Cognitive Computation (2019) 11:31–49
https://doi.org/10.1007/s12559-018-9590-9

Learning with Similarity Functions: a Tensor-Based Framework

Edoardo Ragusa[1] · **Paolo Gastaldo**[1] · **Rodolfo Zunino**[1] · **Erik Cambria**[2]

Received: 10 March 2018 / Accepted: 6 August 2018 / Published online: 31 August 2018

Abstract

Machine learning algorithms are typically designed to deal with data represented as vectors. Several major applications, however, involve multi-way data, such as video sequences and multi-sensory arrays. In those cases, tensors endow a more consistent way to capture multi-modal relations, which may be lost by a conventional remapping of original data into a vector representation. This paper presents a tensor-oriented machine learning framework, and shows that the theory of learning with similarity functions provides an effective paradigm to support this framework. The proposed approach adopts a specific similarity function, which defines a measure of similarity between a pair of tensors. The performance of the tensor-based framework is evaluated on a set of complex, real-world, pattern-recognition problems. Experimental results confirm the effectiveness of the framework, which compares favorably with state-of-the-art machine learning methodologies that can accept tensors as inputs. Indeed, a formal analysis proves that the framework is more efficient than state-of-the-art methodologies also in terms of computational cost. The paper thus provides two main outcomes: (1) a theoretical framework that enables the use of tensor-oriented similarity notions and (2) a cognitively inspired notion of similarity that leads to computationally efficient predictors.

Keywords Tensor data · Similarity functions · Tensor-based learning · Intelligent embedded systems

Introduction

Tensors provide a more convenient and more cognitive-inspired way to describe multi-way data and to suitably capture their multi-linear structure. Unlike other data structures, in fact, tensors take a holistic approach to multi-way data representation, which is more likely to be similar to the way the human brain processes such data. This basic aspect becomes relevant when dealing with machine learning (ML): multi-spectral images [1], computer vision [2], authorship identification [3], and multi-sensory arrays [4, 5] provide relevant examples of domains in which ML-based predictors are required to deal with multi-way data. Accordingly, one needs specific learning methods that can explicitly exploit the multi-mode relations that connect the various data associated with one pattern. These specialized methods [6–9] can outperform conventional approaches, which remap multi-way data into a classical vector representation for the subsequent application of ML models. In this regard, the crucial point is that the geometrical information inherently embedded in the tensor structure can be exploited to boost the learning process, i.e., to improve convergence speed. Conversely, traditional models based on vector space—which actually ensures universal approximation capabilities—face the risk of not capturing such geometrical information. As a major consequence, the convergence speed can be substantially different. The literature provides several tensor-oriented ML frameworks, which might be categorized into two groups.

"Projections learning" approaches focus on multi-linear projections, to remap tensor data down to a lower-dimensional space. The underlying goal is to find a projection schema that can capture the (unknown) structure

✉ Erik Cambria
cambria@ntu.edu.sg

Edoardo Ragusa
edoardo.ragusa@edu.unige.it

Paolo Gastaldo
paolo.gastaldo@unige.it

Rodolfo Zunino
rodolfo.zunino@unige.it

[1] Department of Electrical, Electronic and Telecommunications Engineering and Naval Architecture, DITEN, University of Genoa, Genoa, Italy

[2] School of Computer Science and Engineering, Nanyang Technological University, Nanyang, Singapore

of the problem at hand, so that one may eventually apply ML methods to learn a decision function in the lower-dimensional space. Those methods mostly address the problem of finding the projection that satisfies an optimality criterion [10, 11], with the partial exception of [12], which relies on higher-order random projections.

Conversely, "function learning" frameworks aim to learn a decision function that is designed to accept tensors as inputs. The related training algorithms often involve an iterative process [13–16] to solve convex optimization problems; training stops when some pre-set condition is fulfilled. Otherwise, training procedures rely on standard convex optimization; this case includes support higher-order tensor machines (SHTMs) [17] and the kernel-based framework proposed by Signoretto et al. [18], based on a tensor-oriented kernel.

This paper addresses function learning frameworks and shows that the theory of learning with similarity functions [19] can stimulate a novel design strategy, which may represent an alternative to standard kernel-based methods. Two main reasons justify the focus on similarity-based models: first, these frameworks usually better succeed in balancing the generalization performance and the computational complexity of the trained predictors. This is a crucial aspect when implementing the classifiers on embedded systems. Secondly, similarity-based models provide a viable option for avoiding the computationally intensive learning procedures that affect iterative training.

Similarity functions in principle implement a notion of similarity between two generic patterns that lie in a domain space $\mathcal{X}$, just as any conventional kernel function. Nonetheless, the theory of learning with similarity functions [19] proves that a similarity-based model is no longer tied to functions that span high-dimensional spaces implicitly and require positive semi-definite matrices. This in turn means that a similarity function is not necessarily a valid kernel function. Indeed, similarity functions support a straightforward, yet powerful learning paradigm [19] that relies on (1) a given notion of similarity and (2) landmarks, i.e., a subset of reference patterns randomly extracted from the available dataset. Thus, as a first step, each training sample is remapped according to the similarity of the sample itself with respect to the landmarks. As a result, one remaps the input space $\mathcal{X}$ to a L-dimensional space, where L is the number of landmarks. Then, a conventional learning algorithm sets a linear classifier in the remapped space, where a low-error large-margin separator is guaranteed to exist [19]. The eventual predictor classifies a new pattern by computing the weighted sum of the similarities between the pattern and the landmarks that already supported the training phase.

This research exploits the similarity-based learning paradigm to setup a novel tensor-oriented ML framework. To achieve this goal, the present paper introduces a similarity function that can suitably process multi-way data; thus, it computes the similarity between a given sample and a given landmark that lie in a tensor space. The function embeds a similarity notion that involves a three-step processing. First, multi-linear singular value decomposition (MLSVD) [20] is used to represent the landmark tensor as a set of orthonormal bases with their associate eigenvalues. Then, the sample tensor is projected according to the landmark bases, to emphasize the intrinsic structural differences between the sample and the landmark. Finally, similarity is assessed by a metric function, which processes the eigenvalues of both the landmark and the (projected) sample.

The design of a suitable similarity function that can process patterns represented as tensors is the major novel contribution of this research. The paper characterizes both the geometrical properties and the cognitive properties of the underlying similarity notion, which indeed inherently embeds noise filtering capabilities. Such attribute makes the similarity function particularly suitable for applications that may suffer from noisy acquisition processes. Besides, this work shows that the availability of such similarity function leads to a tensor-oriented ML framework that features the following advantages:

- A training algorithm that does not require iterative processes, unlike most of the state-of-the-art function learning frameworks.
- A prediction function that—in terms of computational cost—bests the prediction function supported by a state-of-the-art kernel-based framework for tensorial data [18]; such aspect becomes relevant when considering implementations on resource-constrained devices [21].
- A predictor that compares positively with state-of-the-art tensor-oriented ML frameworks in terms of classification accuracy.

The proposed ML framework was evaluated on a variety of real-world, pattern-recognition problems involving multi-way data. In the experimental tests, the approach based on tensor-oriented similarity functions was compared with three alternative approaches: SHTM, a kernel machine that can inherently process tensors by exploiting the tensor-oriented kernel introduced in [18], and a standard SVM that remaps multi-way data into vector-based representation. These approaches actually represent state-of-the-art solutions when one wants training procedures that only involve standard convex optimization. Experimental results showed that the proposed approach compared favorably with the others techniques in terms of classification performance.

The remainder of the paper is organized as follows: Section "Learning with Similarity Functions" briefly

reviews the theory of learning with similarity functions; Section "A Tensor-Based Similarity Notion" introduces the novel notion of tensor-based similarity; Section "Tensor-Based Learning with Similarity Functions" frames the tensor-based scheme within the theory of learning with similarity functions, and compares the proposed framework with state-of-the-art tensor-oriented approaches; a formal analysis of the computational cost of the proposed framework is also provided. Section "Experimental Results" presents experimental results; finally, some concluding remarks are discussed in "Conclusion".

Learning with Similarity Functions

Let $\mathcal{X}$ be the input space and let $\mathcal{T} = \{(\boldsymbol{x}, y)_i; \boldsymbol{x} \in \mathcal{X}; y \in \{-1, 1\}; i = 1, ..., Z\}$ be a labeled training set drawn from a domain distribution, $p(\mathcal{X}, \mathcal{Y})$. Then, let $K : \mathcal{X} \times \mathcal{X} \rightarrow R$ be the function that expresses a notion of similarity $K(\boldsymbol{x}_m, \boldsymbol{x}_n)$ between a pair of samples, $\{\boldsymbol{x}_m, \boldsymbol{x}_n \in \mathcal{X}\}$. A similarity-based classifier assigns a label to an input test sample, $\boldsymbol{x}^*$, depending on (1) the pairwise similarities between training patterns and (2) the similarities between $\boldsymbol{x}^*$ and each training pattern.

In principle, the well-known K-nearest neighbors algorithm belongs to such family of classifiers. However, such algorithm in its conventional form does not involve a proper training phase, as the predictor requires the availability of the entire training set. Kernel machines [22] indeed represent a popular, effective implementation of similarity-based classifiers. They express the notion of similarity by kernel functions, which embed an implicit mapping of data into a Hilbert space and lead to symmetric, positive semi-definite matrices. On the other hand, the theory of learning with similarity functions formalized in [19] indicates that viable alternatives exist to the realization of powerful similarity-based classifiers.

The following pair of definitions summarize the crucial elements of the theoretical framework; the first one addresses the notion of pairwise similarity function, K:

Definition 1 [19] A similarity function over $\mathcal{X}$ is any pairwise function $K : \mathcal{X} \times \mathcal{X} \rightarrow [-1, 1]$. K is defined as a symmetric similarity function if $\forall \boldsymbol{x}, \boldsymbol{x}' \in K(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x}', \boldsymbol{x})$.

The second definition formalizes the notion of "good similarity function":

Definition 2 [19] A similarity function, K, is an (ϵ, γ)-good similarity function for a learning problem P if there exists a bounded weighting function ω over $\mathcal{X}(\omega(x) \in [0, 1])$ for all $\boldsymbol{x} \in \mathcal{X}$ such that at least a $(1 - \epsilon)$ probability mass of examples $\boldsymbol{x}$ satisfy the following:

$$E_{\boldsymbol{x}' \sim \boldsymbol{P}}[y(\boldsymbol{x}')y(\boldsymbol{x})\omega(\boldsymbol{x}')K(\boldsymbol{x}, \boldsymbol{x}')] \geq \gamma$$

Algorithm 1 The learning scheme that exploits the theory of learning with (ϵ, γ)-good similarity functions

Input

- a labeled training set $\mathcal{T} = \{(\boldsymbol{x}, y)_i; i = 1, ..., Z\}$
- a similarity function K
- number of landmarks L

0. **Initialize**
extract L random samples $\mathcal{L} = \{\boldsymbol{l}_n; n = 1, .., L\}$ from $\mathcal{T}$

1. **Mapping**
remap all the patterns $\boldsymbol{x} \in \mathcal{T}$ by using the following mapping function

$$\boldsymbol{\phi}(\boldsymbol{x}) = \{\frac{1}{\sqrt{L}}K(\boldsymbol{x}, \boldsymbol{l}_1), ..., \frac{1}{\sqrt{L}}K(\boldsymbol{x}, \boldsymbol{l}_L)\}$$

2. **Learning**
train a linear predictor in the space $\boldsymbol{\phi} : \mathcal{X} \rightarrow R^L$

The above definition uses a weighting function, ω, to balance the relative significance of each sample, $\boldsymbol{x}'$. When considering the eventual learning algorithm, the definition requires that a bounded weighting function, ω, exists, although this does not imply that such a function is known *a-priori*. In other terms, any similarity expression is, in principle, a good similarity function, and one needs a suitable criterion to find the weighting function ω that minimizes ϵ and maximizes γ.

Definition 2 suggests that -in practice- one should replace the expectation with an average over a set of landmarks, i.e., the examples $\boldsymbol{x}'$. As a result, the weighting function ω will satisfy Definition 2 in correspondence of the landmarks at hand. The outcome of this observation is that the notion of good similarity function allows one to setup a learning scheme based on a hypothesis space. To this purpose, one needs the following:

- L landmarks, i.e., a subset of the original dataset which is randomly drawn from the domain distribution $p(\mathcal{X})$ that characterizes P. Both labeled and unlabeled patterns provide an admissible source of landmarks.
- A similarity function K.

To build the hypothesis space, the domain space $\mathcal{X}$ is first remapped into a new space R^L. Accordingly, for every pattern, $\boldsymbol{x}$, one computes the similarities, K, between $\boldsymbol{x}$

and each landmark. In the second step, a linear predictor is trained in the new space, R^L. The eventual hypothesis space can be formalized as follows:

$$f(\boldsymbol{x}) = \sum_{j=1}^{L} \omega_j K(\boldsymbol{x}, \boldsymbol{l}_j) \tag{1}$$

where the weights, ω_j, are computed by adjusting a linear predictor. Algorithm 1 outlines the associate learning algorithm.

Algorithm 1 relies on the similarity function K to remap the original space into a new space where data are separated by a (possibly large) margin with error, ϵ. Then, the task of tuning the weighting function, ω, is assigned to the linear predictor. The learning abilities of this procedure have been formally analyzed in [19]: if one set $L = 16 \cdot \ln(4/\epsilon*)/\gamma^2$, then with probability at least $(1 - \epsilon^*/2)$ there exists a low-error $(\epsilon + \epsilon^*)$, large-margin $(\gamma/2)$ separator in the feature space.

A Tensor-Based Similarity Notion

The above learning scheme sets a suitable baseline to develop a tensor-oriented ML model, as its distinguishing feature is the mapping stage that explicitly remaps the input space into the space R^L. This process is supported by a function K, which in Algorithm 1 supports a measure of similarity between a pattern, $\boldsymbol{x}$, and a landmark, $\boldsymbol{l}$. This learning scheme can extend to multi-way data by introducing a similarity notion that processes tensors instead of conventional vectors. Overall, one might say that the eventual framework belongs to the "function learning" context. The tensor-oriented similarity notion is clearly crucial in the development of the overall framework and is subject to a pair of requirements. First, it should be able to capture the structural properties of multi-way patterns; secondly, it should lead to a computationally efficient implementation of the eventual predictor. The latter constraint gets relevant when addressing hardware-friendly tensor-based implementations. This section shows that tensor decomposition can be profitably adopted to satisfy the above requirements.

In the following, Section "Multi-linear Singular Value Decomposition" introduces basic tensor notations and then illustrates the principles of MLSVD, i.e., the tensor decomposition adopted to support the proposed similarity notion. Section "Multi-linear Extremal Energy" briefly explains how the concept of extremal energy can be extended to nth order tensors. Section "A Novel Similarity Notion for Multi-way Data" analyzes the actual design of the novel tensor-based similarity notion. Section "Toy Examples" illustrates the properties of the similarity notion by exploiting practical examples.

Multi-linear Singular Value Decomposition

A tensor can be formalized as a multi-way array of numbers. In the following, $\mathcal{T} \in R^{I_1 \times I_2 \times \ldots \times I_N}$ will refer to a real-valued tensor of order N. Accordingly, a vector and a matrix can be regarded as tensors of orders 1 and 2, respectively. The notation $\mathcal{T}_{(n)}$ will denote the mode-n unfolding of $\mathcal{T}$; i.e., $\mathcal{T}_{(n)} \in R^{I_n \times I_1 I_2 \ldots I_{n-1} I_{n+1} I_N}$ (Appendix provides details about mode-n unfolding). Finally, $\mathcal{Q} = \mathcal{T} \times_n \boldsymbol{B}$ will denote the mode-n product of $\mathcal{T}$ and $\boldsymbol{B} \in R^{J_n \times I_n}$; this product yields a tensor $\mathcal{Q} \in R^{I_1 \times I_2 \times I_{n-1} \times J_n \times I_{n+1} \times I_N}$, whose entries can be expressed as follows:

$$q_{i_1 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} t_{i_1 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} b_{j_n i_n} \tag{2}$$

As a consequence, $\mathcal{Q} = \mathcal{T} \times_n \boldsymbol{B}$ can also be rewritten as $\mathcal{Q}_{(n)} = \boldsymbol{B}\mathcal{T}_{(n)}$.

MLSVD is a multi-linear generalization of the well-known singular value decomposition (SVD), supported by the model originally introduced in the Tucker decomposition [20, 23].

MLSVD represents a tensor $\mathcal{T}$ as a multi-linear transformation of a core tensor, $\bar{\mathcal{T}} \in R^{I_1 \times I_2 \times I_N}$, by means of the matrices $\boldsymbol{T}_n = [\boldsymbol{t}_1^{(n)} \boldsymbol{t}_2^{(n)} \ldots \boldsymbol{t}_{I_n}^{(n)}] \in R^{I_n \times I_n} (n = 1, N)$; here, the column vectors $\boldsymbol{t}_{I_n}^{(n)}$ define an orthonormal basis in R^{I_n}. As a result,

$$\mathcal{T} = \bar{\mathcal{T}} \times_1 \boldsymbol{T}_1 \times_2 \boldsymbol{T}_2 \ldots \times_N \boldsymbol{T}_N \tag{3}$$

Each orthonormal basis, $\boldsymbol{T}_n$, can be obtained by applying conventional SVD to $\mathcal{T}_{(n)}$. Hence, one has

$$\mathcal{T}_{(n)} = \boldsymbol{U}_n \boldsymbol{\Sigma}_n \boldsymbol{V}_n^t \tag{4}$$

with $\boldsymbol{T}_n = \boldsymbol{U}_n$. Indeed, the core tensor $\bar{\mathcal{T}}$ can be computed as follows:

$$\bar{\mathcal{T}} = \mathcal{T} \times_1 \boldsymbol{T}_1^t \times_2 \boldsymbol{T}_2^t \ldots \times_N \boldsymbol{T}_N^t \tag{5}$$

where $\bar{\mathcal{T}}$ is an all-orthogonal and ordered tensor [20]. "All-orthogonal" means that all the rows of any mode-n unfolding of $\bar{\mathcal{T}}$, $\bar{\mathcal{T}}_{(n)}$, are mutually orthogonal. "Ordered" means that the following condition holds for such rows:

$$\|\bar{\mathcal{T}}_{(n)}(1,:)\| \leq \|\bar{\mathcal{T}}_{(n)}(2,:)\| \leq \ldots \leq \|\bar{\mathcal{T}}_{(n)}(I_n,:)\| \tag{6}$$

It is worth noting that the Frobenius norm $\|\bar{\mathcal{T}}_{(n)}(i_n,:)\|$ corresponds to the i_nth n-mode eigenvalue of $\mathcal{T}$, i.e., the i_nth eigenvalue of $\mathcal{T}_{(n)}$ [20]. Thus,

$$(\sigma_{i_n})_{(n)}^{\mathcal{T}} = \boldsymbol{\Sigma}_n(i_n, i_n) = \|\bar{\mathcal{T}}_{(n)}(i_n,:)\| \tag{7}$$

In addition, the columns of $\boldsymbol{T}_n$ are the n-mode eigenvectors of $\mathcal{T}$. Furthermore, it holds:

$$\|\mathcal{T}_{(n)}\|^2 = \|\bar{\mathcal{T}}_{(n)}\|^2 = \sum_{i}^{I_n} \left((\sigma_{i_n})_{(n)}^{\mathcal{T}}\right)^2 \tag{8}$$

Multi-linear Extremal Energy

The concept of extremal energy plays a crucial role in SVD-based signal separation algorithms.

Definition 3 [24] (Extremal directions of oriented energy): Let $\boldsymbol{A}$ be a $m \times n$ matrix with SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^t$ where $\boldsymbol{\Sigma} = \texttt{diag}\{\sigma_i\}$. Then, each direction of extremal oriented energy is generated by a left singular vector $\boldsymbol{u}_i$ with extremal energy equal to the corresponding singular value σ_i^2.

Definition 3 states that all the energy contained in a matrix can be described by means of the pairs (eigenvector, eigenvalues). In practice, the extremal directions of oriented energy express the hidden geometrical properties of the matrix. These properties indeed lead to the principle of optimal reconstruction in the minimal square-error sense [24].

The following definition paves the way to the extension of the concept of extremal directions of oriented energy to the multivariate case.

Definition 4 [20] (n-mode oriented energy): the n-mode oriented energy of an Nth-order tensor $\mathcal{T}$ in the direction of a unit-norm vector $\boldsymbol{x}$, denoted by $\text{OE}_n(\boldsymbol{x}, \mathcal{T})$, is the oriented energy of the set of n-mode vectors, i.e.,

$$\text{OE}_n(\boldsymbol{x}, \mathcal{T}) \doteq \|\boldsymbol{x}^t \mathcal{T}_{(n)}\|^2 \tag{9}$$

Definition 4 actually allows one to reinterpret Eq. 8 according to the following property:

Property 1 [20] (oriented energy). The directions of extremal n-mode oriented energy correspond to the n-mode singular vectors, with extremal energy value equal to the corresponding squared n-mode singular value.

Thus,

$$\|\mathcal{T}_{(n)}\|^2 = \|\bar{\mathcal{T}}_{(n)}\|^2 = \sum_{i}^{I_n} \left((\sigma_{i_n})_{(n)}^{\mathcal{T}}\right)^2 = \sum_{i}^{I_n} \text{OE}_n(\boldsymbol{U}_{(n)}(i_n, :), \mathcal{T}) \tag{10}$$

As a result, let $\hat{\mathcal{T}}$ be a tensor obtained by removing from $\mathcal{T}$ the components of the n-mode vectors in the direction of an n-mode singular vector $\boldsymbol{U}_{(n)}(I_n, :)$ (i.e., the singular vector that corresponds to the smallest n-mode singular value). One has that

$$\|\mathcal{T} - \hat{\mathcal{T}}\|^2 = (\sigma_{I_n}^{(n)})^2 = OE_n(\boldsymbol{U}_{(n)}(I_n, :), \mathcal{T}) \tag{11}$$

In general, $\hat{\mathcal{T}}$ is not the best approximation in a square-error sense. However, this approximation involves an error that is known a-priori, i.e., $((\sigma_{I_n})_{(n)}^{\mathcal{T}})^2$. Equation 10 proves that the MLSVD actually provides a detailed geometrical description of the energy distribution over the n-mode vector space. In practice, for each mode, the associate matrix defines the orthogonal directions in which the energy is more propagated. Therefore, the MLSVD can be suitably exploited in the design of a similarity notion.

A Novel Similarity Notion for Multi-way Data

The definition of a notion of similarity between a generic multi-way pattern and a multi-way landmark is the first step in designing the tensor-based similarity function. For the sake of clarity, to exemplify the rationale behind the proposed similarity notion and without loss of generality, let a landmark, $\mathcal{L}$, be a tensor of order 2: $\mathcal{L} \in R^{I_1 \times I_2}$. As per (3), $\mathcal{L}$ can be represented as follows:

$$\mathcal{L} = \bar{\mathcal{L}} \times_1 \boldsymbol{L}_1 \times_2 \boldsymbol{L}_2 \tag{12}$$

The properties of the SVD also allow to rewrite $\mathcal{L}$ as follows:

$$\mathcal{L} = \sum_{i=1}^{p} \Sigma(i,i) \cdot \boldsymbol{u}_i \otimes \boldsymbol{v}_i = \sum_{i=1}^{p} \sigma_i^L \hat{\boldsymbol{L}}_i \tag{13}$$

where p is the rank of $\mathcal{L}$, the eigenvalues σ_i are ordered in decreasing level of magnitude, and the matrices $\hat{\boldsymbol{L}}_i$ define an orthonormal basis in $R^{I_1 \times I_2}$ for $\mathcal{L}$. In general, a matrix is univocally identified by the combination of eigenvalues σ_i and the corresponding orthonormal basis. Moreover, the matrices $\hat{\boldsymbol{L}}_i$ identify the most informative basis for the original matrix in a square-error sense. As a result, the following provides an optimal approximation of $\boldsymbol{L}$ (in a square-error sense).

$$\hat{\mathcal{L}} = \sum_{i=1}^{q} \sigma_i^L \hat{\boldsymbol{L}}_i, \text{ with } q < p, \tag{14}$$

Conventional approaches to the assessment of the similarity between a generic pattern $\mathcal{X} \in R^{I_1 \times I_2}$ and a landmark, $\boldsymbol{L}$, usually rely on some metrics that compares the eigenvalues and the orthonormal basis of $\mathcal{X}$, $\{\sigma_i^X, \hat{\boldsymbol{X}}_i\}$, with the eigenvalues and the orthonormal basis of $\mathcal{L}$, $\{\sigma_i^L, \hat{\boldsymbol{L}}_i\}$. These approaches would require the computation of two SVDs; instead, the similarity notion presented in this paper exploits the basis of $\mathcal{L}$ as a reference, thus projecting the pattern $\mathcal{X}$ on the most informative set of basis for the landmark. Accordingly, similarity is assessed by comparing the eigenvalues σ_i^L of $\mathcal{L}$ with the *pseudo*-eigenvalues of $\mathcal{X}$.

The latter quantities can be formalized as the coefficients $\tilde{\sigma}_i^X$ that support the following reconstruction of $\mathcal{X}$:

$$\mathcal{X} = \sum_{i=1}^{p} \tilde{\sigma}_i^X \hat{\boldsymbol{L}}_i, \tag{15}$$

Extending this approach to tensors of any order is feasible, under the hypothesis that the landmark $\mathcal{L} \in R^{I_1 \times I_2 \times \ldots \times I_N}$ has a core tensor $\bar{\mathcal{L}}$ with non-null elements only for $i_1 = i_2 = \ldots I_n$. In general, though, this condition does not hold. Then, one can represent $\mathcal{L}$ either by relying on Eq. 3, or as follows:

$$\begin{aligned}\mathcal{L} &= \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \ldots \sum_{i_N=1}^{I_N} \bar{\mathcal{L}}(i_1, i_2, \ldots, i_N) \cdot \boldsymbol{l}_{i_1}^{(1)} \otimes \boldsymbol{l}_{i_2}^{(2)} \otimes \ldots \otimes \boldsymbol{l}_{i_N}^{(I_N)} \\ &= \sum_{i=1}^{I} \sigma_i^{\mathcal{L}} \hat{\mathcal{L}}_i \text{ with } I = I_1 I_2 \ldots I_N. \end{aligned} \tag{16}$$

In Eq. 16, $\boldsymbol{l}_{i_n}^{(n)}$ represents the i_nth column vector of the orthonormal basis $\boldsymbol{L}_n$; the resulting tensors $\hat{\mathcal{L}}_i$ define a basis in $R^{I_1 \times I_2 \times \ldots \times I_N}$. Equation 16 shows that to project a generic pattern $\mathcal{X} \in R^{I_1 \times I_2 \times \ldots \times I_N}$ onto the bases of $\mathcal{L}$ one needs to align the corresponding bases $\{\boldsymbol{X}_n; n = 1, \ldots, N\}$ with the bases $\boldsymbol{L}_n; n = 1, \ldots, N$. Accordingly, let $\tilde{\mathcal{X}}$ be a *pseudo*-core tensor obtained by the multi-linear transformation of $\mathcal{X}$ by the orthonormal bases $\boldsymbol{L}_n$, i.e., the bases that characterizes $\mathcal{L}$:

$$\tilde{\mathcal{X}} = \mathcal{X} \times_1 \boldsymbol{L}_1^t \times_2 \boldsymbol{L}_2^t \ldots \times_N \boldsymbol{L}_N^t \tag{17}$$

$\tilde{\mathcal{X}}$ is a *pseudo*-core tensor, since it cannot be considered the core tensor of $\mathcal{X}$ in a rigorous sense. According to Eq. 5, the core tensor $\bar{\mathcal{X}}$ is obtained from the associate orthonormal basis, $\boldsymbol{X}_n$:

$$\bar{\mathcal{X}} = \mathcal{X} \times_1 \boldsymbol{X}_1^t \times_2 \boldsymbol{X}_2^t \ldots \times_N \boldsymbol{X}_N^t \tag{18}$$

Anyway, $\tilde{\mathcal{X}}$ can be very useful if one addresses the goal of assessing the similarity between $\mathcal{L}$ and $\mathcal{X}$. In this regard, it is convenient to rewrite (17) as follows:

$$\begin{aligned}\tilde{\mathcal{X}} = \mathcal{X} \times_1 \boldsymbol{L}_1^t \times_2 \boldsymbol{L}_2^t \ldots \times_N \boldsymbol{L}_N^t &= \bar{\mathcal{X}} \times_1 \boldsymbol{X}_1 \times_2 \boldsymbol{X}_2 \ldots \\ \times_N \boldsymbol{X}_N \times_1 \boldsymbol{L}_1^t \times_2 \boldsymbol{L}_2^t \ldots \times_N \boldsymbol{L}_N^t &= \bar{\mathcal{X}} \times_1 \boldsymbol{L}_1^t \boldsymbol{X}_1 \\ \times_2 \boldsymbol{L}_2^t \boldsymbol{X}_2 \ldots \times_N \boldsymbol{L}_N^t \boldsymbol{X}_N \end{aligned} \tag{19}$$

Equation 19 formalizes the relationship between $\tilde{\mathcal{X}}$ and $\mathcal{X}$. It is interesting to understand the role played by the terms $\{\boldsymbol{L}_n^t \boldsymbol{X}_n\}$. One recalls that both $\boldsymbol{L}_n$ and $\boldsymbol{X}_n$ define orthonormal bases; the corresponding basis vectors $\boldsymbol{l}_{i_n}^{(n)}$ and $\boldsymbol{x}_{i_n}^{(n)}$ are indeed the eigenvectors that characterize the corresponding subspaces. As a result,

$$\boldsymbol{L}_n^t \boldsymbol{x}_n = \begin{bmatrix} |\boldsymbol{l}_1||\boldsymbol{x}_1|\cos(\theta_{11}) & \ldots & |\boldsymbol{l}_1||\boldsymbol{x}_{I_n}|\cos(\theta_{1 I_n}) \\ \vdots & \ddots & \vdots \\ |\boldsymbol{l}_{I_n}||\boldsymbol{x}_1|\cos(\theta_{I_n 1}) & \ldots & |\boldsymbol{l}_{I_n}||\boldsymbol{x}_{I_n}|\cos(\theta_{I_n I_n}) \end{bmatrix} \tag{20}$$

where θ_{ij} is the angle between the basis vector $\boldsymbol{l}_i$ and the basis vector $\boldsymbol{x}_j$. The subscript n has been omitted in the single matrix elements for the sake of conciseness and readability.

Equation 20 can be further revised by taking into account that– by definition—$|\boldsymbol{l}_{i_n}^{(n)}| = 1$ and $|\boldsymbol{x}_{i_n}^{(n)}| = 1$. Therefore, one can conclude that the following equation holds for the mode-n unfolding of $\tilde{\mathcal{X}}$, $\tilde{\mathcal{X}}_{(n)}$:

$$\tilde{\mathcal{X}}_{(n)} = \boldsymbol{L}_n^t \boldsymbol{X}_n \bar{\mathcal{X}}_{(n)} = \begin{bmatrix} \cos(\theta_{11}) & \ldots & \cos(\theta_{1 I_n}) \\ \vdots & \ddots & \vdots \\ \cos(\theta_{I_n 1}) & \ldots & \cos(\theta_{I_n I_n}) \end{bmatrix} \cdot \bar{\mathcal{X}}_{(n)} \tag{21}$$

The above equation clarifies that the discrepancies between $\tilde{\mathcal{X}}$ and the "actual" core tensor $\bar{\mathcal{X}}$ stem from the divergences between the bases $\boldsymbol{L}_n$ and $\boldsymbol{X}_n$. In particular, one has that

$$\boldsymbol{L}_n^t \boldsymbol{X}_n = \begin{bmatrix} \cos(\theta_{11}) & \ldots & \cos(\theta_{1 I_n}) \\ \vdots & \ddots & \vdots \\ \cos(\theta_{I_n 1}) & \ldots & \cos(\theta_{I_n I_n}) \end{bmatrix} = \boldsymbol{I} \tag{22}$$

only if the eigenvectors $\boldsymbol{l}_i$ and $\boldsymbol{x}_i$ are parallel for any $i = 1, .., I_n$ and if $\boldsymbol{l}_i$ is orthogonal to $\boldsymbol{x}_j$ for any couple (i, j). It is worth noting that $\tilde{\mathcal{X}}$ encompasses the information provided by the n-mode eigenvalues of $\mathcal{X}$ (as per (7)). As a consequence, one expects that $\tilde{\mathcal{X}}$ combines this information with the relative alignment of the eigenvectors of $\mathcal{L}$ and $\mathcal{X}$. This is an important issue because eigenvalues and eigenvectors may reveal details on geometric patterns in tensors [25]. In the following, Section "Toy Examples" analyzes this aspect by providing some practical examples that help to further clarify the characteristics of the proposed similarity notion. In terms of energy realignment the pseudo-core tensor $\tilde{\mathcal{X}}$ has a simple and clear explanation. It provides all the energy contribution of the tensor $\mathcal{X}$ along the extremal energy directions of $\mathcal{L}$.

Interestingly, Eq. 20 resembles the well-known cosine similarity function, as it involves the angle between the two basis vectors. However, in the proposed similarity notion, the terms $\{\boldsymbol{L}_n^t \boldsymbol{X}_n\}$ implement a realignment of the tensor along the directions selected by the eigenvalues, which is an intermediate step in the computation of the similarity notion. Thus, such terms do not implement the notion of similarity by themselves.

Toy Examples

In principle, the pseudo-core tensor of $\mathcal{X}$, $\tilde{\mathcal{X}}$, cannot be expected to inherit the properties that characterize an actual

core tensor. Thus, $\tilde{\mathcal{X}}$ usually is not an all-orthogonal, ordered tensor, nor can the Frobenius norm $\|\tilde{\mathcal{X}}_{(n)}(i_n,:)\|$ be interpreted as a proper eigenvalue. The expression (21), however, suggests that the quantities $\|\tilde{\mathcal{X}}_{(n)}(i_n,:)\|, (i_n = 1,..,I_n)$ convey useful information on the similarity between $\mathcal{L}$ and $\mathcal{X}$ along the nth mode. A practical example may prove useful.

First, let $\mathcal{L}$ and $\mathcal{X}$ be images, i.e., tensors of order 2 ($\mathcal{L}, \mathcal{X} \in R^{I_1 \times I_2}$). Then, one has

$$\mathcal{L} = \bar{\mathcal{L}} \times_1 \boldsymbol{L}_1 \times_2 \boldsymbol{L}_2, \tag{23}$$

$$\mathcal{X} = \bar{\mathcal{X}} \times_1 \boldsymbol{X}_1 \times_2 \boldsymbol{X}_2, \tag{24}$$

The images in Fig. 1a, b were used as the landmark, $\mathcal{L}$, and the datum, $\mathcal{X}$, respectively. In fact, the second tensor (image) was obtained by flipping vertically and rotating the first one. These operations did not affect the eigenvalues of the matrix $\mathcal{L}$. Figure 1c, d shows two different reconstructions of $\mathcal{X}$. Figure 1c was obtained as follows:

$$\mathcal{X}' = \bar{\mathcal{X}} \times_1 \boldsymbol{L}_1 \times_2 \boldsymbol{L}_2, \tag{25}$$

The reconstruction $\mathcal{X}'$ exploits the original bases provided by $\mathcal{L}$ (i.e., the eigenvectors of $\mathcal{L}$); the eigenvalues obviously stem from $\bar{\mathcal{X}}$. Figure 1d, instead, gives the eventual reconstruction of $\mathcal{X}$ obtained as follows:

$$\mathcal{X}'' = \tilde{\mathcal{X}} \times_1 \boldsymbol{L}_1 \times_2 \boldsymbol{L}_2, \tag{26}$$

The reconstruction $\mathcal{X}''$ still exploits the bases provided by $\mathcal{L}$, but the eigenvalues stem from $\tilde{\mathcal{X}}$ (in fact, they are "pseudo-eigenvalues"). Figure 1c shows that $\mathcal{X}' \equiv \mathcal{L}$. The reconstruction $\mathcal{X}'$ matches $\mathcal{L}$ since the landmark and the datum only differ in rotation. Therefore, $\mathcal{L}$ and $\mathcal{X}$ share the same eigenvalues ($\bar{\mathcal{L}}$ and $\tilde{\mathcal{L}}$, respectively), although they do not share the same bases. Conversely, the reconstruction $\mathcal{X}''$ matches $\mathcal{X}$ (as per Fig. 1d). This, in turn, means that $\bar{\mathcal{L}}$ and $\tilde{\mathcal{X}}$ do not convey the same information; i.e., they may reveal the dissimilarity between $\mathcal{L}$ and $\mathcal{X}$.

Figure 1e allows for further understanding of the role played by $\tilde{\mathcal{X}}$ in the representation formalized in Eq. 23. The reconstructions $\tilde{\mathcal{X}}'$ and $\tilde{\mathcal{X}}''$ correspond to two different implementations of Eq. 23, which differ in the settings of coefficients $\tilde{\sigma}_i^X$.

In $\mathcal{X}$, one uses the values $\|\bar{\mathcal{X}}_{(1)}(i,:)\|$; instead, in $\mathcal{X}''$, the coefficients are given by $\|\tilde{\mathcal{X}}_{(1)}(i,:)\|$. In the graph in Fig. 1e, the x axis gives the index i, with $i = 1,..,10$, whereas black, white, and gray bars gives the values $\|\tilde{\mathcal{X}}_{(1)}(i,:)\|$, $\|\bar{\mathcal{X}}_{(1)}(i,:)\|$, and $\|\bar{\mathcal{L}}_{(1)}(i,:)\|$ respectively. The values $\|\bar{\mathcal{L}}_{(1)}(i,:)\|$ and $\|\bar{\mathcal{X}}_{(1)}(i,:)\|$ correspond to the ten largest eigenvalues of $\bar{\mathcal{L}}_{(1)}$ and $\bar{\mathcal{X}}_{(1)}$, respectively. As expected, those values are ordered in decreasing order and are pairwise identical. On the other hand, the components $\|\tilde{\mathcal{X}}_{(1)}(i,:)\|$ can be regarded as some "special" eigenvalues that provide information about the process of reconstructing $\mathcal{X}$ by using the set of basis $\mathcal{L}_n$; in fact, the ten components are not strictly ordered from the largest to the smallest. A pairwise comparison between the values $\|\bar{\mathcal{X}}_{(1)}(i,:)\|$ and $\|\tilde{\mathcal{X}}_{(1)}(i,:)\|$ confirms that the relative weights assumed by the corresponding ten components differ. This explains the differences between $\mathcal{X}''$ and $\mathcal{X}'$.

Figure 2 presents the results of a similar experiment. In this case, $\mathcal{L}$ (Fig. 2a) and $\mathcal{X}$ (Fig. 2b) are two completely dissimilar images. Figure 2c, d shows the two different reconstruction of $\mathcal{X}$: $\mathcal{X}'$, and $\mathcal{X}''$, respectively. $\mathcal{X}'$ reveals that $\mathcal{L}$ and $\mathcal{X}$ do not share the eigenvalues (as expected); as a result, the reconstruction somewhat mixes the landmark with the datum. On the other hand, by definition the reconstruction $\mathcal{X}''$ matches $\mathcal{X}$, since a realignment of the basis has been involved. Thus, yet again, it is confirmed that $\bar{\mathcal{X}}$ and $\tilde{\mathcal{X}}$ convey different conclusions about the degree of similarity between $\mathcal{L}$ and $\mathcal{X}$. In this regard, Fig. 2e compares the values $\|\hat{\mathcal{L}}_{(1)}(i,:)\|$, $\|\bar{\mathcal{X}}_{(1)}(i,:)\|$, and $\|\tilde{\mathcal{X}}_{(1)}(i,:)\|$ by using the same format of Fig. 1e.

The availability of the similarity function introduced above, in turn, allows one to extend the hypothesis space (1) to tensorial data.

Tensor-Based Learning with Similarity Functions

In this section, the overall tensor-oriented framework inherited from the definition of (ϵ, γ)—good similarity function is proposed. In "Tensor-Based Similarity Function," the notion of similarity that support step 1 of Algorithm 1 is presented. Section "Computational Cost" provides a computational analysis of the framework. Finally, a comparison with the existing tensor-based frameworks is proposed.

Tensor-Based Similarity Function

The non-symmetric tensor-based similarity function $K(\mathcal{X}, \mathcal{L})$ evaluates the degree of similarity between a pattern, $\mathcal{X}$, and a landmark, $\mathcal{L}$, by actually processing $\tilde{\mathcal{X}}$ and $\bar{\mathcal{L}}$. By definition, any mode-n unfolding of $\bar{\mathcal{L}}$ encloses the n-mode eigenvalues of $\mathcal{L}$ (7); at the same time, Section "A Tensor-Based Similarity Notion" proved that it is convenient to interpret the Frobenius norm $\|\tilde{\mathcal{X}}_{(n)}(i_n,:)\|$ as an i_nth n-mode eigenvalue. As a result, the two sets of eigenvalues provide the inputs for a specific measure of similarity between $\mathcal{X}$ and $\mathcal{L}$.

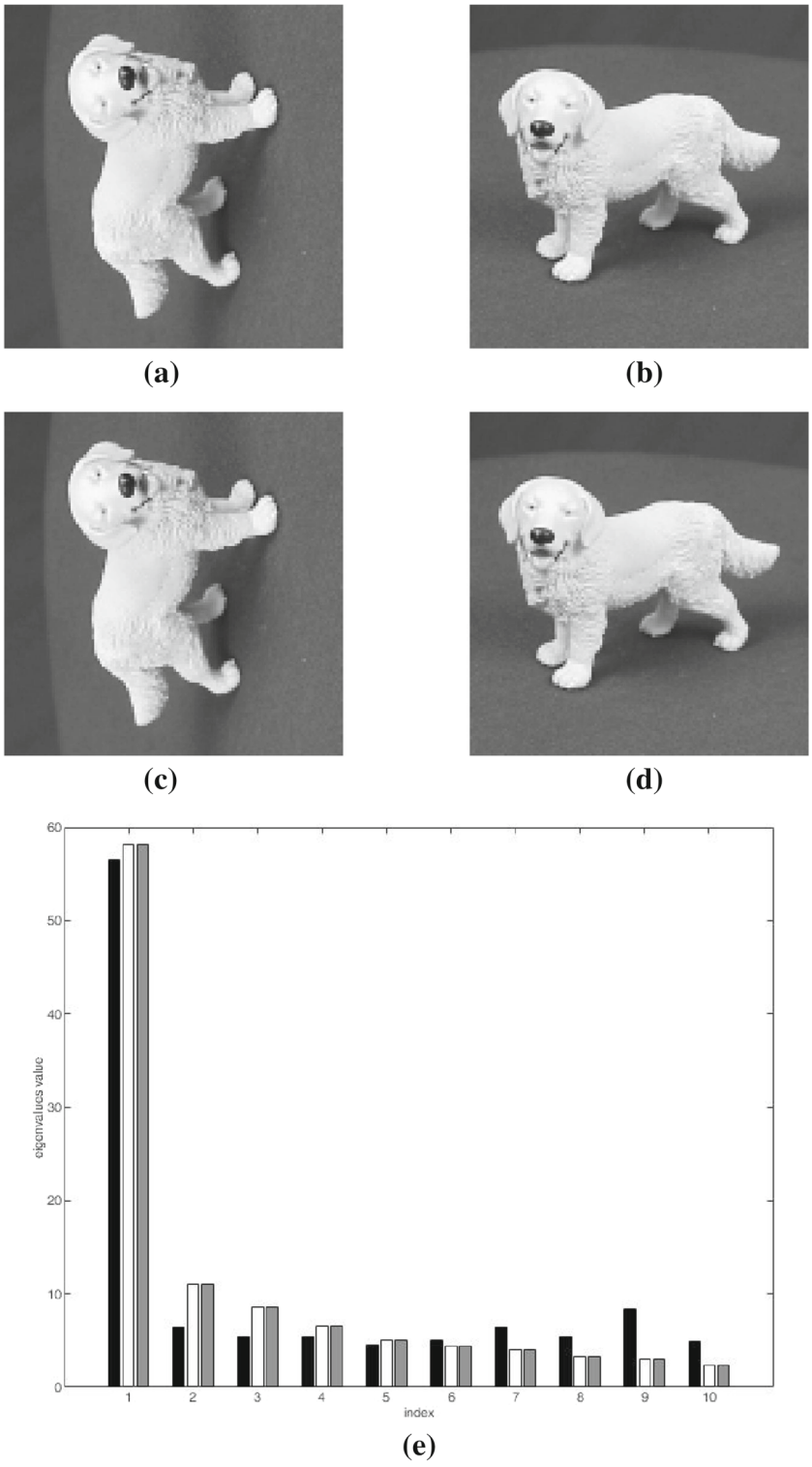

Fig. 1 Assessment of the degree of similarity between a landmark and a pattern according to the proposed notion of similarity, first example: **a** landmark $\mathcal{L}$; **b** pattern $\mathcal{X}$; **c** $\mathcal{X}'$; **d** $\mathcal{X}''$; **e** a plot that shows the ten largest eigenvalues of $\tilde{\mathcal{X}}_{(1)}$ (black bars), $\bar{\mathcal{X}}_{(1)}$ (white bars), and $\bar{\mathcal{L}}_{(1)}$ (gray bars)

The Algorithm 2 outlines the computation of $K(\mathcal{X}, \mathcal{L})$. After working out $\bar{\mathcal{L}}$ and $\tilde{\mathcal{X}}$ as per steps 0 and 1, step 2 collects the associate n-mode eigenvalues in $\mathbf{\Sigma}^{\mathcal{L}}_{(n)} = \{\sigma_{i_n(n)}; i_n = 1, ..., I_n\}$ and $\mathbf{\Sigma}^{\mathcal{X}}_{(n)} = \{\tilde{\sigma}_{i_n(n)}; i_n = 1, ..., I_n\}$, A pruning procedure at step 3 discards the least $\zeta_{(n)}$ informative eigenvalues. To ensure flexibility, an adaptive set size, $\zeta_{(n)}$, should be set for each mode, n. In step 4, the resulting eigenvalues feed the metric procedure that works out the actual result. The goal of the proposed similarity measure is the measure of the extremal energy magnitude

along the directions of the landmark. With this rationale, it is sufficient a comparison between the eigenvalues and the pseudo-eigenvalues:

$$K(\mathcal{X}, \mathcal{L}) = \frac{2}{1+\delta} - 1 \tag{27}$$

Algorithm 2 Computation of the tensor-based similarity function

Input

- a multiway pattern $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$
- a multiway landmark $\mathcal{L} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$
- pruning parameters $\zeta_{(n)} (n = 1, ..., N)$

0. **Initialize**

decompose $\mathcal{L}$ to obtain

- a core tensor $\bar{\mathcal{L}}$
- the orthonormal bases $\boldsymbol{L}_n (n = 1, ..., N)$

1. **Psudo-tensor**

compute the pseudo-tensor of $\mathcal{X}$

$$\tilde{\mathcal{X}} = \mathcal{X} \times_1 \boldsymbol{L}_1^t \times_2 \boldsymbol{L}_2^t ... \times_N \boldsymbol{L}_N^t$$

2. **Eigenvalues**

1. extract the n-mode eigenvalues from $\bar{\mathcal{L}} (n = 1, ..., N)$

$$\Sigma_{(n)}^{\mathcal{L}} = \{\sigma_{i_n(n)}; i_n = 1, ..., I_n\}, \quad \texttt{where } \sigma_{i_n(n)} = \|\bar{\mathcal{L}}_{(n)}(i_n, :)\|$$

2. extract the n-mode eigenvalues from $\tilde{\mathcal{X}} (n = 1, ..., N)$

$$\Sigma_{(n)}^{\mathcal{X}} = \{\tilde{\sigma}_{i_n(n)}; i_n = 1, ..., I_n\}, \quad \texttt{where } \tilde{\sigma}_{i_n(n)} = \|\tilde{\mathcal{X}}_{(n)}(i_n, :)\|$$

3. **Pruning**

for each mode, drop the last $\zeta_{(n)}$ eigenvalues (with $\zeta_{(n)} < I_n$)

$$\Sigma_{(n)}^{\mathcal{L}} = \{\sigma_{i_n(n)}; i_n = 1, ..., I_n - \zeta_{(n)}\},$$
$$\Sigma_{(n)}^{\mathcal{X}} = \{\sigma_{i_n(n)}; i_n = 1, ..., I_n - \zeta_{(n)}\}$$

4. **Similarity**

compute the similarity

$$K(\mathcal{X}, \mathcal{L}) = \frac{2}{1+\delta} - 1$$

where

$$\delta = \sqrt{\sum_{n=1}^{N} \sum_{i_n=1}^{I_n - \zeta_{(n)}} \frac{(\sigma_{i_n(n)} - \tilde{\sigma}_{i_n(n)})^2}{\sigma_{i_n(n)} \cdot \tilde{\sigma}_{i_n(n)}}}$$

where

$$\delta = \sqrt{\sum_{n=1}^{N} \sum_{i_n=1}^{I_n - \zeta_{(n)}} \frac{(\sigma_{i_n(n)} - \tilde{\sigma}_{i_n(n)})^2}{\sigma_{i_n(n)} \cdot \tilde{\sigma}_{i_n(n)}}} \tag{28}$$

The above metric compares the pairs $(\sigma_{i_n(n)}, \tilde{\sigma}_{i_n(n)})$, which stem from aligned basis constituted by the extremal energy directions, and thus can characterize the degree of similarity between $\mathcal{L}$ and $\mathcal{X}$, as showed in "A Tensor-Based Similarity Notion". The normalization term plays a crucial role in (28), as the eigenvalues in general can span a wide range of values. Thus, without a normalization term, one would face the risk of assessing δ by using only the 2 or 3 largest eigenvalues for each mode.

Remarkably, the proposed similarity function involves only one hyper-parameter per mode, i.e., the pruning parameter $\zeta_{(n)}$. It is worth noting that the parameters $\{\zeta_{(n)}; n = 1, .., N\}$ support a dimensionality reduction process that applies to the MLSVD-based characterizations of the pattern and the landmark. Such approach differs from the one implemented by standard dimensionality reduction process, such as latent semantic analysis (LSA) [26], which addresses vector spaces. In that case, SVD is used to capture the global characteristics of the whole training set. Conversely, the proposed similarity notion exploits MLSVD to obtain a suitable representation of the pattern itself (in a tensor space). This in turn means that one can better capture the specific properties of the single pattern. On the other hand, the risk of facing over fitting is higher. The pruning parameters are indeed designed to inhibit such problem.

The availability of the tensor-based function extends the hypothesis space (1) to tensorial data. Accordingly, one can apply Algorithm 1 to tensor-oriented problems directly and without any adjustment. In the research presented here, Step 2 in Algorithm 1 completes by solving a standard regularized least squares (RLS) [22] problem in the space ϕ.

Computational Cost

The above framework exhibits a common schema with the kernel-based framework for tensorial data introduced in [18]. The latter approach virtually extends any kernel machine to a tensor-based kernel machine. This goal is achieved by designing a suitable kernel that can exploit the algebraic structure of tensors. The decision function can then be learned by solving a single convex optimization problem. The proposed framework, in fact, exploits similarity functions to replace kernels, and the decision function can learned by solving a single RLS problem.

A comparison of the relative computational complexities reveals the basic difference between the two approaches. In

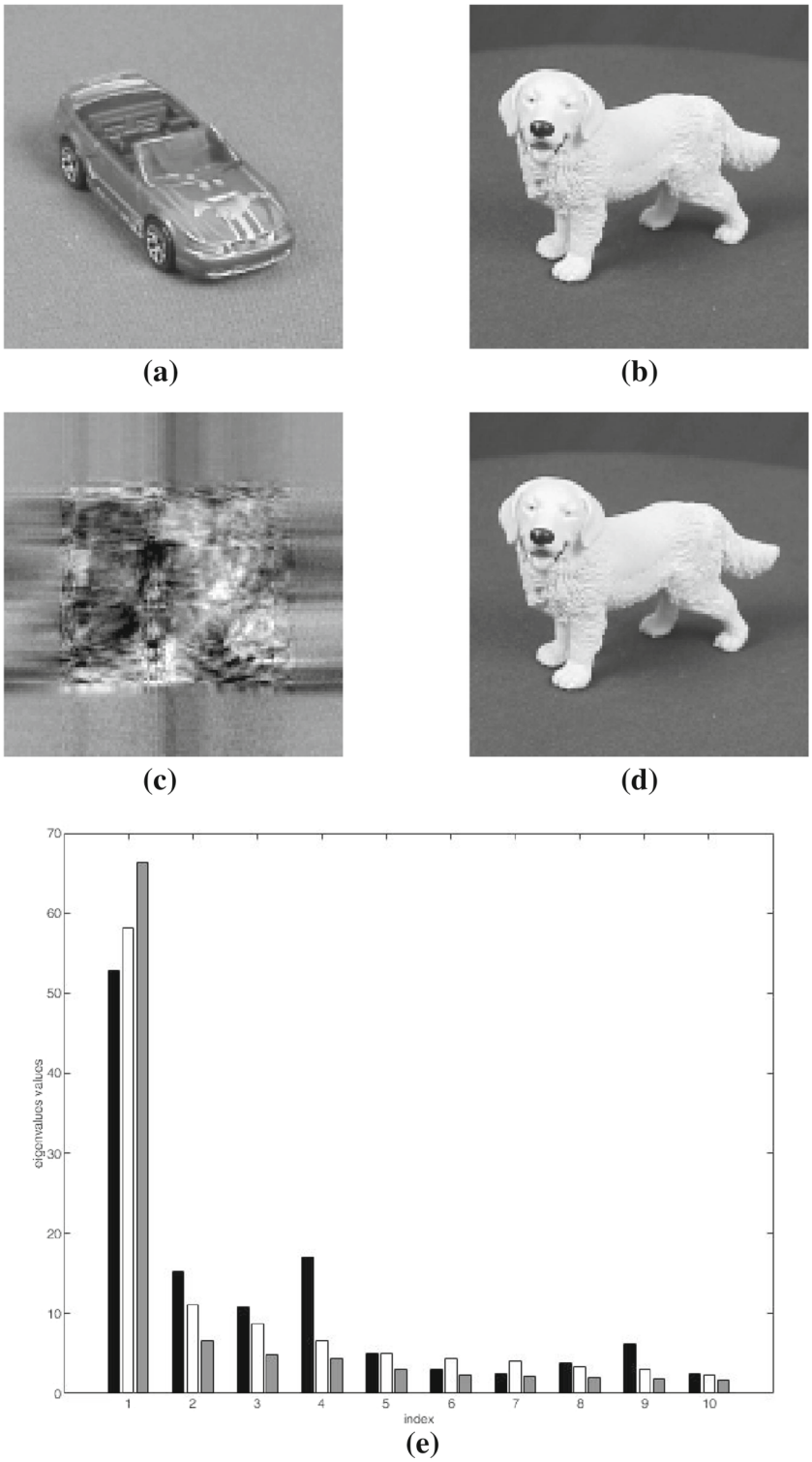

Fig. 2 Assessment of the degree of similarity between a landmark and a pattern according to the proposed notion of similarity, second example: **a** landmark $\mathcal{L}$; **b** pattern $\mathcal{X}$; **c**; $\mathcal{X}'$ **d** $\mathcal{X}''$ **e** a plot that shows the ten largest eigenvalues of $\tilde{\mathcal{X}}_{(1)}$ (black bars), $\bar{\mathcal{X}}_{(1)}$ (white bars), and $\bar{\mathcal{L}}_{(1)}$ (gray bars)

[18], the kernel function $\varphi(\mathcal{X}, \mathcal{P})$ that processes two generic tensors $\mathcal{X}, \mathcal{P}$ is formulated as follows:

$$\varphi(\mathcal{X}, \mathcal{P}) = \prod_{n=1}^{N} \varphi^{(n)}(\mathcal{X}, \mathcal{P}) \tag{29}$$

where

$$\varphi^{(n)}(\mathcal{X}, \mathcal{P}) = \exp\left(-\frac{1}{\sigma^2}(I_n - \text{trace}(\boldsymbol{Z}^t \boldsymbol{Z}))\right), \tag{30}$$

$$\boldsymbol{Z} = (\hat{\boldsymbol{V}}^{\mathcal{X}}_{(n)})^t (\hat{\boldsymbol{V}}^{\mathcal{P}}_{(n)}), \tag{31}$$

In the expression (31), $\hat{\boldsymbol{V}}_{(n)}^{\mathcal{X}}$ is the matrix computed by applying SVD to $\mathcal{X}_{(n)}$ as per (4), and is obtained by picking the first r columns of $\hat{\boldsymbol{V}}_{(n)}^{\mathcal{X}}$, with $r = \texttt{rank}(\mathcal{X}_{(n)})$. A similar procedure applies to $\hat{\boldsymbol{V}}_{(n)}^{\mathcal{P}}$. Thus, the computation of $\psi(\mathcal{X}, \mathcal{P})$ requires a set of $2N$ SVDs (as both $\mathcal{X}$ and $\mathcal{P}$ are processed). Then, two matrix products for each $\varphi(\mathcal{X}, \mathcal{P})$ must be performed. Therefore, the cost O_{KF} associated to the computation of $\varphi(\mathcal{X}, \mathcal{P})$ is as follows:

$$O_{\text{KF}} \cong 2 \cdot N \cdot O_{\text{SVD}} + 2 \cdot N \cdot O_{\text{MP}} \tag{32}$$

For the training of the overall kernel machine, one uses $\varphi(\bullet, \bullet)$ to compute the symmetric kernel matrix. Thus, the computational cost O_{TGK} associated to the kernel building can be estimated as follows:

$$O_{\text{TGK}} \cong Z \cdot N \cdot O_{\text{SVD}} + N \cdot Z^2 \cdot O_{\text{MP}} \tag{33}$$

The implementation of the kernel-based decision function requires a set of Z inner products $\varphi(\bullet, \bullet)$, involving the test pattern versus all training patterns. On one hand, the MLSVD result of each training pattern is computed offline and is stored in memory; however, the MLSVD of the test pattern has to be worked out online. Thus, the computational cost O_{TTK} associated to such step is as follows:

$$O_{\text{TTK}} \cong N \cdot O_{\text{SVD}} + 2 \cdot N \cdot Z \cdot O_{\text{MP}} \tag{34}$$

In the proposed framework, the computational complexity associated to the similarity function $K(\mathcal{X}, \mathcal{L})$ can be estimated easily (as per Algorithm 2). One first needs a set of N SVD procedures to complete Step 0. Then, N matrix products are required to compute $\tilde{\mathcal{X}}$ (Step 1). The computations involved by subsequent steps are negligible in terms of computational complexity as compared with Step 0 and Step 1. The overall cost, O_{SF}, associated with the computation of $(\mathcal{X}, \mathcal{L})$ is as follows:

$$O_{\text{SF}} \cong N \cdot O_{\text{SVD}} + N \cdot O_{\text{MP}} \tag{35}$$

One might argue that associating a uniform cost, O_{SVD}, to any SVD is incorrect, since each unfolding of $\mathcal{L}$ has its own size, which in turn affects the subsequent SVD. The same consideration applies to O_{MP}. This issue, however, only gets relevant when one wants to formalize the relationship between O_{SF} and $(I_1, I_2, .., I_n)$.

To address the training of eventual predictor (1), one applies $K(\mathcal{X}, \mathcal{L})$ to complete the mapping of the input patterns in the reduced space R^L. Thus, as per Step 1 of Algorithm 1, one completes a set of L MLSVDs (one for each landmark). Then, the Z input patterns are finally remapped. In summary, the cost of the prediction training is as follows:

$$O_{\text{TGS}} \cong L \cdot N \cdot O_{\text{SVD}} + L \cdot Z \cdot N \cdot O_{\text{MP}} \tag{36}$$

This means that $O_{\text{TGS}} = O_{\text{TGK}}$ when, in Algorithm 1, one sets all training patterns as landmarks (i.e., $L = Z$).

After training is completed, the implementation of the decision function (1) only requires the availability of the landmark bases $\boldsymbol{L}_n$, which are stored in memory. This is a major difference with respect to the decision function proposed in [13]. Thus, when a test pattern is classified, the computational cost associated to the mapping stage is

Clearly, O_{TTS} can prove significantly lower than O_{TTK}, which is made heavier by the need of computing the MLSVD of the test pattern [24]. This aspect becomes crucial when targeting the implementation of the predictor on a digital architecture, since the SVD can prove quite demanding in terms of computational complexity.

$$O_{\text{TTS}} \cong L \cdot N \cdot O_{\text{MP}}. \tag{37}$$

Comparison with Existing Literature

The proposed framework exhibits distinctive features as compared with existing approaches to tensor-based learning. First, the framework differs from conventional approaches that complete learning by an iterative process [13–16]. In general, these approaches solve a convex optimization problem at each step; hence, the learning processes are characterized by a huge computational cost. Nonetheless, one should deal with the setup of free parameters related to the convergence criterion.

SHTM [17] reformulates the iterative approach implemented by supervised tensor learning (STL) [14] to the purpose of obtaining a framework that inherits the linear C-SVM format [17]; as a result, the eventual learning process involves a single, standard convex optimization problem. The SHTM framework exploits CANDECOMP/PARAFAC (CP) [27] decomposition to suitably implement the inner product between tensors, which are replaced by their rank-one decomposition. Thus, a comparison between SHTM and the proposed framework reveals two basic differences: first, SHTM uses tensor decomposition specifically to address the inner product computation in a SVM-based learning scheme; secondly, SHTM yields a predictor that requires a (computationally demanding) decomposition also for classifying a test pattern at run time. Tensor decomposition also plays a role in projection learning methods [10–12], which aim at learning bases that can suitably capture manifolds in a training set. Most of such approaches convey a minimization problem over the entire training set. The proposed framework, instead, by applying MLSVD in conjunction with landmarks attains two goals: the manifold proprieties of the training corpus are characterized, and the peculiar properties of each pattern are preserved. Algorithm 2 tackles the risk of overfitting by a pruning mechanism on the eigenvectors (as per Step 3).

Convolutional neural networks (CNNs) [28] are indeed designed to deal with nth order tensors by exploiting

convolutional operations. The main advantage of CNNs with respect to shallow architectures (i.e., all the paradigms considered in this paper) is the hierarchical organization of the information supported by the multi-layer outline. This feature proves very effective in domains where patterns embeds structured information, e.g., image processing and video processing. On the other hand, CNNs are computationally demanding and requires very large datasets for the training phase. Thus, shallow architectures represent the only viable option when one targets implementations on resource-constrained devices.

Experimental Results

The experimental verification of the proposed paradigm aimed at evaluating the accuracy of the tensor-based learning scheme derived from the theory of similarity function. The overall framework was tested on seven classification problems involving multi-way data. The first problem used a synthetic dataset that had already been used in [18] for estimating the performance of the tensor kernel (29). The other six problems addressed real-world domains, namely, ETH-80 [29], Yale faces [30], Flower [31], KTH [32], Cambridge hand festure [33], and gas sensors [34].

The experiments were all designed to assess the generalization performance of the proposed tensor-based classifier. At the same time, the tests supported comparisons with three alternative solutions: SHTM [17]; a support vector machine (SVM) that processed multi-way data by relying on the tensor kernel (29); a standard SVM that first remapped multi-way data into vector representations and then adopted the conventional RBF kernel. The first two solutions refer to state-of-the-art approaches that address tensor-based learning by involving a single, standard convex optimization problem. The latter solution allows one to understand the performance of a conventional approach to the problem.

In each experiment, the proposed framework, SHTM, the tensor-based SVM, and the basic SVM were compared by defining a common set up for the range of admissible values of the following:

- λ, i.e., the regularization parameter ($C = 1/\lambda$ for SVM's)
- σ, i.e., the hyper-parameter of both the standard RBF kernel and the tensor kernel (29)
- R, i.e., the rank parameter of SHTM

The following settings were adopted:

- $\lambda \in \{1 \cdot 10^{-6}, 1 \cdot 10^{-5}, 1 \cdot 10^{-4}, 1 \cdot 10^{-3}, 1 \cdot 10^{-2}, 1 \cdot 10^{-1}, 1, 1 \cdot 10^{1}, 1 \cdot 10^{2}, 1 \cdot 10^{3}, 1 \cdot 10^{4}, 1 \cdot 10^{5}, 1 \cdot 10^{6}\}$
- $\sigma \in \{1 \cdot 10^{-6}, 1 \cdot 10^{-5}, 1 \cdot 10^{-4}, 1 \cdot 10^{-3}, 1 \cdot 10^{-2}, 1 \cdot 10^{-1}, 1, 1 \cdot 10^{1}, 1 \cdot 10^{2}, 1 \cdot 10^{3}, 1 \cdot 10^{4}, 1 \cdot 10^{5}, 1 \cdot 10^{6}\}$
- $R \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

Conventional model-selection procedures always drove the run-time settings of $\{\lambda, \sigma\}$ (or $\{\lambda, R\}$) for evaluating generalization performance on unseen test data. In the case of the proposed framework, model selection also supported the setting of the pruning parameters, $\zeta_{(n)}$, used in Algorithm 2; in addition, all the patterns included in the training set served as landmarks (i.e., $L = Z$).

The following sections present the results of the experimental sessions and address the performance of the three different solutions on each dataset.

Synthetic Dataset: Classification of Sparsity Patterns

The first domain addressed the classification of third-order tensors characterized by two different types of sparsity patterns. This synthetic dataset was used in [18] for demonstrating the generalization performance of a SVM adopting the tensor kernel (29).

The patterns were generated as follows. Let $\boldsymbol{e}_j \in R^I$ denote the jth canonical basis vector with $e_{\text{ij}} = 1$ if $i = j$ and $e_{\text{ij}} = 0$ otherwise. In addition, let $\mathcal{D}_j \in R^{I \times I \times I}$ be the rank-1 tensor defined as: $\mathcal{D}_j = \boldsymbol{e}_j \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_j$. Then, the mth pattern, $\mathcal{X}_m$, was computed as follows:

$$\mathcal{X}_m = \begin{cases} a_m \mathcal{D}_1 + b_m \mathcal{D}_2 + c_m \mathcal{D}_3 + \mathcal{G}_m, & \text{if } y_m = 1. \\ a_m \mathcal{D}_4 + b_m \mathcal{D}_5 + c_m \mathcal{D}_6 + \mathcal{G}_m, & \text{if } y_m = -1. \end{cases} \quad (38)$$

In Eq. 38, a_m, b_m, and c_m were i.i.d. from a zero-mean Gaussian distribution with variance $1 - \beta^2$; $\mathcal{G}_m$ was a noise tensor, whose entries were i.i.d. from a zero-mean Gaussian distribution with variance β^2.

The experimental dataset was generated by adopting $I = 3$ and $\beta^2 = 0.05$. Each experiment conveyed a binary classification problem with a balanced training set and a balanced test set. The size of the training set took on the values $\{10, 20, 50, 100, 200\}$; the test set always included 100 samples. For each size of the training set, ten different runs were completed, requiring ten different pairs of training/test sets.

Table 1 gives the experimental outcomes obtained: rows relate to the size of the training set; columns refer to the four classifiers. The performance was assessed by averaging the classification error (expressed in the range $[0, 1]$) over the ten runs; the associate standard deviation is given in brackets. Empirical results point out that the proposed framework outperformed both SHTM and the standard SVM. The SVM based on the tensor kernel only outperformed the proposed framework when very limited training sets were involved.

Table 1 Synthetic dataset: results of the experimental session: the four algorithms are compared in term of average accuracy (%) paired with standard deviation

Size	Proposed framework	SHTM	Multi-way SVM	Standard SVM
10	0.005 (0.009)	0.3275 (0.087)	*0.000* (0.002)	0.172 (0.085)
20	0.003 (0.004)	0.3285 (0.058)	*0.000* (0.002)	0.081 (0.056)
50	*0.000* (0.001)	0.3185 (0.057)	0.000 (0.002)	0.023 (0.019)
100	*0.000* (0.000)	0.3595 (0.067)	*0.000* (0.000)	0.012 (0.010)
200	*0.000* (0.000)	0.3605 (0.063)	*0.000* (0.000)	0.006 (0.007)

ETH-80

The ETH-80 database [29] contains images of objects grouped into eight categories (apples, pears, tomatoes, cars, cups, cows,horses, dogs). Each category covers ten objects that span large in-class variations, while still clearly belonging to the category. Each object is represented by 41 color images, which correspond to as many viewpoints equally spaced over the upper viewing hemisphere. All images are sized 128 × 128 pixels. Accordingly, each of them is represented as a third-order tensor $R^{128\times128\times3}$.

The session was designed to evaluate the performance on the several pairwise binary problems that stem from this dataset. Each experiment targeted the classification problem "category A" versus "category B"; hence, a total of 28 tests were completed to cover all combinations.

In each experiment, six objects were randomly drawn and included in the training set (three for each category); thus, the training set held $6 \times 41 = 246$ patterns. The test set included one randomly extracted object per class, for a total of $2 \times 41 = 82$ test patterns. The training set and the test set never shared any pattern. Ten runs per experiment were completed, requiring ten different pairs of training/test sets.

Figure 3 shows the results of the overall set of experiments. In the graph, the x-axis marks each binary classification problem, whereas the y-axis gives the average classification error over the ten runs (expressed in the range [0, 1]). For the sake of clarity, the graph plots—for each single experiment— two quantities: the classification error scored by the proposed framework (black thick line) and the best classification error achieved among the three remaining frameworks (dashed gray line). The graph points out that the proposed approach was never outperformed by state-of-the-art approaches. In several cases, the classification error scored by the proposed approach was significantly smaller. To amplify on that, Table 2 provides a detailed analysis of these results, aimed at assessing the statistical significance of the results from each experiment. In a given test, predictor A was considered "better than" predictor B only if

$$\bar{\mu}_A + \bar{\sigma}_A/\sqrt{10} < \bar{\mu}_B - \bar{\sigma}_B/\sqrt{10} \tag{39}$$

where $\bar{\mu}$ and $\bar{\sigma}$ are the sample mean and the sample standard deviation, respectively, worked out on the ten classification errors. The expression (39) takes into account the standard error in the computation of the average classification error (i.e., the sample mean). The columns of Table 2 give the following quantities:

- ν_B: the number of experiments in which the proposed framework proved to be the best predictor, as per (39);
- ν_{BNE}: the number of experiments in which the proposed framework still scored the lowest classification error, but (39) did not hold.
- ν_{BH}: the number of experiments in which the proposed framework proved better than SHTM, as per (39);
- ν_{BT}: the number of experiments in which the proposed framework proved better than tensor kernel SVM, as per (39).
- ν_{BS}: the number of experiments in which the proposed framework proved better than standard SVM, as per (39).
- μ_G: the average improvement in classification error attained by the proposed framework. This quantity was computed by considering only the ν_B experiments in which the method proved to be the best predictor. The gain always referred to the second-best comparison.

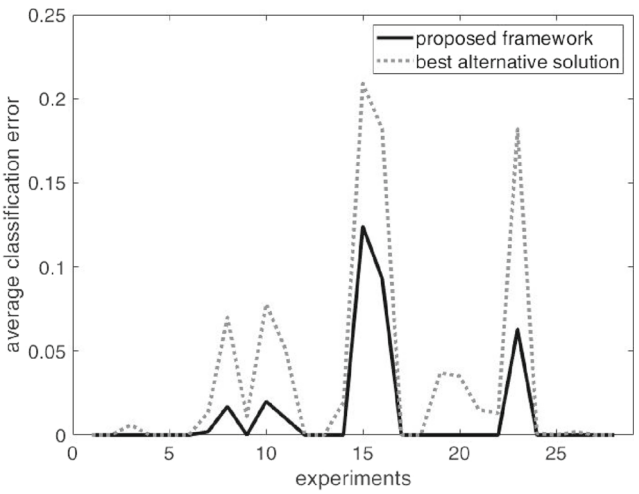

Fig. 3 ETH-80 dataset: results of the 28 experiments on binary classification. The graph compares, for each experiment, the proposed framework with the best solution achieved among state-of-the-art frameworks

Table 2 ETH-80 dataset: analysis of the results of the experimental session

ν_B	ν_{BNE}	ν_{BH}	ν_{BT}	ν_{BS}	μ_G
12(42.8%)	16(57.1%)	16(57.1%)	17(60.7%)	15(53.5%)	0.05

Table 3 Yale faces dataset: analysis of the results of the experimental session

ν_B	ν_{BNE}	ν_{BH}	ν_{BT}	ν_{BS}	μ_G
15(14.3%)	81(77.1%)	22(20.9%)	105(100%)	25(23.8%)	0.06

The table also provides—in brackets—the same quantities expressed as percentage over the total number of experiment. The results confirm the effectiveness of kernel-based learning with similarity functions, which in almost half of the experiments scored as the best overall predictor; the average gain in classification error was 0.05.

Yale Faces

The Yale face database [30] contains 165 grayscale images of 15 individuals. There are 11 images per subject, 1 per different facial expression or configuration. All images have size 243 × 320 pixels; the present experimental session, tough, exploited the processed data set already used in [35]. Accordingly, each image was represented as a second-order tensor $R^{64\times 64}$.

The experimental session addressed the binary classification problems "subject A" versus "subject B" (105 experiments in total). In each experiment, the training set was generated by randomly drawing nine images per class; the test set included the remaining two images per class. The training set and the test set never shared any pattern. A total of ten different runs per experiment were completed; hence, ten different training/test pairs of sets were generated. Figure 4 gives the results of the overall set of experiments and uses the same the format as per Fig. 3. The x-axis marks single experiments; the y-axis gives the classification error on the test sets. The plot shows that the proposed approach was never outperformed by state-of-the-art approaches. Indeed, Table 3 provides further details on the experimental outcomes by using the same descriptors as per Table 2. The proposed framework often scored the lowest classification error; nonetheless, frequently, the second-best predictor was very close to such performance (as per BNE). This result may indicate that the Yale faces database does not provide a very challenging problem in general.

Flower

The flower database [31] gathers images for 17 different categories of flowers; 80 color images per category are provided. The images were characterized by changes in scale, pose, and light variations; in addition, a single class may include images with a single flower in the

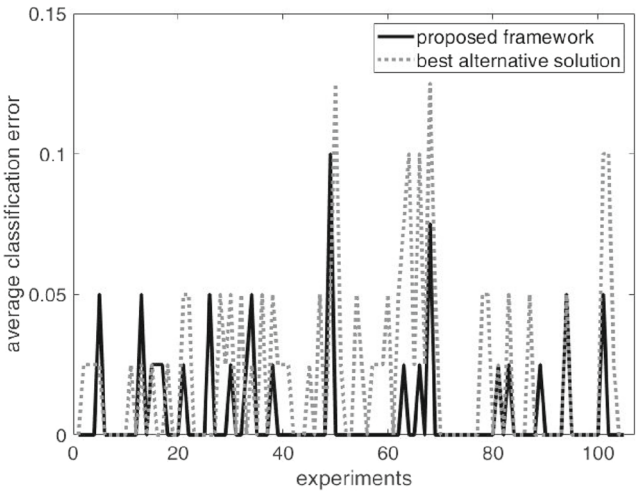

Fig. 4 Yale faces dataset: results of the 105 experiments on binary classification. The graph compares, for each experiment, the proposed framework with the best solution achieved among state-of-the-art frameworks

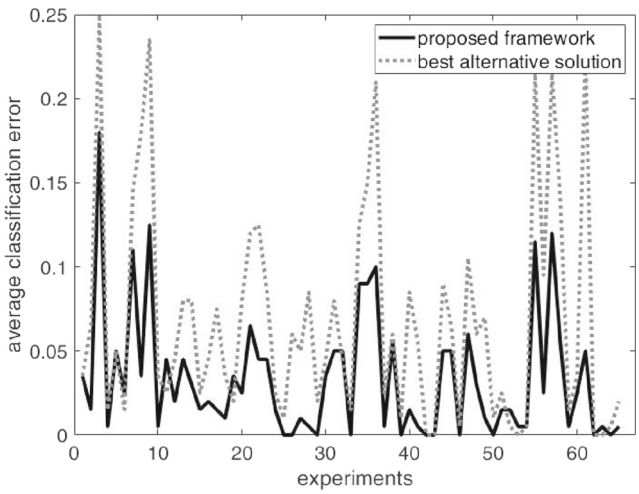

Fig. 5 Flower dataset: results of the 66 experiments on binary classification. The graph compares, for each experiment, the proposed framework with the best solution achieved among state-of-the-art frameworks

Table 4 Flower dataset: analysis of the results of the experimental session

ν_B	ν_{BNE}	ν_{BH}	ν_{BT}	ν_{BS}	μ_G
36 (55.4%)	22 (33.8%)	47 (72.3%)	58 (89.2%)	46 (70.8%)	0.06

foreground, as well as images with groups of flowers of the same class. In the experimental session, the database was pruned by only keeping those images having one flower in the foreground. As a result, five categories had to be removed from the database, since the number of acceptable pictures was too small. All the pictures were resized to obtain a unique format (150 × 150 pixels). Thus, each pattern was represented as a third-order tensor $R^{150\times150\times3}$.

The experimental session evaluated the performances on the binary classification problems "category A" versus "category B" (66 experiments in total). In each test, the training set was generated by randomly drawing 20 patterns per class; the test set included 10 random patterns. The training set and the test set never shared any pattern. A total of ten different runs per experiment were completed; hence, ten different training/test pairs of sets were generated.

Figure 5 gives the results of the overall set of experiments with the same format used in Fig. 3. The graph points out that the proposed framework often compared favorably with state-of-the-art frameworks. Table 4 provides a deeper statistical analysis of the experimental outcomes with the usual descriptors. The table confirms the effectiveness of proposed framework, which scored as the best overall predictor in more than half of the experiments. In addition, the average gain classification error was 0.06.

KTH

The KTH database [32] includes six types of human actions (walking, jogging, running, boxing, hand waving and hand clapping) performed by 25 subjects. Four different scenarios were involved: outdoor, outdoor with scale variation, outdoor with different clothes, indoor. All sequences were taken over homogeneous backgrounds with a static camera (frame rate: 25 fps). In the original database, the frame size was 160 × 120 pixels; the length of the videos varied. In the present experiment, the videos were all resized to 30 × 30 pixels. Accordingly, each pattern was represented as a third-order tensor $R^{30\times30\times I}$, with I varying according to each video.

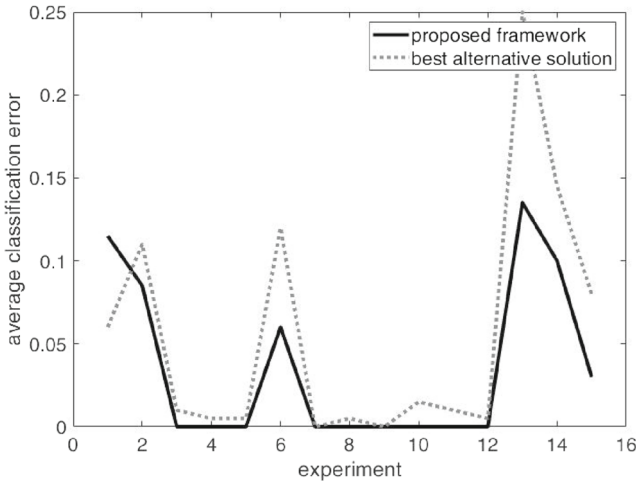

Fig. 6 KTH dataset: results of the 15 experiments on binary classification. The graph compares, for each experiment, the proposed framework with the best solution achieved among state-of-the-art frameworks

The experimental session evaluated the performance of the proposed framework on the binary classification problems "action A" versus "action B" (15 experiments in total). In each experiment, the training set was generated by randomly drawing 30 patterns per class; the test set included 10 random patterns per class. The training set and the test set never shared any pattern. I was always set according to the shortest sequence included in the experiment (training and test); all the sequences originally including a number of frames greater than I were edited by removing all frames after the Ith one. Ten different runs per experiment were completed; hence, ten different pairs of training/test sets were generated.

Figure 6 gives the results for the overall set of experiments with the same format used in Fig. 3. The graph confirms the reliability of the proposed framework. Table 5 gives further statistical parameters and points out that the proposed framework scored the lowest classification error in 14 experiments out of 15 (best predictor in 7 experiments out of 15 if one only consider B); the average improvement in classification error was 0.04. Table 5 indeed shows that only the tensor-based SVM did proved better than the proposed approach in a few tests. Conversely, both SHTM and the basic SVM never attained the performances scored by the proposed approach.

Cambridge Hand Gesture

The Cambridge hand gesture database [33] includes 900 image sequences that stemmed from 9 gesture classes.

Table 5 KTH dataset: analysis of the results of the experimental session

ν_B	ν_{BNE}	ν_{BH}	ν_{BT}	ν_{BS}	μ_G
7 (46.7%)	7 (46.7%)	14 (93.3%)	7 (46.7%)	15 (100%)	0.04

Table 6 Hand gesture dataset: analysis of the results of the experimental session

ν_B	ν_{BNE}	ν_{BH}	ν_{BT}	ν_{BS}	μ_G
3(8.3%)	15 (41.7%)	35 (97.2%)	3 (8.3%)	34 (94.4%)	0.04

The 9 gestures were defined by 3 primitive hand shapes and 3 primitive motions. Each class contained 100 image sequences, which had been recorded by involving 2 different subjects, 10 arbitrary motions, and 5 different illuminations. Each sequence was recorded in front of a fixed camera having roughly isolated gestures in space and time. In the present experiment, images were all resized to the uniform format 40 × 40 pixels. As a result, each image sequence was represented as a third-order tensor $R^{40\times 40\times I}$. Actually, I varied for each sequence.

The session followed the approach commonly applied in the literature for this dataset and evaluated the performance of the proposed framework on the binary classification problems "gesture A" versus "gesture B" (36 experiments in total). For each experiment, a training set was generated by including all the image sequences captured with the plain illumination setting (20 patterns per class). The corresponding test set included all the image sequences captured with the remaining illuminations (80 patterns per class). In each experiment, I was set according to the shortest sequence included in the dataset (training and test). As a result, all the sequences originally including a number of images greater than I were subsampled.

Figure 7 gives the results of the overall set of experiments and uses the same the format as per Fig. 3. The x-axis marks single experiments; the y-axis gives the classification error on the test sets. The results show that, in this case, the proposed framework gained the role of best predictor only in few experiments. Table 6 provides further insights on the experimental outcomes. In the present setup, however, the standard error associated to the classification error was null, since experiments did not involve multiple runs. The quantity B just counted the experiments in which the proposed framework scored the lowest classification error; the quantity BNE counted the cases in which the proposed framework proved to the best predictor as well as another predictor (featuring the same classification error). The table proves that, again, only the tensor-based SVM was able to compete with the proposed framework.

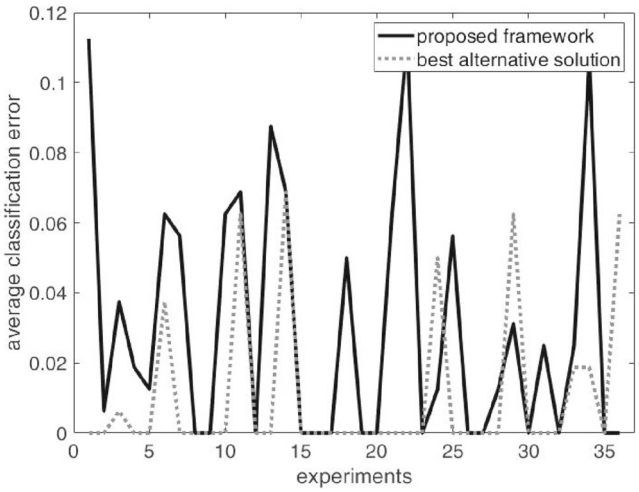

Fig. 7 Hand Gesture dataset: results of the 36 experiments on binary classification. The graph compares, for each experiment, the proposed framework with the best solution achieved among state-of-the-art frameworks

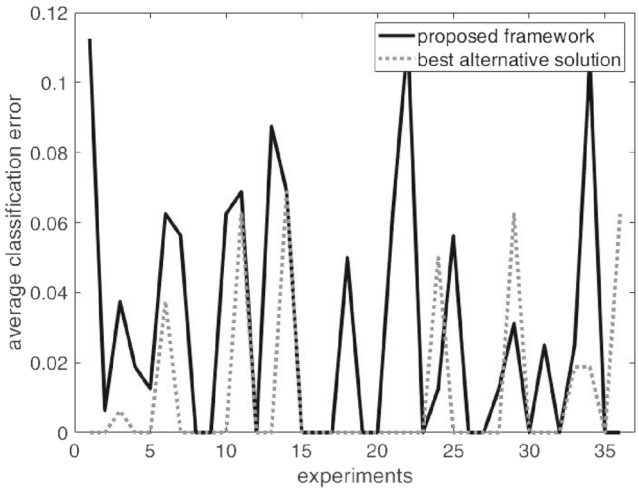

Fig. 8 Gas sensor dataset: results of the 16 experiments on binary classification. The graph compares, for each experiment, the proposed framework with the best solution achieved among state-of-the-art frameworks

Table 7 Gas sensor dataset: analysis of the results of the experimental session

ν_B	ν_{BNE}	ν_{BH}	ν_{BT}	ν_{BS}	μ_G
0 (0.0%)	4 (26.7%)	7 (46.7%)	0 (0.0%)	2 (13.3%)	0.0

Gas Sensors

The gas sensors database [34] includes 13,910 measurements from 16 chemical sensors exposed to 6 gases at different concentration levels. The single measurement provided the response of the sensors when exposed to a given gas at a given concentration; 8 features per sensor were used. The measurements were gathered during 36 months. In the present experimental session, the data stemming from a single measurement were organized as a second-order tensor $R^{16\times 8}$ (i.e., the 16 sensors with the corresponding 8 features). Experiments addressed the binary classification problem "gas A" versus "gas B" (16 experiments in total). Thus, concentration levels were not involved in the experiment. The balanced training set always included 25 samples per class; the test set included 100 samples per class. Ten runs were performed; in each run, the training set and the test set were generated by randomly sampling the available dataset. The training set and the test set never shared any pattern.

Figure 8 gives the results of the overall set of experiments and uses the same the format as per Fig. 3. The x-axis marks single experiments; the y-axis gives the classification error on the test sets. The results show that, in general, the proposed framework achieved the performance scored by the best solution among the three competitors. In fact, only in a few cases one of the competitors was able to slightly outperform the proposed framework. Table 7 gives further statistical parameters and confirms that the proposed framework never proved to be the best overall predictor with a statistical significance.

Conclusion

This paper investigated how the theory of learning with similarity functions can support the development of a cognitive-inspired framework that deals with multi-way data inherently. The design of an original, effective notion of similarity between tensors indeed represents the core feature of the proposed research.

The notion of similarity is built on the decomposition of a tensor into two components: the core tensor and the corresponding orthonormal basis. The degree of similarity between a pattern and a landmark is then assessed by taking into due account the alignment between the respective basis. This, in turn, means that similarity is not just estimated by conventionally comparing the standard n-mode eigenvalues of the two tensors; this procedure in fact can only provide partial information on the degree of resemblance between the tensors. Nonetheless, it is worth to note that the proposed framework can utilize MLSVD to characterize the intrinsic structural properties of each single landmark; as a major result, one can more reliably capture the underlying domain distribution.

Experimental results confirm the effectiveness of the proposed framework, which compared favorably with both a basic SVM and two state-of-the-art frameworks that can inherently process tensors: SHTM and the tensor-based SVM classifier model presented in [18]. Only the latter predictor proved able to compete with the proposed framework in terms of classification performances. This paper, though, showed that the proposed framework can be more effective in terms of computational costs.

Future research work will address the design of new tensor-based similarity functions, to the purpose of enlarging the collection of available similarity notions. This will increase the ability of the proposed framework to tackle an even larger variety of application domains that require real-time multi-way data processing, including OCR [36], image processing [37], and common sense reasoning [38].

Compliance with Ethical Standards

Conflict of Interest The authors declare that they have no conflict of interest.

Informed Consent Informed consent was not required as no human or animals were involved.

Human and Animal Rights This article does not contain any studies with human or animal subjects performed by any of the authors.

Appendix

Unfolding is the process of reordering the elements of a tensor $\mathcal{T} \in R^{I_1\times I_2\times \ldots \times I_N}$ into a matrix $\mathcal{T} \in R^{P\times Q}$, where $P \times Q = I_1 \times I_2 \times \ldots \times I_N$.

The mode-n unfolding of $\mathcal{T}$, $\mathcal{T}_{(n)}$, defines a specific instance of such a procedure [20]. Accordingly, $\mathcal{T}_{(n)} \in R^{I_n\times I_1 I_2 I_{n-1} I_{n+1}\ldots I_N}$ and the reordering process follows the scheme exemplified in Fig. 9. In the example, a third-order tensor $\mathcal{T}$ is unfolded in the matrices $\mathcal{T}_{(1)}$, $\mathcal{T}_{(2)}$, and $\mathcal{T}_{(3)}$.

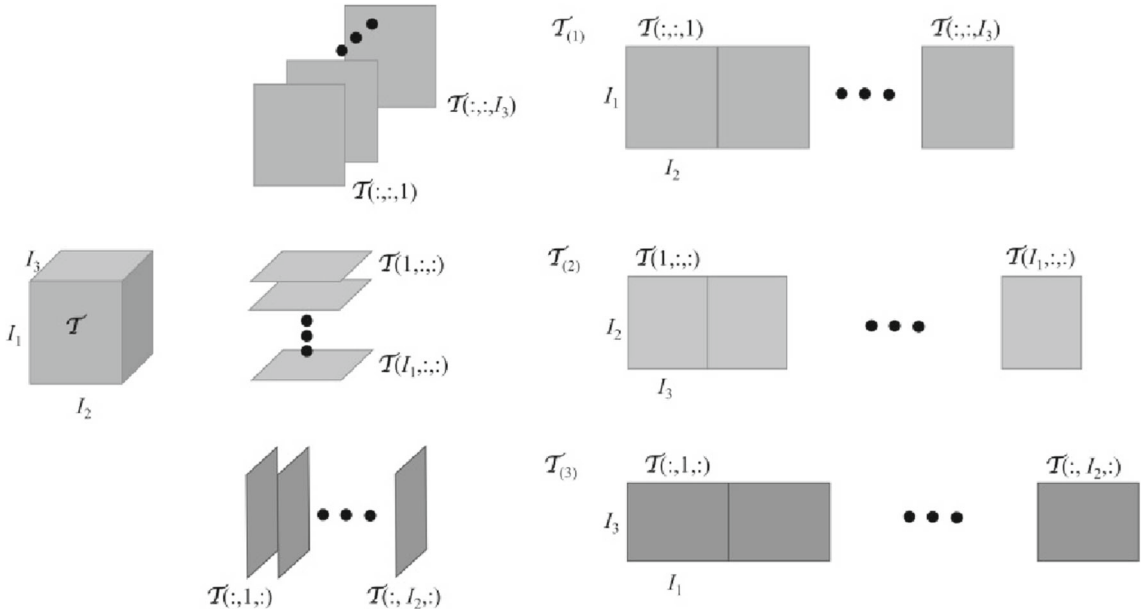

Fig. 9 Unfolding of a third-order tensor

The mode-1 unfolding $\mathcal{T}_{(1)}$ is obtained by first partitioning the original tensor according to the slices $\mathcal{T}(:,:,i3)$. Then, the eventual matrix is created by placing side-by-side the slices in the original order. Likewise, $\mathcal{T}_{(2)}$ and $\mathcal{T}_{(3)}$ are obtained by partitioning the original tensor according to the slices $\mathcal{T}(:,i_2,:)$ and $\mathcal{T}(i_1,:,i3)$, respectively. Such scheme can be easily extended to a generic tensor of order *N*.

References

1. Cambria E, White B. Jumping nlp curves: A review of natural language processing research. IEEE Comput Intell Mag. 2014;9(2):48–57.
2. Liu X, Lin S, Fang J, Xu Z. Is extreme learning machine feasible? A theoretical assessment (part I). IEEE Trans Neural Netw Learn Syst. 2015;26(1):7–20.
3. Plakias S, Stamatatos E. Tensor space models for authorship identification. In: Hellenic conference on artificial intelligence, Springer, p. 239–249. 2008.
4. Gastaldo P, Pinna L, Seminara L, Valle M, Zunino R. A tensor-based pattern-recognition framework for the interpretation of touch modality in artificial skin systems. IEEE Sensors J. 2014;14(7):2216–25.
5. Wang X-W, Nie D, Lu B-L. Emotional state classification from EEG data using machine learning approach. Neurocomputing. 2014;129:94–106.
6. Zhao Q, Zhou G, Adali T, Zhang L, Cichocki A. Kernelization of tensor-based models for multiway data analysis: Processing of multidimensional structured data. IEEE Signal Process Mag. 2013;30(4):137–48.
7. Zhao Q, Zhang L, Cichocki A. Multilinear and nonlinear generalizations of partial least squares: an overview of recent advances. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2014;4(2):104–15.
8. Liu Y, Liu Y, Chan KC. Tensor distance based multilinear locality-preserved maximum information embedding. IEEE Trans Neural Netw. 2010;21(11):1848–54.
9. Iwasaki T, Furukawa T. Tensor SOM and tensor GTM: nonlinear tensor analysis by topographic mappings. Neural Netw. 2016;77:107–25.
10. Lai Z, Wong WK, Xu Y, Zhao C, Sun M. Sparse alignment for robust tensor learning. IEEE Trans Neural Netw Learn Syst. 2014;25(10):1779–92.
11. Lu H, Plataniotis KN, Venetsanopoulos AN. A survey of multilinear subspace learning for tensor data. Pattern Recogn. 2011;44(7):1540–51.
12. Sanguansat P. Higher-order random projection for tensor object recognition. In: 2010 International Symposium on Communications and Information Technologies (ISCIT). IEEE, p. 615–619. 2010.
13. Signoretto M, Dinh QT, De Lathauwer L, Suykens JA. Learning with tensors: a framework based on convex optimization and spectral regularization. Mach Learn. 2014;94(3):303–51.
14. Tao D, Li X, Hu W, Maybank S, Wu X. Supervised tensor learning. In: 5th IEEE international conference on data mining, IEEE, p. 8. 2005.
15. Guo X, Huang X, Zhang L, Zhang L, Plaza A, Benediktsson JA. Support tensor machines for classification of hyperspectral remote sensing imagery. IEEE Trans Geosci Remote Sens. 2016;54(6):3248–64.
16. Wimalawarne K, Tomioka R, Sugiyama M. Theoretical and experimental analyses of tensor-based regression and classification. Neural Comput. 2016;28(4):686–715.
17. Hao Z, He L, Chen B, Yang X. A linear support higher-order tensor machine for classification. IEEE Trans Image Process. 2013;22(7):2911–2920.
18. Signoretto M, De Lathauwer L, Suykens JA. A kernel-based framework to tensorial data analysis. Neural Netw. 2011;24(8):861–74.
19. Balcan M-F, Blum A, Srebro N. A theory of learning with similarity functions. Mach Learn. 2008;72(1-2):89–112.
20. De Lathauwer L, De Moor B, Vandewalle J. A multilinear singular value decomposition. SIAM J Matrix Anal Appl. 2000;21(4):1253–78.

21. Ragusa E, Gianoglio C, Gastaldo P, Zunino R. A digital implementation of extreme learning machines for resource-constrained devices, IEEE Transactions on Circuits and Systems II: Express Briefs.
22. Hofmann T, Schölkopf B, Smola AJ. Kernel methods in machine learning. Ann Stat. 2008;36:1171–20.
23. Cichocki A, Mandic D, De Lathauwer L, Zhou G, Zhao Q, Caiafa C, Phan HA. Tensor decompositions for signal processing applications: from two-way to multiway component analysis. IEEE Signal Process Mag. 2015;32(2):145–63.
24. Golub GH, Van Loan CF. Matrix computations, Vol 3. Baltimore: JHU Press; 2012.
25. Peeters T, Rodrigues P, Vilanova A, ter Haar Romeny B. Analysis of distance/similarity measures for diffusion tensor imaging. In: Visualization and Processing of Tensor Fields, Springer, p. 113–136. 2009.
26. Landauer TK. Latent semantic analysis. New York: Wiley Online Library; 2006.
27. Harshman RA. Foundations of the parafac procedure: models and conditions for an "explanatory" multimodal factor analysis.
28. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, p. 1097–1105. 2012.
29. Leibe B, Schiele B. Analyzing appearance and contour based methods for object categorization. In: 2003 IEEE computer society conference on computer vision and pattern recognition, 2003. Proceedings., vol 2, IEEE, p. II–409. 2003.
30. Belhumeur PN, Hespanha JP, Kriegman DJ. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. IEEE Trans Pattern Anal Mach Intell. 1997;19(7):711–20.
31. Nilsback M-E, Zisserman A. A visual vocabulary for flower classification. In: 2006 IEEE computer society conference on computer vision and pattern recognition, vol 2, IEEE, p. 1447–1454. 2006.
32. Schuldt C, Laptev I, Caputo B. Recognizing human actions: a local SVM approach. In: Proceedings of the 17th international conference on pattern recognition, 2004. ICPR 2004, vol 3, IEEE, p. 32–36. 2004.
33. Kim T-K, Cipolla R. Canonical correlation analysis of video volume tensors for action categorization and detection. IEEE Trans Pattern Anal Mach Intell. 2009;31(8):1415–28.
34. Rodriguez-Lujan I, Fonollosa J, Vergara A, Homer M, Huerta R. On the calibration of sensor arrays for pattern recognition using the minimal number of experiments. Chemom Intell Lab Syst. 2014;130:123–34.
35. Cai D, He X, Hu Y, Han J, Huang T. Learning a spatially smooth subspace for face recognition. In: Computer vision and pattern recognition, 2007. CVPR'07. IEEE Conference on, IEEE, 2007, p. 1–7. 2007.
36. Wang Q-F, Cambria E, Liu C-L, Hussain A. Common sense knowledge for handwritten chinese text recognition. Cogn Comput. 2013;5(2):234–42.
37. Cambria E, Hussain A. Sentic album: content-, concept-, and context-based online personal photo management system. Cogn Comput. 2012;4(4):477–96.
38. Tran H-N, Cambria E, Hussain A. Towards gpu-based common-sense reasoning: using fast subgraph matching. Cogn Comput. 2016;8(6):1074–86.