 0.213 |
| OSULeaf | 6 | 200 | 242 | 427 | **0.095** | 0.190 | 0.329 | 0.326 | 0.153 | 0.479 | 0.409 | 0.438 | 0.359 |
| Sony | 2 | 20 | 601 | 70 | **0.101** | 0.256 | 0.175 | 0.376 | 0.409 | 0.304 | 0.275 | 0.185 | 0.315 |
| SonyII | 2 | 27 | 953 | 65 | **0.054** | 0.105 | 0.196 | 0.339 | 0.154 | 0.141 | 0.169 | 0.123 | 0.215 |
| StarLight. | 3 | 1000 | 8236 | 1024 | **0.021** | 0.031 | 0.022 | 0.135 | **0.021** | 0.151 | 0.093 | 0.057 | 0.063 |
| Swedish. | 15 | 500 | 625 | 128 | **0.074** | 0.278 | 0.075 | 0.339 | 0.125 | 0.211 | 0.210 | 0.123 | 0.269 |
| Symbols | 6 | 25 | 995 | 398 | **0.026** | 0.034 | 0.034 | 0.184 | 0.088 | 0.101 | 0.050 | 0.084 | 0.068 |
| synthetic. | 6 | 300 | 300 | 60 | 0.013 | 0.033 | 0.008 | 0.067 | 0.017 | 0.120 | **0.007** | 0.050 | 0.081 |
| Trace | 4 | 100 | 100 | 275 | **0.000** | 0.000 | 0.020 | 0.300 | **0.000** | 0.240 | **0.000** | 0.140 | 0.002 |
| TwoPatt. | 4 | 1000 | 4000 | 128 | 0.003 | 0.016 | 0.001 | 0.087 | 0.010 | 0.093 | **0.000** | 0.121 | 0.114 |
| TwoL. | 2 | 23 | 1139 | 82 | **0.002** | 0.018 | 0.046 | 0.257 | 0.248 | 0.253 | 0.096 | 0.232 | 0.090 |
| uWaveX | 8 | 896 | 3582 | 315 | 0.176 | 0.192 | **0.164** | 0.358 | 0.242 | 0.261 | 0.273 | 0.238 | 0.293 |
| uWaveY | 8 | 896 | 3582 | 315 | **0.237** | 0.303 | 0.249 | 0.493 | 0.352 | 0.338 | 0.366 | 0.290 | 0.392 |
| uWaveZ | 8 | 896 | 3582 | 315 | 0.233 | 0.254 | **0.217** | 0.439 | 0.325 | 0.350 | 0.342 | 0.291 | 0.364 |

**Table 1** continued

| Dataset | Cls. | Train | Test | Len. | INFA | INFAI | TSBF | SSSK | BOW | ENN | DTW | CID | FSH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Wafer | 2 | 1000 | 6174 | 152 | **0.002** | 0.003 | 0.004 | 0.029 | 0.010 | 0.005 | 0.020 | 0.006 | 0.004 |
| WordsS. | 25 | 267 | 638 | 270 | 0.307 | 0.386 | **0.302** | 0.553 | 0.371 | 0.382 | 0.351 | 0.357 | 0.594 |
| yoga | 2 | 300 | 3000 | 426 | **0.119** | 0.185 | 0.149 | 0.172 | 0.145 | 0.170 | 0.164 | 0.164 | 0.269 |
| **Absolute total wins** | | | | | **25.16** | | **8.00** | **0.00** | **1.33** | **1.33** | **5.33** | **0.83** | **1.00** |
| **INFAI one-to-one wins** | | | | | | | **18** | **40** | **25** | **31** | **24** | **29** | **39** |
| **INFAI one-to-one draws** | | | | | | | **0** | **0** | **1** | **3** | **1** | **2** | **0** |
| **INFAI one-to-one losses** | | | | | | | **25** | **3** | **17** | **9** | **18** | **12** | **4** |
| **INFA one-to-one wins** | | | | | | | **30** | **42** | **38** | **39** | **35** | **38** | **41** |
| **INFA one-to-one draws** | | | | | | | **0** | **0** | **1** | **1** | **1** | **1** | **0** |
| **INFA one-to-one losses** | | | | | | | **13** | **1** | **4** | **3** | **7** | **4** | **2** |
| **Wilcoxon signed rank ($p$ values)** | | | | | | | **0.004** | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** |

an established benchmark split and is used by the baselines. The metric of comparison is the error rate, i.e. the misclassification rate. Table 1 shows the datasets used for experimentation together with the number of classes, the number of training instances, the number of testing instances and the length of the series.

Our method has a relatively high number of hyper-parameters, however they can be elegantly tuned via a grid hyper-parameter search. First of all, the set of eligible regularization parameters is $\lambda_P \in \{0.001, 1\}$. The sliding window size was searched from $L = \{0.15, 0.2\} \times Q$, and the number of latent dimensions from $K \in \{0.25, 0.5\} \times Q$. The sliding window scale was picked from $\Phi = \{3, 4\}$, while the sliding window offset was searched from $\delta = \{0.00, 0.005\} \times L$. The maximum number of iterations was set to $\mathcal{I} = 15$ in all cases. The applied classifier was a polynomial kernel SVM with a polynomial degree being 3 and the complexity parameter searched among $C \in \{0.1, 1, 10\}$. Out parameter tuning runs the invariant factorization and the subsequent SVM using all the combinations of parameters and selects that parameter combination which achieve the smallest leave-one-out (LOO) error on the training set. We define the parameters that result in the smallest LOO error as the **optimal** values. The error rate values to be detailed in this paper refer to the test set results when run with the optimal parameter combination. Since the algorithm is based on a probabilistic initialization, it might be possible that it converges to different closeby optima in each execution. However, in our experiments, those optima were very close and the final prediction accuracy results have insignificant differences. **The authors are devoted to promote full reproducibility, therefore the source code, the data and instructions are publicly available**.[2]

Most methods increment the sliding window by an offset of a single point at a time. While such an approach is practical and avoids the need to fit the offset parameter, it doesn't provide the optimal accuracy. Figure 5 illustrate the sensitivity of the LOO training error as a result of changing the $\delta$ parameter in a *ceteris-paribus* principle (all

[2] http://fs.ismll.de/publicspace/InvariantFactorization/.

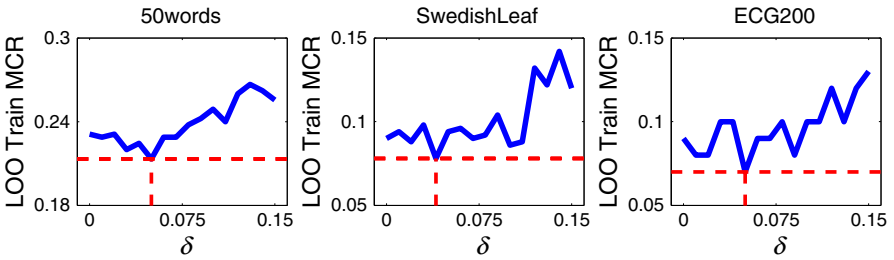


**Fig. 5** Effect of alternating the sliding window threshold $\delta$ on the LOO train miss-classification error. Other parameters: $K = 0.5 \times Q$, $L = 0.2 \times Q$, $\Phi = 4$, $C = 1.0$, $I = 15$

other parameters kept constant). The scale of $\delta$ is the percentage of the sliding window length $L$. As can be observed the optimal offset is a value that is small enough, but not the smallest, i.e. not one. Our method selects the optimal sliding window increment by finding the parameter $\delta$ that minimizes the leave-one-out cross validation search, in a grid search.

### 5.3 Results

The error rate results of the six state of the art baselines and our method INFA are presented in Table 1. The best performing method for each dataset (row) is emphasized in bold. In order to compare multiple classifiers across a large number of datasets we follow the established benchmarks of counting wins and *Wilcoxon's signed-rank* test for statistical significance (Demšar 2006). To be fair with the baselines, we retrieved the results from the baselines' publications (Baydogan et al. 2013; Batista et al. 2014; Rakthanmanon and Keogh 2013) over the **same** data splits as INFA. In addition, we verified the published results of the baselines with our own experimental checkups.

Three comparative figures are conducted, the first of which counts the absolute number of wins. Each dataset awards a total value of 1, which is split into equal fractions in case methods have equal error rate scores. The "Absolute wins" row, in the bottom of the table, counts the datasets where a method has the best prediction accuracy. As can be trivially deduced, our method has a clear superiority in terms of absolute wins, scoring 25.16 wins against 8.00 wins of the second best method. In addition, the transductive INFA outperforms by large margins all the baselines in an one-to-one comparisons of wins. INFA has more wins, yet the predominant analysis is whether or not those wins represent statistically significant differences. Each cell on the bottom row represents the $p$ value of the Wilcoxon signed-rank test on the error rate values of INFA against each baseline. Our method has a statistically significant difference over the error results of all baselines with a two-tailed hypothesis and the standard significance level of 95 % confidence ($p \leq 0.05$).

The transductive mode INFA is obviously always better than the inductive version INFAI, because INFAI learns a partial objective function, as previously explained and demonstrated in Sect. 4.8. Yet, the results for the 'Fold-In' mode (INFAI) indicate that our inductive variant is also very competitive. In a one-to-one comparison the inductive

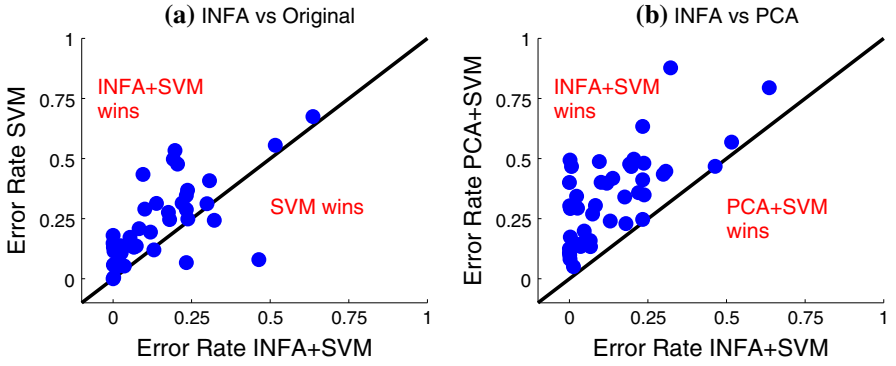


**Fig. 6** **a** Comparison of error rates of INFA+SVM against SVM without dimensionality reduction; **b** comparison of error rates over time-series factorized using two different dimensionality reduction techniques INFA versus PCA

mode outperforms all the baselines, except TSBF. Moreover, the difference of TSBF to INFAI is not significant according to the Wilcoxon signed-rank test ($p = 0.075$). On the other hand, INFAI outperforms strong popular methods such as DTW by 24 wins, 1 draws and 18 losses; or similarly CID by 29 wins, 2 draws and 12 losses.

### 5.4 On the need of INFA as a dimensionality reduction method

In its essence, our method can be perceived as a special variant of a PCA or matrix factorization that is tailored for time-series data. Instead of reducing the dimensionality (a.k.a. factorization) of the full time series, our method INFA factorizes sliding window segments and sums up the factorization coefficients (analogous to PCA weights).

Factorizing local segments has several advantages compared to the factorization of the full segment. First the weights of patterns are captured independent to the locations of those patterns (shift variations of series). Secondly, INFA takes into account the size of patterns by applying a factorization of various sliding window sizes. Furthermore, since the sliding window can be arbitrarily large (up to the full series length), then INFA includes PCA in terms of functionality.

Nevertheless, two experimental tests are required to support our claims. First of all, we should be able to empirically demonstrate whether or not our proposed factorization is required at all, meaning to check how the same classifier (SVM) would perform on the original time-series data. Figure 6, sub-plot a), shows the error rates of INFA+SVM against SVM over the original data using all the 43 time series of our experimental setup. As can be easily deduced from the illustration INFA outperforms largely a SVM without factorization. Yet, we would like to also show that the success belongs to our specific factorization method and not due to any dimensionality reduction method. In sub-plot b) we compare the error rates of SVM over factorized time-series using either our method or a standard PCA. The results indicate that our method, as expected, largely outperform the PCA representation.

### 5.5 Comparison to semi-supervised methods

Whilst our factorization method is unsupervised, yet the transductive operation mode factorizes all predictors of a dataset, including the testing predictors (detailed in Sect. 4.8). The utilization of all predictors (including train and test) for dimensionality reduction, even-though fully unsupervised, raises questions on comparisons against semi-supervised methods. Therefore, this section is dedicated to comparing the prediction quality of our method INFA against state-of-the-art methods that focus on the semi-supervised classification of time series.

We selected two state-of-the-art methods for comparison. The first method by Wei et al. is a well-established alternative in semi-supervised classification of time series (Wei and Keogh 2006). The classifier is trained on an initial training set with positive labeled instances and in an iterative manner the unlabeled instances are mined for enlarging the training set. In addition, a recent method in semi-supervised time-series classification, named SUCCESS, utilizes a combination of constrained hierarchical clustering and DTW (Marussy and Buza 2013).

Table 2 contains the results of our method INFA against the two state-of-the-art methods denoted as Wei et al. (2006) and SUCCESS (Marussy and Buza 2013). The time series belong to the UCR collection of datasets, same as the ones described in Sect. 2. Both leave-one-out training and testing error rate results are shown for all the methods, with the emphasis naturally being on the test scores. The method that achieves the minimum error is highlighted for every dataset. Two different comparative figures are derived. First of all, the total wins indicate that our method INFA has 33.5 wins against the 6.5 wins of SUCCESS, the closest baseline. Secondly, the same superiority is demonstrated by the very high number of one-to-one wins of INFA against both Wei et al. and SUCCESS. All the one-to-one wins are statistically significant as shown in the last row by the Wilcoxon signed rank test (significant for $p < 0.05$).

### 5.6 Prediction ahead of time

Sometimes it is important to be able to tell ahead of time the prediction accuracy of the method, by looking only at the training set (Batista et al. 2014). In order to conduct this comparison, first we define the concept of accuracy gain. The gain is defined as the ration of the accuracies among our proposed method INFA and selected baselines, i.e.: $\mathcal{G}\text{ain} = \frac{1-\text{Error INFA}}{1-\text{Error Baseline}}$. The smaller the error of INFA compared to the baseline, the bigger the gain in accuracy. The 'Expected Gain' is defined as the leave-one-out error on the training set, while the the 'Real Gain' is defined as the test error.

The scatter plot of expected versus real gain against two methods is shown in Fig. 7. We selected two baselines, namely the ED and Complexity-Invariant distance, as used in the original paper (Batista et al. 2014). The scatter plot can be divided into four quadrants that represent regions of True/False Positives/Negatives. As can be seen, our method has a very large number of True Positive results over the baselines using the same 43 datasets of the UCR collections. The semantics of the test is to show that there is a consistent pattern where our method wins, which can be estimated by using only the training series.

**Table 2** Error rate results of INFA against state-of-the-art semi-supervised methods

| Dataset | Classes | Train error | | | Test error | | |
|---|---|---|---|---|---|---|---|
| | | Wei et al. | SUCCESS | INFA | Wei et al. | SUCCESS | INFA |
| 50words | 50 | 0.432 | 0.398 | **0.213** | 0.436 | 0.414 | **0.220** |
| Adiac | 37 | 0.607 | 0.582 | **0.326** | 0.601 | 0.595 | **0.322** |
| Beef | 5 | 0.683 | 0.656 | **0.400** | 0.617 | 0.600 | **0.233** |
| CBF | 3 | 0.007 | 0.002 | **0.000** | 0.005 | 0.003 | **0.001** |
| ChlorineConcentration | 3 | 0.373 | **0.062** | 0.381 | 0.350 | **0.101** | 0.464 |
| CinCECGTorso | 4 | 0.021 | **0.001** | 0.125 | 0.019 | **0.001** | 0.138 |
| Coffee | 2 | 0.429 | 0.368 | **0.000** | 0.460 | 0.440 | **0.000** |
| CricketX | 12 | 0.477 | 0.425 | **0.167** | 0.465 | 0.444 | **0.205** |
| CricketY | 12 | 0.463 | 0.405 | **0.154** | 0.433 | 0.396 | **0.197** |
| CricketZ | 12 | 0.443 | 0.395 | **0.167** | 0.459 | 0.423 | **0.192** |
| DiatomSizeReduction | 4 | 0.018 | **0.017** | 0.063 | 0.031 | 0.025 | **0.003** |
| ECG200 | 2 | 0.237 | 0.225 | **0.080** | 0.239 | 0.195 | **0.130** |
| ECGFiveDays | 2 | 0.051 | 0.021 | **0.000** | 0.053 | 0.030 | **0.001** |
| FaceFour | 4 | 0.201 | 0.191 | **0.000** | 0.182 | 0.200 | **0.000** |
| FacesUCR | 14 | 0.080 | 0.062 | **0.060** | 0.083 | **0.070** | 0.083 |
| Fish | 7 | 0.424 | 0.449 | **0.040** | 0.403 | 0.434 | **0.023** |
| GunPoint | 2 | 0.089 | 0.039 | **0.000** | 0.075 | 0.045 | **0.007** |
| Haptics | 5 | 0.671 | 0.706 | **0.426** | 0.704 | 0.730 | **0.516** |
| InlineSkate | 7 | 0.693 | 0.679 | **0.470** | 0.683 | 0.663 | **0.636** |
| ItalyPowerDemand | 2 | 0.063 | 0.073 | **0.030** | 0.066 | 0.076 | **0.036** |
| Lighting2 | 2 | 0.355 | 0.322 | **0.150** | 0.342 | 0.317 | **0.180** |
| Lighting7 | 7 | 0.463 | 0.477 | **0.243** | 0.536 | 0.529 | **0.233** |
| Mallat | 8 | 0.042 | 0.041 | **0.018** | **0.042** | 0.037 | 0.047 |
| MedicalImages | 10 | 0.379 | 0.386 | **0.252** | 0.394 | 0.393 | **0.299** |
| MoteStrain | 2 | 0.124 | 0.129 | **0.050** | 0.115 | 0.107 | **0.066** |
| OliveOil | 4 | 0.300 | 0.315 | **0.100** | 0.367 | 0.383 | **0.067** |
| OSULeaf | 6 | 0.550 | 0.512 | **0.105** | 0.532 | 0.466 | **0.095** |
| SonyAIBORobotS. | 2 | 0.052 | 0.090 | **0.050** | **0.060** | 0.110 | 0.101 |
| SonyAIBORobotS.II | 2 | 0.088 | 0.094 | **0.074** | 0.079 | 0.087 | **0.054** |
| StarLightCurves | 3 | 0.119 | 0.200 | **0.022** | 0.140 | 0.200 | **0.021** |
| SwedishLeaf | 15 | 0.330 | 0.369 | **0.068** | 0.364 | 0.379 | **0.074** |
| Symbols | 6 | 0.033 | **0.022** | 0.040 | 0.025 | **0.019** | 0.026 |
| SyntheticControl | 6 | 0.051 | 0.029 | **0.010** | 0.065 | 0.045 | **0.013** |
| Trace | 4 | 0.054 | 0.001 | **0.000** | 0.050 | **0.000** | **0.000** |
| TwoPatterns | 4 | **0.000** | **0.000** | 0.002 | 0.000 | **0.000** | 0.003 |
| TwoLeadECG | 2 | 0.004 | 0.001 | **0.000** | 0.003 | **0.001** | 0.002 |
| uWaveGestureX | 8 | 0.276 | 0.284 | **0.173** | 0.284 | 0.286 | **0.176** |
| uWaveGestureY | 8 | 0.356 | 0.368 | **0.211** | 0.377 | 0.377 | **0.237** |
| uWaveGestureZ | 8 | 0.359 | 0.378 | **0.219** | 0.368 | 0.385 | **0.233** |

**Table 2** continued

| Dataset | Classes | Train error | | | Test error | | |
|---|---|---|---|---|---|---|---|
| | | Wei et al. | SUCCESS | INFA | Wei et al. | SUCCESS | INFA |
| Wafer | 2 | 0.009 | 0.009 | **0.000** | 0.009 | 0.009 | **0.002** |
| WordsSynonyms | 25 | 0.414 | 0.378 | **0.247** | 0.410 | 0.382 | **0.307** |
| Yoga | 2 | 0.148 | 0.149 | **0.130** | 0.152 | 0.151 | **0.119** |
| **Absolute total wins** | | **0.50** | **4.50** | **37.00** | **2.00** | **6.50** | **33.50** |
| **INFA one-to-one wins** | | **37** | **40** | **–** | **36** | **37** | **–** |
| **INFA one-to-one draws** | | **0** | **0** | **–** | **0** | **1** | **–** |
| **INFA one-to-one losses** | | **5** | **2** | **–** | **6** | **4** | **–** |
| **Wilcoxon signed rank ($p$ value)** | | **0.000** | **0.000** | **–** | **0.000** | **0.000** | **–** |

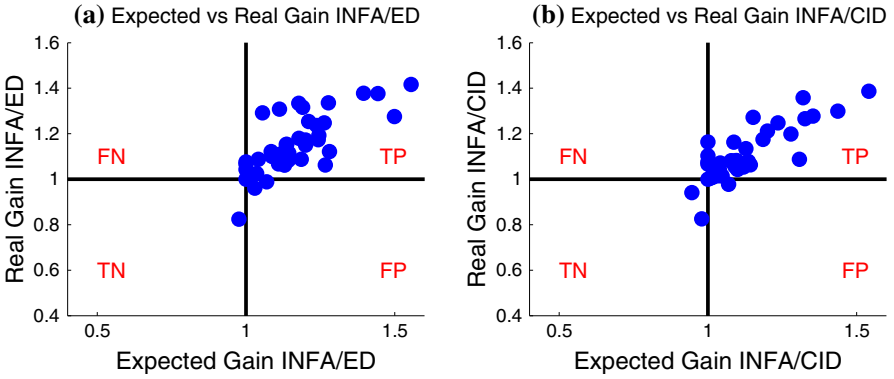


**Fig. 7** Illustration of the expected (train) versus real (test) gain against **a** ED and **b** CID

## 6 Conclusions

In this study we presented invariant factorization, a method that initially decomposes the time series into a set of overlapping segments via a sliding window approach. The segments are approximated by learning a set of latent patterns and degrees of memberships of each segment to each pattern. We formalized the factorization as a constraint objective function and proposed a stochastic coordinate descent method to solve it. The new representation of time series are the sums of the membership weights, which represent frequencies of local patterns. Features from various sliding window sizes were concatenated to encapsulate interaction among patterns of various scales. Finally we conducted a thorough experimental comparison against totally 11 state of the art baselines in 43 real-life time series datasets. Our method introduces state-of-the-art results in the realm of time-series classification, regarding the UCR collection of datasets.

**Acknowledgments** Partially co-funded by the Seventh Framework Programme of the European Comission, through project REDUCTION (# 288254). www.reduction-project.eu.

## References


Arthur D, Vassilvitskii S (2007) k-means++: the advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms. SODA '07, Philadelphia, PA, society for industrial and applied mathematics, pp 1027–1035

Barthelemy Q, Larue A, Mayoue A, Mercier D, Mars J (2012) Shift and 2d rotation invariant sparse coding for multivariate signals. IEEE Trans Signal Process 60(4):1597–1611

Batista GEAPA, Wang X, Keogh EJ (2011) A complexity-invariant distance measure for time series. In: SDM, SIAM / Omnipress, pp 699–710

Batista GEAPA, Keogh EJ, Tataw OM, de Souza VMA (2014) CID: an efficient complexity-invariant distance for time series. Data Min Knowl Disc 28(3):634–669

Baydogan MG, Runger G, Tuv E (2013) A bag-of-features framework to classify time series. IEEE Trans Pattern Anal Mach Intell 35(11):2796–2802

Buza K, Schmidt-Thieme L (2010) Motif-based classification of time series with Bayesian networks and SVMs. In: Fink A, Lausen B, Seidel W, Ultsch A (eds) Advances in data analysis, data handling and business intelligence. Studies in classification, data analysis, and knowledge organization. Springer, Berlin, Heidelberg, pp 105–114

Chen Y, Nascimento M, Ooi BC, Tung A (2007) Spade: on shape-based pattern detection in streaming time series. In: IEEE 23rd international conference on data engineering, 2007. ICDE 2007. pp 786–795

Chen L, Ng R (2004) On the marriage of lp-norms and edit distance. In: Proceedings of the thirtieth international conference on very large data bases—vol 30. VLDB '04, VLDB endowment, pp 792–803

Cuturi M (June 2011) Fast global alignment kernels. In: et al. G. (ed) Proceedings of the ICML 2011. ICML 2011, New York, ACM, pp 929–936

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Ding H, Trajcevski G, Scheuermann P, Wang X, Keogh EJ (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. PVLDB 1(2):1542–1552

Grabocka J, Nanopoulos A, Schmidt-Thieme L (2012a) Classification of sparse time series via supervised matrix factorization. In: Hoffmann J, Selman B (eds) AAAI, AAAI Press

Grabocka J, Nanopoulos A, Schmidt-Thieme L (2012b) Invariant time-series classification. In: Flach PA, Bie TD, Cristianini N (eds) ECML/PKDD (2). Lecture notes in computer science, vol 7524. Springer, pp 725–740

Gudmundsson S, Runarsson TP, Sigurdsson S (2008) Support vector machines and dynamic time warping for time series. In: IJCNN, IEEE, pp 2772–2776

Hills J, Lines J, Baranauskas E, Mapp J, Bagnall A (2013) Classification of time series by shapelet transformation. Data Min Knowl Disc 28:851–881

Huang PS, Yang J, Hasegawa-Johnson M, Liang F, Huang TS (2012) Pooling robust shift-invariant sparse representations of acoustic signals. In: INTERSPEECH, ISCA

Keogh EJ, Pazzani MJ (2000) Scaling up dynamic time warping for datamining applications. In: KDD. pp 285–289

Keogh E, Chakrabarti K, Pazzani M, Mehrotra S (2001) Dimensionality reduction for fast similarity search in large time series databases. Knowl Inf Syst 3(3):263–286

Kuksa P, Pavlovic V (2010) Spatial representation for efficient sequence classification. In: 20th international conference on pattern recognition (ICPR), 2010. pp 3320–3323

Lewicki MS, Sejnowski TJ (1999) Coding time-varying signals using sparse, shift-invariant representations. In: Proceedings of NIPS, Cambridge, MIT Press pp 730–736

Lin J, Khade R, Li Y (2012) Rotation-invariant similarity in time series using bag-of-patterns representation. J Intell Inf Syst 39(2):287–315

Lin J, Li Y (2009) Finding structural similarity in time series data using bag-of-patterns representation. In: Proceedings of the 21st international conference on scientific and statistical database management. SSDBM 2009, Springer, Berlin pp 461–477

Lin J, Keogh E, Wei L, Lonardi S (October 2007) Experiencing sax: a novel symbolic representation of time series. Data Min Knowl Disc 15(2):107–144

Marussy K, Buza K (2013) Success: a new approach for semi-supervised classification of time-series. In: Rutkowski L, Korytkowski M, Scherer R, Tadeusiewicz R, Zadeh L, Zurada J (eds) Artificial intelligence and soft computing. Lecture notes in computer science, vol 7894. Springer, Berlin, pp 437–447

Mueen A, Keogh EJ, Young N. (2011) Logical-shapelets: an expressive primitive for time series classification. In: Apté C, Ghosh J, Smyth P (eds) KDD, ACM, pp 1154–1162

Platt JC (1999) Advances in kernel methods. MIT Press, Cambridge
Rakthanmanon T, Campana B, Mueen A, Batista G, Westover B, Zhu Q, Zakaria J, Keogh E (2012) Searching and mining trillions of time series subsequences under dynamic time warping. In: Proceedings of the 18th ACM SIGKDD. KDD 2012, New York, ACM, pp 262–270
Rakthanmanon T, Keogh E (2013) Fast shapelets: a scalable algorithm for discovering time series shapelets. In: Proceedings of the 13th SIAM international conference on data mining
Vlachos M, Kollios G, Gunopulos D (2002) Discovering similar multidimensional trajectories. In: Proceedings 18th international conference on data engineering, 2002. pp 673–684
Wang X, Mueen A, Ding H, Trajcevski G, Scheuermann P, Keogh E (2013) Experimental comparison of representation methods and distance measures for time series data. Data Min Knowl Disc 26(2):275–309
Wang J, Liu P, She MF, Nahavandi S, Kouzani A (2013) Bag-of-words representation for biomedical time series classification. Biomed Signal Process Control 8(6):634–644
Wang F, Lee N, Hu J, Sun J, Ebadollahi S (2012) Towards heterogeneous temporal clinical event pattern discovery: a convolutional approach. In: Proceedings of ACM SIGKDD. KDD '12, New York, ACM pp 453–461
Wei L, Keogh E (2006) Semi-supervised time series classification. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '06, New York, ACM pp 748–753
Yu HF, Hsieh CJ, Si S, Dhillon IS (2012) Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In: Zaki MJ, Siebes A, Yu JX, Goethals B, Webb GI, Wu X (eds) ICDM, IEEE computer society, pp 765–774
Zhang D, Zuo W, Zhang D, Zhang H (2010) Time series classification using support vector machine with Gaussian elastic metric kernel. In: ICPR, IEEE, pp 29–32