Similarity Preserving Representation Learning for Time Series Analysis

Abstract

A considerable amount of machine learning algorithms take 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 equal or unequal lengths to a matrix format. In particular, we guarantee that the pairwise similarities between time series are well preserved after the transformation. Therefore, the learned feature representation is particularly suitable to the class of learning problems that are sensitive to data similarities. 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 extremely efficient algorithm that solves a highly non-convex and NP-hard problem to learn new features based on the partially observed similarity matrix. We use the learned features to conduct experiments on both data classification and clustering tasks. Our empirical results verify the effectiveness and efficiency of the proposed method by comparing it to several state-of-the-art time series models.

1 Introduction

Modeling time series data is an important but challenging task. It is considered by (Yang and Wu 2006) as one of the 10 most challenging problems in data mining. Although time series analysis has attracted much attention in recent years, the models that analyze time series data are still fewer than the models developed for static data. The latter category of models, which usually take matrices as their inputs, cannot directly analyze time series data due to its temporal nature, usually unequal lengths, and complex properties (Längkvist, Karlsson, and Loutfi 2014). This is a great pity since many static models are effective, robust, efficient, and easy to use. Introducing them to time series analysis can greatly enhance the development of this domain.

In this work, we bridge this gap by proposing an efficient and problem-independent unsupervised representation learning framework that is able to convert a set of time series data with equal or unequal lengths to a matrix format. In particular, the pairwise similarities between the raw time series data are well preserved after the transformation. Therefore, the learned feature representation is particularly suitable to the similarity-based static models in a variety of learning problems such as data clustering, classification, and learning to rank. In this work, we use dynamic time warping (DTW) (Berndt and Clifford 1994) algorithm to measure the pairwise similarities between time series data with equal or unequal lengths. It allows two temporal sequences that are locally out of phase to align in a non-linear manner, and is thus robust to distortion in time axis. As shown in (Rakthanmanon et al. 2013), DTW is the best measure for time series problems in a wide variety of domains.

Given a total of n time series, our first step is to generate an $n \times n$ similarity matrix $\mathbf{A}$ with $\mathbf{A}_{ij}$ equaling to the DTW similarity between the time series i and j. However, computing all the pairwise similarities requires to call the DTW algorithm $O(n^2)$ times, which can be very time-consuming when n is large. As a concrete example, generating a full similarity matrix when $n = 150,000$ takes more than 28 hours on an Intel Xeon 2.40 GHz processor with 256 GB of main memory. In order to significantly reduce the running time, we follow the setting of matrix completion (Sun and Luo 2015) by assuming that the similarity matrix $\mathbf{A}$ is of low-rank. This is a very natural assumption since DTW algorithm captures the co-movements of time series, which has shown to be driven by only a small number of latent factors (Stock and Watson 2005). According to the theory of matrix completion, only $O(n \log n)$ randomly sampled entries are needed to *perfectly* recover an $n \times n$ low-rank matrix. This allows us to only sample $O(n \log n)$ pairs of time series to generate a partially observed similarity matrix $\tilde{\mathbf{A}}$. In this way, the time spent on generating similarity matrix is significantly reduced by a factor of $O(n/\log n)$, a number that scales almost linearly with respect to n. When $n = 150,000$, it only takes about 3 minutes to construct a partially observed similarity matrix with $[20n \log n]$ observed entries,[1] more than 500 times faster than generating a full similarity matrix.

Given the generated partially observed similarity matrix

[1]Since similarity matrix is symmetric, we only need to call the DTW algorithm about $[10n \log n]$ times to generate this matrix.

$\tilde{\mathbf{A}}$, 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 $\tilde{\mathbf{A}}$, i.e., learning a feature representation $\mathbf{X} \in \mathcal{R}^{n\times d}$ such that $P_\Omega(\tilde{\mathbf{A}}) \approx P_\Omega(\mathbf{X}\mathbf{X}^\top)$, where P_Ω is a matrix projection operator defined on the observed entry set Ω. Despite its relatively simple formulation, this optimization problem is hard to solve since it is highly non-convex and NP-hard. To address this challenge, we propose a very efficient exact cyclic coordinate descent algorithm. By wisely updating variables and taking the advantage of sparse observed entries in $\tilde{\mathbf{A}}$, the proposed algorithm enjoys a very low computational cost, and thus can learn new feature representations in an extremely efficient way.

To evaluate the performance of the learned feature representation, we use more than 10 real-world data sets to conduct experiments on both data classification and clustering tasks. Our results show that classical static models that are fed with our learned features outperform the state-of-the-art time series classification and clustering algorithms in both accuracy and computational efficiency. In summary, our main contributions of this work are two-fold:

1. We propose the first, to the best of our knowledge, optimization algorithm that can solve symmetric matrix factorization problem on a partially observed matrix. The proposed algorithm enjoys very low computational cost in each iteration and yields a fast convergence.
2. We bridge the gap between time series data and a great amount of static models by learning a new feature representation of time series. The learned feature representation preserves the pairwise similarities of the raw time series data, and is general enough to be applied to a variety of learning problems.

2 Related Work

In this section, we review the existing work on learning feature representations for time series data. Among them, a family of methods use a set of derived features to represent time series. For instance, (Nanopoulos, Alcock, and Manolopoulos 2001) proposed to use the mean, standard deviation, kurtosis, and skewness of time series to represent control chart patterns. (Wang, Smith, and Hyndman 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. In order to automate the selection of features for time series classification, (Fulcher and Jones 2014) proposed a greedy forward method that can automatically select features from thousands of choices. Besides, several techniques have been proposed to represent time series by a certain types of transformation, such as discrete Fourier transformation (Faloutsos, Ranganathan, and Manolopoulos 1994), discrete cosine transformation (Korn, Jagadish, and Faloutsos 1997), discrete wavelet transformation (Chan and Fu 1999), piecewise aggregate approximation (Keogh et al. 2001), and symbolic aggregate approximation (Lin et al. 2007). In addition, 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. Due to their outstanding performance on a number of applications, they have become increasingly popular in recent years.

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

3 Similarity Preserving Representation Learning for Time Series Analysis

In this section, we first present the general idea of our similarity preserving time series representation learning framework. We then propose an extremely efficient algorithm that is significantly faster than the initial idea.

3.1 Problem Definition and General Framework

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

$$\mathrm{S}(T_i, T_j) \approx \langle f(T_i), f(T_j)\rangle \ \forall i, j \in [n], \tag{1}$$

where $\langle\cdot,\cdot\rangle$ stands for the inner product, one of the most commonly used similarity measure in analyzing static data. $\mathrm{S}(\cdot,\cdot)$ 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 to measure this similarity since it provides the best measure for a wide variety of time series problems (Rakthanmanon et al. 2013). By warping sequences non-linearly in the time dimension, the DTW algorithm 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 many applications, including computer animation (Müller 2007), surveillance (Sempena, Maulidevi, and Aryan 2011), gesture recognition (Celebi et al. 2013), signature matching (Efrat, Fan, and Venkatasubramanian 2007), protein sequence alignment (Vial et al. 2009), and speech recognition (Muda, Begam, and Elamvazuthi 2010). Normally, DTW algorithm 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 space using $\langle x, y\rangle = (\|x\|^2 + \|y\|^2 - \|x - y\|^2)/2$ (Adams 2004), we generate the DTW similarity by

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

where *0* denotes the length one time series with entry 0. We note that the similarity computed via the above equation is a more robust 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.

In order to learn the matrix $\mathbf{X}$, an intuitive idea is to factorize the similarity matrix $\mathbf{A} \in \mathcal{R}^{n\times n}$ where $\mathbf{A}_{ij} = \mathrm{S}(T_i, T_j)$. 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, it constructs a similarity matrix $\mathbf{A}$ as follows

$$\mathbf{A}_{ij} = \begin{cases} [b_i^2 + b_j^2 - \mathrm{DTW}^2(T_i, T_j)]/2, & \text{if } i < j \\ (b_i^2 + b_j^2)/2, & \text{if } i = j \\ \mathbf{A}_{ji} & \text{if } i > j, \end{cases} \quad (2)$$

where $b_i = \mathrm{DTW}(T_i, \mathit{0}),\ i = 1, \cdots, n$. In the second step, it learns an optimal data-feature matrix $\mathbf{X}$ by solving the following optimization problem

$$\min_{\mathbf{X}=(\mathbf{x}_1,\cdots,\mathbf{x}_n)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \|\mathbf{A}_{ij} - \langle \mathbf{x}_i, \mathbf{x}_j \rangle\|^2, \quad (3)$$

which can be further expressed in a matrix form

$$\min_{\mathbf{X}\in\mathcal{R}^{n\times d}} \|\mathbf{A} - \mathbf{X}\mathbf{X}^\top\|_F^2. \quad (4)$$

The problems (3) and (4) can be solved by performing eigen-decomposition on $\mathbf{A}$, i.e.,

$$\mathbf{X} = \mathbf{Q}_{1:n,1:d} \times \sqrt{\mathbf{\Lambda}_{1:d,1:d}}\ ,$$

where $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$ and the notation $\mathbf{Q}_{1:k,1:r}$ represents the sub-matrix of $\mathbf{Q}$ that includes its first k rows and the first r columns.

Although the inner products of the learned data points $\mathbf{x}_1, \cdots, \mathbf{x}_n$ can well approximate the DTW similarities of the raw time series, the idea described above is not practical since both two steps are painfully slow when n is large. In order to generate a $n \times n$ similarity matrix, we need to call the DTW algorithm $O(n^2)$ times.[2] In addition, a naive implementation of eigen-decomposition takes $O(n^3)$ time. Although we can reduce this cost by only computing the d largest eigenvalues and the corresponding eigenvectors, it is still computational intensive with a large n. As a concrete example, when $n = 150,000$ and the length of time series is 30, it takes more than 28 hours to generate the similarity matrix and more than 3 days to compute its 30 largest eigenvalues on an Intel Xeon 2.40 GHz processor with 256 GB of main memory.

3.2 A Much More Efficient Algorithm

In this subsection, we propose an extremely efficient approach that significantly reduces the computational costs of both steps while learns the new feature representation with a high precision.

[2] Indeed, we need to call the DTW algorithm at least $n(n+1)/2$ times to generate full similarity matrix $\mathbf{A}$. This number is achieved when we pre-compute all the $b_i,\ i = 1, \cdots, n$.

To significantly improve the efficiency of the first step, we make the key observation that the similarity matrix $\mathbf{A}$ should be of low-rank. This is due to the reason that the DTW algorithm measures 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). Based on 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. Following this theory, we don't need to compute all the pairwise DTW similarities. Instead, it allows us to randomly sample only $O(n \log n)$ pairs of time series, and then compute the DTW similarities only within the selected pairs. In other words, we generate a partially observed similarity matrix $\tilde{\mathbf{A}}$ with $O(n \log n)$ observed entries as

$$\tilde{\mathbf{A}}_{ij} = \begin{cases} \mathrm{S}(T_i, T_j) & \text{if } \Omega_{ij} = 1 \\ \text{unobserved} & \text{if } \Omega_{ij} = 0, \end{cases} \quad (5)$$

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

Given the partially observed similarity matrix $\tilde{\mathbf{A}}$, our second step aims to learn a new feature representation matrix $\mathbf{X}$. Instead of first completing the full similarity matrix $\mathbf{A}$ and then factorize it, we propose an efficient symmetric factorization algorithm that is able to directly factorize the partially observed similarity matrix $\tilde{\mathbf{A}}$, i.e., find a $\mathbf{X} \in \mathcal{R}^{n\times d}$ that minimizes the following optimization problem

$$\min_{\mathbf{X}\in\mathcal{R}^{n\times d}} \|P_\Omega\,(\tilde{\mathbf{A}} - \mathbf{X}\mathbf{X}^T)\|_F^2, \quad (6)$$

where $P_\Omega : \mathcal{R}^{n\times n} \to \mathcal{R}^{n\times n}$ is a projection operator defined as

$$[P_\Omega\,(\mathbf{B})]_{ij} = \begin{cases} \mathbf{B}_{ij} & \text{if } \Omega_{ij} = 1 \\ 0 & \text{if } \Omega_{ij} = 0. \end{cases} \quad (7)$$

We note that we do not need to add a regularization term to the objective function (6) since it already bounds the Frobenius norm of $\mathbf{X}$. Despite its relatively simple formulation, solving problem (6) is non-trivial since its objective function is highly non-convex and the problem is NP-hard. To address this issue, we propose a very efficient optimization algorithm that solves problem (6) based on exact cyclic coordinate descent (CD). To the best of our knowledge, this is the first algorithm that can solve the symmetric matrix factorization problem on a partially observed matrix.

At each iteration of the exact cyclic CD method, all variables but one are fixed, and that variable is updated to its optimal value. One of the main strengths of our algorithm is its capacity to update variables in an extremely efficient way. Besides, the proposed algorithm takes the advantage of sparse observed entries in $\tilde{\mathbf{A}}$ to further reduce the computational cost. In more detail, our algorithm consists of two loops that iterate over all the entries of $\mathbf{X}$ to update their values. The outer loop of the algorithm traverses through each column of $\mathbf{X}$ by assuming all the other columns known and fixed. At the i-th iteration, it optimizes the i-th column $\mathbf{X}_{1:n,i}$ by minimizing the following subproblem

$$\|\mathbf{R} - P_\Omega(\mathbf{X}_{1:n,i}\mathbf{X}_{1:n,i}^T)\|_F^2, \tag{8}$$

where $\mathbf{R}$ is the residual matrix defined as $\mathbf{R} = P_\Omega(\tilde{\mathbf{A}} - \sum_{j\neq i}\mathbf{X}_{1:n,j}\mathbf{X}_{1:n,j}^T)$.

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 $\mathbf{X}_{ji}$, we solve the following optimization problem

$$\begin{aligned}
&\min_{\mathbf{X}_{ji}} \|\mathbf{R} - P_\Omega(\mathbf{X}_{1:n,i}\mathbf{X}_{1:n,i}^T)\|_F^2 \\
\Longleftrightarrow\quad &\min_{\mathbf{X}_{ji}} \|\mathbf{R}\|_F^2 - 2\langle \mathbf{R}, P_\Omega(\mathbf{X}_{1:n,i}\mathbf{X}_{1:n,i}^T)\rangle \\
&\qquad + \|P_\Omega(\mathbf{X}_{1:n,i}\mathbf{X}_{1:n,i}^T)\|_F^2 \\
\Longleftrightarrow\quad &\min_{\mathbf{X}_{ji}} \mathbf{X}_{ji}^4 + 2(\sum_{k\in\Omega_j,\, k\neq j}\mathbf{X}_{ki}^2 - \mathbf{R}_{jj})\mathbf{X}_{ji}^2 \\
&\qquad - 4(\sum_{k\in\Omega_j,\, k\neq j}\mathbf{X}_{ki}\mathbf{R}_{jk})\mathbf{X}_{ji} + C \\
\Longleftrightarrow\quad &\min_{\mathbf{X}_{ji}} \psi(\mathbf{X}_{ji}) = \mathbf{X}_{ji}^4 + 2p\mathbf{X}_{ji}^2 + 4q\mathbf{X}_{ji} + C,
\end{aligned} \tag{9}$$

where $\Omega_i,\ i = 1,\cdots,n$ contains the indices of the observed entries in the i-th row of matrix $\tilde{\mathbf{A}}$. C is a constant that is independent of $\mathbf{X}_{ji}$. Since the $\psi(\mathbf{X}_{ji})$ is a fourth-degree polynomial function, $\mathbf{X}_{ji}$, as will be shown later, can be updated in a very efficiently way. Algorithm 1 describes the detailed steps of the proposed exact cyclic CD algorithm.

The proposed algorithm enjoys 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 Ω_j. Besides, the derivative $\nabla\psi(\mathbf{X}_{ji})$ is a third-degree polynomial, thus its roots can be computed in closed form. By using Cardano's method (Cardano and Witmer 1993), the optimal solution of $\mathbf{X}_{ji}$ 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 $\mathbf{R}$ can be updated by only considering the observed entries. To sum up, the proposed algorithm has a per-iteration cost of $O(dn\log n)$, thus is significantly faster than the recent matrix factorization algorithms that take at least $O(dn^2)$ time in each iteration (Vandaele et al. 2015; Hsieh and Dhillon 2011; Yu et al. 2012).

In addition to a low per-iteration cost, we also expect that 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 hypothesis 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 $\mathbf{A}$ by computing the DTW similarities between all the time series pairs, and then randomly sample $[20n\log n]$ of its entries to generate a partially observed matrix $\tilde{\mathbf{A}}$. We call the proposed algorithm to factorize matrix $\tilde{\mathbf{A}}$ by setting $d = 30$. To measure the performance of the proposed method, we compute two error rates, i.e., the observed error $\|P_\Omega(\tilde{\mathbf{A}} - \mathbf{X}\mathbf{X}^T)\|_F/\|P_\Omega(\tilde{\mathbf{A}})\|_F$ and the underlying true error $\|\mathbf{A} - \mathbf{X}\mathbf{X}^T\|_F/\|\mathbf{A}\|_F$, at each iteration. Figure 1 shows how they converge as a function of time. This figure clearly shows that the proposed exact cyclic CD algorithm converges very fast – it only takes 1 second and 8 iterations to converge. Besides, the construction 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%, indicating that the inner products of the learned features can well approximate the pairwise DTW similarities of the raw time series. This result also validates the low-rank assumption. It shows that a $3,765\times 3,765$ DTW similarity matrix can be accurately approximated by a rank 30 matrix.

Algorithm 1 Efficient Exact Cyclic Coordinate Descent Algorithm for Solving the Optimization Problem (6)

```
 1: Inputs:
      • $\tilde{\mathbf{A}} \in \mathcal{R}^{n\times n}$: partially observed similarity matrix
      • $\Omega_i,\ i = 1,\cdots,n$: indices of the observed entries in the $i$-th row of matrix $\tilde{\mathbf{A}}$
      • $I$: number of iterations
      • $d$: dimension of features
 2: Initializations:
      • $\mathbf{X}^{(0)} \leftarrow \mathbf{0_{n\times d}}$
      • $\mathbf{R} \leftarrow P_\Omega(\tilde{\mathbf{A}} - \mathbf{X}^{(0)}\mathbf{X}^{(0)\top}) = P_\Omega(\tilde{\mathbf{A}})$
 3: for $t = 1,\cdots,I$ do
 4:     $\mathbf{X}^{(t)} \leftarrow \mathbf{X}^{(t-1)}$
 5:     for $i = 1,\cdots,d$ do
 6:         $\mathbf{R} \leftarrow \mathbf{R} + P_\Omega(\mathbf{X}_{1:n,i}^{(t)}\mathbf{X}_{1:n,i}^{(t)\top})$
 7:         for $j = 1,\cdots n$ do
 8:             $p \leftarrow \sum_{k\in\Omega_j}\mathbf{X}_{ki}^{(t)2} - \mathbf{X}_{ji}^{(t)2} - \mathbf{R}_{jj}$
 9:             $q \leftarrow -\sum_{k\in\Omega_j}\mathbf{X}_{ki}^{(t)}\mathbf{R}_{jk} + \mathbf{X}_{ji}^{(t)}\mathbf{R}_{jj}$
10:             $\mathbf{X}_{ji}^{(t)} \leftarrow \text{argmin}\{\mathbf{X}_{ji}^4 + 2p\mathbf{X}_{ji}^2 + 4q\mathbf{X}_{ji}\}$
11:         end for
12:         $\mathbf{R} \leftarrow \mathbf{R} - P_\Omega(\mathbf{X}_{1:n,i}^{(t)}\mathbf{X}_{1:n,i}^{(t)\top})$
13:     end for
14: end for
15: Output: $\mathbf{X}^{(I)}$
```

4 Experiments

In this section, we evaluate the proposed framework, i.e., Similarity PreservIng RepresentAtion Learning (SPIRAL for short), on both data classification and clustering tasks. For both tasks, we learn new feature representations by setting the number of features $d = 30$, the number of iterations $I = 20$, and the sample size $|\Omega| = [20n\log n]$. We then fed the learned features into some static models and compare them with the state-of-the-art time series classification

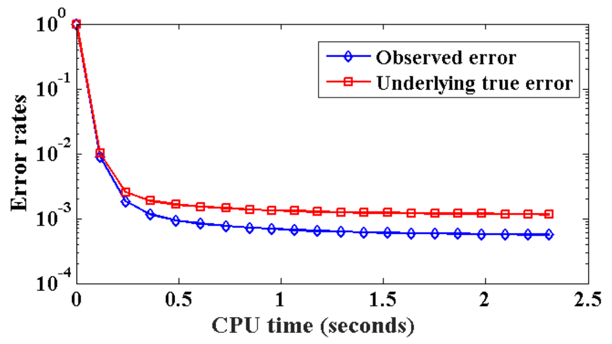

Figure 1: Two error rates as a function of CPU time on UCR Non-Invasive Fetal ECG Thorax1 data set

and clustering algorithms. Since the DTW function is also called by some baseline algorithms, we set a same DTW window size $\min(40, \lceil \text{average time series length}/10 \rceil)$ for all the DTW-related algorithms evaluated here. All the results were obtained on a Linux server with an Intel Xeon 2.40 GHz CPU and 256 GB of main memory.

4.1 Classification Task

Six real-world time series data sets are used in our classification analysis. As a research institute, we have partnered with one of the largest American financial companies on a challenge problem of predicting its clients' propensity of trading options. We are provided with a historical records data of $64,523$ sampled clients with the lengths of the time series range from 1 month to 67 months. The data has 76 dynamic attributes but none of them contains the option trading related information. Given this data, our task is to predict whether the clients will trade options in the next 100 days.[3] Besides, we also conduct experiments on 5 benchmark data sets, i.e., FordA, Wafer, PhalangesOutlinesCorrect (POC), ItalyPowerDemand (IPD), and HandOutlines (HO), from the UCR Time Series Repository (Chen et al. 2015). The data sets used in our experiments have widely varying sizes and encompass multiple domains such as finance, healthcare, and manufacture. Table 1 summarizes the statistics of these data sets.

Given the feature representations learned by the proposed SPIRAL framework, we fed them into the algorithms of *XGBoost* (Chen and Guestrin 2016) and *ℓ_2-regularized logistic regression* that is implemented in LibLinear (Fan et al. 2008). These approaches, denoted as *SPIRAL-XGB* and *SPIRAL-LR*, respectively, are compared with the state-of-the-art time series classification algorithms *NN-DTW* (Wang et al. 2013) and *Long Short-Term Memory (LSTM)* (Hochreiter and Schmidhuber 1997). For the XGBoost algorithm, we use the default parameters specified in its source code. The parameter of logistic regression is determined by a 5-fold cross validation.

[3]We would like to thank the analytics team of this broker since it is their question "*Is it possible to bridge the gap between time series data and the static models we are familiar with?*" that motivates this paper.

Table 1: *Description of Classification Data sets*

Data sets	# training time series	# testing time series	length of time series
Financial Company	51,618	12,905	1 – 67
FordA	1,320	3,601	500
Wafer	1,000	6,174	152
POC	1,800	858	80
IPD	67	1,029	24
HO	370	1,000	2,709

Table 2: *Average AUC Scores of SPIRAL-XGB, SPIRAL-LR and the baseline algorithms NN-DTW, and LSTM*

Data sets	SPIRAL -XGB	SPIRAL -LR	NN -DTW	LSTM
Financial Company	**0.84**	0.83	0.76	0.79
FordA	**0.73**	0.68	0.66	0.56
Wafer	**0.99**	0.93	**0.99**	0.92
POC	**0.77**	0.69	0.69	0.66
IPD	0.95	**0.98**	0.95	0.90
HO	**0.88**	0.82	0.76	0.73

We use AUC (area under the ROC Curve) to measure the classification performance. All the experiments in this study are repeated five times, and the AUCs averaged over the five trials are reported in Table 2. From this table, we first observe that XGBoost and logistic regression algorithms that are fed with our learned features outperform the two baseline algorithms on all the data sets. In particular, the method SPIRAL-XGB yields the best performance on five out of six data sets, and the method SPIRAL-LR performs the best on the other data set. More encouragingly, SPIRAL-LR can achieve similar or better performance than NN-DTW and LSTM on almost all the data sets. This clearly shows that the feature representation learned by the proposed method is powerful – even classical models such as logistic regression can achieve satisfactory performance on it.

On the other hand, although LSTM has been successfully applied to a number of sequence prediction tasks, it fails to deliver strong performance in our empirical study. We conjecture that this is because the data sets are not large enough for LSTM to model complex structures. Besides, NN-DTW is considered as hard-to-beat in the literature (Wang et al. 2013). However, it also yields an overall worse performance than SPIRAL-LR and SPIRAL-XGB. One possible reason is that NN-DTW uses 1-nearest neighbor classifier, thus is sensitive to noises and outliers. This observation shows another advantage of the proposed method: by learning a feature representation instead of directly developing a time series model, our method is more flexible and can exploit the strengths of different static algorithms.

In addition to the superior classification performance, SPIRAL-LR and SPIRAL-XGB are very efficient as well. They have a significantly lower running time than that of the NN-DTW algorithm. For example, it takes NN-DTW more than 5 hours to classify the clients in the Financial Company data set while the running time for SPIRAL-LR and

SPIRAL-XGB are only about 4 minutes.

4.2 Clustering Task

Similar to time series classification, time series clustering is also an important task that has found numerous applications. To further test our learned features on the clustering task, we conduct experiments on another 7 UCR time series data sets that come from multiple domains. Since data clustering is an unsupervised learning task, we combine their training and testing sets together. Statistics of these data sets are summarized in Table 3.

We fed the features learned by the proposed framework into the kMeans algorithm as our clustering method, and compare it to the state-of-the-art time series clustering algorithm *k-Shape* (Paparrizos and Gravano 2015). Study in (Paparrizos and Gravano 2015) have shown that k-Shape performs better than all state-of-the-art partitional, hierarchical, and spectral clustering approaches. Besides, we also compare our method with clustering algorithms *kMeans-DTW* and *CLDS* (Li and Prakash 2011) since our ideas are similar in some respects. kMeans-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-kMeans that also utilizes the DTW and kMeans algorithms, it is less desirable than SPIRAL-kMeans mainly because: (i) kMeans-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 (Niennattrakul 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 representation learning framework SPIRAL successfully addresses both these issues. Similar to the idea of representation learning presented in this paper, CLDS also learns a new feature representation of the time series data, and then partition the learned representation via an EM-like algorithm.

In this study, 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 value implies a better partition. Each experiment is repeated five times, and the performance averaged over the five trials is reported in Table 4. Compared to all the baseline algorithms, our clustering method SPIRAL-kMeans yields the best performance on six out of seven data sets, indicating that it delivers the state-of-the-art performance. In addition, SPIRAL-kMeans is the most efficient algorithm among the four. It takes not more than 15 minutes to partition all the data sets analyzed here. As a comparison, although kMeans-DTW works well on the SLC data set, it spends more than 2 days for a single run. Besides, the CLDS algorithm has a high time and space complexity. It fails to output meaningful partitions on 3 data sets due to running out of memory.

We finally note that kMeans is just one choice of clustering algorithms that can take our learned features as the input. By replacing it with some more powerful clustering algorithms, we can expect to achieve an even better clustering performance.

Table 3: *Description of Clustering Data Sets Swedish Leaf, Cricket_X, uWaveGestureLibrary_X (uWGLX), 50Words, StarLightCurves (SLC), Synthetic Control (SC), and ElectricDevices (ED)*

Data sets	number of time series	length of time series	number of clusters
Swedish Leaf	1,125	128	15
Cricket_X	780	300	12
uWGLX	4,478	315	8
50words	905	270	50
SLC	9,236	1,024	3
SC	600	60	6
ED	16,637	96	7

Table 4: *Average NMI Scores of SPIRAL-kMeans and the baseline algorithms k-Shape, CLDS, and kMeans-DTW. N/A means that the clustering task cannot be completed due to running out of memory.*

Data sets	SPIRAL -kMeans	k-Shape	CLDS	kMeans -DTW
Swedish Leaf	**0.67**	0.58	**0.67**	0.57
Cricket_X	**0.34**	0.32	0.27	0.26
uWGLX	**0.47**	0.42	N/A	0.39
50words	**0.69**	0.57	0.62	0.54
SLC	0.61	0.58	N/A	**0.64**
SC	**0.80**	0.78	0.47	0.76
ED	**0.36**	0.24	N/A	0.22

5 Conclusions

In this paper, we propose a similarity preserving representation learning framework for time series analysis. Given a set of n time series, the key idea is to first generate a partially observed similarity matrix with $O(n \log n)$ observed DTW similarities, and then factorize this matrix to learn a new feature representation. To this end, we propose the first symmetric matrix factorization algorithm on a partially observed matrix. The proposed algorithm updates variables via exact cyclic coordinate descent, and enjoys a very low computational cost in each iteration. The feature representation learned by the proposed framework preserves the DTW similarities of the raw time series data, and is general enough to be applied to a variety of learning problems. Our empirical studies on both classification and clustering tasks verify the effectiveness and efficiency of the proposed method.

References

Adams, C. C. 2004. *The knot book: an elementary introduction to the mathematical theory of knots*. American Mathematical Soc.

Berndt, D. J., and Clifford, J. 1994. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, 359–370. Seattle, WA.

Cardano, G., and Witmer, T. R. 1993. *Ars magna or the rules of algebra*.

Celebi, S.; Aydin, A. S.; Temiz, T. T.; and Arici, T. 2013. Gesture recognition using skeleton data with weighted dynamic time warping. In *VISAPP (1)*, 620–625.

Chan, K.-P., and Fu, A. W.-C. 1999. Efficient time series matching by wavelets. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, 126–133. IEEE.

Chen, T., and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. *arXiv preprint arXiv:1603.02754*.

Chen, Y.; Keogh, E.; Hu, B.; Begum, N.; Bagnall, A.; Mueen, A.; and Batista, G. 2015. The ucr time series classification archive. `www.cs.ucr.edu/~eamonn/time_series_data/`.

Deng, H.; Runger, G.; Tuv, E.; and Vladimir, M. 2013. A time series forest for classification and feature extraction. *Information Sciences* 239:142–153.

Efrat, A.; Fan, Q.; and Venkatasubramanian, S. 2007. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *Journal of Mathematical Imaging and Vision* 27(3):203–216.

Elman, J. L. 1990. Finding structure in time. *Cognitive science* 14(2):179–211.

Faloutsos, C.; Ranganathan, M.; and Manolopoulos, Y. 1994. *Fast subsequence matching in time-series databases*, volume 23. ACM.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. Liblinear: A library for large linear classification. *JMLR* 9(Aug):1871–1874.

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

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hsieh, C.-J., and Dhillon, I. S. 2011. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *KDD*, 1064–1072. ACM.

Keogh, E.; Chakrabarti, K.; Pazzani, M.; and Mehrotra, S. 2001. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems* 3(3):263–286.

Korn, F.; Jagadish, H. V.; and Faloutsos, C. 1997. Efficiently supporting ad hoc queries in large datasets of time sequences. *ACM SIGMOD Record* 26(2):289–300.

Längkvist, M.; Karlsson, L.; and Loutfi, A. 2014. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters* 42:11–24.

Li, L., and Prakash, B. A. 2011. Time series clustering: Complex is simpler! In *ICML*, 185–192.

Lin, J.; Keogh, E.; Wei, L.; and Lonardi, S. 2007. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery* 15(2):107–144.

Muda, L.; Begam, M.; and Elamvazuthi, I. 2010. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*.

Müller, M. 2007. Dtw-based motion comparison and retrieval. *Information Retrieval for Music and Motion* 211–226.

Nanopoulos, A.; Alcock, R.; and Manolopoulos, Y. 2001. Feature-based classification of time-series data. *International Journal of Computer Research* 10(3):49–61.

Niennattrakul, V., and Ratanamahatana, C. A. 2007. Inaccuracies of shape averaging method using dynamic time warping for time series data. In *International conference on computational science*, 513–520. Springer.

Paparrizos, J., and Gravano, L. 2015. k-shape: Efficient and accurate clustering of time series. In *ACM SIGMOD*, 1855–1870. ACM.

Rakthanmanon, T.; Campana, B.; Mueen, A.; Batista, G.; Westover, B.; Zhu, Q.; Zakaria, J.; and Keogh, E. 2013. Addressing big data time series: mining trillions of time series subsequences under dynamic time warping. *ACM TKDD* 7(3):10.

Sempena, S.; Maulidevi, N. U.; and Aryan, P. R. 2011. Human action recognition using dynamic time warping. In *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*, 1–5. IEEE.

Stock, J. H., and Watson, M. W. 2005. Implications of dynamic factor models for var analysis. Technical report, National Bureau of Economic Research.

Sun, R., and Luo, Z.-Q. 2015. Guaranteed matrix completion via nonconvex factorization. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, 270–289. IEEE.

Vandaele, A.; Gillis, N.; Lei, Q.; Zhong, K.; and Dhillon, I. S. 2015. Coordinate descent methods for symmetric nonnegative matrix factorization. *CoRR* abs/1509.01404.

Vial, J.; Noçairi, H.; Sassiat, P.; Mallipatu, S.; Cognon, G.; Thiébaut, D.; Teillet, B.; and Rutledge, D. N. 2009. Combination of dynamic time warping and multivariate analysis for the comparison of comprehensive two-dimensional gas chromatograms: application to plant extracts. *Journal of Chromatography A* 1216(14):2866–2872.

Wang, X.; Mueen, A.; Ding, H.; Trajcevski, G.; Scheuermann, P.; and Keogh, E. 2013. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery* 26(2):275–309.

Wang, X.; Smith, K.; and Hyndman, R. 2006. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery* 13(3):335–364.

Yang, Q., and Wu, X. 2006. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making* 5(04):597–604.

Yu, H.-F.; Hsieh, C.-J.; Si, S.; and Dhillon, I. 2012. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *ICDM*, 765–774. IEEE.