Learning Low-Dimensional Temporal Representations with Latent Alignments

Bing Su, and Ying Wu, *Fellow, IEEE*

Abstract—Low-dimensional discriminative representations enhance machine learning methods in both performance and complexity. This has motivated supervised dimensionality reduction (DR), which transforms high-dimensional data into a discriminative subspace. Most DR methods require data to be i.i.d. However, in some domains, data naturally appear in sequences, where the observations are temporally correlated. We propose a DR method, namely, latent temporal linear discriminant analysis (LT-LDA), to learn low-dimensional temporal representations. We construct the separability among sequence classes by lifting the holistic temporal structures, which are established based on temporal alignments and may change in different subspaces. We jointly learn the subspace and the associated latent alignments by optimizing an objective that favors easily separable temporal structures. We show that this objective is connected to the inference of alignments and thus allows for an iterative solution. We provide both theoretical insight and empirical evaluations on several real-world sequence datasets to show the applicability of our method.

Index Terms—Dimensionality reduction, latent alignment, temporal sequences, discriminant analysis.

1 Introduction

Multivariate temporal sequences arise in a wide range of applications, where the pattern of interest is represented as a sequence of local feature vectors. The features are generally extracted at local temporal frames or spatial regions in an unsupervised way. They often lie in a high-dimensional space and contain noises or useless information for the overall sequence classification. Learning a discriminative subspace for the features in sequences to better reveal the global temporal structures of sequences is desirable. In such a subspace, both the learning and inference of the subsequent temporal modeling methods would not only be faster and require much smaller memory demand since the dimensionality of the features is reduced but also be more accurate and robust because noises are removed and fewer parameters are needed to be estimated.

Various supervised *dimensionality reduction (DR)* methods have been developed for vector data under the i.i.d. assumption, but they cannot be directly applied to the temporal features in sequences that are not independent. DR for sequence data aims at learning a subspace by maximizing the separability among sequence classes, where the separability is embodied in the differences in temporal structures. The temporal structures reflect the common evolutions of all sequences from the same class, and they depend on temporal alignments to establish correspondences among sequences with local temporal differences. The separability and objective are more difficult to formulate and manipulate inherently. For these reasons, DR for sequence data has received limited attention.

- *B. Su is with the Science & Technology on Integrated Information System Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China, 100190. E-mail: subingats@gmail.com.*
- *Y. Wu is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, 60208. E-mail: yingwu@eecs.northwestern.edu.*

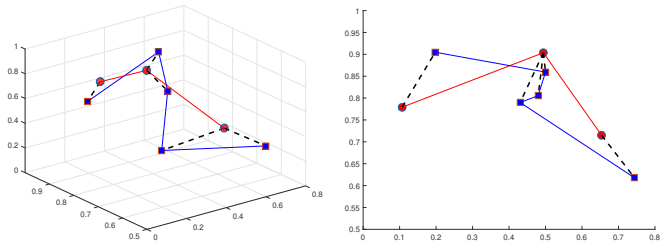
Manuscript received April 19, 2005; revised August 26, 2015.

Existing methods such as *linear sequence discriminant analysis (LSDA)* [1], [2] and *max-min intersequence distance analysis (MMSDA)* [3] construct the separability based on generative models. For each class, they train a left-to-right *hidden Markov model (HMM)* [4] from the original sequences. The mean of the features aligned to each hidden state is calculated, and the means of all ordered states form a mean sequence. The interclass distance is measured as the *dynamic time warping (DTW)* [5] distance between the mean sequences. Such separability depends on the alignments between the sequences and the hidden states, which further rely on the similarities of the features. When projecting the features to a subspace, the local similarities among the transformed features may change, and hence, the alignments may change accordingly. An example is shown in Fig. 1, where the alignments in the subspace change from those in the original space. The alignments in an underlying subspace are called the latent alignments. The latent alignments cannot be inferred before the subspace is determined.

On the other hand, the projection is determined by maximizing the separability, where the separability should be constructed based on the alignments in the subspace. Therefore, learning the projection and inferring the alignments are entangled. To make the projection tractable, existing methods simply fix the alignments in the underlying subspace to those in the original space. However, the resulting separability cannot reflect the real confusion relationship between classes in the subspace. In addition, HMM-based separability requires a large number of sequences for training.

In this paper, we propose a supervised DR method for sequence data called *latent temporal linear discriminant analysis (LT-LDA)*. We learn an abstract template for each class to discover the temporal structures by employing the modified DTW barycenter [6], [7]. We then construct the separability among sequence classes based on the alignments between the abstract templates and the training sequences.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2919303, IEEE Transactions on Pattern Analysis and Machine Intelligence



(a) (b)

Fig. 1. (a) The alignment (the dashed line) of the sequence of 5 points (blue) to the sequence of 3 points (red) in the three-dimensional space; (b) If all points are projected onto the x-y plane, the alignment between the two sequences changes in this two-dimensional subspace.

Although determining the alignments by learning the abstract templates and learning the subspace by maximizing the constructed separability still rely on each other, we show that their objectives are connected, which allows us to jointly learn the most discriminative subspace together with the associated latent alignments, resulting in sequences of low-dimensional discriminative temporal representations.

The main contributions are as follows. (1) Different from the HMM-based separability, our new construction of separability does not require an excessive amount of training data. It can be performed even when only one training sequence per class is available. (2) Different from previous methods where the subspace can only be learned through prefixing the alignments, we propose to learn the subspace and the latent alignments simultaneously and develop an efficient iterative solution. The learned subspace is thereby holistically optimal. (3) We establish a connection between our objective formulation and the abstract template learning, which ensures the convergence of our solution. We further provide theoretical insight on the subspace selection. This paper is an extension of the conference paper [8].

2 Related Work

In this section, we briefly review several related topics: temporal segmentation, latent variable models for sequences, discriminant analysis, discriminative clustering, dimensionality reduction for sequences, and deep learning for sequence data.

Temporal segmentation. Segmenting sequences into different stages is a common strategy for modeling temporal structures. In [9], three approaches based on PCA, probabilistic PCA, and Gaussian mixture model were proposed to segment long motion sequences into distinct high-level behaviors. In [10], kernelized temporal cut incorporated Hilbert space embedding of distributions to change-point detection for segmenting sequences into actions and cyclic motions. In [11], a video sequence was divided into multiple segments by temporal subspace clustering. In [12], a neighborhood graph-based method was proposed to partition a sequence into distinct activities and motion primitives by utilizing self-similar structures in the sequence. In [7], [13], an alignment-based temporal clustering method was proposed to parse a sequence into different stable stages. These methods are unsupervised and work on a single sequence. Our method captures the common temporal structures in all training sequences for each class. Some temporal clustering methods such as [14], [15] partition a sequence into disjoint segments with the constraint that each segment belongs to one of a set of clusters. The orders of clusters that appear in different sequences may be different. A single sequence may not contain all clusters. In contrast, for each class, the temporal clusters learned by our method appear in all the training sequences in the same order.

Latent variable models for sequences. Many models for sequence data involve latent variables. In [16], shape, appearance, and motion states were modeled as latent variables within a Bayesian framework for tracking 3D human figures. In [17], binary latent variables were modeled in an undirected model for human motion data. In [18], the Gaussian process dynamical model for motion modeling was composed of a latent space with associated dynamics. In [19], latent states that include body orientation, activity category, and body pose were jointly estimated for tracking. In [20], efficient optimization algorithms were developed for learning the Gaussian process latent variable model to track multiple activities. In [21], beta-process autoregressive HMM discovered a latent set of behaviors for motion capture segmentation. In this paper, we model the alignments as latent structures for discriminating different classes.

Discriminant analysis. Various supervised linear DR methods have been proposed for numerical data. Linear discriminant analysis (LDA) [22] optimizing the Fisher criterion is perhaps the most widely used method for its simplicity, effectiveness and well-established theory and is attracting consistent interest [23], [24], [25] in machine learning. Many other approaches are dedicated to solve the problems of LDA in some particular cases or improve LDA. Uncorrelated LDA (ULDA) [26] extracts features that are statistically uncorrelated. Generalized ULDA [27] and orthogonal LDA [28] are proposed to solve the undersampled problems when the scatter matrices are singular.

LDA assumes that every class obeys a Gaussian distribution with the same covariance matrix and, hence, is incapable of tackling heteroscedastic data properly. Heteroscedastic LDA (HLDA) [29] optimizes the Chernoff criterion that takes the differences of covariances into account when calculating the between-class scatter. Subclass discriminant analysis [30] clusters each class into a set of subclasses such that each subclass obeys a Gaussian distribution. Based on the graph embedding framework, marginal Fisher analysis [31] constructs the between-class and within-class scatters by a penalty graph that only connects marginal points.

The Fisher criterion maximizes the average of all pairwise between-class distances. As a result, it tends to only preserve the large pairwise distances, but nearby classes may be overlapped after projection. To deal with this problem, weighted pairwise Fisher criteria [32] gives larger weights to close class pairs when calculating the between-class scatter. Max-min distance analysis (MMDA) [33] directly maximizes the minimum pairwise distance in the subspace. Worst-case LDA [34] further redefines the within-class scatter as the maximum average within-class distances among all classes. Heteroscedastic MMDA [35], [36] employs the Chernoff distance as the pairwise between-class distance.

These advances for numerical data cannot directly propagate to structured sequence data because the feature vec-

tors in sequences violate the basic i.i.d. assumption. Our method performs DR for sequence data by lifting the inherent temporal dependencies.

Discriminative clustering. Some methods such as [23], [37], [38] combine LDA and k-means to adaptively learn a discriminative subspace in which vector data can be better clustered. The data have no class labels, and the purpose of dimensionality reduction is to better group data into clusters. There are no temporal relations among clusters. Differently, in our method, dimensionality reduction is applied to temporally correlated vectors in sequences. Data are labeled at the sequence level, and the goal is to better separate different sequence classes. There are temporal constraints when learning the temporal structures for each class.

Dimensionality reduction for sequences. In [39], [40], [41], linear and nonlinear transformations were learned for each sequence pair to perform multimodal alignment. The transformations for different sequence pairs are different. In our method, the projection is for discriminating different classes and stays the same for all sequences from all classes. In [42], a sufficient DR approach was proposed for sequence labeling by building sequence kernels. The labels are associated with the vectors in sequences rather than the whole sequences, and the task is to predict a class label for each vector in the sequences. In [43], the features were transformed by unidimensional convolutions of all dimensions for sequence labeling. Our method focuses on linear projection, and the task is to predict a label for each entire sequence. In [44], a Mahalanobis distance was learned, given the ground truth alignments of training samples, to perform multivariate sequence alignment, while in our method, the alignments of both the training sequences and the test sequences are unavailable. In [45], generalized rank pooling (GRP) encoded a sequence into a subspace representation. GRP is a pooling method, and the subspaces are different for different sequences. Our method is a DR method that learns a common subspace for all sequences.

LSDA [1], [2] and MMSDA [3] are targeted at the same problem as this paper, where the projection was learned by maximizing the separability defined on HMM-based temporal structures. For each class, a left-to-right HMM was trained, and the hidden states of the HMM were considered temporal structures. The alignments of the sequences to the hidden states in the original space and the underlying subspace were assumed to be the same. LSDA optimized the Fisher criterion and made approximations on the interclass scatter to make the optimization tractable; MMSDA optimized the max-min distance criterion, resulting in solving a series of time-consuming semidefinite programming problems, and could not scale to high dimension. Differently, in our method, the discovery of temporal structures is DTW-based and only depends on deterministic operations, which avoids the estimation of massive parameters of HMM. The latent alignments in the subspace can be jointly learned with the projection owing to our construction of separability.

Deep learning for sequence data. Recurrent neural networks (RNNs) [46] can also be used to perform supervised dimensionality reduction for sequences. However, learning a large number of parameters for RNNs such as long short-term memories (LSTMs) [47] requires massive training sequences. Learning low-dimensional features directly from raw frames is often infeasible when only limited training sequences are available. In practice, RNNs are often used as classifiers [46] or for matching [47] by taking handcrafted or CNN-learned frame-wide features as input. For 3D skeletal action sequences, a bidirectional LSTM was directly applied to the sparse coding representations in Kendall's shape space in [48], and temporal sliding LSTM networks were performed on the salient motion features in [49]. For video action sequences, deep 3D CNN architectures such as C3D [50], two-stream I3D [51], and extended versions of ResNet [52] have been shown to be effective. Generally, clips generated from a video were input to 3D CNN models, and the class scores of the clips were averaged for classification. The 3D CNN models can also be used to extract features from the clips, and each video is represented by a sequence of deep features using the sliding window manner. The sequences can be projected first by our proposed method and then input to RNNs or other sequence classifiers for classification. In this way, the input sequences are more discriminative, RNNs need to learn fewer parameters, and the classification complexity is reduced.

3 Latent Temporal Linear Discriminant Analysis

3.1 Learning Abstract Templates

We learn an abstract template $\mathbf{M}$ consisting of ordered temporal structures for each sequence class from all its training sequence samples. Each sequence $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T]$ consists of a series of ordered frame-wide feature vectors, where $\mathbf{x}_t$ is the feature vector extracted from the t-th frame, and T is the length (i.e., the number of vectors) of the sequence. For a specific sequence class, we denote its training sequence sample set by $\{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_N\}$, where N is the number of training sequences in the set, and T_n is the length of $\mathbf{X}_n$. Different sequence samples may have different lengths.

We define the abstract template as a sequence of the abstracted temporal structures $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_L]$, where the element $\mathbf{m}_j$ captures the average frame-wide features of a temporal structure or stage that each sequence must go through. Hence, $\mathbf{M}$ can be considered an atomic sequence. L is the length of $\mathbf{M}$, which is generally shorter than any sequence sample because the learned template only contains the essential temporal structures, and each structure will last several frames in a sequence.

An analogous interpretation of the abstract template is as follows: the temporal structures represent the statistics of frames corresponding to key stages to perform the sequence. Taking action sequence as an example as shown in Fig. 2, where each action video is represented by a sequence of frame-wide features, the basic representative characteristic of an action is the evolution of key poses, and the abstract template is slightly similar to the sequence of key poses. The difference is that the elements of the abstract template are abstracted from the frame-wide features rather than directly from the frames. Each element in $\mathbf{M}$ captures the common characteristics within the corresponding stage in all sequence samples from the same class, and hence, $\mathbf{M}$ captures the shared common evolution of key stages and can be viewed as the temporal template of the class.

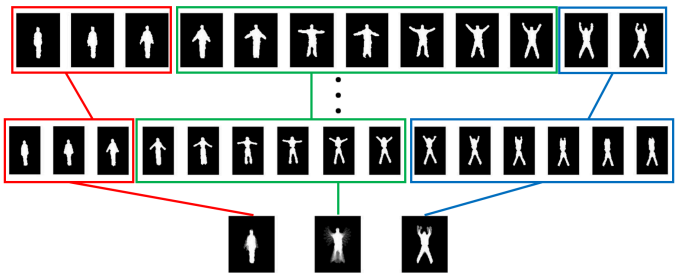


Fig. 2. An illustrative example of the abstract template $\mathbf{M}$ for the "jack" action. The samples are from the WEIZMANN dataset [53]. The alignment between a sequence and $\mathbf{M}$ (at the bottom) actually parses the sequence into different segments. The frames in the segments from different sequences aligned to the same element of $\mathbf{M}$ are bounded in the same color. The element is the mean of all these frames and can be viewed as a temporal structure or a key pose. $\mathbf{M}$ contains the ordered key poses and reflects the basic semantics of the action.

The abstract template $\mathbf{M}$ can be used to divide a sequence sample $\mathbf{X}$ into different temporal regions. This is achieved by aligning $\mathbf{X}$ to $\mathbf{M}$ with a warping function, which maps the elements of $\mathbf{X}$ to the elements of $\mathbf{M}$. The warping map can be defined by a warping path $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_L]$. $\mathbf{p}_t = [s_t, e_t]^T$ means that the $\{s_t, s_t+1, \cdots, e_t\}$-th elements in $\mathbf{X}$ are aligned to the t-th element of $\mathbf{M}$.

Similar to DTW, several constraints are applied to $\mathbf{P}$. (1) The boundary condition $s_1 = 1, e_L = T$: $\mathbf{P}$ should cover the whole sequence $\mathbf{X}$; (2) The monotonicity constraint $e_t < s_{t+1}, \forall t = 1, \cdots, L-1$: if an element $\mathbf{x}_i$ in $\mathbf{X}$ is aligned to $\mathbf{m}_j$ in $\mathbf{M}$, then the elements in $\mathbf{X}$ after $\mathbf{x}_i$ can only be aligned to $\mathbf{m}_j$ or elements after $\mathbf{m}_j$ in $\mathbf{M}$, and no elements in $\mathbf{X}$ can be aligned to more than one element in $\mathbf{M}$. As a result, the sequence $\mathbf{X}$ is divided into L nonoverlapping temporal regions. (3) The continuity constraint $e_t \geq s_t$: for any element in $\mathbf{M}$, there must be at least one element in $\mathbf{X}$ aligned to it; hence, no region divided from $\mathbf{X}$ is empty. (4) The warping constraint $l_t \leq a\frac{T}{L}$: $l_t = e_t - s_t + 1$ is the number of elements in $\mathbf{X}$ that are aligned to the t-th element in $\mathbf{M}$. $a \geq 1$ is a factor that controls the allowed degree of warping, and $\frac{T}{L}$ is the average number of elements in $\mathbf{X}$ aligned to one element in $\mathbf{M}$. This constraint means that the number of elements in $\mathbf{X}$ aligned to any element in $\mathbf{M}$ should not exceed a multiple of the average number. It prevents extremely unbalanced partitioning. With such a constraint, only salient temporal structures that are universal in all training sequences can be captured by the abstract template.

We employ the modified dynamic time warping algorithm [7], [13] to compute the optimal warping path. The sum of all the pairwise Euclidean distances between the matched elements along the warping path is used as the cost score of the path. We denote the cost of a partial path of aligning the first i elements in $\mathbf{X}$ to the first j elements in $\mathbf{M}$ as $c(i, j, l)$, where l means that the last l elements of the first i elements in $\mathbf{X}$ are aligned to the j-th element in $\mathbf{M}$ in the partial path. This partial cost can be determined recurrently:

$$c(i,j,l) = \begin{cases} d(i,j), l = 1, i = j = 1 \\ d(i,j) + \min\limits_{k=1}^{aT/L} c(i-1, j-1, k), l = 1 \\ d(i,j) + c(i-1, j, l-1), l \leq a\frac{T}{L} \\ Inf, otherwise \end{cases} \quad (1)$$

where $d(i, j)$ is the Euclidean distance between the i-th element of $\mathbf{X}$ and the j-th element of $\mathbf{M}$. The minimum alignment cost can be found by this dynamic programming and is achieved at the end of recursion as $\min\limits_{l=1}^{aT/L} c(T, L, l)$. The corresponding optimal warping path is obtained by back tracking.

Based on the dynamic alignment Eq. (1), $\mathbf{M}$ can be obtained by employing the *DTW barycenter averaging (DBA)* [6] as follows. We first use the uniform alignments to initialize $\mathbf{M}$. Specifically, in the n-th training sequence $\mathbf{X}_n$, $l_j^n = \frac{T_n}{L}$ elements in $\mathbf{X}_n$ are aligned to the j-th element of $\mathbf{M}$, $\forall j = 1, \cdots, L$. The initial j-th element $\mathbf{m}_j$ of $\mathbf{M}$ can be computed as follows:

$$\mathbf{m}_j = \frac{1}{\sum\limits_{n=1}^{N} l_j^n} \sum_{n=1}^{N} \sum_{k=s_j^n}^{e_j^n} \mathbf{x}_k^n, \quad (2)$$

where $\mathbf{P}^n = [\mathbf{p}_1^n, \cdots, \mathbf{p}_L^n]$ is the alignment path that aligns $\mathbf{X}_n$ to $\mathbf{M}$, and $\mathbf{p}_j^n = [s_j^n, e_j^n]^T$ records the start and end indexes of elements in $\mathbf{X}_n$ that are aligned to $\mathbf{m}_j$. We then align each training sequence $\mathbf{X}_n$ to the initial $\mathbf{M}$ using Eq. (1) to update the alignment path $\mathbf{P}^n$, for $n = 1, \cdots, N$. We finally recompute the elements in $\mathbf{M}$ using Eq. (2) with the updated $\mathbf{P}^n$ again. This process can be repeated until the difference of $\mathbf{M}$ in the current iteration and $\mathbf{M}$ in the previous iteration is below a threshold or a maximum number of iterations is reached. We summarize the abstract template learning algorithm in Alg. 1.

Algorithm 1 Abstract template learning

Input: $\{\mathbf{X}_1, \cdots, \mathbf{X}_N\}$; L; a;
Output: $\mathbf{M}$; $\mathbf{P}^n, n = 1, \cdots, N$;

1: Initialize the uniform alignment path $\mathbf{P}^n$ for the training sequence $\mathbf{X}_n$, for $n = 1, \cdots, N$;
2: Compute the initial abstract template $\mathbf{M}$ using Eq. (2);
3: **while** $\mathbf{M}$ has not converged **do**
4: Update the alignment paths $\mathbf{P}^n$ by aligning $\mathbf{X}_n$ to $\mathbf{M}$, $n = 1, \cdots, N$ using Eq. (1);
5: Update the abstract template $\mathbf{M}$ with the alignment paths $\mathbf{P}^n$, $n = 1, \cdots, N$ using Eq. (2);
6: **end while**

Alg. 1 extends DBA to multidimensional sequences with a uniform initialization and imposes stricter constraints on the warping path. As a result, any vector in any sequence can only be aligned to one element of $\mathbf{M}$, which facilitates the invariant property of the separability in Section 3.2.

Convergence. Alg. 1 actually minimizes the following objective function:

$$\min_{\mathbf{P}^n, n=1,\cdots,N} \sum_{j=1}^{L} \sum_{n=1}^{N} \sum_{k=s_j^n}^{e_j^n} \|\mathbf{x}_k^n - \mathbf{m}_j\|_2^2. \quad (3)$$

The value of the objective function in Eq. (3) decreases by both alternative procedures in Alg.1, i.e., updating $\mathbf{P}^n, n = 1, \cdots, N$ for given $\mathbf{M}$ using Eq. (1) and recalculating $\mathbf{M}$ for given $\mathbf{P}^n, n = 1, \cdots, N$ using Eq. (2). The objective function also has a lower bound 0. Thus, Alg.1 is guaranteed to converge to a local minimum.

3.2 Separability Construction

We measure the separability among sequence classes based on their abstract templates in two aspects: the within-class scatter and the interclass distance. We define the intraclass scatter of a sequence class as the sum of variances of all component temporal structures in the abstract template:

$$\mathbf{S} = \sum_{j=1}^{L} (\sum_{n=1}^{N} l_j^n / \sum_{n=1}^{N} T_n) \mathbf{S}_j, \tag{4}$$

where l_j^n is the number of features in the n-th sequence aligned to the j-th temporal structure in the abstract template, and $\sum_{n=1}^{N} T_n = \sum_{j=1}^{L} \sum_{n=1}^{N} l_j^n$ is the number of all the features in all the sequences from the class. $\mathbf{S}_j$ is the variance of the j-th temporal structure, which can be estimated as the variance matrix of all feature vectors in all training sequences aligned to the j-th element of $\mathbf{M}$, i.e.,

$$\mathbf{S}_j = \frac{1}{\sum_{n=1}^{N} l_j^n} \sum_{n=1}^{N} \sum_{k=s_j^n}^{e_j^n} (\mathbf{x}_k^n - \mathbf{m}_j)(\mathbf{x}_k^n - \mathbf{m}_j)^T. \tag{5}$$

For the i-th sequence class, we denote its intraclass scatter by $\mathbf{S}^i$. Assuming there are C classes, we define the within-class scatter as the sum of intraclass scatters of all classes weighted by the prior probabilities $p^i, i = 1, \cdots, C$:

$$\mathbf{S}_w = \sum_{i=1}^{C} p^i \mathbf{S}^i, \tag{6}$$

where p^i is estimated as the number of sequences from the i-th class divided by the number of sequences from all classes.

The learned abstract template $\mathbf{M}$ of a sequence class represents the temporal structures and their general evolution of the class. The separability between two sequence classes can be reflected by the differences among the elements of the two corresponding abstract templates. We define the interclass separability as follows:

$$\mathbf{S}_b = \sum_{1 \le i,j \le C} \sum_{1 \le u,v \le L} p_u^i p_v^j (\mathbf{m}_u^i - \mathbf{m}_v^j)(\mathbf{m}_u^i - \mathbf{m}_v^j)^T. \tag{7}$$

$\mathbf{m}_u^i$ and $\mathbf{m}_v^j$ denote the u-th element of $\mathbf{M}^i$ and the v-th element of $\mathbf{M}^j$, respectively. p_u^i and p_v^j denote the prior probabilities of $\mathbf{m}_u^i$ and $\mathbf{m}_v^j$, respectively. p_u^i is estimated as the number of vectors in sequences from the i-th class that are aligned to $\mathbf{m}_u^i$, divided by the number of all vectors in all sequences from all classes.

Constructing the interclass scatter by Eq. (7) is equivalent to viewing each temporal structure as a subclass. Since each sequence class is abstracted by several ordered temporal structures, if all temporal structures from all classes are maximally separated, the separability of different sequence classes increases accordingly. Therefore, Eq. (7) can indeed reflect the separability between sequence classes.

Note that both $\mathbf{S}_w$ (6) and $\mathbf{S}_b$ (7) rely on the alignments of sequence samples to the corresponding abstract templates: $\mathbf{P} = \{\mathbf{P}_n^i, n = 1, 2, \cdots, N^i, i = 1, 2, \cdots, C\}$. We denote them by $\mathbf{S}_w(\mathbf{P})$ and $\mathbf{S}_b(\mathbf{P})$[1] to emphasize the dependencies on alignments.

Compared with HMM-based separability [1], [2], our separability construction has several advantages. (1). It does not require a large amount of training data. Even when each class has only one sequence sample, Alg. 1 can still be performed and meaningful scatters can thereby be constructed. In this case, Alg. 1 degrades to the temporal clustering algorithm [7]. (2). It does not need to estimate any parameter, thus has better scalability. (3). Owing to the constraints on the warping path, calculating $\mathbf{S}_w$ by Eq. (6) is also equivalent to viewing all temporal structures in all classes as subclasses. Therefore, $\mathbf{S}_b(\mathbf{P}) + \mathbf{S}_w(\mathbf{P}) = \mathbf{S}_t$, where $\mathbf{S}_t$ is the total scatter of all features in all sequences and is independent of $\mathbf{P}$. This invariant property ensures the joint optimization in Section 3.3.

3.3 Joint Learning of the Transformation and the Latent Alignments

Our goal is to learn a linear transformation $\mathbf{W} \in \mathbb{R}^{\mathbf{d} \times \mathbf{d}'}$ to project feature vectors in sequences from the original d-dimensional space to the most discriminative d'-dimensional subspace, in which the separability among different sequence classes is maximized. The separability depends on the alignments between the sequences and the abstract templates, which are inferred based on the pairwise distances between feature vectors in the space. When the features are projected to a subspace, the distances among the transformed features may change. The alignments may change accordingly, which should be recalculated using Alg. 1 in the subspace. The updates of the alignments, in turn, affect the determination of the transformation. Existing methods [1], [2] tackle such entanglement by fixing the alignments obtained in the original space, which may lead to suboptimal solutions.

We consider the joint learning of the transformation and the abstract templates together with the corresponding temporal alignments in the latent subspace simultaneously. We optimize the Fisher criterion that maximizes the interclass separability and minimizes the within-class scatter. Due to the invariant property, $\mathbf{S}_b(\mathbf{P}) + \mathbf{S}_w(\mathbf{P}) = \mathbf{S}_t$, the optimal projections of maximizing the ratio of $\mathbf{S}_b$ and $\mathbf{S}_w$ and maximizing the ratio of $\mathbf{S}_b$ and $\mathbf{S}_t$ are the same [54]. Therefore, we formulate our objective function as follows:

$$\max_{\mathbf{W},\mathbf{P}} tr((\mathbf{W}^T \mathbf{S}_t \mathbf{W})^{-1} \mathbf{W}^T \mathbf{S}_b(\mathbf{P}) \mathbf{W}). \tag{8}$$

We solve Eq. (8) by alternatively updating $\mathbf{W}$ and $\mathbf{P}$ to obtain a local optimal solution. We call this method LT-LDA, which is summarized in Alg. 2.

The diagram of LT-LDA is illustrated in Fig. 3. In the first stage, LT-LDA optimizes over $\mathbf{P}$ by fixing $\mathbf{W}$. The first inverse matrix item $(\mathbf{W}^T \mathbf{S}_t \mathbf{W})^{-1}$ in Eq. (8) does not

1. Strictly speaking, they also depend on $\mathbf{M}$, but $\mathbf{M}$ and $\mathbf{P}$ are closely associated, so we omit $\mathbf{M}$ for brevity.

Algorithm 2 LT-LDA

Input: the training sequences of each class $c = 1, \cdots, C$, the length of the abstract template L, the control factor a;
Output: the projection $\mathbf{W}$;

1: Initialize the abstract template $\mathbf{M}^c$ and the associated alignments $\mathbf{P}^c$ in the original space using Alg. 1, for $c = 1, \cdots, C$; calculate $\mathbf{S}_w$ (6) and $\mathbf{S}_b$ (7) according to $\mathbf{P}^c$ and $\mathbf{M}^c$;
2: Initialize $\mathbf{W}$ by solving Eq. (12)
3: **while** $\mathbf{W}$ has not converged **do**
4: Project the training sequences into a subspace by $\mathbf{W}$; update $\mathbf{M}^c$ and $\mathbf{P}^c$ in this subspace using Alg. 1, for $c = 1, \cdots, C$;
5: Recalculate $\mathbf{S}_w$ and $\mathbf{S}_b$ with the updated alignments $\mathbf{P}$ by Eq. (6) and Eq. (7), respectively;
6: Update $\mathbf{W}$ by solving Eq. (12);
7: **end while**

depend on $\mathbf{P}$. We omit this item for the moment to derive an intuitive solution and will explain its effect later. The objective then becomes

$$\max_{\mathbf{P}} tr(\mathbf{W}^T \mathbf{S}_b(\mathbf{P}) \mathbf{W}). \tag{9}$$

Since $\mathbf{S}_b(\mathbf{P}) = \mathbf{S}_t - \mathbf{S}_w(\mathbf{P})$, Eq. (9) is equivalent to the following:

$$\min_{\mathbf{P}} tr(\mathbf{W}^T \mathbf{S}_w(\mathbf{P}) \mathbf{W}). \tag{10}$$

Substituting Eq. (6) into Eq. (10) and expanding, Eq. (10) is transformed into the following:

$$\sum_{i=1}^{C} p^i \min_{\mathbf{P}^{in}, n=1,\cdots,N} \sum_{j=1}^{L} \sum_{n=1}^{N^i} \sum_{k=s_j^{in}}^{e_j^{in}} \left\| \hat{\mathbf{x}}_k^{in} - \hat{\mathbf{m}}_j^i \right\|_2^2, \tag{11}$$

where $\hat{\mathbf{x}}_k^{in} = \mathbf{W}^T \mathbf{x}_k^{in}$ and $\hat{\mathbf{m}}_j^i$ are the projected feature and the element of the abstract template in the subspace, respectively. The superscript i indicates that the variable belongs to the i-th class. To ensure the convergence and compensate the omitted item when deriving Eq. (9), which will be more clear in Section 3.4, the features should first be centered before the start of the iterations, and a whitening preprocessing should be applied to all features in all sequences in this stage.

That is, the mean of all $\mathbf{x}_k^{in}$ is zero, and $\hat{\mathbf{x}}_k^{in} = \mathbf{W}_w \mathbf{W}^T \mathbf{x}_k^{in}$, where $\mathbf{W}_w = \mathbf{\Gamma}_w^{-\frac{1}{2}}$ is the whitening transformation and $\mathbf{\Gamma}_w$ is the total scatter of all projected features in all sequences. In our experiments, we found that the two procedures can be neglected, and LT-LDA still converges, while the computational complexity is reduced.

Each of the C components of minimization in Eq. (11) is exactly the same as those of Eq. (3) in the subspace associated with $\mathbf{W}$ instead of the original space. These minimizations are independent of each other, and hence, we can learn the abstract template and the corresponding alignments of training sequences for each of the C classes using Alg. 1 individually. The learned alignments for all the sequences in all the classes are used to update $\mathbf{S}_w$ and $\mathbf{S}_b$ using Eq. (6) and Eq. (7), respectively.

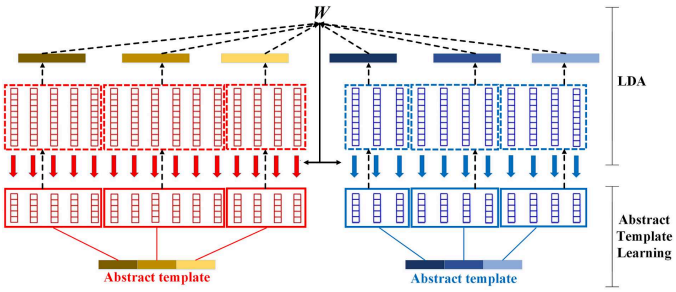


Fig. 3. The diagram of LT-LDA. The upper half: given the alignments, the features in sequences are divided into different subsets, and these subsets are viewed as independent classes; LDA is then applied to update the projection $\mathbf{W}$. The lower half: given $\mathbf{W}$, all features are projected into a subspace; in this subspace, the abstract template learning algorithm is applied to update the alignments. The two procedures are repeated alternatively until convergence.

In the second stage, LT-LDA optimizes over $\mathbf{W}$ for given $\mathbf{P}$. In this case, both $\mathbf{S}_w$ and $\mathbf{S}_b$ are fixed, and the objective function becomes a standard LDA problem:

$$\begin{aligned} &\max_{\mathbf{W}} tr((\mathbf{W}^T \mathbf{S}_t \mathbf{W})^{-1} \mathbf{W}^T \mathbf{S}_b \mathbf{W}) \\ \Leftrightarrow &\max_{\mathbf{W}} tr((\mathbf{W}^T \mathbf{S}_w \mathbf{W})^{-1} \mathbf{W}^T \mathbf{S}_b \mathbf{W}). \end{aligned} \tag{12}$$

The columns of the updated $\mathbf{W}$ are given by the eigenvectors of $\mathbf{S}_w^{-1} \mathbf{S}_b$ with respect to the d' largest eigenvalues.

3.4 Theoretical Analysis

We theoretically provide more insights and show the interest of our method by proving 1) that the abstract template learning algorithm (Alg. 1) can be linked to a trace maximization formulation; 2) that the LT-LDA algorithm (Alg. 2) is guaranteed to converge; and 3) that it is possible to simplify the joint optimization of Eq. (14) under certain conditions.

Let $\mathbf{Z}$ be the matrix consisting of all frame-wide feature vectors in all training sequences. Let $\mathbf{T}$ be the alignment indicator matrix, which is defined as follows: $\mathbf{T} = \{\pi_{i,k}\}_{N_t \times CL}$, where $\pi_{i,k} = 1$ if the frame-wide feature vector $\mathbf{z}_i$ in the i-th column of $\mathbf{Z}$ is in the sequence from the $c = \lceil (k - \frac{1}{2})/L \rceil$-th class and is aligned to the $l = k - (c-1)L$-th stage of the c-th class, and $\pi_{i,l} = 0$ otherwise. N_t is the total number of vectors in all the training sequences from all the samples. Following [23], [55], the weighted indicator matrix is defined as follows:

$$\mathbf{F} = \mathbf{T}(\mathbf{T}^T \mathbf{T})^{-\frac{1}{2}}.$$

It can be shown that

$$\mathbf{F}_{i,k} = \begin{cases} 1/\sqrt{n_k}, \ if \ \ \mathbf{z}_i \in (c,l) \\ 0, otherwise \end{cases},$$

where n_k is the number of 1 in the k-th column of $\mathbf{F}$, *i.e.*, the number of vectors aligned to the l-th stage of the c-th class.

Lemma 1. *Objective function (3) is equivalent to the trace maximization problem*

$$\max_{\mathbf{F}} tr(\mathbf{F}^T \mathbf{Z}^T \mathbf{Z} \mathbf{F}). \tag{13}$$

Proof. It turns out that the k-th column of $\mathbf{Z}\mathbf{F}\mathbf{F}^T$ is exactly the mean $\mathbf{m}_j$ of the j-th stage of the c-th class, where the

k-th column $\mathbf{z}_k$ of $\mathbf{Z}$ is aligned to this stage. Therefore, the objective function (3) can also be written as

$$\begin{aligned} &\min \left\|\mathbf{Z}-\mathbf{Z}\mathbf{F}\mathbf{F}^T\right\|_F^2 \\ \Leftrightarrow &\min tr(\mathbf{Z}^T\mathbf{Z}) - tr(\mathbf{F}^T\mathbf{Z}^T\mathbf{Z}\mathbf{F}) \\ \Leftrightarrow &\max tr(\mathbf{F}^T\mathbf{Z}^T\mathbf{Z}\mathbf{F}) \end{aligned}$$

□

Lemma 2. *Objective function (8) is equivalent to the trace maximization problem*

$$\max_{\mathbf{W},\mathbf{F}} tr(\mathbf{F}^T\mathbf{Z}^T\mathbf{W}(\mathbf{W}^T\mathbf{Z}\mathbf{Z}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{Z}\mathbf{F}). \tag{14}$$

Proof. If the data are zero-centered, $\sum_{i=1}^{N_t} \mathbf{z}_i/N_t = 0$, (otherwise, we can remove the overall mean of all the vectors in all the training sequences), then $\mathbf{S}_t$ and $\mathbf{S}_b$ can be reformulated as

$$\mathbf{S}_t = \mathbf{Z}\mathbf{Z}^T,$$

$$\mathbf{S}_b = \sum_{i=1}^{C}\sum_{u=1}^{L} p_u^i \mathbf{m}_u^i {\mathbf{m}_u^i}^T = \mathbf{Z}\mathbf{F}\mathbf{F}^T\mathbf{Z},$$

Through substituting these formulations, the objective (8) can then be reformulated as

$$\begin{aligned} &tr((\mathbf{W}^T\mathbf{S}_t\mathbf{W})^{-1}\mathbf{W}^T\mathbf{S}_b\mathbf{W}) \\ =\ &tr((\mathbf{W}^T\mathbf{Z}\mathbf{Z}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{Z}\mathbf{F}\mathbf{F}^T\mathbf{Z}^T\mathbf{W}) \\ =\ &tr(\mathbf{F}^T\mathbf{Z}^T\mathbf{W}(\mathbf{W}^T\mathbf{Z}\mathbf{Z}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{Z}\mathbf{F}) \end{aligned}$$

□

Theorem 1. *The LT-LDA algorithm (Alg. 2) is guaranteed to converge.*

Proof. In the first stage, LT-LDA optimizes over $\mathbf{P}$ by fixing $\mathbf{W}$ using Alg. 1. According to Lemma 1, this actually learns the abstract template from the whitened and projected feature sequences by optimizing $\max tr(\mathbf{F}^T\hat{\mathbf{Z}}^T\hat{\mathbf{Z}}\mathbf{F})$. $\hat{\mathbf{Z}} = \mathbf{\Gamma}_w^{-\frac{1}{2}}\mathbf{W}^T\mathbf{Z}$ is the whitened and projected data matrix, and $\mathbf{\Gamma}_w = \mathbf{W}^T\mathbf{Z}\mathbf{Z}^T\mathbf{W}$ is the total scatter of the projected data matrix. Thus, we have

$$tr(\mathbf{F}^T\hat{\mathbf{Z}}^T\hat{\mathbf{Z}}\mathbf{F}) = tr(\mathbf{F}^T\mathbf{Z}^T\mathbf{W}(\mathbf{W}^T\mathbf{Z}\mathbf{Z}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{Z}\mathbf{F})$$

which is exactly the objective (14).

In the second stage, LT-LDA optimizes over $\mathbf{W}$ for given $\mathbf{P}$. According to Lemma 2, this also optimizes Eq. (14).

Both the iterative stages decrease the objective value of Eq. (14) monotonically. Since $\mathbf{F}$ is an orthogonal matrix, the objective (14) is bounded from above. This guarantees the convergence of Alg. 2. □

Similar to [23], in some specific cases, the joint optimization of Eq. (14) can be simplified by factoring out the projection matrix $\mathbf{W}$. The result is summarized as follows.

Theorem 2. *Let $\mathbf{G} = \mathbf{Z}^T\mathbf{Z}$ be the Gram matrix. When the dimensionality is reduced to a specific value $d' = \min(CL, d, N_t)$, and a regularization term $\delta\mathbf{I}_{N_t}$ is added to the total scatter $\mathbf{S}_t$, where $\mathbf{I}_{N_t}$ is the N_t-order identity matrix, if $\mathbf{W}^*$ and $\mathbf{F}^*$ are the optimal solutions of the trace maximization problem (14):*

$$\max_{\mathbf{W},\mathbf{F}} tr(\mathbf{F}^T\mathbf{Z}^T\mathbf{W}(\mathbf{W}^T(\mathbf{Z}\mathbf{Z}^T + \delta\mathbf{I}_{N_t})\mathbf{W})^{-1}\mathbf{W}^T\mathbf{Z}\mathbf{F}) \tag{15}$$

then, $\mathbf{F}^*$ is also the optimal solution of the problem

$$\max_{\mathbf{F}} tr(\mathbf{F}^T(\mathbf{I}_{N_t} - (\mathbf{I}_{N_t} + \frac{1}{\delta}\mathbf{G})^{-1})\mathbf{F}). \tag{16}$$

Proof. We follow the proof in [23]. According to the representer theorem [56], the optimal projection matrix $\mathbf{W}$ has the form $\mathbf{W} = \mathbf{Z}\mathbf{A}$, where $\mathbf{A}$ is a coefficient matrix. The objective (15) is transformed into

$$\begin{aligned} &tr(\mathbf{F}^T\mathbf{Z}^T\mathbf{Z}\mathbf{A}(\mathbf{A}^T\mathbf{Z}^T(\mathbf{Z}\mathbf{Z}^T + \delta\mathbf{I}_{N_t})\mathbf{Z}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Z}^T\mathbf{Z}\mathbf{F}) \\ &= tr(\mathbf{F}^T\mathbf{G}\mathbf{A}(\mathbf{A}^T(\mathbf{G}\mathbf{G} + \delta\mathbf{G})\mathbf{A})^{-1}\mathbf{A}^T\mathbf{G}\mathbf{F}) \\ &= tr(\mathbf{A}^T\mathbf{G}\mathbf{F}\mathbf{F}^T\mathbf{G}\mathbf{A}(\mathbf{A}^T(\mathbf{G}\mathbf{G} + \delta\mathbf{G})\mathbf{A})^{-1}) \end{aligned} \tag{17}$$

By defining $\mathbf{\Gamma}_b = \mathbf{G}\mathbf{F}\mathbf{F}^T\mathbf{G}$ and $\mathbf{\Gamma}_w = \mathbf{G}\mathbf{G} + \delta\mathbf{G}$, we can find that Eq. (17) has a similar form to the generalized LDA problem [28], which can be solved by constructing a matrix $\mathbf{Q}$ that simultaneously diagonalizes $\mathbf{\Gamma}_b$ and $\mathbf{\Gamma}_w$. $\mathbf{Q}$ is constructed as follows.

$\mathbf{G}$ is symmetric and positive semidefinite. If the d-dimensional features are not linearly dependent (otherwise, we can remove the linearly correlated dimensions), the rank is of $\mathbf{G}$ and is $r = \min(d, N_t)$. The SVD of $\mathbf{G}$ has the form

$$\mathbf{G} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T = \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{U}_r^T,$$

where $\mathbf{U}$ is an orthogonal and square matrix, $\mathbf{\Sigma} = diag(\lambda_1, \cdots, \lambda_r, 0, \cdots, 0)$, $\mathbf{U}_r$ consist of the first r columns of $\mathbf{U}$, and $\mathbf{\Sigma}_r = diag(\lambda_1, \cdots, \lambda_r)$ contains only the nonzero singular values.

Define $\mathbf{V} = (\mathbf{\Sigma}_r^2 + \delta\mathbf{\Sigma}_r)^{-\frac{1}{2}}\mathbf{\Sigma}_r\mathbf{U}_r^T\mathbf{F}$. The SVD of $\mathbf{V}$ is denoted as $\mathbf{V} = \mathbf{M}\mathbf{\Sigma}_V\mathbf{N}^T$, where $\mathbf{M}$ and $\mathbf{N}$ are orthogonal matrices, $\mathbf{\Sigma}_V$ is a diagonal matrix of d'-th order, and d' is the rank of $\mathbf{V}$. $d' = \min(rank(\mathbf{S}_b), r) = \min(CL, d, N_t)$. By constructing $\mathbf{Q}$ as

$$\mathbf{Q} = \mathbf{U}diag((\mathbf{\Sigma}_r^2 + \delta\mathbf{\Sigma}_r)^{-\frac{1}{2}}\mathbf{M}, \mathbf{I}_{N_t-r}),$$

$\mathbf{\Gamma}_b$ and $\mathbf{\Gamma}_w$ are simultaneously diagonalized by $\mathbf{Q}$.

$$\mathbf{Q}^T\mathbf{\Gamma}_b\mathbf{Q} = diag(\mathbf{\Sigma}_V^2, \mathbf{0}_{N_t-r}),$$

$$\mathbf{Q}^T\mathbf{\Gamma}_w\mathbf{Q} = diag(\mathbf{I}_r, \mathbf{0}_{N_t-r}).$$

From Theorem 3.1 in [28], the solution of maximizing the objective (17) over $\mathbf{A}$ is given by $\mathbf{A}^*$, consisting of the first d' columns of $\mathbf{Q}$, and the maximum of the objective (17) equals $tr((\mathbf{G}\mathbf{G} + \delta\mathbf{G})^\dagger(\mathbf{G}\mathbf{F}\mathbf{F}^T\mathbf{G}))$. Therefore,

$$\begin{aligned} &tr((\mathbf{A}^T(\mathbf{G}\mathbf{G} + \delta\mathbf{G})\mathbf{A})^{-1}\mathbf{A}^T\mathbf{G}\mathbf{F}\mathbf{F}^T\mathbf{G}\mathbf{A}) \\ \le\ &tr((\mathbf{G}\mathbf{G} + \delta\mathbf{G})^\dagger(\mathbf{G}\mathbf{F}\mathbf{F}^T\mathbf{G})) \\ =\ &tr(\mathbf{F}^T\mathbf{G}(\mathbf{G}\mathbf{G} + \delta\mathbf{G})^\dagger\mathbf{G}\mathbf{F}) \\ =\ &tr(\mathbf{F}^T(\mathbf{I}_{N_t} - (\mathbf{I}_{N_t} + \tfrac{1}{\delta}\mathbf{G})^{-1})\mathbf{F}) \end{aligned}$$

The equality holds when $\mathbf{A} = \mathbf{A}^*$, where the dimensionality is implicitly reduced to d'. □

This theorem provides an upper bound of the objective for the stage of learning the partitions. The bound provides additional insights on the subspace selection of LT-LDA. From Lemma 1, in the original space, the objective function (13) of the abstract template learning actually maximizes $tr(\mathbf{F}^T\mathbf{Z}^T\mathbf{Z}\mathbf{F}) = tr(\mathbf{F}^T\mathbf{G}\mathbf{F})$. While in the d'-dimensional subspace, the objective (16) of LT-LDA actually maximizes a kernel version of Eq. (13), where the kernel

Gram matrix $\mathbf{G}_k = \mathbf{I}_{N_t} - (\mathbf{I}_{N_t} + \frac{1}{\delta}\mathbf{G})^{-1}$ is used instead of the original Gram matrix $\mathbf{G}$ in Eq. (13). $\mathbf{G}_k \rightarrow \mathbf{G}/\delta$ when $\delta \rightarrow \infty$, and hence, objective (16) is equivalent to the standard objective (13). $\mathbf{G}_k \rightarrow \mathbf{U}_r\mathbf{U}_r^T$ when $\delta \rightarrow 0$, and $\mathbf{U}_r$ is the set of the largest r principal components of all the features in all the sequences w.r.t. the nonzero eigenvalues of $\mathbf{G}$. Therefore, objective (16) is equivalent to learning the abstract templates in the subspace determined by PCA.

$\mathbf{G}_k$ can be further expressed as

$$\mathbf{G}_k = \mathbf{U}diag(\lambda_1/(\lambda_1+\delta), \cdots, \lambda_{N_t}/(\lambda_{N_t}+\delta))\mathbf{U}^T.$$

This means that the iterative procedures of LT-LDA essentially construct a kernel matrix for learning the latent alignments w.r.t. the abstract templates. The construction is achieved by performing a transformation to $\mathbf{G}$, such that each eigenvalue λ of $\mathbf{G}$ is transformed to $\lambda/(\lambda+\delta)$, while the eigenvectors of $\mathbf{G}$ remain unchanged. The subspace can be determined easily given the alignments, without the need for the iterative procedures. The nature of the subspace selection by LT-LDA indicates that it may be possible to accelerate the LT-LDA algorithm by fixing the partitions learned by optimizing objective (16) and, hence, removing the time-consuming iterative procedures, without significant degradation in performance.

3.5 Remarks

Initialization. Both the solutions of the LT-LDA algorithm and the abstract learning algorithm are locally optimal and rely on the initializations. We use the alignments $\mathbf{P}^c$ of training sequences in class$c$ ($c = 1, \cdots, C$) to the corresponding abstract template learned in the original space as the initialization of LT-LDA. The uniform alignments are used to initialize Alg. 1 in the original space. In the subspace, the coarse local structures discovered in the original space are generally preserved, and the alignments are refined since noises are removed. When the original features are too noisy or the global optimal alignments are too unbalanced, the learned alignments with such uniform initiation might be quite different from the optimal alignments. In this case, we can adopt similar strategies addressing the local optimality of k-means to improve Alg. 1. For example, we can randomly initialize the alignments several times and choose the best local optimal solution.

Computational complexity. Let N be the average number of training sequences per sequence class and T be the average length per sequence sample. L is the length of the abstract template per class, d is the original dimension, and d' is the reduced dimension. The complexity of aligning each sequence to the corresponding abstract template using the modified DTW is $O(LTd)$. The complexity of updating the alignments and abstract templates using Alg. 1 for all the C classes is $O(ICNLTd)$, where I is the number of iterations in Alg. 1. The complexity of recalculating $\mathbf{S}_t$, $\mathbf{S}_b$, and $\mathbf{W}$ is $O(CNTd^2 + C^2L^2d^2 + d^3)$. The complexity of projecting all training sequences is $O(CNTdd')$, $d' < d$. Thus, the overall complexity of Alg. 2 is $O(I'(ICNLTd + CNTd^2 + C^2L^2d^2 + d^3))$, where I' is the number of iterations in Alg. 2.

Classification with abstract templates. The learned abstract templates directly induce a prototype-based sequence classifier. A test sequence is matched to the abstract templates of all classes by the modified DTW algorithm (Eq. (1)). The class label of the abstract template that has the smallest modified DTW distance is predicted as the class of the test sequence. Furthermore, this *nearest abstract template classifier* can be viewed as a nearest mean classifier. The abstract template of a sequence class can be viewed as the mean of this class. Similarly, we can define the higher-order statistics of a sequence class. For example, for each element in the abstract template, the covariance or higher-order moment of the features that are aligned to it can be obtained, and the covariance or higher-order moment of a sequence class can be defined as the sequence of the covariances w.r.t all the ordered elements. Statistics-based statistical methods such as quadratic discriminant function can then be extended to analyze sequence data.

Selection of L. The optimal L highly depends on the data and can be selected by cross-validation. Since the abstract templates have clear physical interpretations, L can also be set according to the prior knowledge of the data. For instance, when the abstract template captures ordered key poses for action data, intuitively, $L = 8$ key poses can well reveal the evolution of an action and distinguish different actions. We evaluate the influence of L in Section 4.3. The optimal L may be class-specific. Some sequence patterns may contain more complex temporal structures or more fluctuating transitions, so their abstract templates require a larger L than other classes. In Section 4.7, we evaluate a simple adaptation method to set class-specific L. We can also employ some heuristic methods to automatically determine L. For example, for each class, we can learn the abstract template with a large L and keep merging the neighboring elements with the smallest distance until all pairwise distances exceed a threshold, or we can start from $L = 1$ and keep splitting the element with the highest variance. The reduced dimension d' is the same for all classes since LT-LDA learns a single projection for all classes.

Nonlinear extensions. LT-LDA can be extended to learn nonlinear transformations through the kernel trick and employing deep neural networks. Both the kernel extension and the deep extension follow the alternative optimization procedure. When the nonlinear mapping is fixed, the updates of the abstract templates and the corresponding alignments remain the same as Alg. 1. When the alignments are fixed, the partition of subclasses is fixed. For the kernel extension, the kernelized LDA can be directly applied to obtain the implicit nonlinear mapping. For the deep extension, since the gradient w.r.t. the parameters of the network cannot be calculated analytically, we can employ the subgradient-based method following [41] to learn the network for nonlinear mapping.

4 Experimental Results

In this section, we evaluate the proposed LT-LDA on several real-world datasets. The codes are publicly available[2].

4.1 Datasets

ChaLearn Gesture dataset [57], [58] contains Kinect videos from 20 Italian gestures. The dataset has been split into

2. https://github.com/BingSu12/LT-LDA

training, validation and test sets. Following [59], [60], we perform experiments on the segmented sequences and report results on the validation set. There are 6,850 training sequences and 3,454 testing sequences. **MSR Action3D dataset [61]** consists of depth sequences from 20 sports actions. Each action is performed 2 or 3 times by 10 subjects. We follow the same cross-subject setting as in [62], [63] to split the dataset into the training and test set, where the action samples of half of the subjects are used for training, and the samples of the rest of the subjects are used for testing. There are 284 training sequences and 273 testing sequences. **MSR Daily Activity3D dataset [62]** consists of daily activity sequences captured by a Kinect device from 16 activity types. The activities are performed by 10 subjects in the living room, and most of the activities involve the human-object interactions. Each subject performs each activity twice. We follow the cross-subject setting as in [62], [63] again to split the dataset into the training and test set, where the activity samples of half of the subjects are used for training, and the rest of the samples are used for testing. There are 160 training sequences and 160 testing sequences. **"Spoken Arabic Digits (SAD)" dataset from the UCI Machine Learning Repository [64]** consists of 8,800 vector sequences from ten classes.

There are 880 sequences per digit class. The dataset has already been split into training and test sets. The **Olympic Sports dataset [65]** consists of 783 video sequences from 16 actions. The dataset has been split into training and test sets, where 649 videos are used for training and 134 videos are used for testing. The **UCF101 dataset [66]** consists of 13,320 video clips from 101 action classes. This dataset provides three training/testing splits.

4.2 Experimental setup

Frame-wide features. For the SAD dataset, each spoken digit has been represented by a sequence of 13-dimensional MFCC features. These MFCC features were computed using the Hamming window with the sampling rate of 11,025 Hz, 16 bits and the filter preemphasized of 1-0.97Z^{-1}. For action datasets, we extract a feature vector from each frame. Therefore, every action video is represented by a sequence of frame-wide features. For the ChaLearn dataset, we employ the frame-wide features provided by the authors of [59], which are the histograms of quantized relative locations of body joints with a dimensionality of 100. For the Action3D dataset, we employ the frame-wide features provided by the authors of [63], which are the relative angles of the 3D joints with a dimensionality of 192. For the Activity3D dataset, we employ the frame-wide features provided by the authors of [62], which are the relative positions of the 3D joints with a dimensionality of 390.

For the Olympic Sports dataset, we employ the improved dense trajectories [67] and bag-of-words based frame-wide features. Specifically, for each video, the MBH descriptors are first extracted from the tracked trajectories w.r.t. a dense regular grid for all frames. All descriptors from all the training videos are clustered by k-means to form a codebook with $4,000$ visual words. The frame-wide feature is the histogram of the quantized descriptors extracted from each frame, which has a dimensionality of $4,000$.

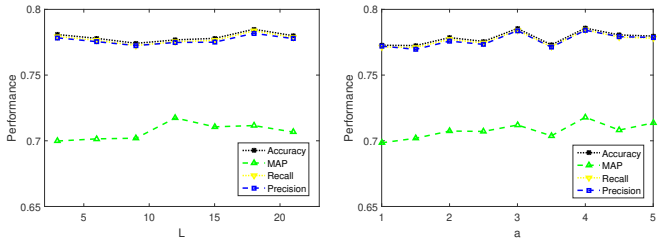


Fig. 4. Different performances by the rank pooling and the SVM classifier as functions of (a) the length L of the abstract template and (b) the control factor a on the ChaLearn Gesture dataset.

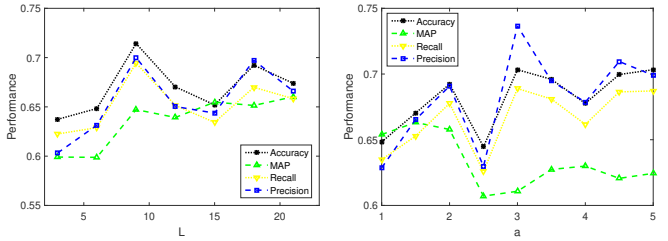


Fig. 5. Different performances by the rank pooling and the SVM classifier as functions of (a) the length L of the abstract template and (b) the control factor a on the MSR Action3D dataset.

Classification in the subspace and evaluation measures. We adopt three classifiers to perform sequence classification in the learned low-dimensional subspace, including the HMM classifier, the DTW classifier, and the SVM classifier with rank pooling [59], [60]. For the HMM classifier, a left-to-right HMM with 4 states and self-loops is trained for each sequence class, and a test sequence is classified to the class whose HMM has the highest probability to generate this sequence. The training and decoding of HMMs are performed by HTK [68]. For the DTW classifier, the training sequence that has the smallest sum of DTW distances, with all other sequences from the same class is selected as the template of this class. A test sequence is classified to the class whose template has the smallest DTW distance from it. The above two classifiers directly take sequences as input, and we use accuracy as the performance measure. For the SVM classifier, we encode each sequence into a vector by rank pooling [59], which learns a linear function that orders the frame-wide features via a ranking machine and utilizes the parameters of the function as the vector representation of the sequence. Linear SVMs are trained on these encoded vectors. The parameter C is selected by cross-validation. A test sequence is encoded into a vector and then input to the SVMs for classification. We use mean average precision (MAP) as the evaluation measure for the SVM classifier.

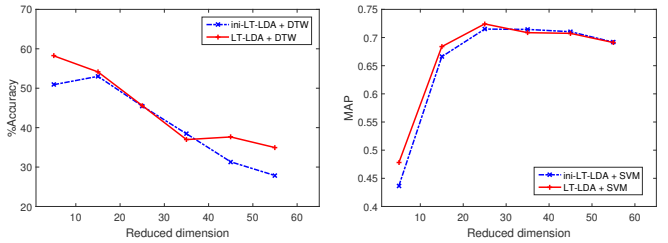
4.3 Influence of hyperparameters

The proposed LT-LDA has two preset hyperparameters: the length of each abstract template L and the factor a controlling the allowed degree of warping. In this section, we evaluate the influence of them on the ChaLearn dataset and the MSR Action3D dataset. Different performance measures including accuracy, MAP, multiclass precision, and recall by the SVM classifier are evaluated by increasing L from 3 to 21 with an interval of 3 while fixing a to 2 and increasing a from 1 to 5 with an interval of 0.5 while fixing L to 8. The

results are shown in Fig. 4 and Fig. 5, where the reduced dimensionality is fixed to 45 and 20, respectively.

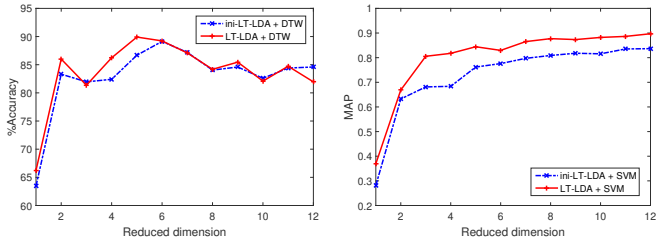
The optimal parameters are generally the same for multiclass indicators, including accuracy, precision, and recall, but are different for MAP. On the ChaLearn dataset, LT-LDA is insensitive to L for multiclass indicators and achieves the highest MAP when $L = 12$. ATs with more stages help to discriminate the subtle differences among fine gesture actions. LT-LDA is also not very sensitive to a on this dataset. It seems that allowing for larger warping leads to better results because the subtle differences can be captured more easily by more flexible alignments.

LT-LDA achieves the best multiclass performances when $L = 9$ on the MSR Action3D dataset. The larger the L is, the longer the template is, and the finer the captured temporal structures are, but the less accurate the estimated statistics of structures are, and the more likely they are to cause overfitting. Therefore, the performances decrease if the length L is too long or too short. Generally, setting L within the range of 6 to 9 leads to satisfactory results. A too large a easily leads to unbalanced alignments. If a is too small, the flexibility of alignments may be restricted. Allowing for appropriate warping leads to satisfactory results. We fix a to 2 in the following experiments and fix L to 8 unless otherwise specified.



(a) (b)

Fig. 6. Comparisons of the proposed LT-LDA without and with the joint learning of the latent alignments. (a) Accuracies by the DTW classifier and (b) MAPs by the rank pooling and the SVM classifier as functions of the dimensionality of the subspace on the ChaLearn Gesture dataset.



(a) (b)

Fig. 7. Comparisons of the proposed LT-LDA without and with the joint learning of the latent alignments. (a) Accuracies by the DTW classifier and (b) MAPs by the rank pooling and the SVM classifier as functions of the dimensionality of the subspace on the SAD dataset.

4.4 Effects of the joint learning

In LT-LDA, the latent alignments are jointly learned with the underlying subspace. If we instead use the alignments in the original space calculated by Alg. 1 directly, LT-LDA degenerates to the initialization of $\mathbf{W}$ in LT-LDA. We denote this algorithm by ini-LT-LDA and compare it with LT-LDA on the ChaLearn dataset, the SAD dataset, and the Olympic Sports dataset. The comparison results by using different classifiers and evaluation measures are shown in Fig. 6, Fig. 7, and Fig. 8, respectively.

On the ChaLearn dataset, the optimal results of LT-LDA among all dimensions are better than those of ini-LT-LDA, but the improvements are quite small. The parameter C of the SVMs is fixed to 100 here. Tuning C by cross-validation can further improve the performances of LT-LDA as shown in Fig 10(c). On the SAD dataset, the improvements of LT-LDA over ini-LT-LDA are still limited on the accuracies by the DTW classifier but are more significant on MAPs by the SVM classifier. This is because LT-LDA optimizes the overall separability between sequence classes, and hence, sequences from different classes are better separated en bloc. However, regarding a specific test sequence, it may be distributed on the boundary of a nearby class and confuse the classifier.

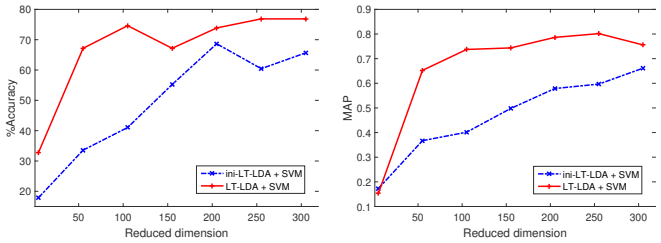
On the large-scale Olympic Sports dataset, LT-LDA significantly outperforms ini-LT-LDA by a much larger margin. This is because the depth information and the locations of human joints are available for the ChaLearn dataset; thus, the alignments in the original space are accurate and should be preserved in the optimal subspace. After the refinement of LT-LDA, the alignments remain nearly unchanged and the performance changes are small. On the Olympic dataset, only the raw videos with complex backgrounds are available. The initial alignments are quite noisy. Refining them by LT-LDA improves the performances significantly.

In summary, in nearly all cases, the best results among all these dimensions of LT-LDA outperform those of ini-LT-LDA. By jointly learning the latent alignments associated with the subspace, the classification performance in the subspace is improved. This is because the temporal structures and the alignments may change from those in the original space. In the learned subspace of ini-LT-LDA, although different classes are better separated under the alignments in the original space, additional confusions may be introduced due to the changes of alignments. However, for LT-LDA, since the separability is maximized in the subspace under the corresponding alignments, the learned subspace achieves joint optimality among all possible subspaces.

We can also observe that on some datasets with some classifiers, the performances of LT-LDA are lower than those of ini-LT-LDA in some dimensions. In addition to overfitting on the training set, another possible reason is that the aligned paths of each class are learned without discriminating other classes, and the objective of LT-LDA is not directly related to a classification performance measure. For some dimensions, although our joint learning leads to better separation of temporal structures, it may interfere with the classifier, as the discrimination of the classifier may not necessarily accord to such separation but rather to other aspects of some local or global properties of sequences. Therefore, updating the paths may cause some fluctuations. However, it generally leads to a more discriminative subspace because the entire sequences are more likely to attain better separation if the temporal structures are better separated.

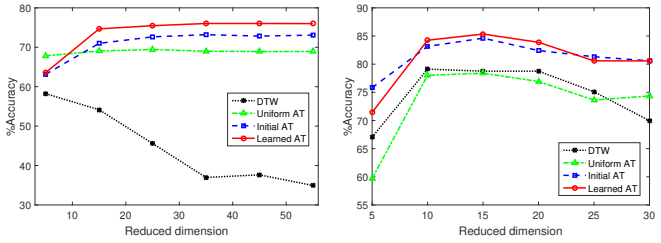
4.5 Evaluation of the nearest AT classifier

We evaluate the performances of the nearest abstract template (AT) classifier on the ChaLearn dataset and the MSR



(a) (b)

Fig. 8. Comparisons of the proposed LT-LDA without and with the joint learning of the latent alignments. (a) Accuracies and (b) MAPs by the rank pooling and the SVM classifier as functions of the dimensionality of the subspace on the Olympic Sports dataset.

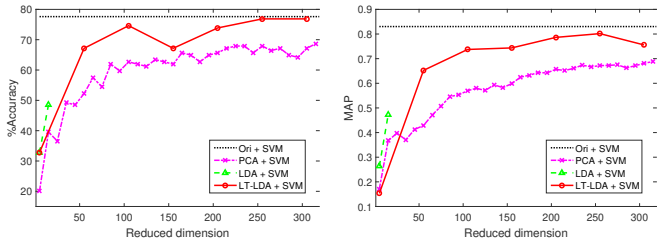


(a) (b)

Fig. 9. Comparisons of the DTW classifier and the nearest AT classifier with the uniform AT, the initial AT, and the learned AT by LT-LDA on (a) the ChaLearn dataset and (b) the Action3D dataset.

Action3D dataset. Since AT depends on the alignments, different alignments lead to different AT. We use uniform AT, initial AT and learned AT to denote the ATs with the uniform alignments, the alignments in the original space, and the latent alignments in the subspace learned by LT-LDA, respectively. We compare the performances of the nearest AT classifier by employing the uniform AT, the initial AT, and the learned AT as the prototype, respectively, in Fig. 9. Both the initial ATs generated by ini-LT-LDA and learned ATs generated by LT-LDA outperform uniform ATs significantly. This indicates that the proposed AT learning method Alg. 1 is able to produce a more discriminative prototype that better captures the common representative temporal structures of a sequence class. We can also observe that the superiorities of learned ATs over initial ATs show similar trends to those of LT-LDA over ini-LT-LDA by using other classifiers in Section 4.4. This verifies the effectiveness of the joint learning of the subspace and the ATs again.

The accuracies of the DTW classifier are also shown as baselines for comparison, where the sequences are projected by LT-LDA, and the transformed sequence that has the smallest sum of DTW distances of all other training sequences from the same class is selected as the prototype. It can be observed that the accuracies are inferior to those of the nearest AT classifiers. This may be because the AT with any alignment encodes the averaged ordered characteristics of a sequence class and acts as a sort of statistic. Therefore, such a prototype is more robust than a single selected training sequence. We can divide a sequence class that has large intraclass variations into some subclasses and extract an AT for every subclass to further improve the AT-induced classifier.



(a) (b)

Fig. 14. (a) Accuracies and (b) MAPs by the rank pooling and the SVM classifier as functions of the dimensionality of the subspace on the Olympic Sports dataset.

4.6 Comparison with different DR methods

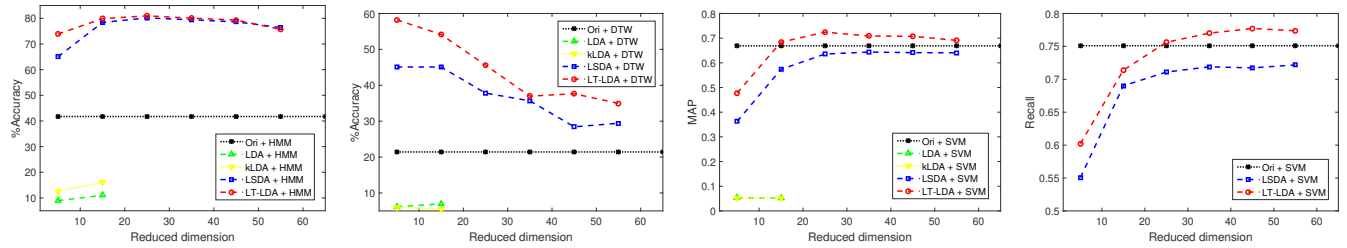
We compare the proposed LT-LDA with other Fisher criterion-based dimensionality reduction methods. These methods include LDA and kernel LDA (kLDA), by viewing the features in sequences as independent samples with the same label, and LSDA. The performances of the original feature sequences with the three classifiers are also presented as baselines. We use the drtoolbox [69] to perform LDA and kLDA. On most datasets, the length of a single sequence is generally comparable to the number of training sequences of a class. Therefore, for kLDA, it is impractical to use all the features in all the training sequences because this will lead to an enormous size of the kernel matrix and very large space and computational overhead. Following [2], we sample 1 to 5 features randomly from each sequence for training. We use the same parameters of LSDA as in [1], [2].

Fig. 10, Fig. 11, Fig. 12 and Fig. 13 depict the performances as functions of the dimensionality of the learned subspace on the ChaLearn dataset, the MSR Action3D dataset, the MSR Activity3D dataset, and the SAD dataset, respectively. We can observe that the proposed LT-LDA achieves the best performances among all these dimensionality reduction methods, in regard to all three classifiers with different evaluation measures on nearly all of the datasets. On the ChaLearn dataset and the MSR Action3D dataset, the accuracies of LT-LDA are consistently better than those of other methods on all the reduced dimensions, especially in regard to the DTW classifier, where LT-LDA outperforms the second LSDA by a margin of more than 10%. With regard to the HMM classifier and the DTW classifier, LT-LDA, which uses less than 15 dimensions, achieves much better results than the original features with hundreds of dimensions.

On the MSR Activity3D dataset, with the SVM classifier implemented with the rank pooling, the best MAP and recall of LT-LDA outperform those of the second LSDA by a margin of more than 5%. On the SAD dataset, regarding the HMM classifier and the SVM classifier, LSDA performs comparatively with LT-LDA. This is because sufficient training samples are available on this dataset and the original dimension of the frame-wide features is low. Therefore, LSDA can reliably train HMMs to obtain sequence statistics. On all the datasets, the worse performances of LDA and kLDA are caused by the dependency of features in sequences, which violates the basic assumption of the two methods, while LT-LDA well exploits such temporal dependencies by learning the latent alignments.

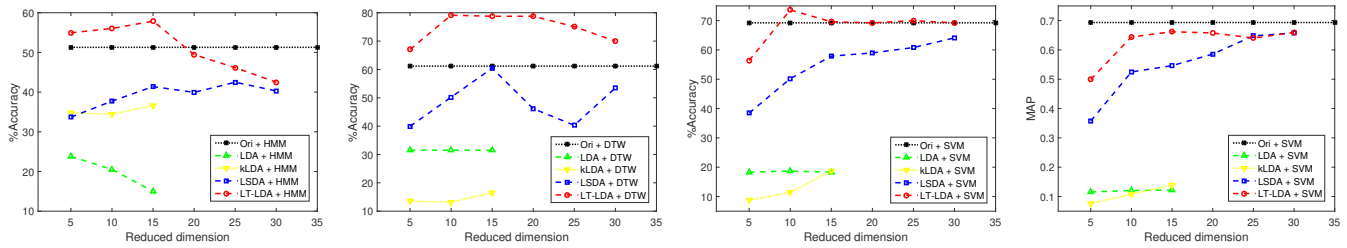
For the Olympic Sports dataset, there are less than 35 training videos per class, but each video generally has hun-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2919303, IEEE Transactions on Pattern Analysis and Machine Intelligence



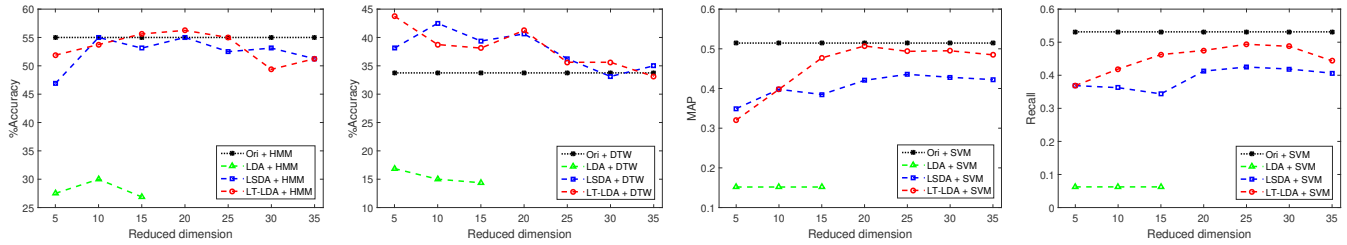
(a) (b) (c) (d)

Fig. 10. (a) Accuracies by the HMM classifier, (b) accuracies by the DTW classifier, (c) MAPs by the SVM classifier and (d) multiclass average recalls by the SVM classifier as functions of the dimensionality of the subspace on the ChaLearn dataset.



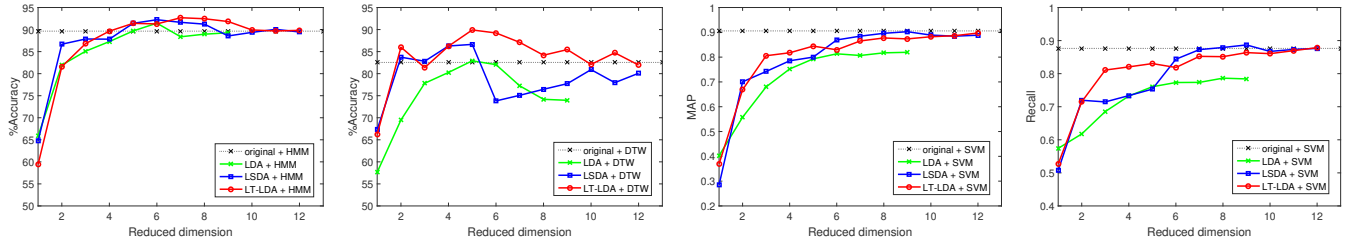
(a) (b) (c) (d)

Fig. 11. (a) Accuracies by the HMM classifier, (b) accuracies by the DTW classifier, (c) accuracies by the SVM classifier and (d) MAPs by the SVM classifier as functions of the dimensionality of the subspace on the MSR Action3D dataset.



(a) (b) (c) (d)

Fig. 12. (a) Accuracies by the HMM classifier, (b) accuracies by the DTW classifier, (c) MAPs by the SVM classifier and (d) multiclass average recalls by the SVM classifier as functions of the dimensionality of the subspace on the MSR Activity3D dataset.



(a) (b) (c) (d)

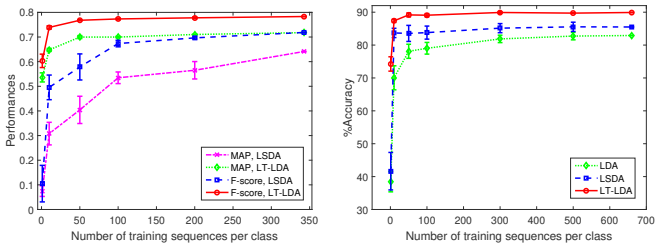
Fig. 13. (a) Accuracies by the HMM classifier, (b) accuracies by the DTW classifier, (c) MAPs by the SVM classifier and (d) multiclass average recalls by the SVM classifier as functions of the dimensionality of the subspace on the SAD dataset.

dreds of frames, and the dimensionality of the feature for each frame is $4,000$. Therefore, it is impractical to train an HMM for each class, and hence, LSDA cannot be employed. kLDA is also computationally prohibited. We compare LT-LDA with PCA and LDA on this dataset. LDA can only preserve $C-1=15$ dimensions at most. We set L to 20 on this dataset such that LT-LDA can preserve $20\times C-1=319$ dimensions at most. As shown in Fig. 14, LT-LDA consistently outperforms PCA and further improves the performances when more than 15 dimensions are preserved. With only 250 dimensions, LT-LDA achieves comparable accuracy and MAP with the original BoW-based distributed features with $4,000$ dimensions. This implies that the BoW features can be greatly compressed by LT-LDA, while the discriminative information is maintained.

4.7 Empirical analysis

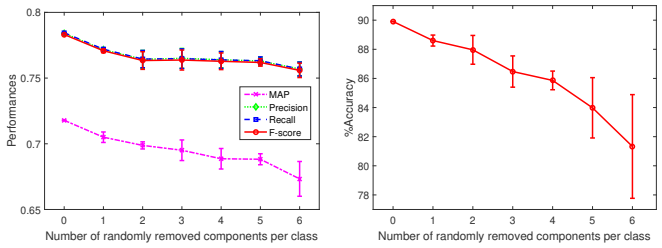
Comparisons of training times. On the ChaLearn dataset, the training times of LDA, LSDA, and LT-LDA are 1087.6, 1113.3, and 12509 seconds, respectively, when reducing the dimension to 45. On the SAD dataset, the training times of LDA, LSDA, and LT-LDA are 128.1, 860.4, and 3742.9 seconds, respectively, when reducing the dimension to 12. LSDA and LT-LDA take longer training times than LDA to tackle the temporal alignments. Compared with LSDA, which fixes the alignments in the original space, LT-LDA achieves the joint learning of the projection and the alignments at the cost

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2019.2919303, IEEE Transactions on Pattern Analysis and Machine Intelligence



(a) (b)

Fig. 15. Performances as functions of the number of training sequences per class (a) for the SVM classifier on the ChaLearn dataset and (b) for the DTW classifier on the SAD dataset.



(a) (b)

Fig. 16. Performances as functions of the number of randomly removed components per class (a) for the SVM classifier on the ChaLearn dataset and (b) for the DTW classifier on the SAD dataset.

of more training time. All these methods only require one matrix multiplication to transform a test sequence.

Influence of the number of training sequences. Fig. 15(a) and Fig. 15(b) plot the performances of LT-LDA and the other methods as functions of the number of training sequences per class on the ChaLearn dataset and the SAD dataset, respectively. We randomly select different numbers of training sequences per class. We repeat the random selection five times and report the average performances and standard deviations. On the ChaLearn dataset, the dimension is reduced to 45, and a is set to 4. MAP and the multiclass F-score for the SVM classifier are used as performance measures. On the SAD dataset, the dimension is reduced to 5. For the DTW classifier, accuracy is used as the performance measure. We can observe that the proposed LT-LDA outperforms other methods significantly when only a few training sequences are available and consistently obtains superior performances with the increase in the number of training sequences. LT-LDA obtains stable performances with only 100 training sequences per class. These results verify that LT-LDA does not require large amounts of training data.

Influence of randomly dropping components. We first use LT-LDA with $L = 8$ to learn the abstract templates w.r.t. the alignments in the subspace. Then, we randomly remove different numbers of components from the abstract template for each class. We use the remaining components to construct $\mathbf{S}_b$ (7) and obtain the projection. The number of removed components per class increases from 0 to 6, and we repeat the random removal five times for each number. Fig. 16(a) shows the average MAP, multiclass precision, recall, and F-score for the SVM classifier as functions of the number of removed components on the ChaLearn dataset. Fig. 16(b) shows the average accuracy for the DTW classifier as a function of the number of removed components on the SAD dataset. As expected, the performances decrease as more components are removed. The learned abstract template captures the essential temporal structures. Each component is necessary and represents a key stage.

Impact of class-specific L. We perform a simple adaptation method to determine class-specific L. For the c-th class, we set L to 0.1 times the average length of all training sequences of this class. In this way, L is larger for classes with longer average lengths because patterns with longer durations are more likely to contain more temporal structures. The results of LT-LDA using this adaptation method (denoted by LT-LDA-var) with the SVM classifier on the ChaLearn dataset are shown in Table 1. a is set to 4, and the reduced dimension is 45. We can observe that this simple class-specific adaptation method outperforms the uniform assignment of L. This indicates that effective adaptation techniques for automatically learning class-specific L benefit the proposed LT-LDA method.

TABLE 1
Comparison with other methods on the ChaLearn dataset.

Method	Precision	Recall	F-score
Rank pooling [59]	0.753	0.751	0.752
GRP [45]	0.753	0.752	0.751
LSDA+rank pooling [2]	0.768	0.767	0.767
LT-LDA+rank pooling	**0.784 (45)**	**0.783 (45)**	**0.783 (45)**
LT-LDA-var+rank pooling	**0.791 (45)**	**0.790 (45)**	**0.789 (45)**

4.8 Comparison with state-of-the-art methods

On the ChaLearn dataset, we evaluate the multiclass precision, recall, and F-score for LT-LDA with fine-tuned $a = 4$ via the rank pooling and the SVM classifier. The comparisons with the gesture recognition methods using the same frame-wide features are shown in Table 1. LT-LDA outperforms other methods using only 45 dimensions. Setting class-specific L further improves the performances of LT-LDA.

LT-LDA can be used to transform arbitrary frame-wide features. Any sequence classification method can be applied to the resulting temporal representations. On the MSR Action3D dataset, we combine LT-LDA with two generalized temporal sliding LSTM (G-TS-LSTM) networks [49], namely, G-TS-LSTM-AM and G-TS-LSTM-GM, where AM and GM indicate that average mean and geometric mean are used for the ensemble of multiterm TS-LSTM networks, respectively. In [49], the frame-wide features are the 60-dimensional motion features. We reduce the dimension to 30 for LT-LDA and use the projected sequences as input to the LSTM networks. The results in comparison with state-of-the-art methods are shown in Tab. 2. LT-LDA improves the accuracies of the original LSTM networks by 1% using only half the dimension.

On the MSR Activity3D dataset, the covariance-based methods such as kernelized-COV [73] work well. The frame-wide features used in [73] are the 120-dimensional velocity and acceleration of joint positions [70], which are publicly available. We employ LT-LDA to reduce the dimension of the features to 80 and apply kernelized-COV [73] to the transformed sequences. The accuracies in comparison with state-of-the-art methods are shown in Tab. 3. LT-LDA improves the accuracy of kernelized-COV by approximately 2% and outperforms other methods. These results indicate

TABLE 2
Comparison with state-of-the-art methods on the MSR Action3D dataset.

Method	Accuracy
Actionlet Ensemble [62]	88.2%
Moving Pose [70]	91.7%
COV-$J_{\mathcal{H}}$-SVM [71]	80.4%
Ker-RP-POL [72]	96.2%
Ker-RP-RBF [72]	**96.9**%
Kernelized-COV [73]	96.2%
SCK+DCK [74]	91.45%
GRP [45]	81.68%
Qiao et al. [75]	95.9%
Ji et al. [76]	90.8%
FTP-SVM [48]	90.01%
Bi-LSTM [48]	86.18%
G-TS-LSTM-AM [49]	92.31%
G-TS-LSTM-GM [49]	91.21%
LT-LDA+G-TS-LSTM-AM	**93.04**%
LT-LDA+G-TS-LSTM-GM	92.67%

TABLE 3
Comparison with state-of-the-art methods on the MSR Activity3D dataset.

Method	Accuracy
Actionlet Ensemble [62]	85.8%
Moving Pose [70]	73.8%
COV-$J_{\mathcal{H}}$-SVM [71]	75.5%
Ker-RP-POL [72]	96.9%
Ker-RP-RBF [72]	96.3%
Kernelized-COV [73]	96.3%
LRTS [77]	80.6%
Qiao et al. [75]	75.0%
Baradel et al. [78]	90.0%
Luo et al. [79]	86.9%
GRP [45]	91.3%
Ji et al. [76]	81.3%
DSSCA SSLM [80]	97.5%
MDMTL [81]	93.8%
LT-LDA+Kernelized-COV	**98.1%**

that LT-LDA is able to produce more discriminative low-dimensional temporal representations and therefore improves various classification methods.

On the UCF101 dataset, we employ the ResNeXt-101 model [52] that is pretrained on the Kinetics dataset [82] to extract frame-wide features. For each video, ResNeXt-101 is applied to every 16 frames successively, resulting in a sequence of 2048-dimensional features. We use LT-LDA to reduce the dimension to 500. We encode each transformed sequence into a vector representation by mean pooling and employ the linear-SVM for classification. The average accuracies over three splits in comparison with state-of-the-art methods are shown in Tab. 4, where "ResNeXt-101+SVM" indicates our reproduced result of classifying the original ResNeXt-101 features by mean pooling and the SVM. C3D, TSN, and I3D fuses multiple networks. LT-LDA is applied to a single deep network taking only RGB data as input. LT-LDA obtains comparable results with other state-of-the-art single deep networks in addition to the fine-tuned ResNeXt-101 with larger inputs (64 frames). Under the same conditions, the temporal representations produced by LT-LDA achieve better results than the original deep representations but only use less than a quarter of the original dimension.

TABLE 4
Comparison with state-of-the-art methods on the UCF101 dataset.

Method	Accuracy
Two-stream networks [83]	88.0%
C3D (3 nets) + IDT [50]	90.4%
TSN (3 modalities) [84]	94.2%
HDPE [13]	83.6%
GRP + IDT [45]	92.3%
Two-steam I3D [51]	**98.0**%
ResNeXt-101 [52]	90.7%
ResNeXt-101 (64f, fine-tuned) [52]	94.5%
ResNeXt-101+SVM	89.2%
LT-LDA+ResNeXt-101	90.3%

5 Conclusion

In this paper, we have presented a dimensionality reduction method for sequence data, called LT-LDA, which learns the subspace and infers the latent alignments within it simultaneously. LT-LDA transforms the sequences of high-dimensional features into new sequences of lower-dimensional features by projecting the features in sequences onto a subspace, where the separability of different temporal structures in all the sequence classes is maximized. The temporal structures are discovered by learning an abstract template and aligning the training sequences to it for each class. The learned subspace is optimal in the sense that the separability under the alignments in this subspace is larger than that in any other subspace under the alignments in that subspace. We show that the learning of the subspace, the latent alignments, and the temporal structures can be formulated into a joint objective function, which can be solved by iteratively repeating the two alternative procedures of applying LDA and learning the abstract templates. The effectiveness of the proposed method is demonstrated on several real-world datasets using various evaluation measures and classifiers.

References

[1] B. Su and X. Ding, "Linear sequence discriminant analysis: a model-based dimensionality reduction method for vector sequences," in *Proc. IEEE Int'l Conf. Computer Vision*, 2013, pp. 889–896.

[2] B. Su, X. Ding, H. Wang, and Y. Wu, "Discriminative dimensionality reduction for multi-dimensional sequences," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 1, pp. 77–91, 2018.

[3] B. Su, X. Ding, C. Liu, H. Wang, and Y. Wu, "Discriminative transformation for multi-dimensional temporal sequences," *IEEE Trans. Image Processing*, vol. 26, no. 7, pp. 3579–3593, 2017.

[4] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[5] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[6] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011.

[7] B. Su, J. Zhou, X. Ding, H. Wang, and Y. Wu, "Hierarchical dynamic parsing and encoding for action recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 202–217.

[8] B. Su and Y. Wu, "Learning low-dimensional temporal representations," in *International Conference on Machine Learning*, 2018, pp. 4768–4777.

[9] J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard, "Segmenting motion capture data into distinct behaviors," in *Proceedings of Graphics Interface 2004*. Canadian Human-Computer Communications Society, 2004, pp. 185–194.

[10] D. Gong, G. Medioni, S. Zhu, and X. Zhao, "Kernelized temporal cut for online temporal segmentation and recognition," in *European Conference on Computer Vision*. Springer, 2012, pp. 229–243.

[11] S. Li, K. Li, and Y. Fu, "Temporal subspace clustering for human motion segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4453–4461.

[12] B. Krüger, A. Vögele, T. Willig, A. Yao, R. Klein, and A. Weber, "Efficient unsupervised temporal segmentation of motion data," *IEEE Transactions on Multimedia*, vol. 19, no. 4, pp. 797–812, 2017.

[13] B. Su, J. Zhou, X. Ding, and Y. Wu, "Unsupervised hierarchical dynamic parsing and encoding for action recognition," *IEEE Trans. Image Processing*, vol. 26, no. 12, pp. 1057–7149, 2017.

[14] M. Hoai and F. De la Torre, "Maximum margin temporal clustering," in *Artificial Intelligence and Statistics*, 2012, pp. 520–528.

[15] F. Zhou, F. De la Torre, and J. K. Hodgins, "Hierarchical aligned cluster analysis for temporal clustering of human motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 3, pp. 582–596, 2013.

[16] H. Sidenbladh, M. J. Black, and D. J. Fleet, "Stochastic tracking of 3d human figures using 2d image motion," in *European conference on computer vision*. Springer, 2000, pp. 702–718.

[17] G. W. Taylor, G. E. Hinton, and S. T. Roweis, "Modeling human motion using binary latent variables," in *Advances in neural information processing systems*, 2007, pp. 1345–1352.

[18] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 283–298, 2008.

[19] T. Jaeggli, E. Koller-Meier, and L. Van Gool, "Learning generative models for multi-activity body pose estimation," *International Journal of Computer Vision*, vol. 83, no. 2, pp. 121–134, 2009.

[20] A. Yao, J. Gall, L. V. Gool, and R. Urtasun, "Learning probabilistic non-linear latent variable models for tracking complex activities," in *Advances in Neural Information Processing Systems*, 2011, pp. 1359–1367.

[21] E. B. Fox, M. C. Hughes, E. B. Sudderth, M. I. Jordan *et al.*, "Joint modeling of multiple time series via the beta process with application to motion capture segmentation," *The Annals of Applied Statistics*, vol. 8, no. 3, pp. 1281–1313, 2014.

[22] R. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, pp. 179–188, 1936.

[23] J. Ye, Z. Zhao, and M. Wu, "Discriminative k-means for clustering," in *Advances in Neural Information Processing Systems (NIPS)*, 2007, pp. 1649–1656.

[24] S. Nikitidis, S. Zafeiriou, and M. Pantic, "Merging svms with linear discriminant analysis: A combined model," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2014, pp. 1067–1074.

[25] K. Lee and J. Kim, "On the equivalence of linear discriminant analysis and least squares," in *AAAI Conference on Artificial Intelligence*, 2015.

[26] Z. Jin, J.-Y. Yang, Z.-S. Hu, and Z. Lou, "Face recognition based on the uncorrelated discriminant transformation," *Pattern recognition*, vol. 34, no. 7, pp. 1405–1416, 2001.

[27] J. Ye, R. Janardan, Q. Li, and H. Park, "Feature extraction via generalized uncorrelated linear discriminant analysis," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 113.

[28] J. Ye, "Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems," *Journal of Machine Learning Research*, vol. 6, no. Apr, pp. 483–502, 2005.

[29] M. Loog and R. Duin, "Linear dimensionality reduction via a heteroscedastic extension of lda: the chernoff criterion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 732–739, 2004.

[30] M. Zhu and A. Martinez, "Subclass discriminant analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1274–1286, 2006.

[31] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: a general framework for dimensionality reduction," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40–51, 2007.

[32] M. Loog, R. P. W. Duin, and R. Haeb-Umbach, "Multiclass linear dimension reduction by weighted pairwise fisher criteria," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 7, pp. 762–766, 2001.

[33] W. Bian and D. Tao, "Max-min distance analysis by using sequential sdp relaxation for dimension reduction," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 1037–1050, 2011.

[34] Y. Zhang and D.-Y. Yeung, "Worst-case linear discriminant analysis," in *Proc.Advances in Neural Information Processing Systems*, 2010.

[35] B. Su, X. Ding, C. Liu, and Y. Wu, "Heteroscedastic max-min distance analysis," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2015, pp. 4539–4547.

[36] ——, "Heteroscedastic maxmin distance analysis for dimensionality reduction," *IEEE Trans. Image Processing*, vol. 27, pp. 4052–4065, 2018.

[37] F. De la Torre and T. Kanade, "Discriminative cluster analysis," in *Proc. IEEE Int'l Conf. Machine Learning*. ACM, 2006, pp. 241–248.

[38] C. Ding and T. Li, "Adaptive dimension reduction using discriminant analysis and k-means clustering," in *Proc. IEEE Int'l Conf. Machine Learning*. ACM, 2007, pp. 521–528.

[39] F. Zhou and F. De la Torre, "Generalized time warping for multimodal alignment of human motion," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2012, pp. 1282–1289.

[40] F. Zhou and F. Torre, "Canonical time warping for alignment of human behavior," in *Proc. Advances in Neural Information Processing Systems*, 2009, pp. 2286–2294.

[41] G. Trigeorgis, M. A. Nicolaou, B. W. Schuller, and S. Zafeiriou, "Deep canonical time warping for simultaneous alignment and representation learning of sequences," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1128–1138, 2018.

[42] A. Shyr, R. Urtasun, and M. I. Jordan, "Sufficient dimension reduction for visual sequence classification," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2010, pp. 3610–3617.

[43] R. Flamary, D. Tuia, B. Labb, G. Camps-Valls, and A. Rakotomamonjy, "Large margin filtering," *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 648–659, 2012.

[44] R. Lajugie, D. Garreau, F. Bach, and S. Arlot, "Metric learning for temporal sequence alignment," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 1817–1825.

[45] A. Cherian, B. Fernando, M. Harandi, and S. Gould, "Generalized rank pooling for activity recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3222–3231.

[46] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[47] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Inform. Process. Syst.*, 2014, pp. 3104–3112.

[48] A. Ben Tanfous, H. Drira, and B. Ben Amor, "Coding kendall's shape trajectories for 3d action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2840–2849.

[49] I. Lee, D. Kim, S. Kang, and S. Lee, "Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1012–1020.

[50] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4489–4497.

[51] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[52] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6546–6555.

[53] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 12, pp. 2247–2253, 2007.

[54] K. Fukunaga, *Introduction to statistical pattern recognition*. NewYork: Academic Press, 1990.

[55] I. Dhillon, Y. Guan, and B. Kulis, "A unified view of kernel k-means. spectral clustering and graph cuts," *Technical report, Department of Computer Sciences, University of Texas at Austin*, 2005.

[56] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[57] S. Escalera, J. Gonzàlez, X. Baró, M. Reyes, O. Lopes, I. Guyon, V. Athitsos, and H. Escalante, "Multi-modal gesture recognition challenge 2013: Dataset and results," in *Proceedings of the 15th ACM on International conference on multimodal interaction*. ACM, 2013, pp. 445–452.

[58] S. Escalera, J. Gonzàlez, X. Baró, M. Reyes, I. Guyon, V. Athitsos, H. Escalante, L. Sigal, A. Argyros, C. Sminchisescu *et al.*, "Chalearn multi-modal gesture recognition 2013: grand challenge and workshop summary," in *Proceedings of the 15th ACM on International conference on multimodal interaction*. ACM, 2013, pp. 365–368.

[59] B. Fernando, E. Gavves, J. O. M., A. Ghodrati, and T. Tuytelaars, "Modeling video evolution for action recognition," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2015, pp. 5378–5387.

[60] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars, "Rank pooling for action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 4, pp. 773–787, 2017.

[61] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *IEEE Int'l Workshop on CVPR for Human Communicative Behavior Analysis*, 2010, pp. 9–14.

[62] J. Wang, Z. Liu, and Y. Wu, "Mining actionlet ensemble for action recognition with depth cameras," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 1290–1297.

[63] J. Wang and Y. Wu, "Learning maximum margin temporal warping for action recognition," in *Proc. IEEE Int'l Conf. Computer Vision*, 2013, pp. 2688–2695.

[64] K. Bache and M. Lichman, *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml, University of California, Irvine, School of Information and Computer Sciences, 2013.

[65] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification," in *European conference on computer vision*. Springer, 2010, pp. 392–405.

[66] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human action classes from videos in the wild," *CRCV-TR-12-01*, November 2012.

[67] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 3551–3558.

[68] S. Young, G. Evermann, D. Kershaw, D. Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK book*. Cambridge University Engineering Dept., 2001.

[69] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[70] M. Zanfir, M. Leordeanu, and C. Sminchisescu, "The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2752–2759.

[71] M. Harandi, M. Salzmann, and F. Porikli, "Bregman divergences for infinite dimensional covariance matrices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1003–1010.

[72] L. Wang, J. Zhang, L. Zhou, C. Tang, and W. Li, "Beyond covariance: Feature representation with nonlinear kernel matrices," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4570–4578.

[73] J. Cavazza, A. Zunino, M. San Biagio, and V. Murino, "Kernelized covariance for action recognition," in *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE, 2016, pp. 408–413.

[74] P. Koniusz, A. Cherian, and F. Porikli, "Tensor representations via kernel linearization for action recognition from 3d skeletons," in *European Conference on Computer Vision*. Springer, 2016, pp. 37–53.

[75] R. Qiao, L. Liu, C. Shen, and A. van den Hengel, "Learning discriminative trajectorylet detector sets for accurate skeleton-based action recognition," *Pattern Recognition*, vol. 66, pp. 202–212, 2017.

[76] X. Ji, J. Cheng, W. Feng, and D. Tao, "Skeleton embedded motion body partition for human action recognition using depth sequences," *Signal Processing*, vol. 143, pp. 56–68, 2018.

[77] C. Jia and Y. Fu, "Low-rank tensor subspace learning for rgb-d action recognition," *IEEE Transactions on Image Processing*, vol. 25, no. 10, pp. 4641–4652, 2016.

[78] F. Baradel, C. Wolf, and J. Mille, "Pose-conditioned spatio-temporal attention for human action recognition," *arXiv preprint arXiv:1703.10106*, 2017.

[79] Z. Luo, B. Peng, D.-A. Huang, A. Alahi, and L. Fei-Fei, "Unsupervised learning of long-term motion dynamics for videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2203–2212.

[80] A. Shahroudy, T.-T. Ng, Y. Gong, and G. Wang, "Deep multimodal feature analysis for action recognition in rgb+ d videos," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1045–1058, 2018.

[81] J. Liu, Y. Li, S. Song, J. Xing, C. Lan, and W. Zeng, "Multi-modality multi-task recurrent neural network for online action detection," *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[82] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[83] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[84] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: towards good practices for deep action recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 20–36.

Bing Su received a B.S. degree in information engineering from Beijing Institute of Technology, Beijing, in 2010, and a Ph.D. degree in Electronic Engineering from Tsinghua University, Beijing, in 2016. From 2016 to 2018, he was an Assistant Professor of the Institute of Software, Chinese Academy of Sciences, Beijing. Currently, he is an Associate Professor in the Institute of Software, Chinese Academy of Sciences. His research interests include pattern recognition, computer vision, and machine learning.

Ying Wu received the B.S. from Huazhong University of Science and Technology, Wuhan, China, in 1994, the M.S. from Tsinghua University, Beijing, China, in 1997, and the Ph.D. in electrical and computer engineering from the University of Illinois at Urbana-Champaign (UIUC), Urbana, Illinois, in 2001.

From 1997 to 2001, he was a research assistant at the Beckman Institute for Advanced Science and Technology at UIUC. During summer 1999 and 2000, he was a research intern with Microsoft Research, Redmond, Washington. In 2001, he joined the Department of Electrical and Computer Engineering at Northwestern University, Evanston, Illinois, as an assistant professor. He was promoted to associate professor in 2007 and full professor in 2012. He is currently full professor of Electrical Engineering and Computer Science at Northwestern University. His current research interests include computer vision, robotics, image and video analysis, pattern recognition, machine learning, multimedia data mining, and human-computer interaction.

He serves as associate editors for IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Image Processing, IEEE Transactions on Circuits and Systems for Video Technology, SPIE Journal of Electronic Imaging, and IAPR Journal of Machine Vision and Applications. He received the Robert T. Chien Award at UIUC in 2001, and the NSF CAREER award in 2003. He is a Fellow of the IEEE.