Time-Warped Sparse Non-negative Factorization for Functional Data Analysis

CHEN ZHANG, Tsinghua University, China
STEVEN C.H. HOI, Singapore Management University
FUGEE TSUNG, Hong Kong University of Science and Technology

This paper proposes a novel time-warped sparse non-negative factorization method for functional data analysis. The proposed method on the one hand guarantees the extracted basis functions and their coefficients to be positive and interpretable, and on the other hand is able to handle weakly correlated functions with different features. Furthermore, the method incorporates time warping into factorization and hence allows the extracted basis functions of different samples to have temporal deformations. An efficient framework of estimation algorithms is proposed based on a greedy variable selection approach. Numerical studies together with case studies on real-world data demonstrate the efficacy and applicability of the proposed methodology.

CCS Concepts: • **Theory of computation → Unsupervised learning and clustering**; • **Computing methodologies → Non-negative matrix factorization**;

Additional Key Words and Phrases: Non-negative functional factorization; Multivariate functional data; Sparse representation; Time warping

ACM Reference Format:
Chen Zhang, Steven C.H. Hoi, and Fugee Tsung. 2019. Time-Warped Sparse Non-negative Factorization for Functional Data Analysis. 1, 1 (June 2019), 23 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Non-negative functional data, where data are in the form of non-negative functions are increasingly popular in many real-word applications. A concrete motivating example is the subway passenger flow data analysis [9, 55], where passenger entry rates of each station in a city's subway system are measured in real time via smart card payment records at the turnstiles. For example, Figure 1 shows the entry rate curves of eight selected stations in the Boston subway system per day from 5AM to next 1AM for in total 30 days. Using these data we can analyze passenger flow patterns from which further inference of the whole subway network can be conducted.

As shown in Figure 1, the passenger enter rates are always non-negative and somewhat smooth over time with strong time dependence. Hence, each curve can be regarded as one non-negative function, and we can use functional factorization [47] methods for feature extraction. One thing to be noted for factorizing non-negative functions is the algorithm's interpretability. To be more

This work was supported in part by Tsinghua University Intelligent Logistics and Supply Chain Research Center Grant THUCSL20182911756-001, in part by the National Natural Science Foundation of China under Grants 71901131, 71931006, 71932006, and in part of Hong Kong Research Grant Council General Research Fund under Grants 16216119 and 16201718.
Authors' addresses: Chen Zhang, Tsinghua University, Beijing, China, zhangchen01@tsinghua.edu.cn; Steven C.H. Hoi, Singapore Management University, Singapore, chhoi@smu.edu.sg; Fugee Tsung, Hong Kong University of Science and Technology, Hong Kong, season@hkust.hk.

XXXX-XXXX/2019/6-ART $15.00
https://doi.org/10.1145/1122445.1122456

specific, for example, as shown in Figure 1, if we treat monitoring peak, evening peak, night peak, bell shape, etc., as different features, these feature functions actually represent different passenger flow patterns and hence are expected to preserve non-negativity for model explanation purpose. Furthermore, since a station's entry rate function can only be a summation, instead of a subtraction of different passenger flow patterns, the representation should be parts-based, i.e., only allow additive, not subtractive, combinations [33]. As such, we should consider this non-negative property and preserve it during the factorization process. This motivates the goal of this paper, non-negative factorization for functional data analysis.

One challenge for such non-negative functional factorization is that different functions usually have deformation (aka misalignment, phase variation), such as shifting and stretching, along the temporal axis with each other. For example, Figure 2a draws the entry rate functions of some stations with morning peaks in a certain day. It is clear to see the peaks of different stations have time deformation with each other. This might be because different residential districts have various distances from the central business district (CBD) and consequently require different travel time, such as stations with longer travel time are expected to have earlier morning peaks. Similar phenomenon also occurs for evening peaks. Furthermore, even for the same station, its evening peaks in different workdays may also deform with each other due to day-to-day variations, as shown in Figure 2c. As such, direct factorization without considering temporal deformations may introduce additional noise, and consequently dislocate and mislead the extracted features[30, 43].

Another challenge is how to describe the rich features (or complex cross-correlations) of different functions in the factorization. In the above example, entry rate functions of stations in the residential areas share similar morning peak features. Yet entry rate functions of stations in the CBD share similar evening peak features, and passenger flows of stations in the commercial districts share similar night peaks. Functions with similar features demonstrate strong cross-correlations with each other, such as stations close to each other or in the same functional distinct. In contrast, functions with different features have very weak cross-correlations, such as stations far from each other or in different functional districts. When the number of functions is large, the chance that all the functions are strongly correlated is quite small. In other words, functions may come from different clusters. Functions in the same cluster have strong cross-correlations, while functions in different clusters have no cross-correlations. Consequently, a good factorization model should be able to extract these abundant features and describe these complex cross-correlation structures accurately.

In addition to the above example, multivariate non-negative functional data with 1) time deformation and 2) different features with weak correlation structures, exist in many other applications. For example, in financial area, real-time transactions of different stocks are recorded every day for analysis [17]. In electroencephalography (EEG) tests, multiple electrodes are placed at different places to record the brain activity over a certain time period for pattern detection [48]. Targeting at the above challenges, this paper proposes a new *time-warped sparse non-negative functional factorization method*. Our contributions are twofold. First, our method can handle the problem of feature deformation, by allowing the learned features of different functional samples to have time warping. Second, our method can describe weak correlations between non-negative functions with different features, and meanwhile can guarantee the extracted features together with their coefficients to be positive and interpretable. An efficient framework of estimation algorithms is proposed based on a greedy variable selection approach. Numerical studies together with case studies on real-world data demonstrate the efficacy and applicability of the proposed methodology.

The remainder of the paper is organized as follows. Section 2 reviews the state-of-the-art methods on functional data analysis. Section 3 introduces our proposed time-warped sparse non-negative functional factorization model in detail. Section 4 discusses the model inference procedure. Section

5 uses some numerical studies to demonstrate the advantages of the proposed model by comparing it with some state-of-the-art methods. We also apply the developed method into two real-data examples in subway passenger flow analysis. Finally, Section 6 concludes this paper with remarks.

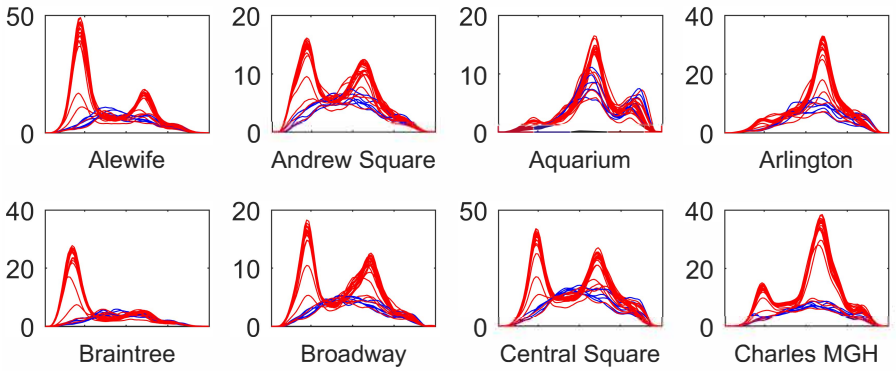


Fig. 1. Entry rates of eight selected stations in the Boston subway data set. Red colors denote entry rates in workdays, and blue colors denote entry rates at weekends.

2 RELATED WORKS

In general, our work is closely related to the following topics.

Negative time series analysis has been addressed in many pioneer works, which can be grouped into two major categories. The first category revises traditional time series models to impose non-negativity regularization [2, 8, 24]. These models usually can only describe stationary and linear time series and hence have limited description powers for functional data, since they are usually nonlinear and nonsationary and have complex features which cannot be captured by any specific parametric model (such as ARIMA, or nonlinear state space models). The second category can describe nonstationary and nonlinear data, by using dimension reduction and feature selection methods. In particular, most of current works use non-negative matrix factorization (NMF [33]) for functional data decomposition and feature learning [12, 32, 38, 48]. Since NMF can decompose high dimensional non-negative data with both non-negative basis functions and coefficients, these methods can return represented basis functions that are easily interpretable in practical applications. In particular, [48] proposed to use NMF to perform model reduction and

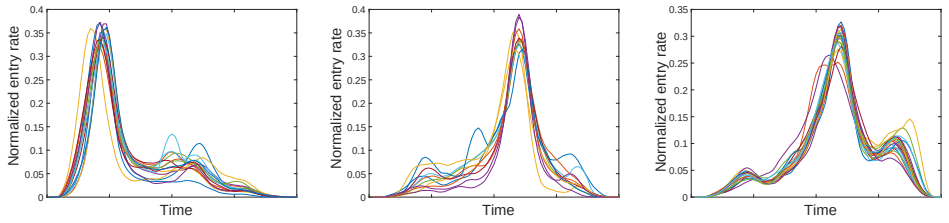


Fig. 2. Entry rates of (a) stations with morning peaks in one working day; (b) stations with evening peaks in one working day; and (c) the station Aquarium in serveral working days.

extract hidden non-negative factors for the intrinsic mode functions of EEG data. [12] derived a NMF that can account for temporal dependencies of sequential coming functional data by explicitly incorporating a temporal constraint for the coefficients into the NMF update rules. [52] proposed a probabilistic NMF for blind signal separation. It regularized the basis functions to be smooth by assuming they evolve according to a Markov process. [11] combined NMF and autoregressive structure and obtained factors that were more smooth and/or sparse. [38] further improved the data recovery by constraining temporal correlation on individual time series (not factors). Recently, [37] also considered including side information into NMF for better time series prediction. Besides time series prediction, NMF has also been studied for other applications, such as feature selection [51], local discriminative analysis[40, 50], semi-supervised learning[39, 41], etc. However, one limitation for the above methods is that they assume all the non-negative functions have similar features and are strongly correlated, since they represent each function as a linear combination of all the extracted basis functions (i.e., features). This assumption leads these methods fail when multiple functions have diverse features with sparse cross-correlations. Furthermore, they are more in favor of traditional time series analysis by treating each function as a collection of data points on an equally spaced grid, instead of considering each function as a single, structured object. Consequently, they cannot be applied for general functions with unequally spaced grids, let alone tackling with the time deformation problem.

Multivariate functional data analysis has been extensively studied in the literature of statistics, machine learning and beyond. Most existing works apply dimension reduction analysis for function feature extraction and depiction through various approaches. In particular, [44] proposed a dynamic factor model to extract common factors from multiple functional data. [13, 18, 45] introduced several multi-channel functional principal component analyses (MFPCA) methods to describe the within-function and between-function correlations. [14] proposed a pairwise interaction model based on the cross-covariance surfaces between functions. For weakly correlated functions, recently [59] proposed a sparse MFPCA (SMFPCA) by representing every function using as few bases as possible. Similar ideas are also proposed in [29], where the sparse coding is used for functional connectivity MRI analysis. However, the extracted basis functions by these methods can be either positive or negative, and functions' coefficients on the basis functions can also be of arbitrary sign. As such, the linear combinations generally involve complex cancellations between positive and negative numbers, which makes the results not interpretable for non-negative functions. In summary, while the literature and available methods for statistical analysis of multivariate functional data have been rapidly increasing during the last two decades, multivariate non-negative functional data analysis has been a largely overlooked topic, especially the sparse representation for high-dimensional non-negative functional data, let alone the non-trivial parts of its extending methodology, i.e., time deformation.

Time deformation is a practical problem generally exists in many real-world applications. It poses severe obstacles to the application of functional versions of commonly used multivariate data analyses, such as computing point-wise means and correlations, feature extraction and data decomposition. To solve the time deformation problem, time warping or curve registration has been proposed in the statistical literature [16, 36, 62]. A warping method can be defined as a process of aligning features of several functions by monotone transformations of their domain. The aligned functions exhibit only amplitude variation, and the domain transformations, called warping functions, capture the phase variation in the original functions. Different time warping procedures have been proposed in statistical literature, such as Landmark registration[21], algorithms based on dynamic programming[31], etc.

However, most of these works focus on merely time warping or its application for multiple time series clustering. Yet for data factorization or representation, the only two works are [1]

and [3], which proposed to first align functions and remove deformations, by assuming that the observed functional samples are deformations from their true signals represented by the dictionary. Then the aligned functions can be used for factorization. However, this assumption may bring additional problems, since in many applications, deformation does not occur in functional samples directly, but in their basis functions. More specifically, it is the basis functions that have misaligned influence on different functional samples, as illustrated in Figure 3. For instance, in the previous example, if we treat "morning peak" as a template feature, the real features of morning peaks of different stations in various days actually deform from this template. This indicates that given a basis dictionary, each function is actually a weighted combination of their own basis functions, which are deformations of the template bases in the dictionary. However, so far no existing work targets at effectively solving this feature deformation problem.

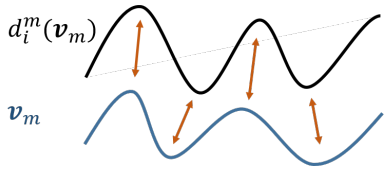


Fig. 3. Illustration of a deformation $d_i^m(\cdot)$ from the template basis function $\mathbf{v}_m$ for a functional sample i.

3 TIME-WARPED SPARSE NON-NEGATIVE FUNCTIONAL FACTORIZATION

Assume we have N functional samples $X_i(t), i = 1, \ldots, N$. Each sample $X_i(t)$ is a non-negative function in the finite time interval $\mathcal{T} = (0, T]$ with $X_i(t) \geq 0, \forall t \in \mathcal{T}$. Following the general dictionary learning frameworks in the literature, we aim to decompose each sample $X_i(t)$ as a linear combination of several basis functions, i.e.,

$$X_i(t) = \sum_{m=1}^{M} \phi_i^m v_m(t) + e_i(t), \tag{1}$$

where $v_m(t), t \in (0, T], m = 1, \ldots, M$ are the basis functions representing features, and ϕ_i^m is the coefficient that $v_m(t)$ takes for $X_i(t)$. As mentioned in Section 1, to make the factorization interpretable, we restrict the basis functions $v_m(t), t \in (0, T], m = 1, \ldots, M$ in (1) together with the coefficients ϕ_i^m to be exactly non-negative, i.e., $v_m(t) \geq 0, \forall t \in (0, T]$, and $\phi_i^m \geq 0, \forall m = 1, \ldots, M, i = 1, 2 \ldots, N$.

As mentioned in Section 1, in reality, the basis functions may have misaligned influence on different samples, due to sample-to-sample variations. Direct feature extraction without considering this misalignment would lead that the extracted basis functions mix the misaligned features from different samples together and result in over-flattened features. This will also cause that one original basis function is misidentified as more with different deformation magnitudes and consequently result in redundant features. To solve this problem, here we propose a new time-warped factorization. Our key idea is to define $v_m(t), m = 1, \ldots, M$ as template bases, and then set the real basis functions of different samples as deformations of these templates, denoted as $d_i^m(v_m(t))$. Then the functional samples are actually combinations of $d_i^m(v_m(t))$ as

$$X_i(t) = \sum_{m=1}^{M} \phi_i^m d_i^m(v_m(t)) + e_i(t), \tag{2}$$

where $d_i^m(\cdot) \in \mathcal{D}$ is the deformation operator of the basis function $v_m(t)$ for sample $X_i(t)$, and $\mathcal{D}$ is the set that contains all considered deformation operators $d(\cdot)$.

Furthermore, recall that the current form of (2) sets each function as a linear combination of all the M basis functions, indicating that all the functions share a common set of basis functions and their cross-correlations are essentially described by the similarity of $\phi_i^m (m = 1, \ldots, M)$. Consequently, this model is only valid when all the functions exhibit strong cross-correlations or share similar features. However, as mentioned earlier, in practice, different functions may have quite different features, and are usually weakly correlated. In other words, for a certain function, it is not necessary to have non-zero coefficients for all the M basis functions. For example, in the previous example of subway entry rate data in Figure 1, the weekday entry rates of station Arlington only have non-zero coefficients on the basis function of "evening peak", while the weekday entry rates of station Braintree only have non-zero coefficients on the basis function of "morning peak". As such, motivated by sparse representations, it is more desirable to set $\boldsymbol{\phi}_i = [\phi_i^1, \ldots, \phi_i^M]$ to be a sparse vector to enforce the coefficients of unrelated basis functions to be zero. Then we get the following objective function

$$\begin{aligned} \min_{v_m(t), \phi_i^m, d_i^m \in \mathcal{D}} & \sum_{i=1}^{N} \int_{\mathcal{T}} \left(X_i(t) - \sum_{m=1}^{M} \phi_i^m d_i^m(v_m(t)) \right)^2 \mathrm{d}t \\ s.t. \quad & v_m(t) \geq 0, \forall m = 1, \ldots, M, t \in \mathcal{T} \\ & \phi_i^m \geq 0, \text{ and } ||\boldsymbol{\phi}_i||_0 < h, \forall i = 1, \ldots, N, m = 1, \ldots, M. \end{aligned} \tag{3}$$

The l_0 sparsity pseudo-norm indicates the number of non-zero elements of a vector. In this way, (3) aims to find the best sparse approximation for $X_i(t)$ by selecting less than h basis functions $v_m(t)$, deforming them by optimizing d_i^m, and scaling them by multiplying with ϕ_i^m.

Remark 1. *The time deformation provides the possibility of solving the feature misalignment problem. However, we should be cautious about choosing $\mathcal{D}$. To prevent it from giving too much flexibility for the estimated $d_i^m(v_m(t))$ and causing over-fitting to some degree, here we recommend to regularize $\mathcal{D}$. Two of the most commonly used constraints are the Sakoe-Chiba Band [49] and the Itakura Parallelogram [27]. In this paper, we use the Sakoe-Chiba band constraint and will explain it in more details in the following section.*

4 ALGORITHMS FOR MODEL ESTIMATION

In this section, we will discuss the proposed solution to optimize (3) in detail. Although our original interest is in the continuous functions, i.e., the underlying stochastic processes, in reality these stochastic processes are often latent and cannot be observed directly, as data can only be collected discretely over time. Therefore, a general assumption commonly used in related literatures [26, 58, 59] is that functional data are recorded on a dense time grid of ordered times. Specifically, we observe $X_i(t)$ at a grid of n_i discrete time points $\{t_{il}, 1 \leq l \leq n_i\}$ with $0 < t_{i1} < \ldots < t_{in_i} \leq T$ (It is to be noted that in this paper, for different samples, these time points t_{il} can be different, while in the literatures above, since they cannot address the problem of deformation, they have to set much stronger regulation on t_{il} such that all the samples should have the same time grid of $t_{il}, i = 1, \ldots, N, l = 1, \ldots, n$ and the sampling grid should be equally spaced, i.e., $t_{l+1} - t_l = t_l - t_{l-1}$). Then every function can be denoted as a vector $\mathbf{X}_i \in \mathcal{R}^{n_i \times 1}$. Similarly, we assume its basis functions $d_i^m(v_m(t)) \in \mathcal{R}^{n_i \times 1}$ are at the same n_i time points as $\mathbf{X}_i$, As to the template basis functions, we aim to estimate them in n fixed equally spaced time points, $0 < t_1 < \ldots < t_n \leq T$, and denote these n points of $v_m(t)$ as $\mathbf{v}_m = [v_m(t_1), \ldots, v_m(t_n)] \in \mathcal{R}^{n \times 1}$. Usually we can set $n = \lfloor \sum_i n_i / N \rceil$. Then (3)

can be reformulated as

$$\begin{aligned} \min_{\mathbf{v}_m, \phi_i^m, d_i^m \in \mathcal{D}} \ & \frac{1}{2} \sum_{i=1}^{N} ||\mathbf{X}_i - \sum_{m=1}^{M} \phi_i^m d_i^m(\mathbf{v}_m)||_2^2 \\ s.t. \quad & v_m(t_l) \geq 0, \forall m = 1, \ldots, M, l = 1, \ldots, n, \\ & \phi_i^m \geq 0, \text{ and } ||\boldsymbol{\phi}_i||_0 < h, \forall i = 1, \ldots, N, m = 1, \ldots, M. \end{aligned} \tag{4}$$

Unlike traditional sparse non-negative matrix factorization methods [20, 25, 42], (4) is a large-scale sparse regression which requires optimization over both $\{\boldsymbol{\phi}_i, \mathbf{v}_m\}$ and d_i^m. The optimization over d_i^m is especially difficult, as it is an operator optimization. Here we consider the block coordinate descent method to iteratively solve $\{\mathbf{v}_m, \phi_i^m, d_i^m\}$, which has the following three main steps:

(i) a greedy variable selection approach to find $\boldsymbol{\phi}_i$;
(ii) a time warping approach to find d_i^m; and
(iii) a projected gradient descent method to update basis $\mathbf{v}_m$.

We will introduce each of them separately as follows, with the detailed algorithm shown in Algorithm 1.

Data: Functional data $\mathbf{X}_i, i = 1, \ldots, N$
Result: Estimated $\mathbf{v}_m, \boldsymbol{\phi}_i, d_i^m$ for $m = 1, \ldots, M, i = 1, \ldots, N$
Initialize $\mathbf{v}_m, m = 1, \ldots, M$
while *Not converge* **do**
  for $i = 1, \ldots, N$ **do**
    Initialize $\mathcal{F} \leftarrow \emptyset$, $\mathcal{M} \leftarrow \emptyset$, $\mathbf{R}_i \leftarrow \mathbf{X}_i$
    while $\max_{m \notin \mathcal{M}, d \in \mathcal{D}} \frac{\langle \mathbf{R}_i, d(\mathbf{v}_m) \rangle}{||d(\mathbf{v}_m)||_2} \geq \epsilon$ *and* $|\mathcal{M}| < h$ **do**
      Find the next deformed basis to be added $\{m, d_i^m\} \leftarrow \arg\max_{m \notin \mathcal{M}, d \in \mathcal{D}} \frac{\langle \mathbf{R}_i, d(\mathbf{v}_m) \rangle}{||d(\mathbf{v}_m)||_2}$
      Calculate the deformed basis $\mathbf{u}_i^m = d_i^m(\mathbf{v}_m) = \mathbf{Z}_i^{m\prime} \mathbf{v}_m$
      Add the deformed basis into the selected basis pool $\mathcal{F} \leftarrow \mathcal{F} \cup \{\mathbf{u}_i^m\}$,
       $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
      Update the coefficients of the selected bases so far
       $\boldsymbol{\phi}_i \leftarrow \arg\min_{\boldsymbol{\phi}_i} ||\mathbf{X}_i - \sum_{\mathbf{u}_i^m \in \mathcal{F}} \phi_i^m \mathbf{u}_i^m||_2^2$
      Update the residual $\mathbf{R}_i \leftarrow \mathbf{X}_i - \sum_{\mathbf{u}_i^m \in \mathcal{F}} \phi_i^m \mathbf{u}_i^m$
    end
  end
  for $m = 1, \ldots, M$ **do**
    Update the m template basis:

$$\mathbf{v}_m = P[\mathbf{v}_m - \rho \sum_{i=1}^{N} \phi_i^m \mathbf{Z}_i^m (\sum_{m=1}^{M} \phi_i^m \mathbf{Z}_i^{m\prime} \mathbf{v}_m - \mathbf{X}_i)]$$

     where ρ is determined by (14).
    Normalize $\mathbf{v}_m = \mathbf{v}_m / ||\mathbf{v}_m||_2$.
  end
end

Algorithm 1: TWSNFF: Time-Warped Sparse Non-negative Functional Factorization algorithm for estimation of $\mathbf{v}_m, \boldsymbol{\phi}_i, d_i^m$ for $m = 1, \ldots, M, i = 1, \ldots, N$

4.1 Solving $\boldsymbol{\phi}_i$ via Orthogonal Matching Projection

Here we temporally assume that d_i^m and $\mathbf{v}_m$ are fixed, and only focus on the estimation of sparse vector $\boldsymbol{\phi}_i$. Note that this is the exact sparse recovery problem with l_0 regularizion, and can be solved efficiently using many standard algorithms [61]. Here we adopt the greedy variable selection with orthogonal matching projections (OMP, [60]). In particular, in each step, suppose the current set of selected bases is $\mathcal{M}$. OMP selects the next basis function to be added in $\mathcal{M}$, whose normalized best deformation can reduce the current representation residual $\mathbf{R}_i$ most, i.e.,

$$m = \arg\max_{m \notin \mathcal{M}} \frac{\langle \mathbf{R}_i, d_i^m(\mathbf{v}_m) \rangle}{||d_i^m(\mathbf{v}_m)||_2}. \tag{5}$$

Here d_i^m is the best deformation. It can be achieved as:

$$d_i^m = \arg\max_{d \in \mathcal{D}} \frac{\langle \mathbf{R}_i, d(\mathbf{v}_m) \rangle}{||d(\mathbf{v}_m)||_2}, \tag{6}$$

which will be solved in detail in the next section. After each selection step, $\mathcal{M}$ is updated as $\{\mathcal{M}, m\}$. Then OMP projects $\mathbf{X}_i$ into the space spanned by the aligned bases selected so far, to find their coefficients as $\boldsymbol{\phi}_i = \arg\min_{\boldsymbol{\phi}_i} ||\mathbf{X}_i - \sum_{m \in \mathcal{M}} \phi_i^m d_i^m(\mathbf{v}_m)||_2^2$. Then the residuals are updated as $\mathbf{R}_i = \mathbf{X}_i - \sum_{m \in \mathcal{M}} \phi_i^m d_i^m(\mathbf{v}_i^m)$ for (5) in the next step. The iteration continues until $\max_{m \notin \mathcal{M}, d \in \mathcal{D}} \frac{\langle \mathbf{R}_i, d(\mathbf{v}_m) \rangle}{||d(\mathbf{v}_m)||_2} < \epsilon$ where ϵ is a small positive constant. In this paper, we follow the stopping rule of [10] and set $\epsilon = \sigma\sqrt{2 \log M}$.

4.2 Time Warping

Denote the components of $\mathbf{R}_i$ as $R_i(t_l), l = 1, \ldots, n_i$, and the components of $\mathbf{v}_m$ as $v_m(t_k), k = 1, \ldots, n$. Estimating d_i^m in (6) is the most crucial step of the whole algorithm. The deformation here (or time warping) can be regarded as a typical mapping function that assigns each point in $\mathbf{v}_m$ to one point in $\mathbf{R}_i$ with preserved time order, and $d(\cdot)$ can be expressed by a list of non-decreasing pairs of indices (t_k, t_l) with constraints on neighboring pairs. However, unlike traditional time warping algorithms, such as dynamic time warping [56], our goal is to only manipulate $\mathbf{v}_m$ to minimize (6), while existing algorithms aim to manipulate both curves to minimize their distance $\langle \mathbf{R}_i, d(\mathbf{v}_m) \rangle$. Furthermore, their distance is not normalized by $||d(\mathbf{v}_m)||_2$, which makes results of different pairs of $(\mathbf{R}_i, d(\mathbf{v}_m))$ incomparable. Thus, we develop an alternative algorithm for quickly computing of the optimal alignment between $(\mathbf{R}_i, d(\mathbf{v}_m))$, which returns a normalized distance of (6).

To be more specific, mathematically, denote a list of binary indicator vectors $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_{n_i}\}$, where $\mathbf{z}_l \in \mathcal{R}^n, l = 1, \ldots, n_i$, with $z_{lk} \in \{0, 1\}$ and $\sum_{k=1}^n z_{lk} = 1$. Then we can represent a mapping as $d(\mathbf{v}_m) = [\mathbf{z}_1' \mathbf{v}_m, \ldots, \mathbf{z}_{n_i}' \mathbf{v}_m]$, or as a matrix multiplication $d(\mathbf{v}_m) = \mathbf{Z}' \mathbf{v}_m$ with $\mathbf{Z} = [\mathbf{z}_1, \cdots, \mathbf{z}_{n_i}] \in \mathcal{R}^{n \times n_i}$ [63]. To further guarantee that the mapping preserves the time order, additional linear constraints need to be imposed: if $v_m(t_k)$ is assigned to $X_i(t_{il})$, $v_m(t_{k'})$ is assigned to $X_i(t_{il'})$, and $l < l'$, then we require $k \leq k'$. To enforce this constraint, it suffices to consider $\{k | z_{lk} = 1\} \leq \{k' | z_{(l+1)k'} = 1\}$ for $l = 1, \ldots, n_i - 1$, where k and k' are the indices satisfying $z_{lk} = 1$ and $z_{(l+1)k'} = 1$ [1]. This constraint can be implemented as a set of linear constraints by considering the positional binary notation.

[1]It is to be noted that deformations we consider here are a bit different from commonly defined deformations in works of dynamic time warping. In particular, first, we add one constraint that no more than one point on $\mathbf{v}_m$ can be mapped to the same point on $\mathbf{R}_i$, i.e., $\sum_{k=1}^n z_{lk} = 1$, since we only want to manipulate $\mathbf{v}_m$, but not $\mathbf{R}_i$. Second, we remove the stepwise constraint that for neighboring pairs each index can change by at most one from one pair to the next. This makes the slope of the admissible warping paths more flexible, but will not introduce overfitting, since we regularize the deformation set by the Sakoe-Chiba band constraint.

Furthermore, to constrain the flexibility of time warping, we regularize the deformation set by the Sakoe-Chiba band constraint, i.e., for any warping, we force its Z to have $|k-l| \leq \max(0.2n_i, 0.2n, |n-n_i|), \forall z_{lk}=1$. Besides preventing overfitting, this constraint further reduces the number of variables in looking for the optimal Z from nn_i to $n_i(2\max(0.2n_i, 0.2n, |n-n_i|)+1)$ and hence accelerates the algorithm significantly, especially when n and n_i are very big. Consequently, we can reformulate the warping selection process as an integer programming:

$$\mathbf{Z}_i^m = \underset{\mathbf{z}_l, l=1,\ldots,n_i}{\arg\min} \frac{\left(\sum_{l=1}^{n_i} R_i(t_l)\mathbf{z}_l'\mathbf{v}_m\right)^2}{\sum_{l=1}^{n_i}(\mathbf{z}_l'\mathbf{v}_m)^2} \tag{7}$$

$$s.t. \quad z_{lk} \in \{0,1\}, \sum_{k=1}^{n} z_{lk} = 1, l = 1,\ldots,n_i, \tag{8}$$

$$\{k|z_{lk}=1\} \leq \{k'|z_{(l+1)k'}=1\}, l=1,\ldots,n_i-1, \tag{9}$$

$$|k-l| \leq \max(0.2n_i, 0.2n, |n-n_i|), \forall z_{lk}=1. \tag{10}$$

By considering the positional binary notation, the position constraint of (9) and (10) can be transferred to a set of linear constraints. Then (7) becomes a 0-1 convex programming problem, and can be solved via many standard algorithms [6], such as branch-and-bound method [7], the outer approximation method [19], etc. However, the computational complexities of these methods are usually high. One alternative way we use in our paper is to relax the integer constraint in (7). Then the objective function becomes convex and can be solved by standard solvers for convex constrained problem with small computational complexity. In our paper we adopt the Frank-Wolfe's algorithm [28]. One thing to be noted is that with the integer relaxation, after solving the real values of $\mathbf{Z}_i^m$ for (7), it is not necessary to re-extract the integer solutions for $\mathbf{Z}_i^m$ anymore. Instead, we can directly use these real values of $\mathbf{Z}_i^m$ to get the optimal $d_i^m(\mathbf{v}_m) = \mathbf{Z}_i^{m\prime}\mathbf{v}_m$ for Algorithm 1, since in other estimation steps we only need $d_i^m(\mathbf{v}_m)$ instead of the alignment path information. Though in this way $d_i^m(\mathbf{v}_m)$ is no longer a time deformation, such relaxation is effective from the computational perspective. Furthermore, since OMP only requires a limited number of calls to (6) in solving the sparse regression problem, the joint estimation of $\boldsymbol{\phi}_i$ and d_i^m would be computationally efficient.

4.3 Solving $\mathbf{v}_m$ via Projected Gradient Descent

After achieving $\boldsymbol{\phi}_i$ and d_i^m for all the samples $i=1,\ldots,N$ based on algorithms in Section 4.1 and Section 4.2, by directly using $\mathbf{Z}_i^{m\prime}\mathbf{v}_m$ to represent d_i^m, we can reformulate the objective function of solving $\mathbf{v}_m$ as

$$\underset{\mathbf{v}_m>0}{\arg\min} f(\mathbf{v}_m) := \frac{1}{2}\sum_{i=1}^{N} ||\mathbf{X}_i - \sum_{m=1}^{M} \phi_i^m \mathbf{Z}_i^{m\prime}\mathbf{v}_m||_2^2. \tag{11}$$

This is part of standard NMF estimation problem, and many algorithms have been extensively studied in the literature, such as the Newton-Type approach [15], the reduced quadratic model approach [34], and the gradient descent methods [35, 57]. Among existing bound-constrained optimization techniques, projected gradient descent (PGD) algorithm is a simple and effective one. It also has a merit that can be applied to online learning case by adopting its conjugate, i.e., stochastic projected gradient descent, which will be detailed in Section 4.4. As such, here we use

PGD to update $\mathbf{v}_m$. Specifically, we first use PGD to update $\mathbf{v}_m$ by

$$\mathbf{v}_m = P[\mathbf{v}_m - \rho \nabla f(\mathbf{v}_m)] \tag{12}$$

$$= P[\mathbf{v}_m - \rho \sum_{i=1}^{N} \phi_i^m \mathbf{Z}_i^m (\sum_{m=1}^{M} \phi_i^m \mathbf{Z}_i^{m\prime} \mathbf{v}_m - \mathbf{X}_i)], \tag{13}$$

where $\nabla(\cdot)$ is the gradient of the function with respect to $\mathbf{v}_m$ evaluated on the current $\mathbf{v}_m$ and ϕ_i^m, and $P[\mathbf{v}] \in \mathcal{R}^n$ is the element-wise gradient projection onto the non-negative constraint set with

$$P[v] = \begin{cases} v, \text{ if } v \geq 0 \\ 0, \text{ otherwise} \end{cases}.$$

Finally, we normalize $\mathbf{v}_m = \mathbf{v}_m / ||\mathbf{v}_m||_2$.

The step size parameter $\rho > 0$ in (13) is set according to the "Armijo rule along the projection arc" in [5]. In particular, given two positive constants $0 < \beta < 1, 0 < \xi < 1$, and any initial feasible $\mathbf{v}_m^0$. Then for iteration $k = 1, 2, \ldots$,

$$\mathbf{v}_m^{k+1} = P[\mathbf{v}_m^k - \rho_k \nabla f(\mathbf{v}_m^k)],$$

where $\rho_k = \beta^{t_k}$, and t_k is the first non-negative integer t for which

$$f(\mathbf{v}_m^{k+1}) - f(\mathbf{v}_m^k) \leq \xi \nabla f(\mathbf{v}_m^k)'(\mathbf{v}_m^{k+1} - \mathbf{v}_m^k). \tag{14}$$

See [35] for detailed discussion. Of course, some more advanced variants of the PGD method can be applied in our case as well, such as the Nesterov optimal gradient method. In general, a more complicated update rule will achieve faster convergence rate yet with more computation complexity. In reality, practitioners can choose their most suitable PGD method case by case. Besides, other methods, such as the multipliactive method or the alternative least square method, are also possible candidates.

Proposition 1. *Algorithm 1 finally converges to a stationary point of (4).*

Proof. Proposition 1 can be guaranteed since in each iteration the loss function of (4) decreases monotonously, and it has a lower bound [54]. In particular, in every greedy selection step, we first update d_i^m. By the integer relaxation, (7) becomes a strictly convex function. So solving $\mathbf{Z}_i^m$ optimally decreases the conditional objective function. As mentioned previously, since we directly use real values of $\mathbf{Z}_i^m$ to calculate $d_i^m(\mathbf{v}_m)$, this monotone decrease (optimal) property will keep in solving $\boldsymbol{\phi}_i$ and $\mathbf{v}_m$. Second, we update ϕ_i^m, whose conditional loss function given $d_i^m(\mathbf{v}_m) = \mathbf{Z}_i^{m\prime}\mathbf{v}_m$ decreases as the algorithm iterates, according to the property of the OMP algorithm [60]. Finally, we update $\mathbf{v}_m$, whose conditional loss given ϕ_i^m and $\mathbf{Z}_i^m$ also decreases as the algorithm iterates, guaranteed by the property of the PGD algorithm [4]. □

4.4 Online Estimation Algorithm

For some applications, functional samples $\mathbf{X}_i, i = 1, \ldots$ may come in a sequential way. For such case, it is more desirable to develop an incremental learning algorithm that can learn $\mathbf{v}_m$ from functional data streams in an online manner, which is presented as below.

In particular, for the estimation of $\boldsymbol{\phi}_i$ and d_i^m, since the minimum values of (5) and (6) are only related to the current sample $\mathbf{X}_i$, their estimation for each online sample sequentially will be the same as the batch algorithm. However, as to the update of $\mathbf{v}_m$, for the batch version, in every iteration, we update $\mathbf{v}_m$ based on all the N samples, as shown in (13), whereas for the online version, this is infeasible since not all the N samples have been observed so far. Instead, we can update $\mathbf{v}_m$

Data: Functional data streams $\mathbf{X}_i, i = 1, \ldots$
Result: Estimated $\mathbf{v}_m, \boldsymbol{\phi}_i, d_i^m$ for $m = 1, \ldots, M, i = 1, \ldots$
Initialize $\mathbf{v}_m, m = 1, \ldots, M$
for *online samples* $\mathbf{X}_i, i = 1, \ldots$ **do**
  Initialize $\mathcal{F} \leftarrow \emptyset$, $\mathcal{M} \leftarrow \emptyset$, $\mathbf{R}_i \leftarrow \mathbf{X}_i$
  while $\max_{m \notin \mathcal{M}, d \in \mathcal{D}} \frac{\langle \mathbf{R}_i, d(\mathbf{v}_m) \rangle}{||d(\mathbf{v}_m)||_2} \geq \epsilon$ *and* $|\mathcal{M}| < h$ **do**
    Find the next deformed basis to be added $\{m, d_i^m\} \leftarrow \arg\max_{m \notin \mathcal{M}, d \in \mathcal{D}} \frac{\langle \mathbf{R}_i, d(\mathbf{v}_m) \rangle}{||d(\mathbf{v}_m)||_2}$
    Calculate the deformed basis $\mathbf{u}_i^m = d_i^m(\mathbf{v}_m) = \mathbf{Z}_i^{m\prime}\mathbf{v}_m$
    Add the deformed basis into the selected basis pool $\mathcal{F} \leftarrow \mathcal{F} \cup \{\mathbf{u}_i^m\}, \mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
    Update the coefficients of the selected bases so far
    $\boldsymbol{\phi}_i \leftarrow \arg\min_{\boldsymbol{\phi}_i} ||\mathbf{X}_i - \sum_{\mathbf{u}_i^m \in \mathcal{F}} \phi_i^m \mathbf{u}_i^m||_2^2$
    Update the residual $\mathbf{R}_i \leftarrow \mathbf{X}_i - \sum_{\mathbf{u}_i^m \in \mathcal{F}} \phi_i^m \mathbf{u}_i^m$
  end
  for $m = 1, \ldots, M$ **do**
    Update the basis function

$$\tilde{\mathbf{v}}_m \leftarrow \mathbf{v}_m - r\phi_i^m \mathbf{Z}_i^m \Big(\sum_{m=1}^{M} \phi_i^m \mathbf{Z}_i^{m\prime} \mathbf{v}_m - \mathbf{X}_i\Big)$$

    Normalize $\mathbf{v}_m = \tilde{\mathbf{v}}_m / ||\tilde{\mathbf{v}}_m||_2$.
  end
end

Algorithm 2: OTWSNFF: online algorithm for estimation of $\mathbf{v}_m, \boldsymbol{\phi}_i, d_i^m$ for $m = 1, \ldots, M, i = 1, \ldots$

according to the stochastic projected gradient descent algorithm, which has been employed for online NMF in many previous works [22], i.e.,

$$\tilde{\mathbf{v}}_m \leftarrow \mathbf{v}_m - r\phi_i^m \mathbf{Z}_i^m \Big(\sum_{m=1}^{M} \phi_i^m \mathbf{Z}_i^{m\prime} \mathbf{v}_m - \mathbf{X}_i\Big), \tag{15}$$

$$\mathbf{v}_m = \tilde{\mathbf{v}}_m / ||\tilde{\mathbf{v}}_m||_2, \tag{16}$$

where r is the updating step size. Here we use the classic step size as $r = 1/(\theta i)$ where θ is the largest eigenvalue of the Hessian matrix $\nabla^2 f(\mathbf{v}_m)$. According to [23], this can guarantee a convergence rate of $\mathbf{v}_m$ at $O(i^{-1})$. Of course, more complicated update rules can be used. Some practical guidance on choosing the step size in several variants of the stochastic PGD method is given in [53].

The schematic online estimation algorithm is summarized as below:

- Initialize the basis functions $\mathbf{v}_m, m = 1, \ldots, M$.
- Repeat until running out samples. For each online sample i,
  - Compute its optimal $(\phi_i^m, d_i^m), m = 1, \ldots, M$ based on the methods in Section 4.1 and Section 4.2.
  - Update $\mathbf{v}_m, m = 1, \ldots, M$ by (15) and (16).

The detailed algorithm is shown in Algorithm 2. Subsequently, we denote the proposed batch time-warped sparse non-negative functional factorization as TWSNFF, and the online version as OTWSNFF. It is to be noted that when sequential samples come in mini batch, it is also flexible to adapt Algorithm 2 with slight modification to mini-batch gradient descent flavor, which will be demonstrated in our real case studies.

4.5 Computational Complexity

For Algorithm 1, in each inner iteration of estimating $\boldsymbol{\phi}_i$ and d_i^m, evaluating (7) for each base function $\mathbf{v}_m, m = 1, \ldots, M$ via the Frank-Wolfe's algorithm will take $O(Kn_ig_i)$ computational complexity, where K is number of steps in the Frank-Wolfe's algorithm and $g_i = \max(0.2n_i, 0.2n, |n - n_i|)$. Hence the complexity for solving $\{m, d_i^m\}$ would be at most $O(|\mathcal{M}^c|Kn_ig_i)$ where $\mathcal{M}^c$ is the complementary set of $\mathcal{M}$. Then for every selected $\mathbf{v}_m$, the complexity of solving ϕ_i by the linear least square method would be $O(|\mathcal{M}|n_i^2)$. Since at most all the M base functions have non-zero weights on $\mathbf{X}_i$, the total complexity for analyzing $\mathbf{X}_i$ in every iteration of Algorithm 1 would be $O(M(M-1)Kn_ig_i/2 + M(M-1)n_i^2/2)$. Assume for all the samples, n_i is in the same magnitude of n and $g_i = O(n)$. Then for updating $\mathbf{v}_m$ using PGD, the computational complexity is $O(Nn^2)$. As, such, the total computational complexity for one iteration of Algorithm 1 is $O(NKM^2n^2)$. Similarly, we can calculate the computational complexity of Algorithm 2 as $O(KM^2n^2)$ for each online update step.

5 EXPERIMENTS

To examine the effectiveness of TWSNFF and OTWSNFF, we conduct extensive experiments, including experiments on synthetic data and case studies on real functional data set. For intensive evaluation, we compare the proposed TWSNFF and OTWSNFF algorithms with the following baselines:

NMF: the non-negative matrix factorization for time series [38].
SMFPCA: the sparse multi-channel functional PCA [59].
SC: the sparse coding for functional data [29].
TWPCA: the time-warped PCA [3].
In addition, to allow us in further examining the efficacy of exploiting the time warping approach, we also implement a simplified version of our method without time warping as
SNFF: A variant of the proposed batch sparse non-negative functional factorization without time warping. To the best of our knowledge, no existing work has attempted this baseline for functional data analysis.

5.1 Experimental Results on Synthetic Data

We begin with synthetic data experiments to investigate how TWSNFF and OTWSNFF perform on data generated from the assumed deformation model as described in previous section. For each experiment replication, first, we generate three template basis functions from the B-spline basis functions. In particular, we set $M = 3$, $n = 60$ and $N = 500$. We generate $\mathbf{v}_m, m = 1, \ldots, M$ from certain basis space. Then we consider the set of deformation operators $\mathcal{D}$ as all possible time warpings with maximum window of length six. We generate $d_i^m(\mathbf{v}_m)$ by randomly selecting an operator $d_i^m \in \mathcal{D}$. Then we generate ϕ_i^m from a threshold distribution, i.e., $\phi_i^m = b_i^m I(b_i^m \geq \rho)$ where b_i^m is randomly generated from the standard half-normal distribution. $I(\cdot)$ is the indicator function with $I(\cdot) = 1$ if the condition is satisfied and $I(\cdot) = 0$ otherwise. ρ is used to enforce small ϕ_i^m to be 0 and hence regularizes the sparsity of $\boldsymbol{\phi}_i = [\phi_i^1, \ldots, \phi_i^M]$. Finally we generate $\mathbf{X}_i = \sum_{m=1}^{M} \phi_i^m d_i^m(\mathbf{v}_m) + \mathbf{e}_i$ where $e_i(t_l) \sim N(0, \sigma^2)$ is random noise at time point t_l for $\mathbf{X}_i$. In the simulation we set $\sigma = 0.1$. In particular, we consider two models for $\mathbf{v}_m$. (I) Spline space: we set $\mathbf{v}_m$ to be the 1, 3 and 7 B-spline basis functions of order 3 with a grid of 60 equally spaced knots in $[0, 1]$. $t_l, l = 1, \ldots, n$ are at the same locations as the knots. (II) Fourier space: we set $\mathbf{v}_m$ as $v_m(t_k) = cos(mt_k + m\pi), m = 1, \ldots, M$, with a grid of $n = 60$ equally spaced time points $t_1, \ldots, t_n$ in $[0, 2\pi]$.

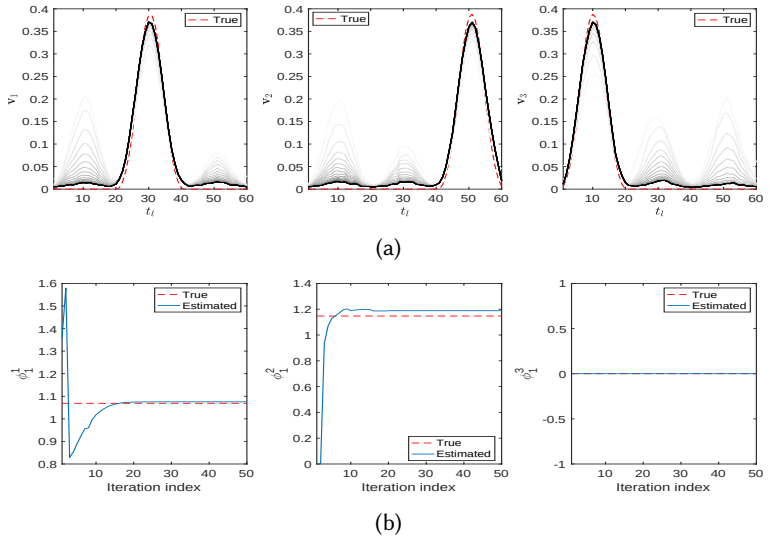


Fig. 4. (a) Estimated basis functions $\mathbf{v}_m, m = 1, \ldots, 3$ by TWSNFF based on the first $i = 1, \ldots, 50$ iterations; The red dot lines are true basis functions; b) The corresponding estimated $\phi_1^m, m = 1, \ldots, 3$ in different iteration steps by Algorithm 1.

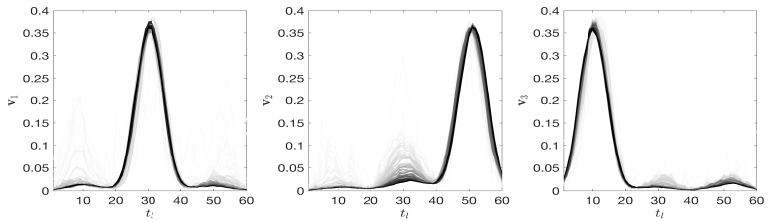


Fig. 5. Estimated basis functions by OTWSNFF using the first $i = 1, \ldots, N$ samples. Functions estimated by later iterations or more samples are drawn in darker colors.

For all the methods, $\mathbf{v}_m$ are initialized by randomly sampling from a standard half-normal distribution. For TWSNFF, NMF, SMFPCA, SC, TWPCA and SNFF, we assume they can access the total N samples since the beginning and use them all for model estimation according to their nature of batch learning. In particular, for all the sparse models of TWSNFF, OTWSNFF, SMFPCA, SC and SNFF, we assume the number of bases is known in advance as $M = 3$. Then for SMFPCA, we set its sparsity tuning parameter ρ according to the Bayesian information criterion (BIC) rule and do the model estimation according to the procedures in [59]. For SC, we set its sparsity parameter by performing the grid-search of [29]. For the online method OTWSNFF, we deliberately assume samples come sequentially and only the first sample is available at the beginning.

First, we illustrate the convergence properties of the proposed two algorithms. In particular, for a certain experiment replication of TWSNFF with Algorithm 1, we draw its updated $\mathbf{v}_m$ for Model (I) in each iteration until convergence in Figure 4a with gradually varied gray lines, where functions estimated by later iterations are drawn in darker colors. Clearly, $\mathbf{v}_m$ can converge to the true basis functions fast within few iterations. Furthermore, Figure 4b shows the estimated $\boldsymbol{\phi}_i$

for sample $i = 1$ on the three B-spline basis functions in the first 50 iterations. The true values of $\boldsymbol{\phi}_1$ are plotted by red dash lines. It shows that the estimated $\boldsymbol{\phi}_i$ can converge very fast to a value close to the true value. These demonstrate the efficiency of Algorithm 1. Similarly, for a certain experiment replication of OTWSNFF with Algorithm 2 in Model (I), we draw its updated $\mathbf{v}_m$ as i increases in Figure 5 with gradually varied gray lines, where functions estimated with larger i are drawn in darker colors. It shows as sample index goes, $\mathbf{v}_m$ can also converge to the true basis functions, though the convergence rate is a bit lower than Algorithm 1.

To evaluate the description performance of TWSNFF and OTWSNFF, Figure 6 compares the fitting results for $\mathbf{X}_{500}$ by TWSNFF, OTWSNFF and other baselines for one experiment replication of Model (I). Clearly, both TWSNFF and OTWSNFF can accurately describe the true signals of $\mathbf{X}_{500}$, while the other methods fail in different ways. In particular, NMF, SMFPCA, SC, and SNFF have deformed features (i.e., temporally shifted peaks) since they fail to consider feature deformation. The features of TWPCA have no peak shifts. However, because its time warping is directly on functional signals instead of features, its extracted features are not very accurate, i.e., flatter than the true ones. In addition, NMF and TWPCA "over-estimate" an additional peak around time point 30. This is because they do not consider feature sparsity and consequently their extracted bases mix the three features (peaks at time points 10, 30 and 50) together. Similar phenomenons also exist for Model (II).

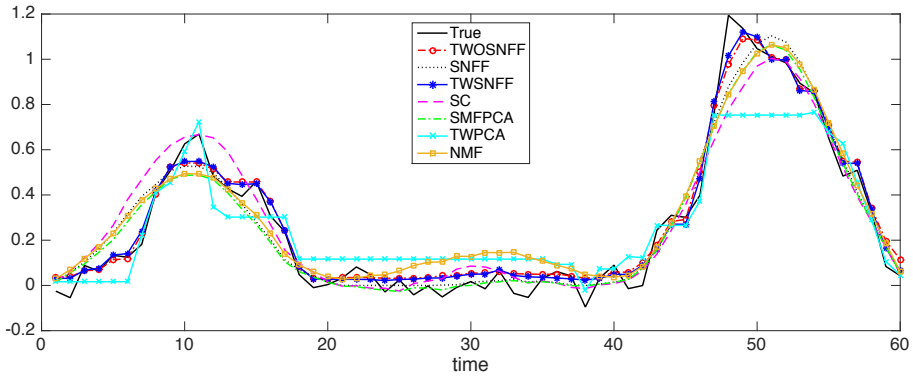
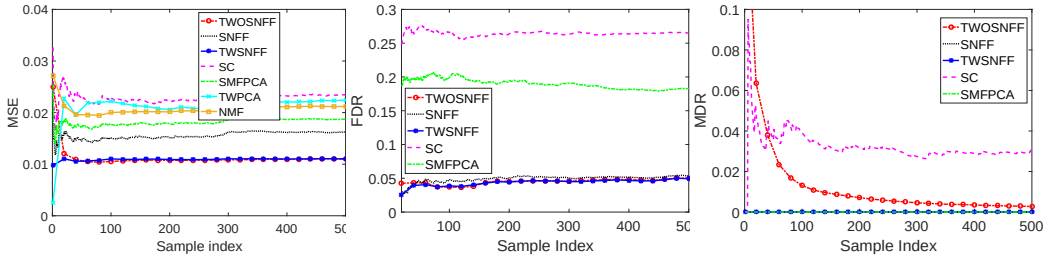


Fig. 6. Description of $\mathbf{X}_{500}$ by different methods, with the estimation MSEs equal to 0.0039 (OTWSNFF), 0.0080(SNFF), 0.0031(TWSNFF), 0.0118(SC), 0.0083(SMFPCA), 0.148(TWPCA), 0.0114(NMF).

We further test their description power in terms of estimation mean square error (MSE) calculated by the first i accumulated samples as i increases for Models (I) and (II). As Figures 7a and 8a show, TWSNFF and OTWSNFF have much smaller MSEs than the other methods. Furthermore, as more samples are received, the MSE of OTWSNFF decreases quickly and finally converges to nearly the same value as TWSNFF, demonstrating the convergence and scalability of the online algorithm.

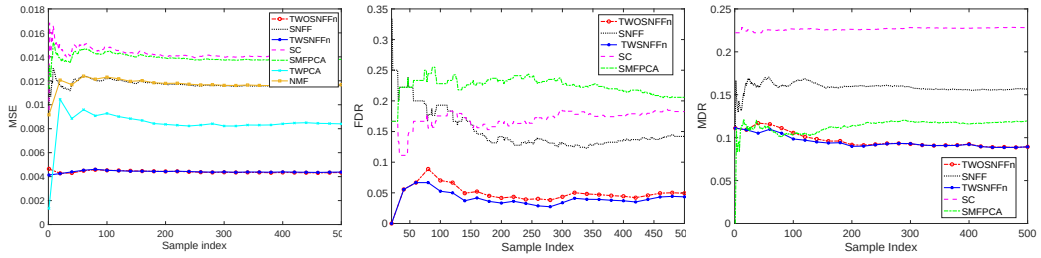
For the models exploiting feature sparsity (i.e., TWSNFF, OTWSNFF, SMFPCA, SC and SNFF), we further test whether their estimated sparse $\boldsymbol{\phi}_i$ can identify bases with non-zero coefficients. In particular, suppose the non-zero support of the true $\boldsymbol{\phi}_i$ as $\mathcal{S}(\boldsymbol{\phi}_i)$ and its complement set as $\mathcal{S}^c(\boldsymbol{\phi}_i)$, and define the corresponding estimated ones by a method as $\hat{\mathcal{S}}(\boldsymbol{\phi}_i)$ and $\hat{\mathcal{S}}^c(\boldsymbol{\phi}_i)$. Then the false detection rate (FDR) and the missed detection rate (MDR) of a method are defined as

$$\text{FDR} = \frac{\sum_i \hat{\mathcal{S}}(\boldsymbol{\phi}_i) \cap \mathcal{S}^c(\boldsymbol{\phi}_i)}{\sum_i \mathcal{S}^c(\boldsymbol{\phi}_i)}, \quad \text{MDR} = \frac{\sum_i \hat{\mathcal{S}}^c(\boldsymbol{\phi}_i) \cap \mathcal{S}(\boldsymbol{\phi}_i)}{\sum_i \mathcal{S}(\boldsymbol{\phi}_i)}.$$



(a) Mean Squared Error (MSE) (b) False Detection Rate (FDR) (c) Missed Detection Rate (MDR)

Fig. 7. Model (I): Performance comparison for the synthetic data for one experiment replication with noise standard deviation $\sigma = 0.1$.



(a) MSE (b) FDR (c) MDR

Fig. 8. Model (II): Performance comparison for the synthetic data for one experiment replication with noise standard deviation $\sigma = 0.1$.

Figures 7b and 7c, and Figures 8b and 8c show the FDRs and MDRs of different algorithms calculated by the first i accumulated samples (as i increases), for Models (I) and (II) respectively. In particular, TWSNFF and OTWSNFF have the smallest FDRs, followed by SMFPCA, while the other two methods have unsatisfactorily large FDRs. As to MDR, except SC, the other three batch methods have satisfactory MDRs. The MDR of OTWSNFF also converges to that of TWSNFF as i increases.

Finally, to have a complete understanding of TWSNFF and OTWSNFF, we evaluate their performance sensitivities to the noise standard deviation σ. We consider different levels of $\sigma = 0.01, 0.025, 0.05, 0.1, 0.15, 0.2$. For each magnitude of σ, we conduct the above experiments for 50 replications. In each replication, we generate $N = 500$ functional data in the same way as above and do estimation based on different algorithms. We record the estimation results in terms of MSEs, FDRs and MDRs of different algorithms, and calculate the mean and standard deviation of these results from the 50 replications in Tables 1-3 for Model (I) and Tables 4-6 for Model (II) respectively. It is clear to see that as σ increases, MSEs and MDRs of all the methods increase, and in the meanwhile the FDRs of all the methods decrease. This is intuitive since larger noise blurs the signals and makes the features of the basis functions less identifiable. Consequently MDRs and MSEs increase, and on the opposite, FDRs decrease. Furthermore, among all the methods, TWSNFF has the smallest MSEs, MDRs, and FDRs all the way, followed by OTWSNFF (though its MDRs are a bit larger than the other batch methods due to its online property for Model (I)), demonstrating the superiority of the proposed methodology. Compared with them, SNFF and SMFPCA have larger MSEs due to their lack of consideration of deformation, but SNFF significantly outperforms

SMFPCA in terms of FDR. This can be attributed to the non-negative constraint of SNFF, which leads to its estimated basis functions more accurate and the identification easier. NFF performs slightly worse than SNFF, due to its failure of consideration of sparsity. SC and TWPCA perform the worst.

Table 1. Model (I): MSEs and their standard deviations (in the parentheses) under different noise magnitudes (10^{-3}).

Noise std	TWSNFF	OTWSNFF	SNFF	NMF	SMFPCA	TWPCA	SC
0.01	1.42(0.09)	3.15(0.15)	7.46(0.32)	9.62(0.38)	9.47(0.31)	16.1(3.48)	15.4(0.80)
0.025	1.90(0.07)	4.20(0.20)	8.88(0.34)	10.5(0.39)	10.1(0.36)	18.0(3.73)	16.5(0.80)
0.05	3.85(0.13)	5.10(0.30)	10.8(0.45)	12.1(0.42)	11.8(0.41)	19.2(4.69)	18.6(1.30)
0.10	11.3(0.13)	12.6(0.34)	16.0(0.49)	20.4(0.44)	19.3(0.43)	22.4(5.02)	24.3(1.40)
0.15	23.5(0.27)	24.9(0.52)	30.8(0.51)	31.4(0.44)	31.8(0.47)	40.1(8.17)	39.6(2.20)
0.2	43.4(0.48)	43.5(0.59)	47.9(0.53)	48.9(0.54)	47.2(0.52)	49.5(10.2)	57.2(2.70)

Table 2. Model (I): FDRs and their standard deviations (in the parentheses) under different noise magnitudes (10^{-2}).

Noise std	TWSNFF	OTWSNFF	SNFF	SMFPCA	SC
0.01	7.17(0.11)	7.41(0.14)	7.95(0.16)	25.4(0.41)	30.9(0.48)
0.025	6.76(0.10)	6.68(0.11)	7.65(0.09)	22.4(0.29)	30.4(0.42)
0.05	6.15(0.05)	6.46(0.08)	6.75(0.09)	20.7(0.12)	28.2(0.42)
0.10	4.75(0.05)	5.34(0.08)	5.34(0.09)	18.5(0.33)	29.7(0.36)
0.15	4.44(0.05)	4.78(0.06)	4.78(0.09)	17.8(0.16)	28.8(0.31)
0.20	4.09(0.03)	4.64(0.05)	4.65(0.08)	17.4(0.24)	20.5(0.25)

Table 3. Model (I): MDRs and their standard deviations (in the parentheses) under different noise magnitudes(10^{-2}).

Noise std	TWSNFF	OTWSNFF	SNFF	SMFPCA	SC
0.01	0(0)	0.02(0.01)	0(0)	0(0)	1.45(0.24)
0.025	0(0)	0.02(0.01)	0(0)	0(0)	2.70(0.27)
0.05	0(0)	0.02(0.01)	0(0)	0(0)	3.98(0.36)
0.10	0.02(0.01)	0.25(0.05)	0.02(0.01)	0.03(0.02)	4.20(0.44)
0.15	0.14(0.06)	0.67(0.17)	0.19(0.03)	0.15(0.06)	4.26(0.45)
0.20	2.54(0.16)	2.98(0.44)	2.55(0.14)	2.15(0.13)	5.46(0.43)

5.2 Case Studies on Real-world Data

5.2.1 Boston Subway. The first data set is provided by the Boston's Massachusetts Bay Transit Authority[2] and contains per-minute passenger entry count at each station of the Boston's subway

[2]https://www.mbta.com/developers

Table 4. Model (II): MSEs and their standard deviations (in the parentheses) under different noise magnitudes (10^{-4}).

Noise std	TWSNFF	OTWSNFF	SNFF	NMF	SMFPCA	TWPCA	SC
0.01	1.58(0.02)	1.86(0.03)	6.56(0.06)	7.11(0.08)	13.5(0.06)	9.09(0.68)	78.4(0.70)
0.025	2.04(0.02)	2.44(0.05)	8.23(0.08)	9.42(0.16)	15.1(0.06)	9.84(0.49)	79.2(0.56)
0.05	3.98(0.03)	4.19(0.04)	14.1(0.07)	16.2(0.27)	21.1(0.07)	13.1(0.43)	87.5(0.45)
0.10	14.0(0.07)	14.2(0.11)	37.8(0.09)	43.3(0.37)	44.8(0.08)	25.0(0.35)	113(0.62)
0.15	34.2(0.14)	35.8(0.18)	77.3(0.11)	85.8(0.46)	84.3(0.11)	43.7(0.60)	154(0.57)
0.20	65.9(1.21)	67.5(0.19)	133(0.15)	146(0.46)	145(0.15)	72.3(0.77)	210(0.48)

Table 5. Model (II): FDRs and their standard deviations (in the parentheses) under different noise magnitudes (10^{-2}).

Noise std	TWSNFF	OTWSNFF	SNFF	SMFPCA	SC
0.01	13.6(0.67)	13.8(0.67)	15.5(0.54)	21.4(0.21)	26.7(0.60)
0.025	11.0(0.63)	12.3(0.63)	14.6(0.43)	21.5(0.20)	19.0(1.00)
0.05	9.13(0.63)	13.5(0.48)	14.2(0.45)	21.3(0.25)	21.1(0.79)
0.10	5.45(0.39)	11.2(0.42)	13.4(0.38)	21.8(0.22)	15.6(0.88)
0.15	4.15(0.36)	10.9(0.49)	13.4(0.37)	22.0(0.25)	13.1(0.73)
0.20	6.03(0.45)	9.66(0.42)	12.8(0.34)	19.1(0.22)	12.4(0.75)

Table 6. Model (II): MDRs and their standard deviations (in the parentheses) under different noise magnitudes(10^{-2}).

Noise std	TWSNFF	OTWSNFF	SNFF	SMFPCA	SC
0.01	3.33(0.13)	3.23(0.14)	3.43(0.10)	13.1(0.05)	7.19(0.11)
0.025	3.92(0.11)	4.15(0.21)	4.67(0.22)	13.1(0.05)	16.9(0.19)
0.05	5.31(0.14)	5.69(0.15)	6.46(0.39)	13.6(0.05)	18.2(0.14)
0.10	9.65(0.14)	10.6(0.30)	11.1(0.53)	13.9(0.05)	22.2(0.08)
0.15	11.5(0.15)	12.1(0.25)	13.5(0.49)	14.0(0.05)	22.2(0.06)
0.20	13.0(0.14)	13.8(0.18)	17.7(0.44)	14.7(0.04)	22.2(0.06)

system measured at the turnstiles used for payment, for totally 30 days in 2014. 63 stations of the Red, Orange, Blue, and Green lines are included. Every day the subway system starts at 5AM and continues through 1AM on the next day (denoted as $1AM^+$). We treat the entry rate of each station in each day as a function, and in total we have $63 \times 30 = 1890$ functions. We apply our proposed models in these entry rate curves and extract features (such as morning peaks, evening peaks, etc.), to learn and interpret passenger flow patterns.

From data preprocessing, we can test that for each station, the entry count per minute follows a non-homogeneous Poisson process. Therefore we first use the classic nonparametric approach based on smoothing kernels [46] to estimate the entry rate function for each station every day, i.e.,

$$\lambda_{ij}(t_s) = \frac{1}{Th} \sum_{l=1}^{n_{ij}} N_{ij}(t_l) \mathcal{K}(\frac{t_l - t_s}{h}), 5\text{AM} \le t_s < 1\text{AM}^+,$$

where $N_{ij}(t_l)$ is the passenger entry count in the minute $t_l, l = 1, \ldots, n_{ij}$, and n_{ij} is the total number of observed minutes for station j in day i; $t_s, s = 1, \ldots, n$ are n equally spaced time points in $[5AM, 1AM^+)$. $\mathcal{K}$ is the kernel smoothing function and h is the kernel width. The entry rate functions of eight selected stations with $n = 240$ are shown in Figure 1 for illustration.

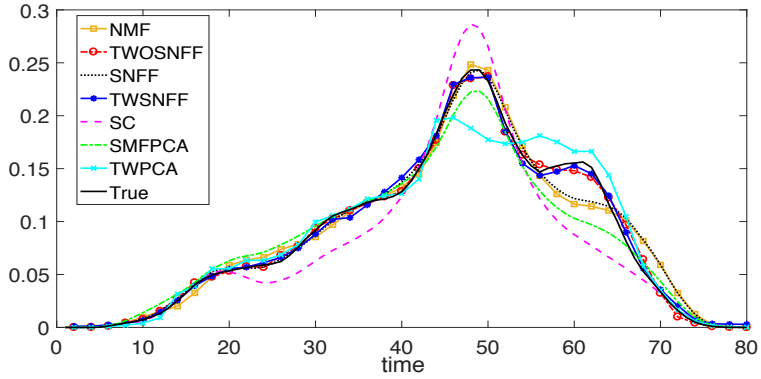


Fig. 9. Description of a real functional sample, i.e., passenger entry rates of a subway station in one day of the Boston subway data set. The MSE results ($\times 10^{-5}$) on this sample are 2.295 (TWSNFF), 13.67(SNFF), 2.310(OTWSNFF), 46.30(SC), 27.79(SMFPCA), 28.72(TWPCA), and 17.13(NMF).

Table 7. Comparison of overall MSE results by different models for the four real case studies.

Data set	TWSNFF	OTWNFF	SNFF	NMF	SMFPCA	TWPCA	SC
Boston($\times 10^{-4}$)	0.482	0.741	1.07	1.66	2.53	2.81	5.05
Hong Kong($\times 10^{-4}$)	0.414	0.515	0.752	0.89	1.06	3.64	1.45
NYC Bike($\times 10^{-1}$)	0.50	0.53	1.52	1.37	1.39	2.68	2.45
Electricity($\times 10^{-1}$)	0.30	0.31	1.41	1.25	1.17	1.50	3.21

We apply the two proposed models and the baselines to analyze the passenger entry rate functions. For the six batch methods, their model settings follow the same procedure as Section 5. For OTWSNFF, a more natural way is to treat the passenger entry rate functions of all the stations per day as a mini batch with in total 30 mini batches and use the mini-batch stochastic gradient descent in Algorithm 2 for estimation. Their description results for one randomly selected station in one particular day is shown in Figure 9. Specifically, TWSNFF and OTWSNFF describe the data best, because their time-warped bases can track the station's deformed passenger patterns in that specific day accurately. Table 7 further gives a summary of overall MSEs of different models, which are calculated based on the whole data set of all stations in all days. The results are generally consistent with the above inference and further demonstrate the superiority of the proposed methods. Finally, Figure 10 plots the $M = 6$ extracted basis functions by TWSNFF. These features are easy to interpret, which in order represent the weekend bells with either evening or morning skewness (first column), the morning peak and the normal flat travel pattern in the day (second column), the night peak and the evening peak (third column).

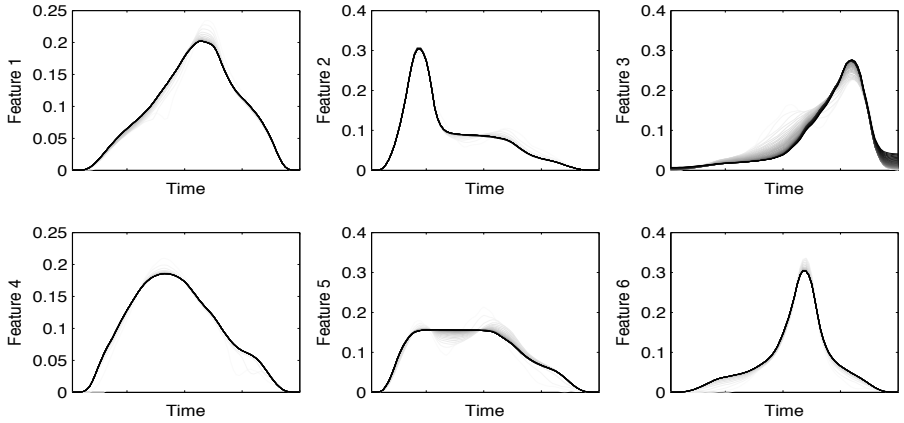
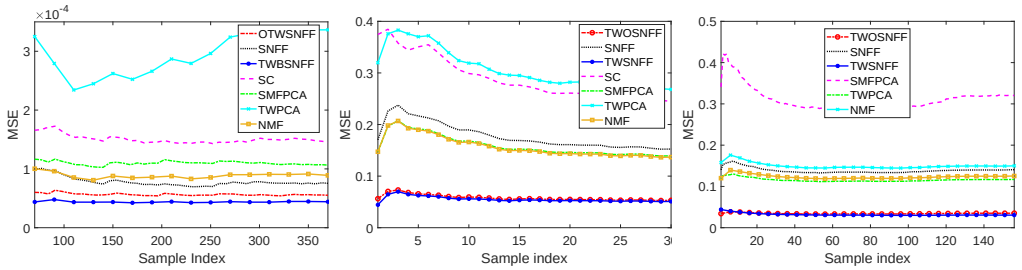


Fig. 10. Examples of extracted bases of the Boston subway data set by the proposed TWSNFF method.

5.2.2 Hong Kong MTR.. Similar as the first data set, the second data set contains daily passenger entry rates of 88 subway stations for seven weekdays in the Hong Kong MTR system, with a total of $88 \times 7 = 616$ functional samples. The data are provided by the Hong Kong MTR Corporation Limited. Due to confidential reason, we cannot disclose the data set to the public. The data have similar characteristics as the Boston subway data, so we preprocess the data in the similar way as Section 5.2.1. Then we apply the two proposed models together with the baselines for feature extraction and modeling. Their model settings follow the same settings as the previous experiments, except OTWSNFF. For it, unlike Section 5.2.1, since seven mini batches cannot guarantee convergence, here we directly use Algorithm 2 and treat each station's entry rate function in each day as an individual on-line sample. Their estimation MSEs calculated by the first i accumulated samples (i increases from 50 to 616) are shown in Figure 11a, and the MSEs calculated by all the 616 samples are shown in Table 7. Similar to the Boston subway analysis, our two models have the best performance in general.



(a) Hong Kong MTR (b) NYC bike (c) Electricity

Fig. 11. Model comparison in terms of MSE.

5.2.3 NYC Bike Trip Data Set. The data set[3] records all trip data in the CitiBike share program in New York City every day. In each trip, a user picks up (unlocks) a sharing bike from a certain station, rides it to somewhere else and returns it to another station. Because the daily data in each

[3]https://www.citibikenyc.com/system-data

station are quite sparse, we divide New York City into 17×13 grids, and aggregate trip data in stations of the same region together. In this way, each trip data contain the information of the regions and time stamps of a bike being picked up and returned. We treat the real-time bike pickup rate of each region in each day as a function, and apply our proposed models for analysis. The same preprocessing procedure is implemented as before with $n = 24$ for data of in total 30 days in November 2017. The model settings are the same as the previous experiments. For OTWSNFF, we also consider treating the bike pickup rate functions of all the regions per day as a mini batch of size $B = 221$ with in total 30 mini batches, and use the mini-batch stochastic gradient descent in Algorithm 2 for estimation. Their estimation MSEs calculated by the first i accumulated batches (i increases from 1 to 30) are shown in Figure 11b, and the MSEs calculated by all the 6330 functional samples are shown in Table 7. In general, our two models have the best performance.

5.2.4 Electricity Consumption Data Set. The data set[4] includes the electricity consumption of each 15 minutes of 370 Portuguese clients from 2011 to 2014. For each client, all days present 96 measures (time stamps). Yet among them there are a lot of zero values. Therefore, we first average the consumption data at the same time stamp over seven days of one week, and get one averaged daily consumption curve in each week for each client, with in total $N = 156$ weeks in the three years. For each curve, we further apply the similar preprocessing procedure as before and set $n = 24$. Then we implement our proposed algorithms for the overall 370×156 functions for analysis. The model settings are the same as the previous experiments. In particular, for OTWSNFF, we also consider treating the average daily consumption functions of all the clients in each week as a mini batch of size $B = 370$ with in total 156 mini batches, and use the mini-batch stochastic gradient descent in Algorithm 2 for estimation. Their estimation MSEs calculated by the first i accumulated batches (i increases from 1 to 156) are shown in Figure 11c, and the MSEs calculated by all the 6330 samples are shown in Table 7. In general, our two models have the best performance.

5.3 Computation Complexity

Last, we would like to show the computation time of TWSNFF and OTWSNFF for the synthetic data and four case studies in Table 8. The results of TWSNFF are the average computation time for one estimation iteration in Algorithm 1. The results of OTWSNFF are the average computation time for one online sample in Algorithm 2. All the simulations are run on a laptop using a single Intel Core i5 processor. In all the simulations, we set the number of steps in the Frank-Wolfe's algorithm as $K = 50$. For OTWSNFF, we can see that in most scenarios with mini batch of $B = 1$, the computation time for one online sample is very small. When B increases, the computation time increases, but still less than one hour. For TWSNFF, its computation time for one iteration is moderate on a minute scale. Though for the largest scale experiment of Section 5.2.4, the computation time is about five hours, it is still tolerable considering the algorithm's fast convergence rate. Besides, the computation time can be further reduced with the help of parallel computing. This can be easily implemented, since the estimation of $\boldsymbol{\phi}_i$ for each sample is independent and can be run on different processors simultaneously.

6 CONCLUSIONS

This paper proposes a novel time-warped sparse non-negative functional factorization method for functional data analysis. Compared with the existing state-of-the-art methods, our method on one hand guarantees the extracted bases together with their coefficients to be positive and interpretable, and on the other allows functions to be weakly correlated with different features. Furthermore, our method can handle feature deformation of different functions by incorporating time warping into

[4]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014#

Table 8. Computation time of TWSNFF for one iteration and OTWSNFF for one online sample in numerical studies.

Model	Synthetic data		Real case studies			
	Model(I)	Model (II)	Boston	HK	NYC	Electricity
	$M = 3, N = 500,$ $n = 60, B = 1$		$M = 6, N = 1890,$ $n = 240, B = 63$	$M = 6, N = 616,$ $n = 240, B = 1$	$M = 4, N = 6330,$ $n = 24, B = 221$	$M = 6, N = 57720,$ $n = 24, B = 370$
TWSNFF	53(0.12)		970(34.2)	540(27.8)	2237(86.3)	17829(154.3)
OTWSNFF	0.10(0.08)		65.4(4.54)	1.18(0.02)	78(2.51)	87.1(1.24)

factorization. We present encouraging results from both experiments on synthetic data and case studies with real-world functional data analysis.

REFERENCES

[1] Mohammad Taha Bahadori, David Kale, Yingying Fan, and Yan Liu. 2015. Functional subspace clustering with application to time series. In *International Conference on Machine Learning*. 228–237.

[2] A Bartlett and WP McCormick. 2013. Estimation for non-negative time series with heavy-tail innovations. *Journal of Time Series Analysis* 34, 1 (2013), 96–115.

[3] Poole Ben, Alex H. Williams, Niru Maheswaranathan, Byron Yu, Gopal Santhanam, Stephen I. Ryu, Stephen A. Baccus, Krishna Shenoy, and Surya Ganguli. 2017. Time-warped PCA: simultaneous alignment and dimensionality reduction of neural data. In *Computational and Systems Neuroscience*.

[4] Dimitri Bertsekas. 1976. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on automatic control* 21, 2 (1976), 174–184.

[5] Dimitri P Bertsekas. 1999. *Nonlinear programming*. Athena scientific Belmont.

[6] Pierre Bonami, Mustafa Kilinç, and Jeff Linderoth. 2012. Algorithms and software for convex mixed integer nonlinear programs. In *Mixed integer nonlinear programming*. Springer, 1–39.

[7] Brian Borchers and John E Mitchell. 1994. An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers & Operations Research* 21, 4 (1994), 359–367.

[8] Philippe Bougerol and Nico Picard. 1992. Stationarity of GARCH processes and of some nonnegative time series. *Journal of econometrics* 52, 1-2 (1992), 115–127.

[9] Anne-Sarah Briand, Etienne Côme, K Mohamed, and Latifa Oukhellou. 2015. A mixture model clustering approach for temporal passenger pattern characterization in public transport. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*. IEEE, 1–10.

[10] T Tony Cai and Lie Wang. 2011. Orthogonal matching pursuit for sparse signal recovery with noise. *IEEE Transactions on Information theory* 57, 7 (2011), 4680–4688.

[11] Zhe Chen and Andrzej Cichocki. 2005. Nonnegative matrix factorization with temporal smoothness and/or spatial decorrelation constraints. *Laboratory for Advanced Brain Signal Processing, RIKEN, Tech. Rep* 68 (2005).

[12] Vincent CK Cheung, Karthik Devarajan, Giacomo Severini, Andrea Turolla, and Paolo Bonato. 2015. Decomposing time series data by a non-negative matrix factorization algorithm with temporally constrained coefficients. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*. IEEE, 3496–3499.

[13] Jeng-Min Chiou, Yu-Ting Chen, and Ya-Fang Yang. 2014. Multivariate functional principal component analysis: A normalization approach. *Statistica Sinica* (2014), 1571–1596.

[14] Jeng-Min Chiou and Hans-Georg Müller. 2016. A pairwise interaction model for multivariate functional and longitudinal data. *Biometrika* 103, 2 (2016), 377.

[15] Moody Chu, Fasma Diele, Robert Plemmons, and Stefania Ragni. 2004. Optimality, computation, and interpretation of nonnegative matrix factorizations. In *SIAM Journal on Matrix Analysis*. Citeseer.

[16] Marco Cuturi and Mathieu Blondel. 2017. Soft-DTW: a differentiable loss function for time-series. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 894–903.

[17] Ruairí de Fréin, Konstantinos Drakakis, Scott Rickard, and Andrzej Cichocki. 2008. Analysis of financial data using non-negative matrix factorization. In *International Mathematical Forum*, Vol. 3. Journals of Hikari Ltd, 1853–1870.

[18] Chong-Zhi Di, Ciprian M Crainiceanu, Brian S Caffo, and Naresh M Punjabi. 2009. Multilevel functional principal component analysis. *The annals of applied statistics* 3, 1 (2009), 458.

[19] Marco A Duran and Ignacio E Grossmann. 1986. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical programming* 36, 3 (1986), 307–339.

[20] Julian Eggert and Edgar Korner. 2004. Sparse coding and NMF. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, Vol. 4. IEEE, 2529–2533.
[21] Theo Gasser and Alois Kneip. 1995. Searching for structure in curve samples. *Journal of the american statistical association* 90, 432 (1995), 1179–1188.
[22] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. 2012. Online nonnegative matrix factorization with robust stochastic approximation. *IEEE Transactions on Neural Networks and Learning Systems* 23, 7 (2012), 1087–1099.
[23] Elad Hazan, Amit Agarwal, and Satyen Kale. 2007. Logarithmic regret algorithms for online convex optimization. *Machine Learning* 69, 2-3 (2007), 169–192.
[24] Yongmiao Hong and Yoon-Jin Lee. 2011. Detecting misspecifications in autoregressive conditional duration models and non-negative time-series processes. *Journal of Time Series Analysis* 32, 1 (2011), 1–32.
[25] Patrik O Hoyer. 2004. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research* 5, Nov (2004), 1457–1469.
[26] Jianhua Z Huang, Haipeng Shen, Andreas Buja, et al. 2008. Functional principal components analysis via penalized rank one approximation. *Electronic Journal of Statistics* 2 (2008), 678–695.
[27] Fumitada Itakura. 1975. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 23, 1 (1975), 67–72.
[28] Martin Jaggi. 2013. Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization.. In *ICML (1)*. 427–435.
[29] Seongah Jeong, Xiang Li, Jiarui Yang, Quanzheng Li, and Vahid Tarokh. 2017. Dictionary Learning and Sparse Coding-based Denoising for High-Resolution Task Functional Connectivity MRI Analysis. In *International Workshop on Machine Learning in Medical Imaging*. Springer, 45–52.
[30] Eamonn Keogh and Chotirat Ann Ratanamahatana. 2005. Exact indexing of dynamic time warping. *Knowledge and information systems* 7, 3 (2005), 358–386.
[31] A Kneip, Xiaochun Li, KB MacGibbon, and JO Ramsay. 2000. Curve registration by local regression. *Canadian Journal of Statistics* 28, 1 (2000), 19–29.
[32] Cosmin Lazar, Daniel Nuzillard, and Andrei Doncescu. 2009. Non Negative Matrix Factorization for Time Series of Medical Images Analysis. In *Complex, Intelligent and Software Intensive Systems, 2009. CISIS'09. International Conference on*. IEEE, 918–923.
[33] Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 6755 (1999), 788.
[34] Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*. 556–562.
[35] Chih-Jen Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural computation* 19, 10 (2007), 2756–2779.
[36] Jennifer Listgarten, Radford M Neal, Sam T Roweis, and Andrew Emili. 2005. Multiple alignment of continuous time series. In *Advances in neural information processing systems*. 817–824.
[37] Jiali Mei, Yohann De Castro, Yannig Goude, Jean-Marc Azais, and Georges Hébrail. 2018. Nonnegative matrix factorization with side information for time series recovery and prediction. *IEEE Transactions on Knowledge and Data Engineering* 31, 3 (2018), 493–506.
[38] Jiali Mei, Yohann De Castro, Yannig Goude, and Georges Hébrail. 2017. Nonnegative Matrix Factorization for Time Series Recovery From a Few Temporal Aggregates. In *International Conference on Machine Learning*. 2382–2390.
[39] Yang Meng, Ronghua Shang, Licheng Jiao, Wenya Zhang, and Shuyuan Yang. 2018. Dual-graph regularized non-negative matrix factorization with sparse and orthogonal constraints. *Engineering Applications of Artificial Intelligence* 69 (2018), 24–35.
[40] Yang Meng, Ronghua Shang, Licheng Jiao, Wenya Zhang, Yijing Yuan, and Shuyuan Yang. 2018. Feature selection based dual-graph sparse non-negative matrix factorization for local discriminative clustering. *Neurocomputing* 290 (2018), 87–99.
[41] Yang Meng, Ronghua Shang, Fanhua Shang, Licheng Jiao, Shuyuan Yang, and Rustam Stolkin. 2019. Semi-supervised graph regularized deep NMF with bi-orthogonal constraints for data representation. *IEEE transactions on neural networks and learning systems* (2019).
[42] Morten Morup, Kristoffer Hougaard Madsen, and Lars Kai Hansen. 2008. Approximate L 0 constrained non-negative matrix and tensor factorization. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*. IEEE, 1328–1331.
[43] Kanchan Mukherjee, ROBERT H Shumway, and KENNETH L Verosub. 2007. On the alignment of multiple time series fragments. *Biometrika* 94, 2 (2007), 347–358.
[44] Jiazhu Pan and Qiwei Yao. 2008. Modelling multiple time series via common factors. *Biometrika* (2008), 365–379.
[45] Kamran Paynabar, Changliang Zou, and Peihua Qiu. 2016. A Change-Point Approach for Phase-I Analysis in Multivariate Profile Monitoring and Diagnosis. *Technometrics* 58, 2 (2016), 191–204.

[46] Henrik Ramlau-Hansen. 1983. Smoothing counting process intensities by means of kernel functions. *The Annals of Statistics* (1983), 453–466.
[47] James Ramsay. 2005. Functional data analysis. *Encyclopedia of Statistics in Behavioral Science* (2005).
[48] Tomasz M Rutkowski, Rafal Zdunek, and Andrzej Cichocki. 2007. Multichannel EEG brain activity pattern analysis in time–frequency domain with nonnegative matrix factorization support. In *International Congress Series*, Vol. 1301. Elsevier, 266–269.
[49] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing* 26, 1 (1978), 43–49.
[50] Ronghua Shang, Yang Meng, Wenbing Wang, Fanhua Shang, and Licheng Jiao. 2019. Local discriminative based sparse subspace learning for feature selection. *Pattern Recognition* 92 (2019), 219–230.
[51] Ronghua Shang, Wenbing Wang, Rustam Stolkin, and Licheng Jiao. 2017. Non-negative spectral learning and sparse regression-based dual-graph regularized feature selection. *IEEE transactions on cybernetics* 48, 2 (2017), 793–806.
[52] Paris Smaragdis, Cedric Fevotte, Gautham J Mysore, Nasser Mohammadiha, and Matthew Hoffman. 2014. Static and dynamic source separation using nonnegative factorizations: A unified view. *IEEE Signal Processing Magazine* 31, 3 (2014), 66–75.
[53] James C Spall. 2005. *Introduction to stochastic search and optimization: estimation, simulation, and control*. Vol. 65. John Wiley & Sons.
[54] Paul Tseng. 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications* 109, 3 (2001), 475–494.
[55] Laura L Tupper, David S Matteson, and John C Handley. 2016. Mixed data and classification of transit stops. In *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2225–2232.
[56] Taras K Vintsyuk. 1968. Speech discrimination by dynamic programming. *Cybernetics* 4, 1 (1968), 52–57.
[57] Fei Wang, Chenhao Tan, Ping Li, and Arnd Christian König. 2011. Efficient document clustering via online nonnegative matrix factorizations. In *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM, 908–919.
[58] Jane-Ling Wang, Jeng-Min Chiou, and Hans-Georg Muller. 2016. Functional Data Analysis. *Annual Review of Statistics and Its Application* 3, 1 (2016), 257–295.
[59] Chen Zhang, Hao Yan, Seungho Lee, and Jianjun Shi. 2018. Weakly correlated profile monitoring based on sparse multi-channel functional principal component analysis. *IISE Transactions* 50, 10 (2018), 878–891.
[60] Tong Zhang. 2011. Sparse recovery with orthogonal matching pursuit under RIP. *IEEE Transactions on Information Theory* 57, 9 (2011), 6215–6221.
[61] Zheng Zhang, Yong Xu, Jian Yang, Xuelong Li, and David Zhang. 2015. A survey of sparse representation: algorithms and applications. *IEEE access* 3 (2015), 490–530.
[62] Feng Zhou and Fernando De la Torre. 2015. Generalized canonical time warping. *IEEE transactions on pattern analysis and machine intelligence* 38, 2 (2015), 279–294.
[63] Feng Zhou and Fernando Torre. 2009. Canonical time warping for alignment of human behavior. In *Advances in neural information processing systems*. 2286–2294.