Similarity Preserving Representation Learning for Time Series Clustering

Qi Lei\textsuperscript{1*}, Jinfeng Yi\textsuperscript{2*}, Roman Vaculin\textsuperscript{3}, Lingfei Wu\textsuperscript{3} and Inderjit S. Dhillon\textsuperscript{4,1}

\textsuperscript{1}University of Texas at Austin
\textsuperscript{2}JD AI Research
\textsuperscript{3}IBM Research
\textsuperscript{4}Amazon

leiqi@oden.utexas.edu.com, yijinfeng@jd.com

Abstract

A considerable amount of clustering algorithms take instance-feature matrices as their inputs. As such, they cannot directly analyze time series data due to its temporal nature, usually unequal lengths, and complex properties. This is a great pity since many of these algorithms are effective, robust, efficient, and easy to use. In this paper, we bridge this gap by proposing an efficient representation learning framework that is able to convert a set of time series with various lengths to an instance-feature matrix. In particular, we guarantee that the pairwise similarities between time series are well preserved after the transformation, thus the learned feature representation is particularly suitable for the time series clustering task. Given a set of $n$ time series, we first construct an $n \times n$ partially-observed similarity matrix by randomly sampling $O(n \log n)$ pairs of time series and computing their pairwise similarities. We then propose an efficient algorithm that solves a non-convex and NP-hard problem to learn new features based on the partially-observed similarity matrix. By conducting extensive empirical studies, we show that the proposed framework is more effective, efficient, and flexible, compared to other state-of-the-art time series clustering methods.

1 Introduction

Modeling time series data is important but highly challenging. It is considered by [Yang and Wu, 2006] as one of the 10 most challenging problems in data mining. Although time series clustering has attracted increasing attention, the time series clustering algorithms are still much fewer than the clustering algorithms developed for static data. The latter category of models, which usually take instance-feature matrices as their inputs, cannot directly partition time series data due to its temporal nature, typically unequal lengths, and complex properties [Långkvist \textit{et al.}, 2014]. This is a great pity since many static clustering algorithms are effective, robust, efficient, and easy to use. Introducing them to time series clustering can significantly advance this field.

\*Contact Author

In this work, we bridge this gap by proposing an efficient unsupervised representation learning framework that can convert a set of uni- or multi-dimensional time series data with equal or unequal lengths to an instance-feature matrix. In particular, the learned features preserve the pairwise similarities between the raw time series data, thus are particularly suitable to the time series clustering problem. Notably, the proposed framework is flexible to any time series distance or similarity measures such as Mikowski distance, cross-correlation, Kullback-Leibler divergence, dynamic time warping (DTW) similarity, move-split-merge (MSM) distance, and short time series (STS) distance. Here, we slightly abuse the notation and use the term similarity measure to denote both similarity and distance measures, as they are usually interchangeable. In this paper, we use widely-used DTW similarity as an example to illustrate our approach, and then empirically show that the proposed approach also works well with other time series similarity measures.

Given a total of $n$ uni- or multi-dimensional time series, our first step generates an $n \times n$ similarity matrix $A$ with $A_{ij}$ equaling to the DTW similarity between the time series $i$ and $j$. However, computing all the pairwise similarities requires calling the DTW $O(n^2)$ times, which is time-consuming when $n$ is large. As a concrete example, generating a full similarity matrix for $n = 150,000$ time series of length $30$ takes more than 28 hours on an Intel Xeon 2.40 GHz processor with 256 GB of main memory. To significantly reduce the running time, we note that time series similarity measures, including DTW, usually capture the co-movements of time series, which has shown to be driven by only a small number of latent factors [Stock and Watson, 2005]. This suggests that the similarity matrix $A$ can be well approximated by a low-rank matrix. According to the theory of (noisy) matrix completion [Lafond, 2015; Sun and Luo, 2015], only $O(n \log n)$ randomly sampled entries are needed to recover an $n \times n$ low-rank matrix. This allows us to sample only $O(n \log n)$ pairs of time series to generate a partially-observed similarity matrix $\hat{A}$. In this way, the time spent on generating similarity matrix is significantly reduced by a factor of $O(n/ \log n)$. For $n = 150,000$ time series of length $30$, it only takes 3 minutes to construct a partially observed similarity matrix with $[20n \log n]$ observed entries.

Given the generated partially-observed similarity matrix $\hat{A}$, our second step learns a new feature representation for $n$ time series such that their pairwise DTW similarities can be well
approximated by the inner products of new features. To this end, we solve a symmetric matrix factorization problem to factorize $\mathbf{A}$, i.e., learn a $d$-dimensional matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ such that $P_{\Omega}(\mathbf{A}) \approx P_{\Omega}(\mathbf{X} \mathbf{X}^T)$, where $P_{\Omega}$ is a matrix projection defined on the observed set $\Omega$. Despite its relatively simple formulation, this optimization problem is NP-hard and non-convex. To address this challenge, we propose a highly efficient and parameter-free exact cyclic coordinate descent algorithm. By wisely updating variables with the sparsely observed entries in $\mathbf{A}$, the proposed algorithm incurs a very low computational cost, and thus can learn new feature representations in an extremely efficient way. For example, with $n = 150,000$ and $d = 15$, it only takes the proposed algorithm 25 seconds to learn new features. We summarize our contributions as below:

1. We bridge the gap between time series data and static clustering algorithms by learning a feature representation that preserves the pairwise similarities of the raw time series data. The underlying low-rank assumption of the similarity matrix is verified both theoretically and empirically.
2. We propose a parameter-free algorithm for symmetric matrix factorization on a partially-observed matrix.
3. We conduct extensive experiments on over 80 real-world time series datasets. The results show that our learned features working with some simple static clustering methods like $k$-Means can significantly outperform the state-of-the-art time series clustering algorithms in both accuracy and efficiency.

2 Related Work

In this section, we briefly review the existing work on learning feature representations for time series data. A majority of them were designed for time series classification instead of clustering. A family of methods uses a set of derived features to represent time series. For instance, [Nanopoulos et al., 2001] proposed to use the mean, standard deviation, kurtosis, and skewness of time series to represent control chart patterns. The authors in [Wang et al., 2006] introduced a set of features such as trend, seasonality, serial correlation, chaos, nonlinearity, and self-similarity to partition different types of time series. [Deng et al., 2013] used some easy to compute features such as mean, standard deviation and slope temporal importance curves to guide time series classification. To automate the selection of features for time series classification, the authors in [Fulcher and Jones, 2014] proposed a greedy forward method that can automatically select features from thousands of choices. Shapelet-based classifiers [Hills et al., 2014] extract shapelet features from the time series, and then convert the time series to a regular feature table. [Kate, 2016] proposed to use the DTW distances from a training example to create new features for time series data. Besides, several techniques have been proposed to represent time series by a certain types of transformation, such as discrete Fourier transformation [Faloutsos et al., 1994], discrete cosine transformation [Korn et al., 1997], Laplace transformation [Hayashi et al., 2005], discrete wavelet transformation [Chan and Fu, 1999], piecewise aggregate approximation [Keogh et al., 2001], and symbolic aggregate approximation [Lin et al., 2007]. Also, deep learning models such as Elman recurrent neural network [Elman, 1990] and long short-term memory [Hochreiter and Schmidhuber, 1997] are capable of modeling complex structures of time series data and learn a layer of feature representations.

Despite the remarkable progress, most feature representations learned by these algorithms are problem-specific and are not general enough for applications in multiple domains. Besides, these learned features cannot preserve similarities of the raw time series data, thus are not suitable for the clustering problem that is sensitive to the data similarity. These limitations inspire us to propose a problem-independent and similarity preserving representation learning framework for time series clustering.

3 Similarity Preserving Representation Learning for Time Series Clustering

In this section, we first present the general framework of our similarity preserving time series representation learning method, and then propose an extremely efficient algorithm that is significantly faster than a naive implementation.

3.1 Problem Definition and General Framework

Given a set of $n$ time series $\mathcal{T} = \{T_1, \ldots, T_n\}$ with equal or unequal lengths, our goal is to convert them to a matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ such that the time series similarities are well preserved after the transformation. Specifically, we aim to learn a mapping function $f : T \rightarrow \mathbb{R}^d$ that satisfies

$$S(T_i, T_j) \approx \langle f(T_i), f(T_j) \rangle \quad \forall i, j \in [n],$$

where $\langle \cdot , \cdot \rangle$ stands for the inner product, a common similarity measure in analyzing static data. $S(\cdot , \cdot)$ denotes the pairwise time series similarity that can be computed by a number of functions. In this work, we use dynamic time warping (DTW) algorithm as an example to illustrate our approach. By warping sequences non-linearly in the time dimension, DTW can calculate an optimal match between two given temporal sequences with equal or unequal lengths. Due to its superior performance, DTW has been successfully applied to a variety of applications, including surveillance [Sempena et al., 2011], gesture recognition [Celebi et al., 2013], signature matching [Efrat et al., 2007], and speech recognition [Muda et al., 2010].

Normally, DTW outputs a pairwise distance between two temporal sequences, thus we need to convert it to a similarity score. Since the inner product space can be induced from the normed vector space using $(x, y) = (\|x\|^2 + \|y\|^2 - \|x - y\|^2)/2$, we generate the DTW similarity by

$$S(T_i, T_j) = \frac{\text{DTW}(T_i, 0)^2 + \text{DTW}(T_j, 0)^2 - \text{DTW}(T_i, T_j)^2}{2},$$

where $0$ denotes the length one time series with entry 0. The similarity computed via the above equation is a more numerically-stable choice than some other similarity measures such as the reciprocal of distance. This is because when two time series are almost identical, their DTW distance is
close to 0 and thus its reciprocal tends to infinity. An alternative way to convert DTW is via the Laplacian transformation $e^{-dtw^2(t_i, t_j)/t}$ [Hayashi et al., 2005], where $t$ is a hyperparameter. However, our experimental results shows that it yields a worse performance than our proposed method.

In order to learn the matrix $X$, an intuitive idea is to factorize the similarity matrix $A \in \mathbb{R}^{n \times n}$ where $A_{ij} = S(t_i, t_j)$. In more detail, this idea consists of two steps, i.e., a similarity matrix construction step and a symmetric matrix factorization step. In the first step, we construct $A$ by calling the DTW oracle at least $(n + 1)n/2$ times to compute the pairwise similarities and the distance. In the second step, we learn an optimal data-feature matrix $X$ by solving the following optimization problem

$$
\min_{X \in \mathbb{R}^{n \times d}} \| A - XX^T \|^2_F, \tag{2}
$$

where $i$-th row of $X$ indicates the feature vector of sample $i$. The problem (2) has a closed form solution, i.e.,

$$
X = Q_{1:n,1:d} \times \sqrt{A_{1:d,1:d}},
$$

where $A = QAQ^T$ and the notation $Q_{1:k,1:r}$ represents the upper left $k$ by $r$ sub-matrix of $Q$.

Although the inner products of features in $X$ well preserve the DTW similarities of the raw time series, the idea described above is impractical since both construction and factorization steps are extremely time-consuming when $n$ is large. To generate an $n \times n$ similarity matrix, we need to call the DTW algorithm $O(n^2)$ times. Meanwhile, a naive implementation of eigen-decomposition takes $O(n^3)$ time. We hereby introduce our Similarity Preserving RepresentAtion Learning (SPIRAL) framework that only calls DTW $O(n \log n)$ times for similarity matrix construction, with an additional $O(n^d \log n)$ flops on learning $d$ features from the similarity matrix.

### 3.2 The Low-rankness of the Similarity Matrix

To significantly improve the efficiency of the first step, we make a key observation that the similarity matrix $A$ can be well approximated by a low-rank matrix. In the following, we show that this observation is valid when the time series are generated from some distinguishable clusters.

**Assumption 1.** Suppose all the time series belong to $k$ clusters $C_1, C_2, \cdots, C_k$. For simplicity, let cluster $C_0$ to be the set of a unique length-one time series with entry $0$. Define the cluster distance $d_{ab}$ between clusters $C_a$ and $C_b$ to be $d_{ab} = \min_{T_1 \in C_a, T_2 \in C_b} DTW(T_1, T_2), 0 \leq a \neq b \leq k$. We consider the following conditions:

1. **Definition of clusters:** For each cluster $C_a$, we have $DTW(T_1, T_2) \leq \epsilon \ll d_{ab}^2, \forall T_1, T_2 \in C_a, 1 \leq a \leq k$;
2. **Clusters are distinguishable:** $\epsilon \ll d_{ab}^2, \forall 0 \leq a \neq b \leq k$;
3. **Proxy for triangle inequality:** For any two different clusters $C_a$ and $C_b$, $DTW(T_1, T_2) \leq d_{ab}^2 + O(\epsilon), \forall T_1 \in C_a, T_2 \in C_b, 0 \leq a \neq b \leq k$;

Under such assumptions, we have:

**Theorem 2.** Let $T_1, T_2, \cdots, T_n$ be $n$ time series generated from $k (\ll n)$ clusters satisfying Assumption 1, then the generated similarity matrix $A$ can be written as $L + N$, where matrix $L$ has a small rank of at most $k(k - 1) + 2$, and $N$ is a noise matrix satisfying $|N_{ij}| \leq O(\epsilon)$.

We defer the proof to the full version [Lei et al., 2017]. Some alternative intuitions on the low-rankness of matrix $A$ are: (i) time series similarity functions, including DTW, usually measure the level of co-movement between time series, which has shown to be dictated by only a small number of latent factors [Stock and Watson, 2005]; and (ii) since the matrix $A$ is a special case of Wigner random matrix, the gaps between its consecutive eigenvalues should not be small [Marčenko and Pastur, 1967]. This implies the low-rank property since most of its energy is concentrated in its top eigenvalues [Erdős et al., 2009]. Our experiments on an extensive set of real-world time series datasets also verify the low-rankness of the similarity matrix $A$.

### 3.3 A Parameter-free Scalable Algorithm

Given a low-rank similarity matrix, we are now able to significantly reduce the computational costs of both steps. According to the theory of matrix completion [Sun and Luo, 2015], only $O(n \log n)$ randomly sampled entries are needed to perfectly recover an $n \times n$ low-rank matrix. Thus, we don’t need to compute all the pairwise DTW similarities. Instead, we randomly sample $O(n \log n)$ pairs of time series, and then compute the DTW similarities within the selected pairs. This leads to a partially-observed similarity matrix $\tilde{A}$ with $O(n \log n)$ observed entries:

$$
\tilde{A}_{ij} = \begin{cases} 
S(t_i, t_j) & \text{if } \Omega_{ij} = 1 \\
\text{unobserved} & \text{if } \Omega_{ij} = 0,
\end{cases}
$$

where $\Omega \in \{0, 1\}^n \times \{0, 1\}^n$ is a binary matrix indicating the indices of sampled pairs. In this way, the running time of the first step is reduced by a significant factor of $O(n/\log n)$. Since this factor scales almost linearly with $n$, we can significantly reduce the running time of generating the similarity matrix when $n$ is large. As an example when $n = 150,000$, it now only takes 194 seconds to construct $A$ with $[20n \log n]$ observed entries, more than 500 times faster than generating a full similarity matrix.

Given the partially-observed similarity matrix $\tilde{A}$, our second step aims to learn a new feature representation matrix $X$. To this end, we propose an efficient and parameter-free algorithm that can directly learn the feature matrix $X \in \mathbb{R}^{n \times d}$ from minimizing the following optimization problem

$$
\min_{X \in \mathbb{R}^{n \times d}} \| P_{\Omega}(\tilde{A} - XX^T) \|^2_F, \tag{4}
$$

where $P_{\Omega} : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times n}$ is a projection operator on $\Omega$. The objective function (4) does not have a regularization term since it already bounds the Frobenius norm of $X$. Despite its relatively simple formulation, solving problem (4) is non-trivial since its objective function is non-convex. To address this issue, we propose a very efficient optimization algorithm that solves problem (4) based on exact cyclic coordinate descent (CD). Our proposed coordinate descent algorithm has

---

1 We omit the details due to space limitation, but instead provide one example in Figure 1.
the following two advantages: (i) for each iteration, our CD algorithm directly updates each coordinate to the optimum. Thus, we do not need to select any hyper-parameter such as the learning rate; and (ii) by directly updating coordinates to the optimums using the most up-to-date information, our CD algorithm is efficient and converges at a very fast rate.

At each iteration of the exact cyclic CD method, all variables but one are fixed, and that variable is directly updated to its optimal value. To be precise, our algorithm consists of two loops that iterate over all the entries of $X$ to update their values. The outer loop of the algorithm traverses through each column of $X$ by assuming all the other columns known and fixed. At the $i$-th iteration, it optimizes the $i$-th column $X_{i:n,i}$ by minimizing the following subproblem

$$\min_{X_{i:i}} \| R - P_\Omega(X_{i:n,i}X_{i:n,i}^\top) \|^2_F,$$

where $R$ is the residual matrix defined as $R = P_\Omega(\tilde{A} - \sum_{j \neq i} X_{1:n,j}X_{1:n,j}^\top)$. In the inner loop, the proposed algorithm iterates over each coordinate of the selected column and updates its value. Specifically, when updating the $j$-th entry $X_{j:i}$, we solve the following optimization problem:

$$\min_{X_{j:i}} \| R - P_\Omega(X_{1:n,i}X_{1:n,i}^\top) \|^2_F$$

$$\leftarrow \min_{X_{j:i}} \| R \|^2_F - 2\langle R, P_\Omega(X_{i:n,i}X_{1:n,i}^\top) \rangle + \| P_\Omega(X_{1:n,i}X_{1:n,i}^\top) \|^2_F$$

$$\leftarrow \min_{X_{j:i}} \| \sum_{k \in \Omega_1, k \neq j} X_{k:i}^2 - R_{j:j} \|^2_F$$

$$- 4\langle \sum_{k \in \Omega_1, k \neq j} X_{k:i}R_{j:j} \rangle X_{j:i} + C$$

$$\leftarrow \min_{X_{j:i}} \psi(X_{j:i}) := X_{j:i}^4 + 2pX_{j:i}^2 + 4qX_{j:i} + C,$$

where $\Omega_1, i = 1, \ldots, n$ contains the indices of the observed entries in the $i$-th row of matrix $A$, and $C$ is a constant. Algorithm 1 describes the detailed steps of the proposed exact cyclic CD algorithm that updates $\psi(X_{j:i})$ to its minimum.

The proposed algorithm incurs a very low computational cost in each iteration. Lines 7-11 of the algorithm can be computed in $O(n \log n)$ operations. This is because the costs of computing $p$ and $q$ are only proportional to the cardinality of $\Omega_1$. Besides, the derivative $\psi'(X_{j:i})$ is a third-degree polynomial, thus its roots can be computed in a closed form. By using Cardano’s method [Cardano and Witmer, 1993], the optimal solution of $X_{j:i}$ can be calculated in a constant time given the computed $p$ and $q$. Likewise, lines 6 and 12 of the algorithm also take $O(n \log n)$ time since matrix $R$ can be updated by only considering the observed entries. To sum up, the proposed algorithm has a per-iteration cost of $O(n \log n)$, which is significantly faster than direct factorization of the whole matrix that take at least $O(n^2 \log n)$ time in each iteration [Vandaele et al., 2015].The following theorem shows that Algorithm 1 guarantees to converge to a stationary point of (4). We defer the proof to the full version [Lei et al., 2017].

**Theorem 3.** Let $(X^{(0)}, X^{(1)}, \ldots)$ be a sequence of iterates generated by the Algorithm 1. If it converges to a unique accumulation point, then the point is a stationary point of Problem (4).

---

**Algorithm 1 Efficient Exact Cyclic Coordinate Descent Algorithm for Solving the Optimization Problem (4)**

1: **Inputs:**
- $A \in \mathbb{R}^{n \times n}$, partially-observed similarity matrix.
- $\Omega_i, i = 1, \ldots, n$: indices of the observed entries in the $i$-th row of matrix $A$.
- $I$: # iterations, $d$: # features

2: **Initializations:**

- $X^{(0)} \leftarrow 0_{n \times d}$, $R \leftarrow P_\Omega(\tilde{A} - X^{(0)}X^{(0)^\top})$.

3: **for** $t = 1, \ldots, I$ **do**

4: $X^{(t)} \leftarrow X^{(t-1)}$.

5: **for** $i = 1, \ldots, d$ **do**

6: $R \leftarrow R + P_\Omega(X_{1:n,i}X_{1:n,i}^\top)$.

7: **for** $j = 1, \ldots, n$ **do**

8: $p \leftarrow \sum_{k \in \Omega_1} X_{k:j}^{(t)2} - X_{j:j}^{(t)2} - R_{j:j}$.

9: $q \leftarrow -\sum_{k \in \Omega_1} X_{k:j}^{(t)2}R_{j:j} + X_{j:j}^{(t)2}R_{j:j}$.

10: $X_{j:j}^{(t)} \leftarrow \arg\min\{X_{j:j}^4 + 2pX_{j:j}^2 + 4qX_{j:j} + C\}$.

11: **end for**

12: $R \leftarrow R - P_\Omega(X_{1:n,i}X_{1:n,i}^\top)$.

13: **end for**

14: **end for**

15: **Output:** $X^{(I)}$.

In addition to a low per-iteration cost, the proposed algorithm yields a fast convergence. This is because our algorithm always uses the newest information to update variables and each variable is updated to the optimum in a single step. This is verified by a convergence test conducted on the UCR Non-Invasive Fetal ECG Thorax1 testbed [Chen et al., 2015]. This testbed contains a total of 3,765 time series with a length of 750. In this test, we generate a full similarity matrix $A$ by computing the DTW similarities between all time series pairs, and then randomly sample $[20n \log n]$ of its entries to generate a partially-observed matrix $\tilde{A}$. We call the proposed algorithm to factorize matrix $\tilde{A}$ by setting the dimensionality $d = 15$. To measure the performance of the proposed method, we compute two error rates, i.e., the observed error $\| P_\Omega(\tilde{A} - XX^\top) \|_F / \| P_\Omega(A) \|_F$ and the underlying true error rate $\| A - XX^\top \|_F / \| A \|_F$ at each iteration. Figure 1 shows how they converge as a function of time. This figure clearly demonstrates that the proposed exact cyclic CD algorithm converges very fast – it only takes 1 second and 8 iterations to converge. Besides, the reconstruction accuracy is also very encouraging. The observed error and the underlying true error rates are close to each other and both of them are only about 0.1%. This result not only indicates that the inner products of the learned features well approximate the pairwise DTW similarities, but also verifies that we can learn accurate enough features by only computing a small portion of pairwise similarities. In addition, this test validates the low-rank assumption. It shows that a rank 30 matrix can accurately approximate a 3,765 $\times$ 3,765 DTW similarity matrix. To further demonstrate the effectiveness and efficiency of our proposed algorithm, we compare our model with the naive gradient descent [Ge et al., 2016]. For the 85 UCR time series datasets [Chen et al., 2015].
By designing an efficient algorithm that only needs to call work, i.e., Similarity Preserving RepresentAtion Learning (SPIRAL for short), on the time series clustering task. We conduct extensive experiments on all the 85 datasets in the UCR time series classification and clustering repository [Chen et al., 2015] – the largest public collection of class-labeled time-series datasets.

These datasets have widely varying characteristics: their training and testing set sizes vary from 16 to 8,926 and 20 to 8,236, respectively; the numbers of classes/clusters are between 2 and 60; and the lengths of the time series range from 24 to 2,709. Besides, this repository encompasses a wide range of domains such as medicine, engineering, astronomy, entomology, finance, and manufacture.

In our experiments, we set \( |\mathcal{D}| = [20 \log n] \), and \# features \( d = 15 \). The convergence criteria is defined as the objective decreases to be less than 1e-5 in one iteration. Given the learned features, we then feed them into some static clustering models and compare them with some state-of-the-art time series clustering methods. To conduct fair comparisons, in all DTW related algorithms and all datasets, we set the DTW window size to be the best warping size reported in [Chen et al., 2015].

To further verify that the proposed SPIRAL framework is flexible enough and works well on other similarity measures, we also conduct experiments that use move-split-merge (MSM) distance [Stefan et al., 2013] in our framework. We denote them as SPIRAL-DTW and SPIRAL-MSM, respectively. All the results were averaged from 5 trials and obtained on a Linux server with an Intel Xeon 2.40 GHz CPU and 256 GB of main memory. Our source code and the detailed experimental results are publicly available.

Since data clustering is an unsupervised learning problem, we merge the training and testing sets of all the datasets. Given the features learned by the proposed framework SPIRAL with

\[ \text{DTW similarity, we feed them into the } k\text{-Means algorithm as our clustering method, denoted as SPIRAL-DTW-}k\text{-Means.} \]

4.2 Baseline Methods

We feed the features learned by our proposed framework into the \( k\text{-Means} \) algorithm as our clustering method, denoted as SPIRAL-DTW-\( k\text{-Means} \). To compare it with the existing time series representation methods, we respectively use the Laplacian transformation [Hayashi et al., 2005] and the complex-valued linear dynamical systems (CLDS) [Li and Prakash, 2011] for extracting the same number of features as ours. We then feed the learned features into the \( k\text{-Means} \) algorithm, forming two baselines \( \text{Laplace-DTW-}k\text{-Means} \) and \( \text{CLDS-}k\text{-Means} \).

Another baseline is the state-of-the-art time series clustering algorithm \( k\text{-Shape} \) [Paparrizos and Gravano, 2015], which has been shown to outperform many state-of-the-art partitional, hierarchical, and spectral time series clustering approaches. Besides, we also compare our method with clustering algorithms \( k\text{-Means-DTW} \) and \( k\text{Medoids-DTW} \) since our ideas are similar in some respects. \( k\text{Means-DTW} \) is a popular time series clustering algorithm that uses DTW algorithm to measure pairwise distances between data points. Although it looks similar to the idea of our SPIRAL-\( k\text{Means} \) that also utilizes the DTW and \( k\text{Means} \) algorithms, it is less desirable than SPIRAL-\( k\text{Means} \) mainly because: (i) \( k\text{Means-DTW} \) suffers from a very high computational cost since it needs to compute the pairwise DTW distances between all the time series and all the cluster centers at each iteration; and (ii) the DTW distance does not satisfy the triangle inequality, thus can make the cluster centers computed by averaging multiple time series drift out of the cluster [Nienattrakul and Ratanamahatana, 2007]. By designing an efficient algorithm that only needs to call
the DTW function $O(n \log n)$ times and by embedding time series data to the Euclidean space while preserving their original similarities, the proposed method SPIRAL successfully addresses both these issues.

### 4.3 Experimental Results

We use the normalized mutual information (NMI for short) to measure the coherence between the inferred clustering and the ground truth categorization. NMI scales from 0 to 1, and a higher NMI score implies a better partition. Figure 2 (a)(b) show that SPIRAL-DTW-$k$Means performs better in 62 (which is 72.9%) and 52 (61.2%) out of 85 datasets in comparison to $k$-Shape and CLDS, respectively. Though not plotted, it is also higher in 57 (67.1%), 57 (67.1%) and 66 (77.6%) datasets compared with Laplace-DTW-$k$Means, $k$Medoids-DTW, and $k$Means-DTW. In all these comparisons, the statistical test demonstrates the superiority of SPIRAL-DTW-$k$Means. In addition to using DTW similarity in the SPIRAL framework, we also test our framework with another similarity measure move-split-merge (MSM), and denote this clustering method as SPIRAL-MSM-$k$Means. Figure 2 (c)(d) summarize the performance of all the MSM-based algorithms. The figures show that SPIRAL-MSM-$k$Means performs better in 69 (which is 81.2%) and 64 (75.3%) out of 85 datasets in comparison to Laplace-MSM-$k$Means and $k$Medoids-MSM, respectively. Table 1 reports the average NMI of all the algorithms, and the percentage of datasets over which the baseline methods perform worse than our method SPIRAL-MSM-$k$Means.

#### Table 1: The overall clustering performance of all the proposed and baseline methods. The notation < denotes the number (percentage in parenthesis) of datasets over which the baseline methods perform worse than our method SPIRAL-MSM-$k$Means.

<table>
<thead>
<tr>
<th>method</th>
<th>SPIRAL-DTW-$k$Means</th>
<th>SPIRAL-MSM-$k$Means</th>
<th>Laplace-DTW-$k$Means</th>
<th>Laplace-MSM-$k$Means</th>
<th>$k$-Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI (&lt;%)</td>
<td>0.332 (69.4%)</td>
<td>0.365 (75.3%)</td>
<td>0.171 (60.4%)</td>
<td>0.179 (60.4%)</td>
<td>0.281 (72.9%)</td>
</tr>
<tr>
<td>$k$Medoids-DTW</td>
<td>0.291 (76.5%)</td>
<td>0.294 (76.5%)</td>
<td>0.217 (71.4%)</td>
<td>0.235 (71.4%)</td>
<td>0.285 (71.4%)</td>
</tr>
<tr>
<td>$k$Medoids-MSM</td>
<td>0.237 (75.3%)</td>
<td>0.237 (75.3%)</td>
<td>0.217 (71.4%)</td>
<td>0.235 (71.4%)</td>
<td>0.285 (71.4%)</td>
</tr>
<tr>
<td>$k$Means-DTW</td>
<td>0.285 (73.5%)</td>
<td>0.285 (73.5%)</td>
<td>0.217 (71.4%)</td>
<td>0.235 (71.4%)</td>
<td>0.285 (71.4%)</td>
</tr>
<tr>
<td>$k$Means-MSM</td>
<td>0.285 (73.5%)</td>
<td>0.285 (73.5%)</td>
<td>0.217 (71.4%)</td>
<td>0.235 (71.4%)</td>
<td>0.285 (71.4%)</td>
</tr>
<tr>
<td>CLDS-$k$Means</td>
<td>0.217 (71.4%)</td>
<td>0.217 (71.4%)</td>
<td>0.217 (71.4%)</td>
<td>0.235 (71.4%)</td>
<td>0.285 (71.4%)</td>
</tr>
</tbody>
</table>

The proposed framework SPIRAL is flexible to multiple time series distance or similarity measures, including DTW and MSM. In more detail, the MSM-based method usually yields a better performance while the DTW-based method is more efficient. Both of them are orders of magnitude faster than the baseline algorithms.

Our experimental results show that even simple clustering algorithms like $k$Means can yield strong performance on our learned representations. By using some more advanced clustering algorithms, we may achieve an even better performance.

The proposed framework learns a feature representation instead of directly developing a time series model. In this way, our method is more flexible and can exploit the strengths of different clustering algorithms.

To sum up, the proposed framework is effective, efficient, and flexible. In addition to the superior performance, it is flexible in the choice of similarity measures and static learning algorithms. This enables the design of problem-specific algorithms to tackle various time series clustering problems.

### 5 Conclusions

In this paper, we show that the pairwise time series similarity matrix is close to low-rank, and propose a scalable representation learning framework that preserves pairwise similarities for time series clustering task. Our extensive empirical studies verify the effectiveness and efficiency of the proposed method.

### References


