Order Preserving Sparse Coding

Bingbing Ni, Pierre Moulin, *Fellow, IEEE,* and Shuicheng Yan, *Senior Member, IEEE*

Abstract

In this paper, we investigate order-preserving sparse coding for classifying structured data whose atomic features possess ordering relationships. Examples include time sequences where individual frame-wise features are temporally ordered, as well as still images (landscape, street view, etc.) where different regions of the image are spatially ordered. Classification of these structured data is often tackled by first decomposing the input data into individual atomic features, then performing sparse coding or other processing for each atomic feature vector independently, and finally aggregating individual responses to classify the input data. However, this heuristic approach ignores the underlying order of the individual atomic features within the input data, and results in suboptimal discriminative capability. In this work, we introduce an order preserving regularizer which aims to preserve the ordering structure of the reconstruction coefficients within the sparse coding framework. An efficient Nesterov-type smooth approximation method is developed for optimization of the new regularization criterion, with theoretically guaranteed error bound. We perform extensive experiments for time series classification on a synthetic dataset, several machine learning benchmarks, and an RGB-D human activity dataset. We also report experiments for scene classification on a benchmark image dataset. The encoded representation is discriminative and robust, and our classifier outperforms state-of-the-art methods on these tasks.

Index Terms

sparse coding, order preserving, time sequence classification, scene classification

Part of this work was presented at ECCV 2012.

B. Ni is with the Advanced Digital Sciences Center, Singapore 138632. E-mail: bingbing.ni@adsc.com.sg.

P. Moulin is with the Department of Electrical and Computer Engineering, Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801.
E-mail: moulin@ifp.uiuc.edu.

S. Yan is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576. E-mail: eleyans@nus.edu.sg.

Manuscript received xx xx, 2013; revised xx xx, 2013.

I. INTRODUCTION

Sparse coding has been successfully used in various computer vision applications [1] [2] [3]. Using sparse coding for classification tasks presents two advantages: 1) the method is training-free, if the dictionary is constructed by simply sampling from the training data; and 2) the method provides flexibility in handling dynamic training settings, i.e., new training samples and new class labels. Sparse coding compactly represents an objects as a linear combination of a small number of elements in a dictionary.

Data are usually structured. For example, there may exist *group* structures in the data. To handle such structures, two alternative methods, Elastic Net [4] and group Lasso [5] have been proposed. They favor the selection of a small number of groups of correlated dictionary samples to represent the testing data. Another example is when data are described by several types of features (modalities). In object recognition, we may extract K types of image representations from K different feature types associated with the same visual input. By minimizing the sum of ℓ_2 norms of the blocks of coefficients associated with each covariate group across different feature representations, we favor similar sparsity patterns in all modalities [6]. In [7], the objective of joint sparsity is achieved by imposing an $\ell_{2,1}$ mixed-norm penalty on the reconstruction coefficients.

In this work, we are interested in a different problem setting, i.e., to deal with a different type of structured data, under the sparse coding framework: the input is a sequence of feature vectors instead of a single feature vector, and there exist dependencies among the input feature vectors. An example is a multidimensional time series such as audio data, which admits a natural **temporal ordering** structure across instantaneous feature vectors at successive time stamps. Another example is a still image (in this work, still images include landscapes, street views, etc., but not general objects), which is usually composed of regions with underlying spatial arrangement (**spatial ordering** structure) rules. For example, the sky region should always be above the sea region, and the street region should always be between buildings on both sides. For time sequence classification, a classical solution is to apply sparse coding to the feature vector in each input frame (i.e., time stamp) individually, and then compute and aggregate reconstruction coefficients for individual frames over the entire audio sequence for classification [8]. However, the temporal structure of the sequence conveys discriminative information. Treating each frame independently would discard this important information. Similarly, for scene classification, treating individual image regions independently and ignoring their underlying spatial ordering property would also discard information.

The objective of this work is therefore to explore a novel encoding method which can not only

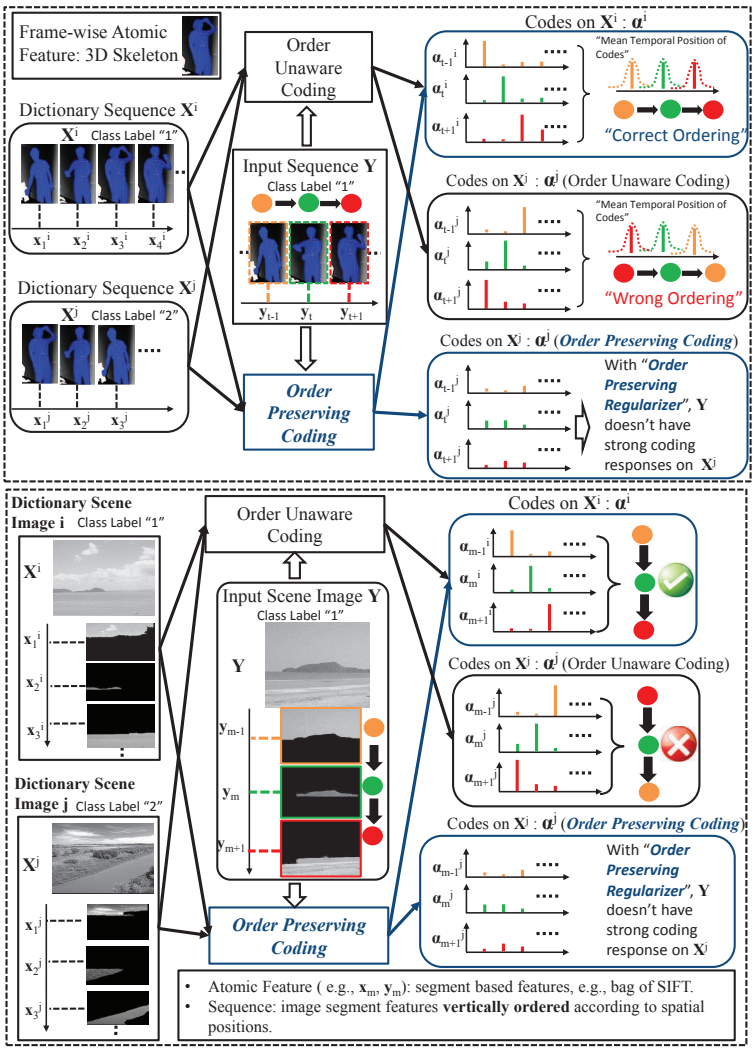


Fig. 1. Application of order preserving sparse coding to human motion sequence classification (upper panel) and still image classification (lower panel). Here still images include landscape and street view images. For human motion sequence classification, we extract 3D skeleton features for each frame, denoted by $\mathbf{x}_t$ for dictionary sample and $\mathbf{y}_t$ for testing sample. The frame-wise features within each sequence are ordered temporally. For still image classification, we first segment each image into regions. Then for each region, we extract a frame-based feature vector, e.g., color histogram or bag of SIFT features, denoted by $\mathbf{x}_m$ for dictionary sample and $\mathbf{y}_m$ for testing sample. These region-based features are then ordered according to the vertical position of the region centers in a sequence. Coding coefficients for different frame/segment-wise features are color-coded on the right. The mean squared reconstruction residual is used as part of the classification criterion. Note that performing sparse coding for individual features independently makes input sequence $\mathbf{Y}$ have strong coding coefficients on dictionary sequence $\mathbf{X}^j$, which leads to misclassification. In contrast, our order preserving coding will have weak response from $\mathbf{X}^j$ because $\mathbf{X}^j$ has different ordering structure than $\mathbf{Y}$ (even though the atomic feature vectors from both $\mathbf{Y}$ and $\mathbf{X}^j$ are similar).

encode structured data (e.g., time sequence and still image) discriminatively but also preserve the ordering structure of the input data. The main idea of our work is illustrated in Figure 1, which shows **why order preservation is important**. Figure 1 shows two examples of structured (ordered) data: a $3D$-skeleton feature time sequence (top panel) and an image segment-based feature sequence in the vertically ordered way (bottom panel). For each example, there is an input sequence $\mathbf{Y}$ and two dictionary sequences $\mathbf{X}^i$ and $\mathbf{X}^j$. In both examples, $\mathbf{X}^i$ and $\mathbf{X}^j$ contain similar frame/segment-wise features; however, these individual features are ordered in different ways. Therefore, the class labels for both dictionary sequences are different. The input sequence $\mathbf{Y}$ has the same class label as dictionary sequence $\mathbf{X}^i$. Moreover, $\mathbf{Y}$ and $\mathbf{X}^i$ have similar frame-wise features with similar temporal or spatial ordering structures. The reconstruction (coding) coefficients are shown on the right. If we ignore the ordering structure of the input sequence and perform sparse coding for each frame/segment-wise feature independently, individual input frame-wise features will receive strong response from similar dictionary items. In this sense, even if the class labels of $\mathbf{X}^j$ and $\mathbf{Y}$ are different, $\mathbf{Y}$ still receives strong reconstruction (coding) coefficients from $\mathbf{X}^j$ since their individual elements are very similar. As a consequence, $\mathbf{Y}$ will be most probably classified into the same class as $\mathbf{X}^j$. Therefore, we need a regularization mechanism which can discourage strong coding coefficients on those dictionary sequences whose ordering structures are different from those of the input sequence (even if their individual elements are similar).

However, how to utilize the ordering information for structured data has been largely ignored in current sparse coding literature. Previously designed regularization schemes [4], [5], [6], [7] cannot be directly applied to address the order preserving problem. A smoothness regularizer (e.g., total variation) is sometimes paired with a sparse regularizer such that correlated features in the input space map to similar coefficients. An example is fused lasso [39], where the reconstruction coefficients learned for highly correlated features have similar values. Another example is Zhao et al. [21], which applied a weighted smoothness regularizer for predicting unusual local video events. Since similar motion features at neighboring patches are more likely to be involved in a usual event, it is reasonable to enforce that similar reconstruction weight vectors be assigned to neighboring cuboids. However, *smooth codes* do not necessarily correspond to *order-preserving codes*.

Our order preserving regularizer penalizes the misfit of the temporal or spatial order of the reconstruction coefficients for individual atomic features with respect to the underlying order of the atomic features within the input data. The resulting optimization problem is convex but non-smooth. We develop an efficient Nesterov-type smooth approximation method [9] for optimization with theoretically guaranteed error bound. To the best of our knowledge, this is the first attempt to address this ordering consistency

issue for sparse coding. The classical smooth regularization framework encourages smoothness of the target function, but this does not guarantee that the reconstruction coefficients for the test sequence follow the same ordering pattern as in the dictionary sequence. Our method inherits the discriminative capability of conventional sparse coding and further boosts this capability by explicitly encoding temporal or spatial ordering structures.

The rest of this paper is organized as follows. Section II discusses related work on time series and still image classification. Section III presents the problem formulation and our optimization procedure. Extensive experimental results and discussions are presented in Section IV. Section V concludes the paper.

II. Related Work

In this section, we discuss the disadvantages of previous attempts to utilize the ordering structure information and then the motivations our approach.

A. Temporally-Ordered Structured Data

Time series classification is an active research topic. Hidden Markov models (HMMs) [10] is the most popular approach to sequence classification. A segmental hidden Markov model (HMM) was recently used to characterize the waveform shape for recognition [11]. Dynamic Time Warping (DTW) [12] is also widely used for time series classification. Rodrigues et al. [13] proposed a DTW decision tree method for time series classification. Hayashi et al. [14] used DTW distances to embed time series into a lower dimensional space by Laplacian eigenmap. The computational burden of DTW-based methods is generally high. Xi et al. [15] proposed numerosity reduction to accelerate nearest-neighbor DTW. Another tool for sequence classification is Recurrent Neural Networks (RNN) [16], which is a modification of a general Neural Network architecture that considers the temporal structure. However, training the model using back propagation is also complex and slow [16]. In contrast, our method uses fixed dictionary sequences which are sampled from the training data and thus it requires no dictionary learning. Owing to an efficient approximate optimization scheme, the testing time complexity of our method is comparable with the above mentioned sequence classification methods. The *n-gram* model [18] is also used for sequence modeling for speech recognition; however, it lacks explicit representation of long range dependency, and it is very sensitive to temporal scaling. In contrast, our proposed regularization scheme does not have these limitations. Although some other works [19] [20] [21] also study sparse coding for temporally varying signals (events, images), their regularization framework only uses traditional temporal smoothness

constraints. None of these works directly considers temporal ordering information. Our method inherits the good discriminative capability of sparse coding and also preserves the ordering structure of the input data. Therefore, it achieves better classification performance.

B. Spatially-Ordered Structured Data

The bag-of-words (BoW) model has been widely used in still image classification due to its good accuracy, simplicity and robustness to image variations. In this work, still images mean images with strong spatial ordering structure such as landscape, street view etc. However general images are not considered here. Each visual feature (e.g., SIFT [37]) extracted from the image is assigned to a visual feature prototype (i.e., visual word) by nearest neighbor searching, and then a histogram representation is computed by counting the occurrence of each visual word. As the BoW model cannot encode spatial information, Lazebnik *et al.* [33] extended the BoW model with Spatial Pyramid Matching Kernel (SPM) by exploiting the spatial information of location regions. Yang *et al.* [35] proposed an extension of SPM by using Sparse Coding (ScSPM), which achieved the state-of-the-art performance in image classification. By replacing k-means with sparse coding, the accuracy of the quantization process is improved. Object positions and scales in the image usually have large variations and therefore the fixed spatial pyramid feature pooling approach is not always optimal. To address this issue, Sadeghi and Tappen [29] proposed a latent pyramidal regions (LPR) representation for scenery image classification and all possible sub-windows of the images are fed into a latent SVM training procedure to make the classification robust to different spatial configurations of images. Yan et al. [32] introduced a two-level feature extraction and pooling framework for image classification that includes a comprehensive set of windows densely sampled over location, size and aspect ratio. To explore the dependence information among nearby local features, Gao et al. [28] proposed a Laplacian sparse coding method by taking into account the correlation among the local features. The kNN method is used to construct a Laplacian matrix that characterizes the similarity of local features. This Laplacian matrix is incorporated into the objective function of sparse coding to preserve the consistence in sparse representation of similar local features. Recently, Balasubramanian et al. [30] proposed a smooth sparse coding framework based on kernel-smoothing for incorporating feature similarity information between the samples into sparse coding.

These works only consider the object position/scale variations in the images and the local consistency between neighboring visual features in feature coding. The important spatial ordering structures between elements (segments) in still images were not considered.

III. Order Preserving Sparse Coding

A. Notation

We denote scalars and vectors by lower case roman and bold fonts respectively, and matrices by upper case letters. The input is a sequence of ordered atomic feature vectors $Y = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_t]$ where each $\mathbf{y}_i$ is a D-dimensional feature vector, and t denotes the length of the sequence. For a time sequence, i denotes the temporal index and $\mathbf{y}_i$ is the atomic feature vector extracted from the i-th frame. For a still image, we first segment an image into t regions (segments). We extract an atomic feature vector $\mathbf{y}_i$ from each region (e.g., color histogram or bag of SIFT features). Then Y is ordered according to the spatial coordinates of the centers of these regions (patches, image segments). We order atomic feature vectors extracted from all regions either horizontally according to the x-coordinate or vertically according to the y-coordinate. We are given a basis dictionary denoted by $\mathcal{X} = \{(X_1, y_1), (X_2, y_2), \cdots, (X_S, y_S)\}$, where S is the number of dictionary sequences. Each sequence X_j is also an ordered atomic feature vector sequence represented by $X_j = [\mathbf{x}_1^j, \mathbf{x}_2^j, \cdots, \mathbf{x}_{t_j}^j]$, where t_j is the length of the j-th sequence (i.e., number of time stamps in a time sequence or number of image patches/segments in an image) and y_j is the corresponding class label. Each $\mathbf{x}_i^j$ is a D-dimensional feature vector normalized to be of unit ℓ_2 norm. Note that t_j may vary from sequence to sequence. The class label y_j has $\mathcal{C}$ possible values $\{1, 2, \cdots, \mathcal{C}\}$. For example, in human action classification we have $\mathcal{C}$ categories for actions such as walking, running, and standing. We stack all the sequences (i.e., for each sequence the temporal order or horizontal/vertical spatial order within the sequence is retained) in the dictionary one by one, and represent the dictionary by a $D \times N$ matrix $X = [X_1, X_2, \cdots, X_S]$, where $N = \sum_{j=1}^{S} t_j$ is the total number of feature vectors. We denote by $\boldsymbol{\alpha}_i$ the N-dimensional vector of reconstruction coefficients for the input vector $\mathbf{y}_i$, which is a linear combination of all dictionary entries. Let $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1; \boldsymbol{\alpha}_2; \cdots; \boldsymbol{\alpha}_t)$, and denote by $\boldsymbol{\alpha}_i^j$ the t_j-dimensional reconstruction coefficients for input vector i from the dictionary sequence j. Using this convention, we further denote by $\boldsymbol{\alpha}^j = (\boldsymbol{\alpha}_1^j; \boldsymbol{\alpha}_2^j; \cdots; \boldsymbol{\alpha}_t^j)$ the reconstruction coefficients from the t_j feature vectors of the dictionary sequence j.

B. Order Preserving Regularizer

A straightforward way to represent an input sequence within the sparse coding framework is to first decompose the input sequence into atomic feature representations, e.g., frame-based feature vectors at individual time stamps for time sequence or region-based feature vectors for individual image regions/patches for still image. Then sparse coding is performed for each atomic feature representation vector independently. Finally individual responses (reconstruction coefficients) are aggregated for

classification. However, this approach is suboptimal because these types of input data have very strong ordering structures across individual atomic feature vectors. Our proposed regularization scheme explicitly addresses this problem and is formulated as

$$F(\boldsymbol{\alpha}) = f(\boldsymbol{\alpha}) + \lambda_1 G(\boldsymbol{\alpha}) + \lambda_2 P(\boldsymbol{\alpha}), \tag{1}$$

where $f(\boldsymbol{\alpha})$, $G(\boldsymbol{\alpha})$ and $P(\boldsymbol{\alpha})$ represent the least squared data fitting term, the sparsity term, and the order preserving regularization term, respectively, and λ_1 and λ_2 are the corresponding weighting factors.

We first require that the reconstruction coefficients for all individual feature vectors of the input sequence be nonzero on only a few dictionary sequences. As the input sequence normally matches only a few dictionary sequences, spreading the reconstruction coefficients throughout all dictionary sequences degrades the representation discriminative capability. To this end, we adopt an $\ell_{2,1}$ norm group-sparsity regularizer as in multitask joint sparse coding [7]. Thus the first term of our regularization criterion is

$$G(\boldsymbol{\alpha}) = \sum_{j=1}^{S} \|\boldsymbol{\alpha}^j\|_2. \tag{2}$$

The second term of our regularization criterion is the order preserving regularizer. For a time sequence, the goal is to prevent the reconstruction coefficients of the input feature vector i (i.e., at the time stamp i of the input sequence) of the j-th dictionary sequence from being *behind* those of the input vector $i+1$ (at the time stamp $i+1$ of the input sequence). Similarly, for a still image, the goal is to prevent the reconstruction coefficients of the input feature vector i (i.e., the i-th patch-based feature vector in the horizontally (left to right) or vertically (top to bottom) ordered image patch feature sequence) of the j-th dictionary sequence from being *on the right of* or *below* those of the input vector $i+1$. In other words, the nonzero reconstruction coefficients for individual feature vectors obey the same order as the corresponding feature vectors in the input sequence. To encourage such ordering, we consider the expression

$$(\mathbf{w}_j^T(\boldsymbol{\alpha}_i^j - \boldsymbol{\alpha}_{i+1}^j))_+ \triangleq \max\{\mathbf{w}_j^T(\boldsymbol{\alpha}_i^j - \boldsymbol{\alpha}_{i+1}^j), 0\}, \tag{3}$$

where $\mathbf{w}_j$ is a vector that has the same length as the corresponding dictionary sequence j and is element-wise increasing. $(.)_+$ denotes $\max(., 0)$. Thus, $\mathbf{w}_j(i) < \mathbf{w}_j(i'), \forall i < i'$. The dot product $\mathbf{w}_j^T \boldsymbol{\alpha}_i^j$ approximates the temporal/spatial ordering position of the responses for the i-th input vector $\mathbf{y}_i$ from dictionary sequence j. When the sum of all entries in $\boldsymbol{\alpha}_i^j$ is one, the sum $\mathbf{w}_j^T \boldsymbol{\alpha}_i^j$ can be considered as the approximated ordering position. Otherwise $\mathbf{w}_j^T \boldsymbol{\alpha}_i^j$ can be considered as the importance weighted version of the approximated ordering position. Therefore $\mathbf{w}_j^T \boldsymbol{\alpha}_i^j > \mathbf{w}_j^T \boldsymbol{\alpha}_{i+1}^j$ means that the approximated ordering position of the response for the $(i+1)$-th input vector on dictionary sequence j precedes that

of the i-th input vector, which is the case to penalize. We call $\mathbf{w}_j$ an ordering-prior-multiplier, and in this work we choose the following simple choice for $\mathbf{w}_j$, namely element-wise linear:

$$\mathbf{w}_j = \left(\frac{1}{t_j}, \frac{2}{t_j}, \cdots, \frac{t_j - 1}{t_j}, 1\right)^T. \tag{4}$$

Recall that t_j denotes the length (number of atomic features) of sequence j. Note that the ordering-prior-multiplier reflects the increase of time stamp or the increase of spatial position index from left to right (top to bottom). The **order preserving regularization** term is obtained by summing (3) over all consecutive elements of the input sequence and over all dictionary sequences, namely,

$$P(\boldsymbol{\alpha}) = \sum_{i=1}^{t-1} \sum_{j=1}^{S} \max(\mathbf{w}_j^T \boldsymbol{\alpha}_i^j - \mathbf{w}_j^T \boldsymbol{\alpha}_{i+1}^j, 0). \tag{5}$$

An illustration of the effect of the regularizer (5) in the case of time sequence classification (e.g., human action classification) is given in Figure 2. The input is a $3D$-skeleton feature sequence with three frames, $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3]$. There are two dictionary sequences $\mathbf{X}^i$ and $\mathbf{X}^j$. We see that $\mathbf{X}^i$ and $\mathbf{X}^j$ contain the same frame-wise features. However, their frame-wise features are temporally ordered in different ways. Therefore the action labels for both dictionary sequences are different. The input sequence $\mathbf{Y}$ has the same action label as dictionary sequence $\mathbf{X}^i$, i.e., the same frame-wise features in the same temporal order. The right side of Figure 2 shows the reconstruction coefficients for all frame-wise features on dictionary sequences $\mathbf{X}^i$ (upper) and $\mathbf{X}^j$ (bottom). The reconstruction coefficients corresponding to different frame-wise features are color-coded. For the reconstruction coefficients on $\mathbf{X}^i$, we see that the mean approximate temporal position of the reconstruction coefficients follows the input ordering structure, thus $\mathbf{w}_i^T \boldsymbol{\alpha}_t^i - \mathbf{w}_i^T \boldsymbol{\alpha}_{t+1}^i < 0$ for $t = 1, 2$ and the penalty of our regularizer is zero, i.e., $\sum_{t=1}^{2} \max(\mathbf{w}_i^T \boldsymbol{\alpha}_t^i - \mathbf{w}_i^T \boldsymbol{\alpha}_{t+1}^i, 0) = 0$. For the reconstruction coefficients on $\mathbf{X}^j$, while individual input frame-wise features receive strong response from similar items in the dictionary, there is a strong ordering difference between $\mathbf{Y}$ and $\mathbf{X}^j$. This results in a penalty, i.e., $\sum_{t=1}^{2} \max(\mathbf{w}_j^T \boldsymbol{\alpha}_t^j - \mathbf{w}_j^T \boldsymbol{\alpha}_{t+1}^j, 0) > 0$. In this case, the proposed regularizer will most likely attenuate the coding coefficients, which results in a small penalty value, i.e., $\sum_{t=1}^{2} \max(\mathbf{w}_j^T \boldsymbol{\alpha}_t^j - \mathbf{w}_j^T \boldsymbol{\alpha}_{t+1}^j, 0) \approx 0$. Figure 3 shows the coding effects using different regularization schemes including: 1) sparse regularizer, 2) sparse and smooth regularizer (e.g., fused lasso [39] of the form $\sum_{j=1}^{S} \|\boldsymbol{\alpha}^j\|_1 + \lambda \sum_{j=1}^{S} \sum_{i=1}^{t-1} \|\boldsymbol{\alpha}_i^j - \boldsymbol{\alpha}_{i+1}^j\|_1$), and 3) our order preserving regularizer. We see that the sparse regularizer simply yields sparse reconstruction coefficients. The sparse and smooth regularizer favors similar reconstruction values for temporally nearby input features. However, these regularizers cannot enforce the order preserving property. In contrast, our regularization scheme achieves that goal. We note from the mathematical formulation that our proposed order preserving regularizer seems

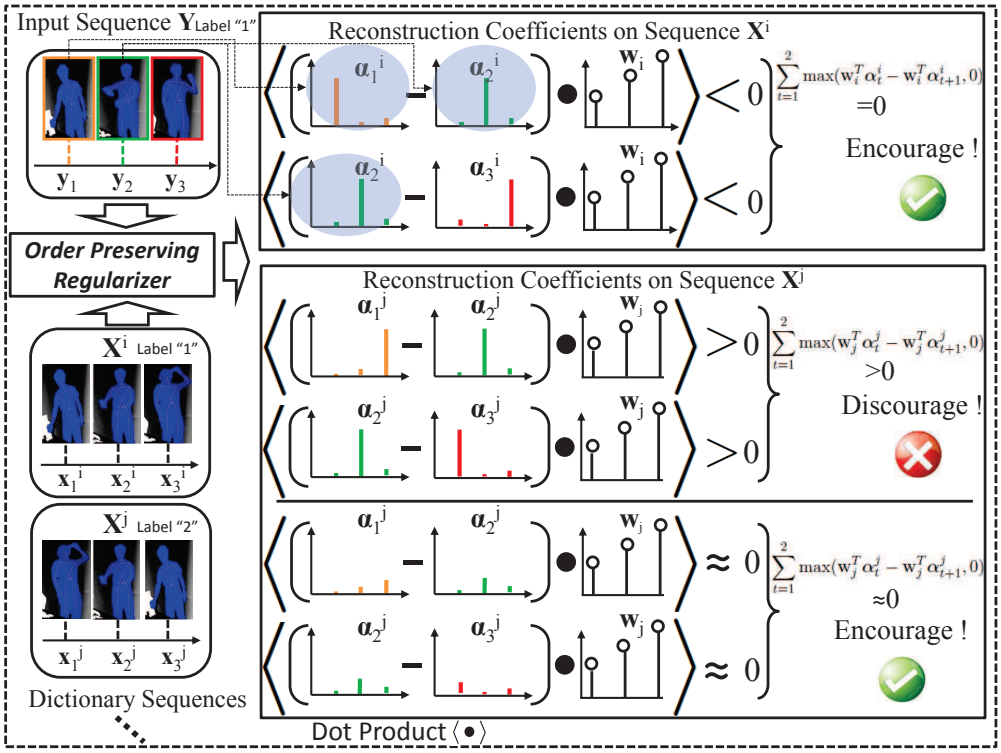


Fig. 2. Application of the regularizer (5) to human action sequence classification. The reconstruction coefficients that follow the temporal order of the input sequence are not penalized by the order-preserving regularizer (upper row). Those do not follow the ordering structure of the input sequence are penalized (middle row). In this case, the proposed regularizer will weaken the coding response on the dictionary sequence $\mathbf{X}^j$ which has a different ordering structure than the input sequence $\mathbf{Y}$ (bottom row).

like an *one-sided* version of the fused lasso regularizer. However, the sparse and smooth regularizer (e.g., fused lasso) is not able to force the reconstruction coefficients of the input feature vector at frame i to appear before those of the $i+1$-th frame. Therefore, it cannot enforce the order preserving property.

Discussion: 1) The proposed regularizer preserves the ordering structure approximately. When the sum of all entries in $\boldsymbol{\alpha}_i^j$ is one, the sum $\mathbf{w}_j^T \boldsymbol{\alpha}_i^j$ can be considered as the expected relative ordering position. When the sum of the entries in $\boldsymbol{\alpha}_i^j$ is not equal to one, $\mathbf{w}_j^T \boldsymbol{\alpha}_i^j$ can be considered as the importance weighted sum of $\boldsymbol{\alpha}_i^j$. Then $\mathbf{w}_j^T \boldsymbol{\alpha}_i^j / |\boldsymbol{\alpha}_i^j|$ can still be considered as the expected relative ordering position. If we constrain each element $\boldsymbol{\alpha}_i^j$ to take binary values 0 or 1 to enforce exactly matching a dictionary element, the optimization becomes infeasible (NP-hard). Regarding the trade-off between accuracy and efficiency, the proposed relaxation (approximation) of $\boldsymbol{\alpha}_i^j$ is quite standard in computer vision. 2) Although our approximation is based on the assumption that the learned reconstruction coefficients are non-negative,

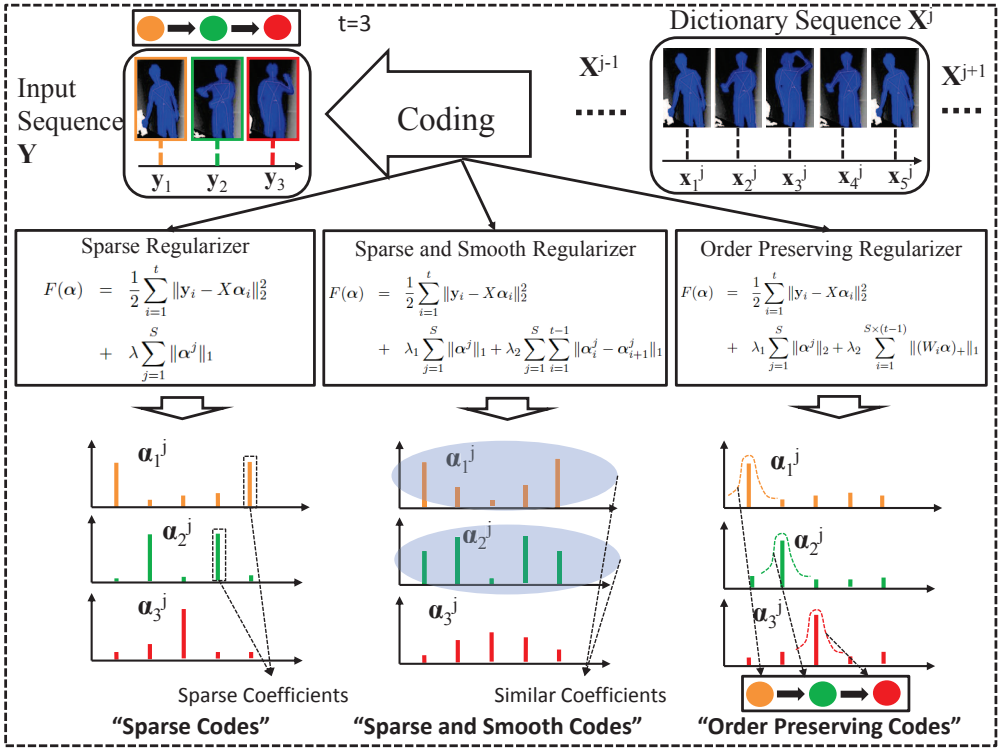


Fig. 3. Application of different types of regularizers. The input time sequence has three frames, i.e., $t = 3$. The reconstruction coefficients corresponding to the feature vectors from each input frame are color-coded.

we can regard that a negative value for the entry of $\boldsymbol{\alpha}_i^j$ means there is out-of-segment extrapolation. In the experimental part, we will empirically show that even if we do not explicitly enforce this non-negative constraint in our objective function formulation, most of the learned coefficients are non-negative and the few negative coefficients mostly do not affect the algorithmic performance. We will also provide a *positivity correction* version of our coding algorithm by setting the negative reconstruction coefficients to zero after each iteration.

C. Objective Function and Optimization

Here we formally state our regularization criterion. The reconstruction error term is

$$f(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{t} \|\mathbf{y}_i - X\boldsymbol{\alpha}_i\|_2^2, \tag{6}$$

which is a convex, smooth, differentiable function with Lipschitz constant $L_f = \|X^T X\|_F$, where $\|.\|_F$ denotes the Frobenius norm.

The group-sparsity regularizer of (2) is an $\ell_{2,1}$ mixed-norm, which is convex and nonsmooth. However, the norm $\|\boldsymbol{\alpha}^j\|_2$ can be written as

$$\|\boldsymbol{\alpha}^j\|_2 = \max_{\|\mathbf{v}_j\|_2 \leq 1} \langle \boldsymbol{\alpha}^j, \mathbf{v}_j \rangle. \tag{7}$$

We use the Nesterov smooth approximation method of [9] and approximate (7) by the smooth function

$$q_{\mu,j}(\boldsymbol{\alpha}^j) = \max_{\|\mathbf{v}_j\|_2 \leq 1} \left\{ \langle \boldsymbol{\alpha}^j, \mathbf{v}_j \rangle - \frac{1}{2}\mu \|\mathbf{v}_j\|_2^2 \right\}, \tag{8}$$

where μ is a parameter that controls the approximation accuracy. The unique maximizer of (8), denoted by $\mathbf{v}_j(\boldsymbol{\alpha}^j)$, is derived as

$$\mathbf{v}_j(\boldsymbol{\alpha}^j) = \begin{cases} \frac{\boldsymbol{\alpha}^j}{\mu}, & 0 \leq \|\boldsymbol{\alpha}^j\|_2 \leq \mu, \\ \frac{\boldsymbol{\alpha}^j}{\|\boldsymbol{\alpha}^j\|_2}, & \|\boldsymbol{\alpha}^j\|_2 > \mu. \end{cases} \tag{9}$$

The smooth approximation of (2) is therefore obtained as

$$G_\mu(\boldsymbol{\alpha}) = \sum_{j=1}^{S} q_{\mu,j}(\boldsymbol{\alpha}^j). \tag{10}$$

The order preserving regularization function is given by $P(\boldsymbol{\alpha})$ in (5). By simple manipulation, we rewrite $P(\boldsymbol{\alpha})$ more compactly as (we note that $W_i\boldsymbol{\alpha}$ is a scalar):

$$P(\boldsymbol{\alpha}) = \sum_{i=1}^{S\times(t-1)} \|(W_i\boldsymbol{\alpha})_+\|_1 = \sum_{i=1}^{S\times(t-1)} (W_i\boldsymbol{\alpha})_+, \tag{11}$$

Here each W_i, $i = 1, 2, \cdots, (t-1)\times S$, is an N-dimensional row vector given by

$$W_{(h-1)\times S+j} = (0, \cdots, 0, \mathbf{w}_j^T, 0, \cdots, 0, -\mathbf{w}_j^T, 0, \cdots, 0), \tag{12}$$

for $h = 1, 2, \cdots, t-1$, $j = 1, 2, \cdots, S$. The positions of $\mathbf{w}_j^T$ and $-\mathbf{w}_j^T$ correspond to those of $\boldsymbol{\alpha}_h^j$ and $\boldsymbol{\alpha}_{h+1}^j$ in $\boldsymbol{\alpha}$, respectively. As mentioned above, the regularizer (11) encourages order preserving for the reconstruction coefficients for individual feature vectors of the input sequence. The function $(W_i\boldsymbol{\alpha})_+$ is convex and nonsmooth. Moreover,

$$(W_i\boldsymbol{\alpha})_+ = \max_{0\leq v_i\leq 1} (W_i\boldsymbol{\alpha})v_i. \tag{13}$$

Using the Nesterov smooth approximation method again, we approximate (13) by the following smooth function

$$p_{\mu,i}(\boldsymbol{\alpha}) = \max_{0\leq v_i\leq 1} \left\{ (W_i\boldsymbol{\alpha})v_i - \frac{1}{2}\mu v_i^2 \right\}, \tag{14}$$

where μ is the parameter that controls the approximation accuracy. For fixed $\boldsymbol{\alpha}$, the unique maximizer of (14) is derived as

$$v_i(\boldsymbol{\alpha}) = \min\left\{1, \max\left(0, \frac{W_i\boldsymbol{\alpha}}{\mu}\right)\right\}. \tag{15}$$

We then construct a Nesterov-type smooth approximation of $P(\boldsymbol{\alpha})$ as

$$\begin{aligned} P_\mu(\boldsymbol{\alpha}) &= \sum_{i=1}^{S\times(t-1)} p_{\mu,i}(\boldsymbol{\alpha}), \\ &= \max_{\|\mathbf{v}\|_\infty\leq 1,\ \mathbf{v}\succeq 0} \left\{ \langle W\boldsymbol{\alpha}, \mathbf{v}\rangle - \frac{1}{2}\mu\|\mathbf{v}\|_2^2 \right\}, \end{aligned} \tag{16}$$

where the matrix W is given by $W = (W_1; \cdots; W_{(t-1)\times S})$, and the vector $\mathbf{v}$ by $\mathbf{v} = (v_1; \cdots; v_{(t-1)\times S})$.

The *nonsmoothed* objective function is defined as

$$F(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{t} \|\mathbf{y}_i - X\boldsymbol{\alpha}_i\|_2^2 + \lambda_1 \sum_{j=1}^{S} \|\boldsymbol{\alpha}^j\|_2 + \lambda_2 \sum_{i=1}^{S\times(t-1)} (W_i\boldsymbol{\alpha})_+, \tag{17}$$

for problems with one-dimensional ordering structure, e.g., a time sequence with temporal ordering.

A still image has a multi-dimensional ordering structure. That is, the segments of the image have spatial ordering in both horizontal and vertical directions. We extend (17) to incorporate order-preserving regularizers for both directions as

$$\begin{aligned} F_s(\boldsymbol{\alpha}, \boldsymbol{\alpha}') &= \frac{1}{2}\sum_{i=1}^{t}(\|\mathbf{y}_i - X\boldsymbol{\alpha}_i\|_2^2 + \|\mathbf{y}'_i - X'\boldsymbol{\alpha}'_i\|_2^2) \\ &+ \lambda_1 \sum_{j=1}^{S}(\|\boldsymbol{\alpha}^j\|_2 + \|\boldsymbol{\alpha}'^j\|_2) + \lambda_2 \sum_{i=1}^{S\times(t-1)} ((W_i\boldsymbol{\alpha})_+ + (W_i\boldsymbol{\alpha}')_+). \end{aligned} \tag{18}$$

Here, $Y = [\mathbf{y}_1, \cdots, \mathbf{y}_t]$ and $Y' = [\mathbf{y}'_1, \cdots, \mathbf{y}'_t]$ denote the input sequence which is horizontally and vertically ordered. Namely, the image is first segmented into t segments (patches) and we extract a patch-based feature (atomic feature representation vector $\mathbf{y}_i$ or $\mathbf{y}'_i$) for each patch. We then order $\{\mathbf{y}_i\}$ and $\{\mathbf{y}'_i\}$ into horizontally and vertically ordered sequences Y and Y' based on the image coordinates of the centers of these patches, i.e., the image patch corresponding to $\mathbf{y}_i$ is spatially situated on the left of the patch corresponding to $\mathbf{y}_{i+1}$. Accordingly, X and X' denote the dictionary sequences which are horizontally or vertically ordered. $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ are the corresponding reconstruction (coding) coefficients for preserving the horizontal and vertical spatial orders, respectively.

The above objective functions are convex but nonsmooth. According to (10) and (16), $F(\boldsymbol{\alpha})$ can be approximated by the convex and smooth function

$$F_\mu(\boldsymbol{\alpha}) = f(\boldsymbol{\alpha}) + \lambda_1 G_\mu(\boldsymbol{\alpha}) + \lambda_2 P_\mu(\boldsymbol{\alpha}). \tag{19}$$

We adopt (19) as our regularization criterion. The gradient of $F_\mu(\boldsymbol{\alpha})$ is $\nabla F_\mu(\boldsymbol{\alpha}) = \nabla f(\boldsymbol{\alpha}) + \lambda_1 \nabla G_\mu(\boldsymbol{\alpha}) + \lambda_2 \nabla P_\mu(\boldsymbol{\alpha})$, and the corresponding Lipschitz constant is $L_{F_\mu} = L_f + \lambda_1 L_{G_\mu} + \lambda_2 L_{P_\mu}$.

In the following, we introduce the approximate optimization technique for $F(\boldsymbol{\alpha})$. The optimization method for $F_s(\boldsymbol{\alpha}, \boldsymbol{\alpha}')$ is similar. To minimize $F_\mu(\boldsymbol{\alpha})$, we use the efficient accelerated proximal gradient

(APG) method [22], which has rate of convergence $O(1/n^2)$, where n is the iteration number. In terms of the desired residue ϵ, i.e., $|F_\mu - \min F_\mu| \leq \epsilon$, by choosing $\mu \approx \epsilon$ we have the rate of convergence $O(1/\epsilon)$. Algorithm 1 shows the optimization procedure. The regularization parameters λ_1 and λ_2 are chosen by cross-validation on a validation subset sampled from the training data. In particular, we consider values of λ_1 and λ_2 in the set $\{0.001, 0.01, 0.1, 1.0, 10, 100, 1000\}$. The testing time complexity of our order preserving sparse coding method (MTO-SC) is $\mathcal{O}(n(\overline{t}^2 SD + \overline{t}^2 S^2))$, where $\overline{t}$ denotes the average length of sequence, S the number of training sequences and D the dimensionality of the frame-wise features. The gradient computation step in Algorithm 1 could be implemented using parallel computing, e.g., GPU. Because the gradient of $F_\mu(\boldsymbol{\alpha})$ is a summation over the frame index of the testing sequence, this would further accelerate the gradient computation by a factor of $\overline{t}$.

Algorithm 1: Optimization procedure for (19)

Inputs : $X \in \mathbb{R}^{D\times N}$, $\{W_i \in \mathbb{R}^{1\times N}, i = 1, 2, \cdots, S(t-1)\}$, λ_1, λ_2, μ, $\{\mathbf{y}_i, i = 1, \cdots, t\}$.

Output: $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1; \boldsymbol{\alpha}_2; \cdots; \boldsymbol{\alpha}_t) \in \mathbb{R}^{tN}$.

Initialization: Calculate $L_{F_\mu} = L_f + \lambda_1 L_{G_\mu} + \lambda_2 L_{P_\mu}$. Initialize $\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0 \in \mathbb{R}^{tN}$ to be zero vectors, and let $\gamma_0 = 0$, $k = 0$.

repeat

- $\boldsymbol{u}_k = (1 - \gamma_k)\boldsymbol{\alpha}_k + \gamma_k \boldsymbol{\beta}_k$,
- Calculate the gradient $\nabla F_\mu(\boldsymbol{u}_k)$.
- $\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k - \frac{1}{\gamma_k L_{F_\mu}} \nabla F_\mu(\boldsymbol{u}_k)$,
- $\boldsymbol{\alpha}_{k+1} = (1 - \gamma_k)\boldsymbol{\alpha}_k + \gamma_k \boldsymbol{\beta}_{k+1}$,
- $\gamma_{k+1} = \frac{2}{k+1}$, $k \leftarrow k + 1$.

until *convergence*;

D. Analysis

Here, we analyze the approximations (10) and (16) and bound the approximation errors.

Proposition 1. *$G_\mu(\boldsymbol{\alpha})$ is a μ-accurate approximation to $G(\boldsymbol{\alpha})$, that is*

$$G_\mu(\boldsymbol{\alpha}) \leq G(\boldsymbol{\alpha}) \leq G_\mu(\boldsymbol{\alpha}) + \frac{1}{2}\mu S. \tag{20}$$

This result is a special case of Nesterov's approximation theorem [9] [Equation 2.7]. See proof in Appendix A in the supplemental material.

Theorem 1. *The function $G_\mu(\boldsymbol{\alpha})$ is convex and continuously differentiable. Moreover, its gradient $\nabla G_\mu(\boldsymbol{\alpha}) = \sum_{j=1}^{S} \mathbf{v}_j(\boldsymbol{\alpha}^j)$ is Lipschitz continuous with constant $L_{G_\mu} = \frac{tN}{\mu}$.*

This result is a special case of Nesterov's approximation theorem [9] [Theorem 1]. See proof in Appendix B in the supplemental material.

Proposition 2. *$P_\mu(\boldsymbol{\alpha})$ is a μ-accurate approximation to $P(\boldsymbol{\alpha})$, that is*

$$P_\mu(\boldsymbol{\alpha}) \leq P(\boldsymbol{\alpha}) \leq P_\mu(\boldsymbol{\alpha}) + \frac{1}{2}\mu S(t-1). \tag{21}$$

This result is a special case of Nesterov's approximation theorem [9] [Equation 2.7]. See proof in Appendix C in the supplemental material.

Theorem 2. *The function $P_\mu(\boldsymbol{\alpha})$ is convex and continuously differentiable. Moreover, its gradient $\nabla P_\mu(\boldsymbol{\alpha}) = \sum_{i=1}^{S(t-1)} W_i^T v_i(\boldsymbol{\alpha})$ is Lipschitz continuous with constant $L_{P_\mu} = \frac{1}{\mu}\sum_{i=1}^{S(t-1)} \|W_i\|_2^2$.*

This result is a special case of Nesterov's approximation theorem [9] [Theorem 1]. See proof in Appendix D in the supplemental material.

E. Out-of-Sample Classification Rules

For unseen testing data such as a time sequence or a still (e.g., scenery) image, we extract frame-wise or segment-wise feature vectors and arrange them in a temporally/vertically/horizontally ordered way. We then calculate the reconstruction coefficients on the dictionary sequences using Algorithm 1. We use the same classification criterion as in [7]. The input sequence will most likely receive strong coding responses (reconstruction coefficients) from those closely related dictionary sequences (i.e., of the same class), and the reconstruction coefficients on the unrelated dictionary sequences will be likely smaller. Therefore if we reconstruct the input sequence using only one dictionary sequence (or a subset of dictionary sequences grouped by their corresponding class labels), the reconstruction error on the related sequence (or group) will most likely be smaller than that of the unrelated sequence (or group). Thus we can use the reconstruction error to perform classification. The classification procedure is as follows.

Time Sequence Classification: We first arrange the dictionary sequences into groups according to the class labels of the sequences. We denote by $X^{(j)} = [X_{j_1}, X_{j_2}, \cdots]$ the set of dictionary sequences from the j-th class. Here X_{j_i} denotes the i-th sequence in $X^{(j)}$ that belongs to the j-th class, i.e., $j \in \{1, \cdots, \mathcal{C}\}$ assuming $\mathcal{C}$ classes. Let $\boldsymbol{\alpha}^{(j)}$ denote the corresponding reconstruction coefficients for $X^{(j)}$. One can approximate the input sequence Y by using only the optimal coefficients associated with the j-th class

as $X^{(j)}\boldsymbol{\alpha}^{(j)}$. Following [7], the predicted class label is the one with the lowest total reconstruction error:

$$j_{opt} = \arg\min_j \|Y - X^{(j)}\boldsymbol{\alpha}^{(j)}\|_F^2. \tag{22}$$

Still Image Classification: We extract region-based features from the segmented image regions and order them horizontally and vertically as sequences Y and Y' respectively. Similarly, we denote by $X^{(j)}$ and $X'^{(j)}$ the sets of horizontally and vertically ordered dictionary sequences from the j-th class and by $\boldsymbol{\alpha}^{(j)}$ and $\boldsymbol{\alpha}'^{(j)}$ the corresponding reconstruction coefficients for $X^{(j)}$ and $X'^{(j)}$, respectively. The predicted class label is the one with the lowest joint total reconstruction error:

$$j_{opt} = \arg\min_j \|Y - X^{(j)}\boldsymbol{\alpha}^{(j)}\|_F^2 + \|Y' - X'^{(j)}\boldsymbol{\alpha}'^{(j)}\|_F^2. \tag{23}$$

IV. Experiments

To evaluate the effectiveness of our proposed method, we conduct experiments on multidimensional time series classification on a synthetic dataset, three machine learning benchmarks, and a real-world RGB-D human activity dataset. We also test our algorithm on a scenery image classification dataset. We perform both qualitative and quantitative evaluations to demonstrate the effectiveness of the proposed method on order preserving encoding. We also show the discriminative capability of the proposed method by comparing it with other state-of-the-art algorithms for time sequence and scenery image classification.

A. Synthetic Dataset

Consider the following eight polynomial functions:

$$\begin{aligned} &f^1(i) = \frac{\pi}{4}, \quad f^2(i) = \frac{\pi}{5}, \quad f^3(i) = \frac{\pi}{10}, \quad f^4(i) = \frac{\pi}{10}i, \quad f^5(i) = \frac{\pi}{10}(4-i), \\ &f^6(i) = \frac{\pi}{10}(i-2)^2, \quad f^7(i) = \frac{2\pi}{5} - \frac{\pi}{10}(i-2)^2, \quad f^8(i) = \frac{\pi}{5}\left|\sin(\frac{\pi}{2}i)\right| + \frac{\pi}{10}. \end{aligned} \tag{24}$$

From these functions, we generate eight length-5 two-dimensional time series

$$\begin{aligned} X_j &= \begin{bmatrix} \sin(f^j(0)) & \sin(f^j(1)) & \cdots & \sin(f^j(4)) \\ \cos(f^j(0)) & \cos(f^j(1)) & \cdots & \cos(f^j(4)) \end{bmatrix}, \\ j &= 1, 2, \cdots, 8. \end{aligned} \tag{25}$$

For each sequence, we add independent Gaussian noise samples with zero mean and variance 0.2 at each time stamp. We select the fourth sequence from the dictionary as the test time series input, i.e., $Y = X_4$. Each dictionary sequence is labeled with a different class.

In our comparisons, we use a) sparse coding (SC) which computes reconstruction coefficients for each frame of the input sequence individually; b) multitask group sparse coding (Group-SC) [7]; and c) the

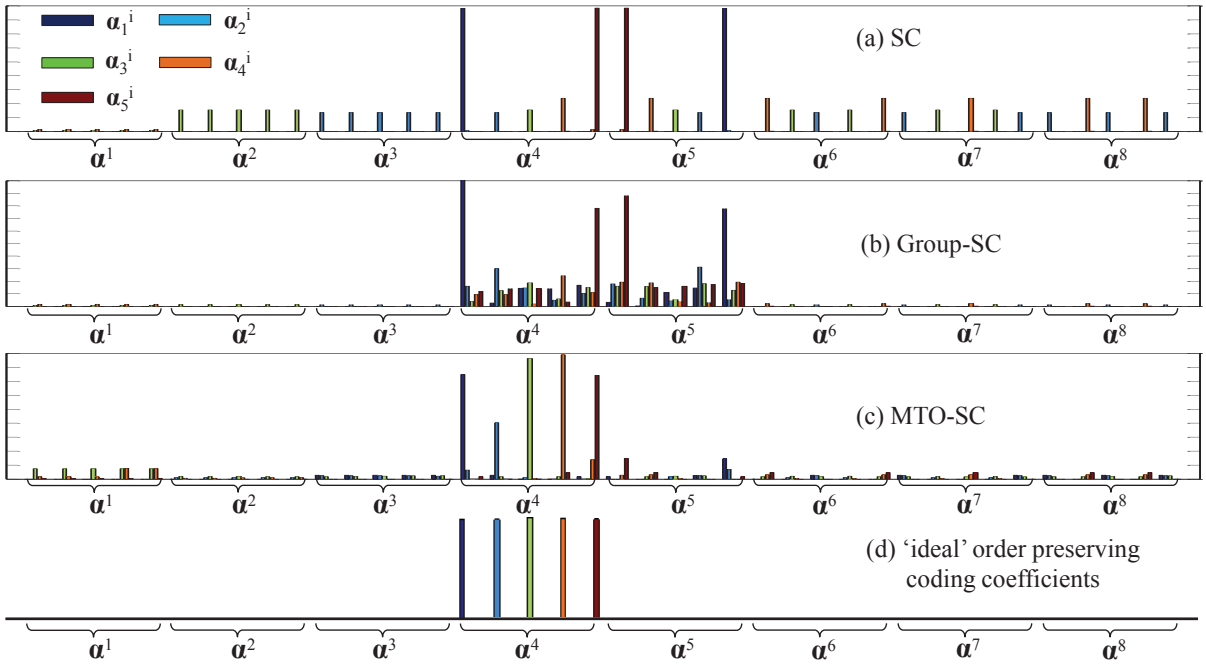


Fig. 4. Reconstruction coefficients of the input time series on the synthetic dataset using three algorithms: (a) sparse coding; (b) group sparse coding, and (c) Order-preserving sparse coding.

proposed order-preserving sparse coding method (MTO-SC). Reconstruction coefficients are shown in Figure 4. We observe that the reconstruction coefficients from sparse coding are similar among different dictionary sequences and therefore the representation is less discriminative. This is because sparse coding does not utilize any structure information of either the dictionary or the input sequence. Moreover, the reconstruction coefficients from multitask group sparse coding [7] are nonzero on few sequences. However, without enforcing the temporal order constraint, dictionary sequences with similar individual features but different ordering structures receive similar reconstruction coefficients, which results in ambiguities. Since our method explicitly encourages temporal order preservation, the reconstruction coefficients tend to follow the temporal ordering of the input sequence.

B. Machine Learning Benchmarks

We apply the proposed algorithm on three benchmark time series datasets:

- **UCI Australian Sign Language signs (High Quality) Dataset** [23]: It consists of 2565 samples of Auslan signs captured from 9 native signers using high-quality position trackers. The dataset contains 95 different signs, with 27 samples per sign. The average length of each sign is about 60

frames. Each frame is represented as a 15 dimensional feature vector consisting of hand position (X, Y, Z), roll, yaw, pitch, bend measurements of different fingers. For the ease of experiment, we randomly selected 4 subsets of the whole dataset with each subset containing 20 categories, denoted by AusLan1, AusLan2, AusLan3 and AusLan4, respectively.

- **UCI Spoken Arabic Digits Dataset** [24]: It contains time series of mel-frequency cepstrum coefficients (MFCCs) corresponding to spoken Arabic digits. The dataset includes data from 44 male and 44 female native Arabic speakers, capturing 8800 (10 digits $\times 10$ repetitions $\times 88$ speakers) time series of 13 Frequency Cepstral Coefficients (MFCCs). The average length of each sample is about 40 frames.
- **CMU Motion Capture Dataset (CMU MoCap)** [25]: We use the same subset as in [26] which includes 5 actions, i.e., jumping, golf swing, running, climbing and walking. The dataset contains 225 sequences with average length of 300 frames. Each frame is represented by rotation angles of 11 joints and end points including head, shoulders, elbows, hands, knees and feet.

We randomly split each dataset into training and testing sets with equal size. The random split is performed 10 times and all the reported testing results are averaged over the 10 random choices of the training and testing partition. For each frame, we normalize the feature vector to be of unit ℓ_2 norm. We compare our method (MTO-SC) (with testing time complexity $\mathcal{O}(n(\overline{t}^2SD + \overline{t}^2S^2))$) with the following state-of-the-art time series classification methods:

1) Segmental Hidden Markov Model [11] (S-HMM). Its testing time complexity is $\mathcal{O}(m^2\overline{t}^3SD)$, where m denotes the number hidden states, $\overline{t}$ the average length of sequences, S the number of training sequences, and D the dimensionality of the frame-wise features.
2) DTW-based decision tree method [13] (DTW-DT). Its testing time complexity is $\mathcal{O}(\overline{t}^2SD)$.
3) DTW-based distance embedding [14] (DTW-DE). Its testing time complexity is $\mathcal{O}(\overline{t}^2SD)$.
4) Numerosity reduction based DTW [15] (NR-DTW). It is an accelerated version of DTW.
5) Multi-resolution symbolic representation [17] (MSR). Its testing time complexity is $\mathcal{O}(\overline{t}DB + BS)$, where B denotes the codebook size.
6) Sparse coding performed for every individual input frame (SC) followed by majority voting. The coding uses an APG-based optimization method similar to that of our MTO-SC algorithm. Its testing time complexity is $\mathcal{O}(n\overline{t}^2SD)$.
7) Multitask group sparse coding [7] (Group-SC). Its testing time complexity is similar to SC.
8) Fused lasso method with the objective function $F(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{t}\|\mathbf{y}_i - X\boldsymbol{\alpha}_i\|_2^2 + \lambda_1\sum_{j=1}^{S}\|\boldsymbol{\alpha}^j\|_1 +$

$\lambda_2 \sum_{j=1}^{S} \sum_{i=1}^{t-1} \|\boldsymbol{\alpha}_i^j - \boldsymbol{\alpha}_{i+1}^j\|_1$ and APG-based optimization method similar to that of our MTO-SC algorithm. Its testing time complexity is $\mathcal{O}(n\bar{t}^2SD)$.

For all competing algorithms, their corresponding parameters (e.g., λ_1 and λ_2 in MTO-SC, number of hidden states for HMM and number of neighborhood samples for DTW, etc.) are set by cross-validation on a validation subset sampled from the training data. For our MTO-SC algorithm, we also implement a *positivity correction* version by setting the negative reconstruction coefficients to zero after each iteration of Algorithm 1. We denote this version of our algorithm by MTO-SC(PC).

We implemented these algorithms using Matlab 2010 on a 2.63 GHz machine with 8GB of memory. Figure 5 shows several typical convergence curves of Algorithm 1 on the AusLan1 dataset. We note that convergence is relatively fast. We empirically set the maximum number of iterations to 200, since for all experiments in this work the change in the objective function value becomes less than 1% after 200 iterations. The average testing time per sample on the AusLan1 dataset for S-HMM, NR-DTW (the

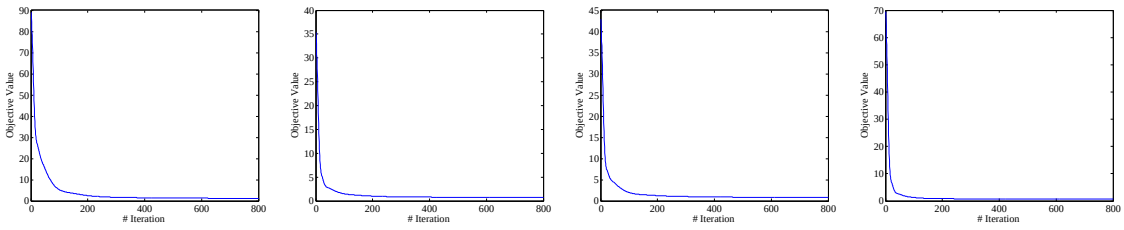


Fig. 5. Convergence curves of the MTO-SC algorithm on the AusLan1 dataset. From left to right: sample ID 10, 20, 30, and 40, respectively.

fastest DTW-based method among the three), MSR, SC, Group-SC, Fused Lasso and MTO-SC is 3.5, 1.5, 2.8, 2.0, 2.2, 2.3 and 2.4 seconds, respectively. We see that MTO-SC is among the most efficient methods in testing. For training, S-HMM, DTW-DT, DTW-DE, NR-DTW and MSR take about 4, 13, 3, 2.5 and 1.5 minutes, respectively, and the rest do not require training.

In Table I, we report the mean classification accuracies averaged over 10 random splits with standard deviations. We make two observations. First, MTO-SC achieves the highest classification accuracy, owing to its ability to encode the temporal ordering structure for time series classification. Second, the *positive correction* version MTO-SC(PC) has very similar classification performance to MTO-SC. Indeed, our experimental results show that about 90% learned coefficients by MTO-SC are non-negative. Therefore the few negative values of the reconstruction coefficients do not affect the coding algorithm too much.

TABLE I

CLASSIFICATION ACCURACY [MEAN(STD. DEV.)] FOR DIFFERENT ALGORITHMS ON DIFFERENT DATASETS.

Dataset	AusLan1	AusLan2	AusLan3	AusLan4	ArabSpokenDigit	CMU MoCap
S-HMM	75.46(1.39)	80.07(2.04)	82.30(1.77)	80.02(1.74)	54.30(1.37)	79.53(1.93)
DTW-DT	79.70(2.01)	82.73(1.98)	84.98(1.49)	83.76(1.91)	58.73(1.76)	82.97(1.64)
DTW-DE	79.93(1.49)	84.96(2.34)	85.41(1.84)	84.11(2.40)	59.60(2.40)	83.21(1.84)
NR-DTW	79.63(1.95)	85.43(1.91)	85.03(2.30)	83.09(2.21)	57.32(1.98)	83.60(1.28)
MSR	80.33(2.20)	89.77(1.35)	87.96(1.90)	85.98(1.90)	64.90(2.45)	85.36(1.92)
SC	83.09(1.89)	90.46(1.09)	90.00(1.66)	86.78(1.72)	45.98(2.10)	84.00(1.57)
Group-SC	84.19(1.76)	89.97(1.12)	91.50(1.89)	87.18(1.92)	46.04(1.99)	84.56(1.73)
Fused Lasso	86.57(2.05)	89.09(1.74)	92.31(1.69)	90.55(2.33)	62.74(2.85)	86.96(2.10)
MTO-SC	$\mathbf{91.40}(1.53)$	$\mathbf{96.73}(1.80)$	$\mathbf{95.53}(1.79)$	$\mathbf{92.45}(1.81)$	$\mathbf{75.80}(2.25)$	$\mathbf{92.65}(1.09)$
MTO-SC(PC)	$\mathbf{90.84}(1.80)$	$\mathbf{95.81}(1.86)$	$\mathbf{95.80}(1.96)$	$\mathbf{91.56}(1.70)$	$\mathbf{76.50}(2.40)$	$\mathbf{93.28}(1.53)$

We have also conducted an empirical study on *how well the proposed regularizer can preserve the order*. We calculate the percentage of *correctly ordered coding coefficient pairs*, i.e., $\mathbf{w}_j^T\boldsymbol{\alpha}_i^j \leq \mathbf{w}_j^T\boldsymbol{\alpha}_{i+1}^j$, over all pairs $\boldsymbol{\alpha}_i^j$ and $\boldsymbol{\alpha}_{i+1}^j$. The percentage numbers from different regularization schemes on Auslan1 and ArabSpokenDigit datasets are compared in Figure 6. Compared with other regularizers (i.e., sparse coding, fused lasso), the proposed regularizer produces more correctly ordered coefficient pairs, which indicates our regularizer better preserves the order.

To evaluate algorithmic robustness, we also add Gaussian noise to the AusLan1 dataset with zero mean and standard deviation in $\{0.1, 0.2, 0.3\}$ in three different experiments. The classification results are shown in Figure 7. Observe that our MTO-SC method is more robust than its competitors.

We also perform a study on the effects of the weighting parameters λ_1 and λ_2 in our MTO-SC algorithm. Figure 8 plots the classification accuracies on AusLan1 dataset by varying the values of λ_1 and λ_2. We note that when $\lambda_1 = 1.0$ and $\lambda_2 = 0.01$ our algorithm achieves the best performance. Too large or small values for both parameters will decrease the classification accuracy, since in those cases the weighting factors on sparsity and order preserving are not well balanced. Also it is noted that when $\lambda_1 = 100$ and $\lambda_2 = 0.001$ the algorithm performance is very similar with that of group sparse coding as indicated in Table I, since the weighting factor for order preserving is much smaller compared to the weighting factor for sparsity.

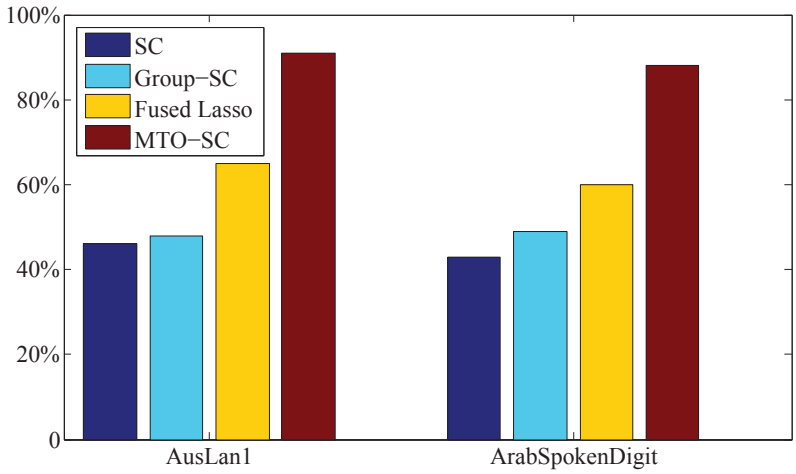


Fig. 6. Percentage of *correctly ordered coding coefficient pairs* from different coding algorithms on AusLan1 and ArabSpokenDigit datasets.

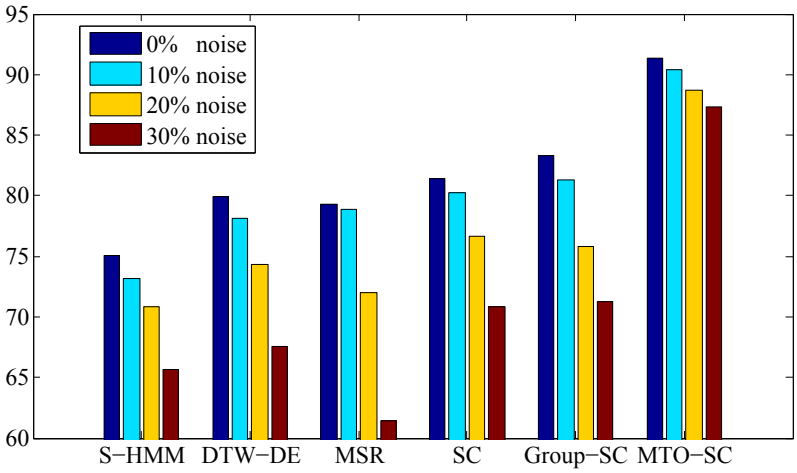


Fig. 7. Classification accuracies for different algorithms on the AusLan1 dataset under different noise conditions.

C. Human Activity Recognition

In this experiment, we use the RGB-D human activity dataset [27]. The video dataset is captured using the Kinect sensor, which produces 640×480 color-depth image sequences with human 3D motion sequences. Namely, each activity sample can be represented as a sequence of 3D joint positions (or angles), similar to those in the CMU MoCap dataset. The dataset consists of five scenarios: office, kitchen, bedroom, bathroom, and living room. Three to four common activities were identified for each

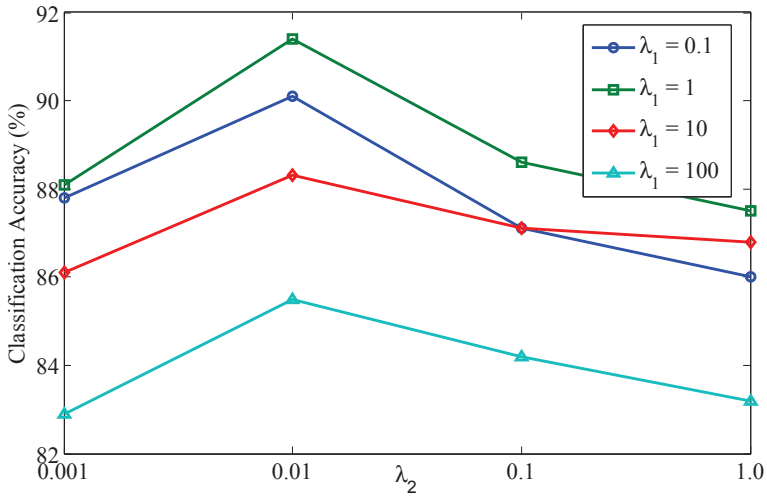


Fig. 8. Classification accuracies of MTO-SC algorithm with different λ_1 and λ_2 values on AusLan1 dataset.

TABLE II

CLASSIFICATION ACCURACY (%) ON RGB-D HUMAN ACTIVITY DATASET.

Method	S-HMM	DTW-DT	DTW-DE	NR-DTW	MSR	SVM
Accuracy	55.78	58.71	57.66	57.02	60.34	50.67
Method	MEMM	HMEMM	SC	Group-SC	Fused Lasso	MTO-SC
Accuracy	61.98	63.75	59.60	58.73	60.95	**65.32**

location, and a total of twelve unique activities were collected from 4 subjects (with an additional *neutral activity* category). We use similar feature representation as in [27], where each frame is represented as a combination of body pose, hand position, motion information and object contextual information. We use the leave-one-subject-out scheme, thus subjects in the testing samples do not occur in the training samples. We compare the multiclass classification accuracies for various algorithms including the proposed MTO-SC method, the time series classification methods compared in the previous experiment and SVM, One-level MEMM and hierarchical maximum entropy Markov model, which are also evaluated in [27]. The classification accuracies are summarized in Table II and the class confusion matrix for our method is illustrated in Figure 9. We observe that our method outperforms the other methods.

D. Still Image Classification

We use the benchmark Scene-15 image dataset [33] for the scene classification experiment. Scene images in this dataset such as landscape images and street view images have strong spatial ordering

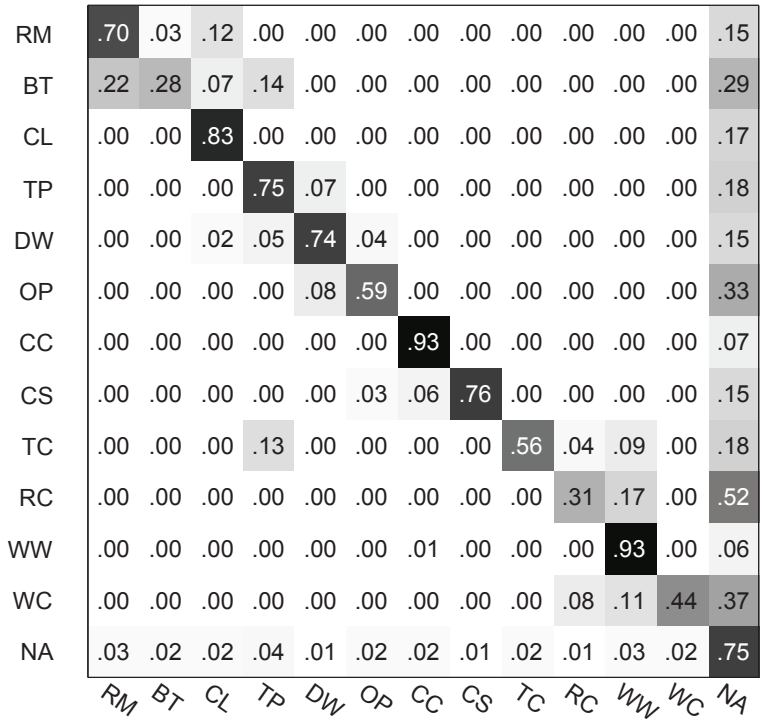


Fig. 9. Class confusion matrix for the proposed MTO-SC method on RGB-D human activity dataset. Acronyms for Action Categories: RM: rinsing mouth; BT: brushing teeth; CL: wearing contact lens; CS: cooking (stirring); WW: writing on white board; WC: working on computer; TP: talking on phone; RC: relaxing on a chair; OP: opening a pill container; DW: drinking water; CC: cooking (chopping); TC: talking on a chair; NA: neutral activity.

structure information. Note that we do not perform experiments on general object image classification datasets since general object images do not have apparent spatial ordering structures among image regions. The Scene-15 dataset contains 15 scenery classes and each class contains 200 to 400 images for a total of 4485 images. The scene categories are street, highway, coast, mountain, kitchen, bedroom etc. We extract SIFT features [37] from densely located patches centered at every 4 pixels on the images. The size of the patches is fixed as 16×16 pixels. As in [33] [35], we randomly select 100 images from each class as training samples to construct a visual word dictionary containing $K = 2048$ words from the training samples via k-means clustering. The random training and test data split is performed 20 times and we report the mean classification accuracy and standard deviation. SIFT features are encoded by:

1) sparse coding (SC) as in [35]; and
2) Laplacian sparse coding (LSC) as in [28].

We segment each image using SLIC [38]. Each image is segmented into about 20 regions on average. We then perform max pooling for the encoded local SIFT features according to the trained dictionary within each image region (segment). Each image region is represented by a K-dimensional feature vector after pooling within the region. Thus an image is represented by a set of K-dimensional feature vectors and the size of the set equals the number of image segments. We then arrange these feature vectors according to the spatial centers of the image segments, resulting horizontally and vertically ordered testing sequences Y and Y', respectively. The dictionary sequences are constructed in the same manner.

The results for our method are denoted by MTO-SC and MTO-LSC respectively, according to whether the segment based features are pooled by the sparse coding (SC) or the Laplacian sparse coding (LSC) method. The baseline methods that perform sparse coding for each segment independently are called SC/LSC-Seg accordingly.

Figure 10 shows the recognition accuracy comparison for each class in the Scene-15 dataset, using 1) sparse coding for each segment based feature independently (denoted as SC-Seg) and 2) our proposed MTO-SC that encourages spatial order preserving for the encoded coefficient for different image segments. We observe that MTO-SC consistently outperforms SC-Seg for all classes. This is because SC-Seg does not consider the ordering structure. We also note that using MTO-SC, the highest improvements are attained for image classes such as *street*, *coast*, *highway*. This is because images in these classes have strong spatial ordering structures. This observation shows that our encoding method effectively encourages spatial ordering structure preserving.

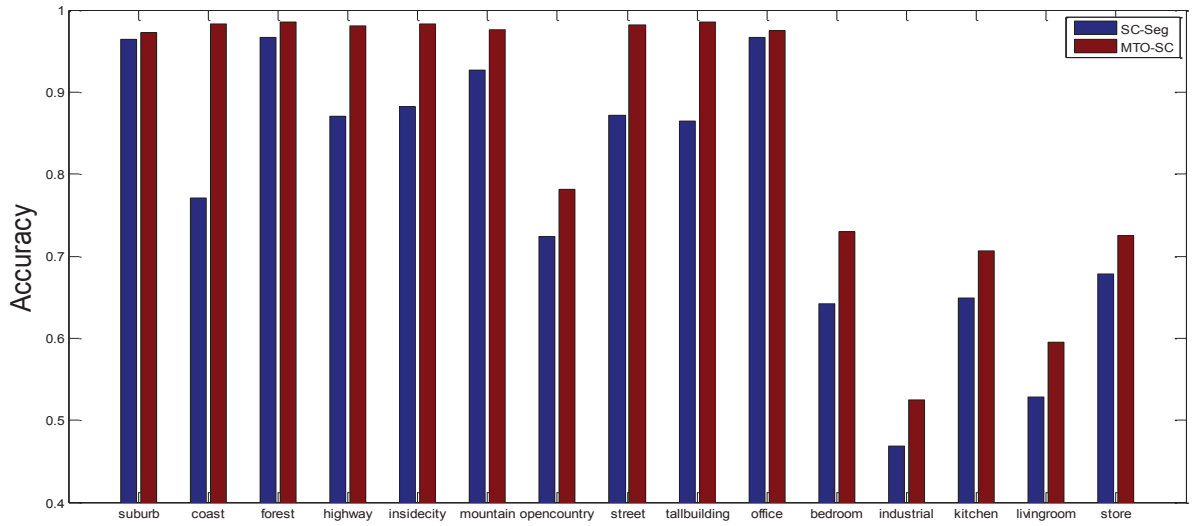


Fig. 10. Recognition accuracy comparison for different classes using SC and MTO-SC methods on Scene-15 dataset.

To directly visualize the advantages of our ordering structure preserving encoding scheme, we show the encoding coefficients for two example test images in Figure 11. The settings are as follows. We use one image as the testing image; and then one image from the same class and two images from other classes (with similar visual elements) as the dictionary images. We perform image segmentation, segment based feature extraction, reconstruction and classification for the test image by SC-Seg and MTO-SC respectively. The arrows from the test image segments to dictionary image segments represent non-zero coding coefficients (after thresholding out small noisy values). From these two examples, we see that using the proposed MTO-SC method, the coding coefficients preserve fairly well the spatial ordering structure of the image segments for the test image. In contrast, ignoring this important ordering structure and performing sparse encoding independently for each image segment encourages matching the testing segments with the dictionary segments only based on visual similarity. For example, *sky* is encoded to match *grass*, *sea* is encoded to match *cloud*, and the *facade* of the building along the street is encoded to match the *facade* of other buildings that do not belong to the street class.

To quantitatively demonstrate the discriminative power of our method, we further compare the recognition accuracies with state-of-the-art image classification methods:

1) the spatial pyramid matching kernel method (KSPM) in [33];
2) the sparse coding + spatial pyramid matching method (ScSPM) [35];
3) the latent pyramidal regions method (LPR) in [29];
4) the dense spatial sampling method (DSS) in [32];
5) the locally linear encoding method (LLC) in [34];
6) the linear distance coding method (LSA) in [36];
7) the histogram of oriented p.d.f gradients method (HOPDFG) in [31];
8) the smooth sparse coding method (SSC) in [30];
9) the Laplacian sparse coding method (LSC) in [28]; and
10) sparse coding for each segment based feature independently (denoted as SC/LSC-Seg).

We directly report the published results for these state-of-the-art methods as all methods follow the same experimental settings. Comparisons in terms of average accuracy and standard deviation from multiple runs are summarized in Table III. We see that 1) both MTO-SC and MTO-LSC significantly outperform their competitors, namely, MTO-SC outperforms SC-Seg as $78.5 \rightarrow 85.9$ and MTO-LSC outperforms LSC-Seg as $86.3 \rightarrow 91.4$, respectively; and 2) MTO-LSC outperforms all these state-of-the-art methods. The second best performance is obtained by the SSC method [30], but this method uses 4096 visual

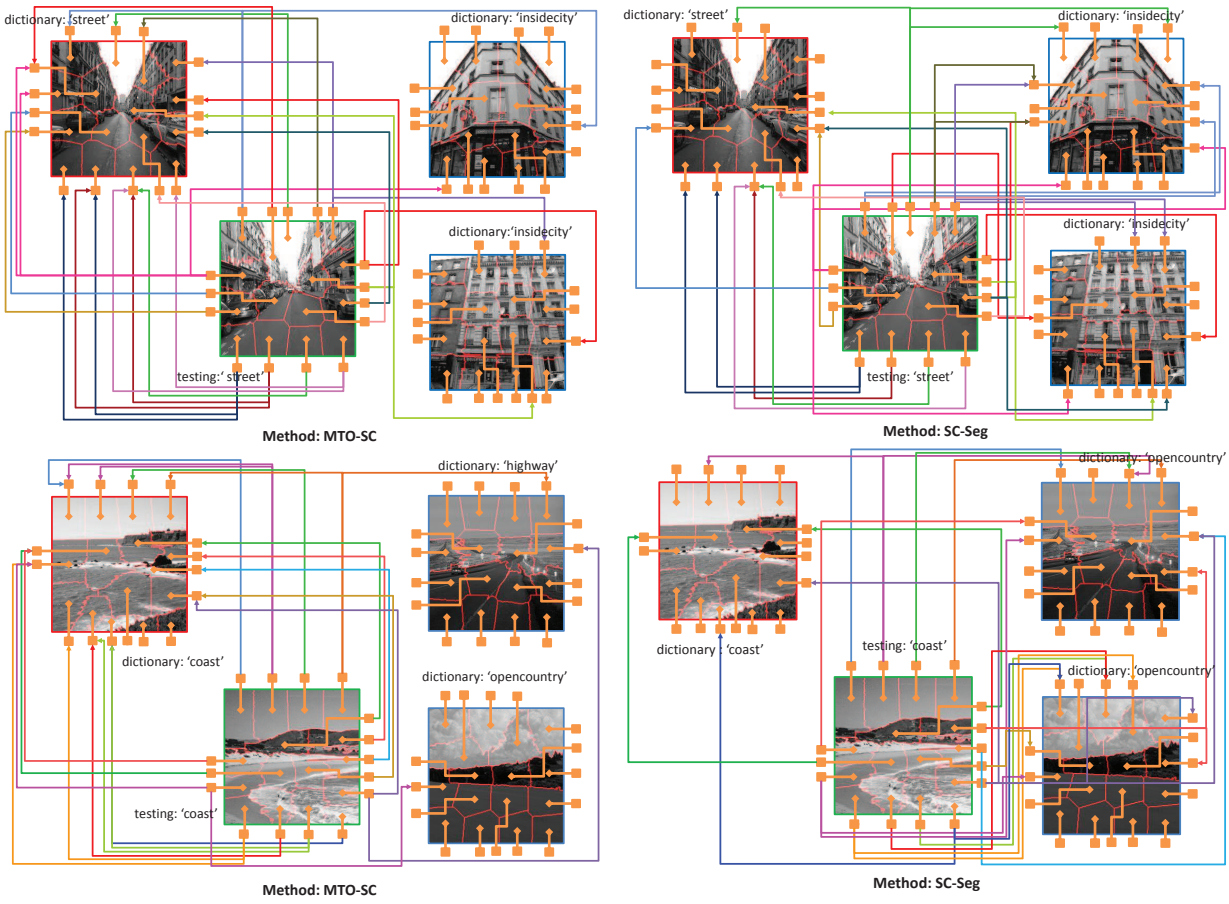


Fig. 11. Coding coefficients for example testing images (upper row and lower row). Arrows show non-zero coding coefficients. Left: using the proposed MTO-SC method; Right: using SC method, i.e., each segment of the testing image is sparsely encoded independently. We see that using MTO-SC, the coding coefficients well preserve the spatial ordering structure of the image segments for the testing image. In contrast, ignoring this important ordering structure and performing sparse encoding independently (SC-Seg) for each image segment encourages matching the testing segments with the dictionary segments only based on visual similarity. For example, *sky* is encoded to match *grass*, *sea* is encoded to match *cloud*, and the *facade* of the building along the street is encoded to match the *facade* of other buildings that do not belong to the street class. Best view in color.

words and ours only 2048.

V. Conclusions

We have proposed an order-preserving sparse coding scheme for time series and still image classification. To this end, we have derived an order-preserving regularization framework. An efficient approximate optimization method for coding is proposed and a guaranteed error bound is derived. Comprehensive

TABLE III

CLASSIFICATION ACCURACY (%) COMPARISON ON SCENE-15 DATASET.

Algorithm	Accuracy	Algorithm	Accuracy
KSPM [33]	81.4 ± 0.5	HOPDFG [31]	85.6 ± 0.7
ScSPM [35]	80.3 ± 0.9	SSC [30]	90.2 ± 2.9
LPR [29]	85.8	LSC [28]	89.8 ± 0.5
DSS [32]	88.0 ± 0.6	SC/LSC-Seg	$78.5 \pm 0.6/86.3 \pm 0.5$
LLC [34]	79.8 ± 0.4	MTO-SC	$\mathbf{85.9 \pm 0.5}$
LSA [36]	82.5 ± 0.5	MTO-LSC	$\mathbf{91.4 \pm 0.4}$

experiments on sign language recognition, spoken digit recognition, human action recognition and scene classification show that our encoded coefficients well preserve the ordering structure of the input sequence. The encoded representation is discriminative and robust, and it outperforms state-of-the-art methods for both classification tasks.

REFERENCES

[1] Wright, J., Yang, A., Ganesh, A., Sastry, S., Ma, Y.: Robust face recognition via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, **31** (2009) 210—227.

[2] Elad, M., Aharon, M.: Image denoising via sparse and redundant representations over learned dictionaries. IEEE Transactions on Image Processing, **15** (2006) 3736–3745.

[3] Rao, S., Tron, R., Vidal, R., Ma, Y.: Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In: Computer Vision and Pattern Recognition, (2008).

[4] Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society B, **67** (2005) 301–320.

[5] Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society B, **68** (2006) 49–67.

[6] Zhang, J.: A probabilistic framework for multi-task learning. Technical report, CMU-LTI-06-006 (2006).

[7] Yuan, X., Yan, S.: Visual classification with multi-task joint sparse representation. In: Computer Vision and Pattern Recognition, (2010).

[8] Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y.: Self-taught learning: Transfer learning from unlabeled data. In: International Conference on Machine Learning, (2007).

[9] Nesterov, Y.: Smooth minimization of non-smooth functions. Mathematical Programming, (2005) 127–152.

[10] Rabiner, L.R., Juang, B.H.: An introduction to hidden markov models. IEEE Magazine on Accoustics, Speech and Signal Processing, **3** (1986) 4–16.

[11] Kim, S., Smyth, P.: Segmental hidden markov models with random effects for waveform modeling. Journal of Machine Learning Research, **7** (2006) 945–969.

[12] Myers, C.S., Rabiner, L.R.: A comparative study of several dynamic time-warping algorithms for connected word recognition. The Bell System Technical Journal, **60** (1981) 1389–1409.

[13] Rodríguez, J.J., Alonso, C.J.: Interval and dynamic time warping-based decision trees. In: ACM Symposium on Applied Computing, (2004) 548–552.

[14] Hayashi, A., Mizuhara, Y., Suematsu, N.: Embedding time series data for classification. Machine Learning and Data Mining in Pattern Recognition, (2005) 356–365.

[15] Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C.A.: Fast time series classification using numerosity reduction. In: International Conference on Machine Learning, (2006) 1033–1040.

[16] Nanopoulos, A., Alcock, R., Manolopoulos, Y.: Feature-based classification of time series data. Information Processing and Technology, (2001) 49–61.

[17] Megalooikonomou, V., Wang, Q., Li, G., Faloutsos, C.: A multiresolution symbolic representation of time series. In: International Conference on Data Engineering, (2005) 668–679.

[18] Manning, C.D., Schuetze, H.: Foundations of statistical natural language processing, MIT press, (1999).

[19] Cadieu, C., Olshausen, B.: Learning transformational invariants from natural movies. In: Advances in Neural Information Processing Systems, (2008) 209–216.

[20] Kim, T., Shakhnarovich, G., Urtasun, R.: Sparse coding for learning interpretable spatio-temporal primitives. In: Advances in Neural Information Processing Systems, (2010). 1117–1125

[21] Zhao, B., Fei-Fei, L., Xing, E.P.: Online detection of unusual events in videos via dynamic sparse coding. In: Computer Vision and Pattern Recognition, (2011).

[22] Tseng, P.: On accelerated proximal gradient methods for convex-concave optimization. SIAM Journal on Optimization, (2008).

[23] Kadous, M.W.: Temporal classification: Extending the classification paradigm to multivariate time series. PhD Thesis, School of Computer Science and Engineering, University of New South Wales, (2002).

[24] Hammami, N., Bedda, M.: Improved tree model for arabic speech recognition. In: Internationl Conference on Computer Science and Information Technology, (2010) 521—526.

[25] : (http://mocap.cs.cmu.edu/).

[26] Shen, Y., Ashraf, N., Foroosh, H.: Action recognition based on homography constraints. In: International Conference on Pattern Recognition, (2008).

[27] Sung, J., Ponce, C., Selman, B., Saxena, A.: Human activity detection from rgbd images. CoRR **abs/1107.0169** (2011).

[28] S. Gao, I. W. Tsang, L.-T. Chia, and P. Zhao.: Local Features Are Not Lonely - Laplacian Sparse Coding for Image Classification. In Computer Vision and Pattern Recognition, 2010.

[29] F. Sadeghi and M. F. Tappen.: Latent pyramidal regions for recognizing scenes. In European conference on Computer Vision, 2012.

[30] K. Balasubramanian, K. Yu, and G. Lebanon.: Smooth sparse coding via marginal regression for learning sparse representations, In International Conference on Machine Learning, 2013.

[31] T. Kobayashi.: BoF meets HOG: Feature Extraction based on Histograms of Oriented p.d.f Gradients for Image Classification, In Computer Vision and Pattern Recognition, 2013.

[32] S. Yan, X. Xu, D. Xu, S. Lin, and X. Li.: Beyond spatial pyramids: a new feature extraction framework with dense spatial sampling for image classification. In European conference on Computer Vision, 2012.

[33] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Computer Vision and Pattern Recognition, 2006.

[34] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In Computer Vision and Pattern Recognition, 2010.

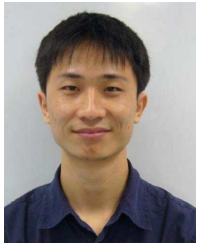
[35] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In Computer Vision and Pattern Recognition, 2009.

[36] Z. Wang, J. Feng, S. Yan, and H. Xi. Linear distance coding for image classification. IEEE Transactions on Image Processing, **22** (2012) 537–548.

[37] D. Lowe. Distinctive image features from scale-invariant keypoints. International Journal on Computer Vision, **60** (2004) 91–110.

[38] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk: SLIC Superpixels Compared to State-of-the-art Superpixel Methods. IEEE Transactions on Pattern Analysis and Machine Intelligence, **34** (2012) 2274–2282.

[39] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight: Sparsity and Smoothness via the Fused Lasso. Journal of Royal Statistical Society, **67** (2005) 91–108.

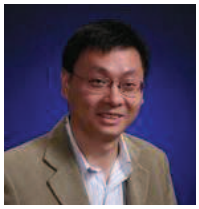


Dr. Bingbing Ni received his B.Eng. degree in Electrical Engineering from Shanghai Jiao Tong University (SJTU), China in 2005 and obtained his Ph.D. from National University of Singapore (NUS), Singapore in 2011. Dr. Ni is currently a research fellow in Advanced Digital Sciences Center, Singapore. His research interests are in the areas of computer vision, machine learning and multimedia. Dr. Ni worked in Microsoft Research Asia, Beijing as a research intern in 2009. He also worked as a software engineer intern in Google Inc., Mountain View, CA in 2010. He received the Best Paper Award from PCM'11 and the Best Student Paper Award from PREMIA'08. He won the first prize in International Contest on Human Activity Recognition and Localization (HARL) in conjunction with International Conference on Pattern Recognition, 2012.

Pierre Moulin received his doctoral degree from Washington University in St. Louis in 1990, after which he joined at Bell Communications Research in Morristown, New Jersey, as a Research Scientist. In 1996, he joined the University of Illinois at Urbana-Champaign, where he is currently Professor in the Department of Electrical and Computer Engineering, Research Professor at the Beckman Institute and the Coordinated Science Laboratory, and affiliate professor in the Department of Statistics.

His fields of professional interest include image and video processing, compression, statistical signal processing and modeling, media security, decision theory, and information theory. Dr. Moulin has served on the editorial boards of the IEEE Transactions on Information Theory and the IEEE Transactions on Image Processing. He is co-founding Editor-in-Chief of the IEEE Transactions on Information Forensics and Security. He has served IEEE in various other capacities and is currently a member of the IEEE Signal Processing Society Board of Governors.

He received a 1997 Career award from the National Science Foundation and an IEEE Signal Processing Society 1997 Senior Best Paper award. He is also co-author (with Juan Liu) of a paper that received an IEEE Signal Processing Society 2002 Young Author Best Paper award. He was 2003 Beckman Associate of UIUCs Center for Advanced Study. He is an IEEE Fellow, recipient of UIUCs 2005 Sony Faculty award, and plenary speaker for ICASSP 2006.



Dr. Yan Shuicheng (M'06, SM'09) is currently an Associate Professor in the Department of Electrical and Computer Engineering at National University of Singapore, and the founding lead of the Learning and Vision Research Group (http://www.lv-nus.org). Dr. Yan's research areas include computer vision, multimedia and machine learning, and he has authored/co-authored over 300 technical papers over a wide range of research topics, with Google Scholar citation $> 9,300$ times and H-index-42. He is an associate editor of IEEE Transactions on Circuits and Systems for Video Technology (IEEE TCSVT) and ACM Transactions on Intelligent Systems and Technology (ACM TIST), and has been serving as the guest editor of the special issues for TMM and CVIU. He received the Best Paper Awards from ACM MM12 (demo), PCM'11, ACM MM10, ICME10 and ICIMCS'09, the winner prizes of the classification task in PASCAL VOC 2010-2012, the winner prize of the segmentation task in PASCAL VOC 2012, the honorable mention prize of the detection task in PASCAL VOC'10, 2010 TCSVT Best Associate Editor (BAE) Award, 2010 Young Faculty Research Award, 2011 Singapore Young Scientist Award, and 2012 NUS Young Researcher Award.