Mach Learn (2014) 94:303–351
DOI 10.1007/s10994-013-5366-3

Learning with tensors: a framework based on convex optimization and spectral regularization

**Marco Signoretto · Quoc Tran Dinh ·
Lieven De Lathauwer · Johan A.K. Suykens**

Received: 5 April 2012 / Accepted: 20 April 2013 / Published online: 17 May 2013

Abstract We present a framework based on convex optimization and spectral regularization to perform learning when feature observations are multidimensional arrays (tensors). We give a mathematical characterization of spectral penalties for tensors and analyze a unifying class of convex optimization problems for which we present a provably convergent and scalable template algorithm. We then specialize this class of problems to perform learning both in a transductive as well as in an inductive setting. In the transductive case one has an input data tensor with missing features and, possibly, a partially observed matrix of labels. The goal is to both infer the missing input features as well as predict the missing labels. For induction, the goal is to determine a model for each learning task to be used for out of sample prediction. Each training pair consists of a multidimensional array and a set of labels each of which corresponding to related but distinct tasks. In either case the proposed technique exploits precise low multilinear rank assumptions over unknown multidimensional arrays; regularization is based on composite spectral penalties and connects to the concept of Multilinear Singular Value Decomposition. As a by-product of using a tensor-based formalism, our approach allows one to tackle the multi-task case in a natural way. Empirical studies demonstrate the merits of the proposed methods.

Editor: Massimiliano Pontil.

M. Signoretto (✉) · J.A.K. Suykens
ESAT-SCD/SISTA, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium
e-mail: marco.signoretto@esat.kuleuven.be

J.A.K. Suykens
e-mail: johan.suykens@esat.kuleuven.be

Q. Tran Dinh
Laboratory for Information and Inference Systems (LIONS), Ecole Polytechnique Federale de Lausanne (EPFL), EPFL STI IEL LIONS ELD 243 Station 11, 1015 Lausanne, Switzerland
e-mail: quoc.trandinh@epfl.ch

L. De Lathauwer
Group Science, Engineering and Technology, Katholieke Universiteit Leuven, Campus Kortrijk E. Sabbelaan 53, 8500 Kortrijk, Belgium
e-mail: lieven.delathauwer@kuleuven-kortrijk.be

Keywords Spectral regularization · Matrix and tensor completion · Tucker decomposition · Multilinear rank · Transductive and inductive learning · Multi-task learning

1 Introduction

Tensors are the higher order generalization of vectors and matrices. They find applications whenever the data of interest have intrinsically many dimensions. This is the case for an increasing number of areas such as econometrics, chemometrics, psychometrics, (biomedical) signal processing and image processing. Regardless of the specific domain, a common task in the data analysis workflow amounts at finding some low dimensional representation of the process under study. Existing tensor-based techniques (Kolda and Bader 2009; Smilde et al. 2004; Coppi and Bolasco 1989; Kroonenberg 2008) mostly consist of decompositions that give a concise representation of the underlying structure of data; this is useful for exploratory data analysis since it often reveals representative low-dimensional subspaces (for Tucker-type decompositions) or sum of rank-1 factors (for Canonic Polyadic Decomposition (CPD) and related techniques). In this work we take a broader perspective and consider a wider set of learning tasks. Our main goal is to extend spectral regularization (Abernethy et al. 2009; Tomioka and Aihara 2007; Argyriou et al. 2007b, 2010; Srebro 2004) to the case where data have intrinsically many dimensions and are therefore represented as higher order tensors.

1.1 Related literature

So far spectral regularization has been advocated mainly for matrices (Tomioka and Aihara 2007; Argyriou et al. 2010, 2008, 2007b; Abernethy et al. 2009). In the important low-rank matrix recovery problem, using a convex relaxation technique proved to be a valuable methodology (Cai et al. 2010; Candès and Recht 2009; Candès and Plan 2010). Recently this approach has been extended and tensor completion has been formulated (Liu et al. 2009; Signoretto et al. 2011b). The authors of Gandy et al. (2011) considered tensor completion and low multilinear rank tensor pursuit. Whereas the former assumes knowledge of some entries, the latter assumes the knowledge of measurements obtained sensing the tensor unknown via a known linear transformation (with the sampling operator being a special case). They provide algorithms for solving constrained as well as penalized versions of this problem. They also discussed formulations suitable for dealing with noisy measurements, in which a quadratic loss is employed to penalize deviation from the observed data.

1.2 Contributions

We present a framework based on convex optimization and spectral regularization to perform learning when data observations are represented as tensors. This includes in particular the cases where observations are vectors or matrices. In addition, it allows one to deal appropriately with data that have a natural representation as higher order arrays. We begin by presenting a unifying class of convex optimization problems for which we present a scalable template algorithm based on an operator splitting technique (Lions and Mercier 1979). We then specialize this class of problems to perform single as well as multi-task learning both in a transductive as well as in an inductive setting. To this end we develop tools extending to higher order tensors the concept of spectral regularization for matrices (Argyriou et al. 2007a). We consider smooth penalties (including the quadratic loss as a special case) and exploit a low multilinear rank assumption over one or more tensor unknowns through spectral regularizers. We show how this connects to the concept of Tucker decomposition (Tucker 1964, 1966) (a particular instance of which is also known as Multilinear Singular Value decomposition (De Lathauwer et al. 2000)). Additionally, as a by-product of using a tensor-based formalism, our framework allows one to tackle the multi-task case (Argyriou et al. 2008) in a natural way. In this way one exploits interdependence both at the level of the data representations as well as across tasks.

Our main contribution is twofold. A first contribution is to apply the framework to supervised transductive and inductive learning problems where the input data can be expressed as tensors. Important special cases of the framework include extensions of multitask learning with higher order observation data. A second main contribution lies within the Inexact Splitting Method that we propose as the template algorithm; we study an adaptive stopping criterion for the solution of a key sub-problem and give guarantees about the convergence of the overall algorithm.

1.3 Outline

In the next section we introduce preliminaries and present our notation. In Sect. 3 we discuss the general problem setting that we are concerned with. We present in Sect. 4 a template algorithm to solve this general class of problems and show its convergence. In Sect. 5 we extend to the tensor setting the existing definition of spectral penalty and develop the analytical tools we need. Section 6 deals with tensor-based transductive learning. Inductive learning is discussed in Sect. 7. We demonstrate the proposed methodologies in Sect. 8 and end the paper with Sect. 9 by drawing our concluding remarks.

2 Notation and preliminaries

We denote both scalars and vectors as lower case letters $(a, b, c, \ldots)$ and matrices as boldface capitals $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \ldots)$. We write 1_N to denote $[1, 1, \ldots, 1]^\top \in \mathbb{R}^N$ and $\boldsymbol{I}_N$ to indicate the $N \times N$ identity matrix. We also use subscript lower-case letters i, j in the meaning of indices and we will use I, J to denote the index upper bounds. Additionally we write $\mathbb{N}_I$ to denote the set $\{1, \ldots, I\}$. We recall that N-th order tensors, which we denote by calligraphic letters $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots)$, are higher order generalizations of vectors (first order tensors) and matrices (second order tensors). More generally, the order N of a tensor is the number of dimensions, also known as ways or modes. We write $a_{i_1,\ldots,i_N}$ to denote the entry $(\mathcal{A})_{i_1,\ldots,i_N}$. Likewise we write a_i to mean $(a)_i$ and a_{ij} to mean $(\boldsymbol{A})_{ij}$.

Next we present basic facts about tensors and introduce the mathematical machinery that we need to proceed further. The level of abstraction that we consider allows one to deal in a unified fashion with different problems and provides a useful toolset for very practical purposes. For instance, a proper characterization of operators and corresponding adjoints allows one to use the chain rule for subdifferentials (see, e.g., Ekeland and Temam 1976) that we extensively use in Sect. 5. Note that this is very useful also from an implementation view point. In fact, it is being used for the automatic derivation of differentials and subdifferentials of composite functions in modern optimization toolboxes (such as Becker et al. 2010).

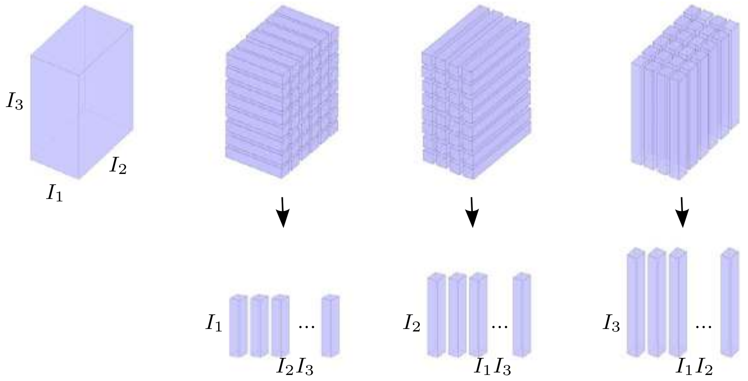

Fig. 1 An illustration of the mode unfoldings for a third order tensor

2.1 Basic facts about tensors

An N-th order tensor $\mathcal{A}$ is rank-1 if it consists of the outer product of N nonzero vectors $u^{(1)} \in \mathbb{R}^{I_1}$, $u^{(2)} \in \mathbb{R}^{I_2}, \ldots,$ $u^{(N)} \in \mathbb{R}^{I_N}$, that is, if $a_{i_1 i_2 \ldots i_N} = u^{(1)}_{i_1} u^{(2)}_{i_2} \cdots u^{(N)}_{i_N}$ for all values of the indices. In this case we write $\mathcal{A} = u^{(1)} \otimes u^{(2)} \otimes \cdots \otimes u^{(N)}$. The linear span of such elements forms a vector space, which once endowed with the inner product

$$\langle \mathcal{A}, \mathcal{B} \rangle := \sum_{i_1} \sum_{i_2} \cdots \sum_{i_N} a_{i_1 i_2 \cdots i_N} b_{i_1 i_2 \cdots i_N}, \tag{1}$$

is denoted by[1] $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. The corresponding Hilbert-Frobenius norm is $\|\mathcal{A}\| := \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$. We use $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$ for any $N \geq 1$, regardless of the specific tuple $(I_1, I_2, \ldots, I_N)$. An *n-mode vector* of $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is an element of $\mathbb{R}^{I_n}$ obtained from $\mathcal{A}$ by varying the index i_n and keeping the other indices fixed. The n-rank of $\mathcal{A}$, indicated by $\text{rank}_n(\mathcal{A})$, is the dimension of the space spanned by the n-mode vectors. A tensor for which $r_n = \text{rank}_n(\mathcal{A})$ for $n \in \mathbb{N}_N$ is called a rank-$(r_1, r_2, \ldots, r_N)$ tensor; the N-tuple $(r_1, r_2, \ldots, r_N)$ is called the *multilinear rank* of $\mathcal{A}$. For the higher order case an alternative notion of rank exists. This is:

$$\text{rank}(\mathcal{A}) := \arg\min \left\{ R \in \mathbb{N} : \mathcal{A} = \sum_{r \in \mathbb{N}_R} u^{(1)}_r \otimes u^{(2)}_r \otimes \cdots \otimes u^{(N)}_r : u^{(n)}_r \in \mathbb{R}^{I_n} \; \forall r \in \mathbb{N}_R, n \in \mathbb{N}_N \right\}. \tag{2}$$

Whereas for second order tensors $\text{rank}_1(\mathcal{A}) = \text{rank}_2(\mathcal{A}) = \text{rank}(\mathcal{A})$ for the general case we can only establish that $\text{rank}_n(\mathcal{A}) \leq \text{rank}(\mathcal{A})$ for any $n \in \mathbb{N}_N$. Additionally the n-ranks differ from each other in the general N-th order case.

Let $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and set

$$J := \prod_{j \in \mathbb{N}_N \setminus \{n\}} I_j.$$

The *n-mode unfolding* (also called matricization or flattening) of $\mathcal{A}$ is the matrix $\mathcal{A}_{\langle n \rangle} \in \mathbb{R}^{I_n \times J}$ whose columns are the n-mode vectors. The ordering according to which the vectors

[1] In the multilinear algebra literature such a space is often denoted by $\mathbb{R}^{I_1} \otimes \mathbb{R}^{I_2} \otimes \cdots \otimes \mathbb{R}^{I_N}$ to emphasize its nature as linear span of rank-1 objects. Here we use $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ for compactness.

are arranged to form $\mathcal{A}_{\langle n\rangle}$ will not matter for our purposes; what matter is that one sticks to a chosen ordering rule.

Remark 1 Assume a second order tensor $\boldsymbol{A} \in \mathbb{R}^{I_1 \times I_2}$. Then if the 2-mode unfolding is defined upon the lexicographic ordering, we have

$$\boldsymbol{A}_{\langle 2\rangle} = \boldsymbol{A}^\top$$

where $\cdot^\top$ denotes matrix transposition.

In our setting the use of unfoldings is motivated by the elementary fact that[2]

$$\operatorname{rank}_n(\mathcal{A}) = \operatorname{rank}(\mathcal{A}_{\langle n\rangle}). \tag{3}$$

Note that the n-mode unfolding as introduced above defines the linear operator

$$\cdot_{\langle n\rangle} : \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \rightarrow \mathbb{R}^{I_n \times J}.$$

The *refolding* or tensorization, denoted as $\cdot^{\langle n\rangle}$, is defined as its adjoint $\cdot^{\langle n\rangle} : \mathbb{R}^{I_n \times J} \rightarrow \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ satisfying

$$\left\langle \boldsymbol{A}^{\langle n\rangle}, \mathcal{B}\right\rangle = \langle \boldsymbol{A}, \mathcal{B}_{\langle n\rangle}\rangle.$$

Finally we recall that the *n-mode product* of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ by a matrix $\boldsymbol{U} \in \mathbb{R}^{J_n \times I_n}$, denoted by $\mathcal{A} \times_n \boldsymbol{U}$, is defined by

$$\mathcal{A} \times_n \boldsymbol{U} := (\boldsymbol{U}\mathcal{A}_{\langle n\rangle})^{\langle n\rangle} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N}. \tag{4}$$

2.2 Sampling operator and its adjoint

Assume $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and consider the ordered set

$$\mathscr{S} = \left\{ s^p := \left(i_1^p, i_2^p, \ldots, i_N^p\right) \in \mathbb{N}_{I_1} \times \mathbb{N}_{I_2} \times \cdots \times \mathbb{N}_{I_N} : p \in \mathbb{N}_P \right\}$$

identifying P entries of the N-th order tensor $\mathcal{A}$. In the following we denote by $S_{\mathscr{S}} : \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \rightarrow \mathbb{R}^P$ the *sampling operator* defined by

$$(S_{\mathscr{S}}\mathcal{X})_p := x_{i_1^p i_2^p \cdots i_N^p} \quad \text{for any } p \in \mathbb{N}_P.$$

Note that $S_{\mathscr{S}}$ is linear and it can be equivalently restated as $(S_{\mathscr{S}}\mathcal{X})_p = \langle \mathcal{X}, \mathcal{E}_{s^p}\rangle$ where $\mathcal{E}_{s^p}$ is that element of the canonical basis of $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ defined as

$$(\mathcal{E}_{s^p})_{i_1 i_2 \cdots i_N} := \begin{cases} 1, & \text{if } (i_1, i_2, \ldots, i_N) = s^p \\ 0, & \text{otherwise.} \end{cases}$$

Based upon this fact one can show that the adjoint of $S_{\mathscr{S}}\mathcal{X}$, namely that unique operator $S_{\mathscr{S}}^* : \mathbb{R}^P \rightarrow \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ satisfying $\langle b, S_{\mathscr{S}}\mathcal{X}\rangle = \langle S_{\mathscr{S}}^* b, \mathcal{X}\rangle$, is:

$$S_{\mathscr{S}}^* : b \mapsto \sum_{p \in \mathbb{N}_P} b_p \mathcal{E}_{s^p}.$$

[2] Note that the right hand side of (3) is in fact invariant with respect to permutations of the columns of $\mathcal{A}_{\langle n\rangle}$.

It is immediate to check that $S_{\mathscr{S}} S^*_{\mathscr{S}} = I_P$ and hence, $S_{\mathscr{S}}$ is a *co-isometry* in the sense of Halmos (1982, Sect. 127, page 69).

Remark 2 From this fact it follows that any solution of $S_{\mathscr{S}}\mathcal{X} = b$ can be written as $S^*_{\mathscr{S}} b + \mathcal{Z}$ where $\mathcal{Z} \in \ker(S_{\mathscr{S}}) := \{\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N} : S_{\mathscr{S}}\mathcal{X} = 0\}$.

Remark 3 Sampling operators in line with $S_{\mathscr{S}}$ abound in learning theory and algorithms. For instance (Smale and Zhou 2005) considers a sampling operator on a *reproducing kernel Hilbert space* of functions $\mathscr{H}$ (Aronszajn 1950) based on a set of *evaluation functionals* of the type

$$E_x : f \mapsto f(x) \tag{5}$$

where $f \in \mathscr{H}$ is a function on a certain domain $\mathscr{X}$ and $x \in \mathscr{X}$. It is worth remarking that an N-th order array $\mathcal{A} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ can be regarded as a function from $\mathbb{N}_{I_2} \times \cdots \times \mathbb{N}_{I_N}$ to $\mathbb{R}$. Correspondingly, $S_{\mathscr{S}}$ can be restated in terms of evaluation functionals of the same type as (5), namely

$$E_{i_1^p i_2^p \cdots i_N^p} : \mathcal{A} \mapsto a_{i_1^p i_2^p \cdots i_N^p}.$$

This is no surprise as any finite dimensional space (such as $\mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$) is isomorphic to a reproducing kernel Hilbert space of functions, see e.g. Berlinet and Thomas-Agnan (2004, Chap. 1).

2.3 Abstract vector spaces

In this paper we consider optimization problems on abstract finite dimensional inner product spaces that represent a generalization of $\mathbb{R}^P$. We are especially interested in the case where such an abstract space, denoted by $\mathscr{W}$, is obtained by endowing the Cartesian product of Q module spaces of tensors of different orders:

$$\left(\mathbb{R}^{I_1\times I_2\times\cdots\times I_{N_1}}\right) \times \left(\mathbb{R}^{J_1\times J_2\times\cdots\times J_{N_2}}\right) \times \cdots \times \left(\mathbb{R}^{K_1\times K_2\times\cdots\times K_{N_Q}}\right) \tag{6}$$

with the *canonical inner product* formed upon the uniform sum of the module spaces' inner products:

$$\big\langle (\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_Q), (\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_Q)\big\rangle_{\mathscr{W}} := \langle \mathcal{W}_1, \mathcal{V}_1\rangle + \langle \mathcal{W}_2, \mathcal{V}_2\rangle + \cdots + \langle \mathcal{W}_Q, \mathcal{V}_Q\rangle. \tag{7}$$

Note that we denoted the q-th component using the notation reserved for higher order tensors. When $N_q = 2$ (second order case) we stick with the notation for matrices introduced above and finally we denote it as a vector if $N_q = 1$. We denote $(\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_Q)$ by $\mathcal{W}$. The norm associated to (7) is $\|\mathcal{W}\|_{\mathscr{W}} = \sqrt{\langle \mathcal{W}, \mathcal{W}\rangle_{\mathscr{W}}}$.

Remark 4 As an example, assume $\mathscr{W}$ is formed upon the module spaces $\mathbb{R}^{2\times 3\times 3}$, $\mathbb{R}^{4\times 4}$ and $\mathbb{R}^5$. A generic element of $\mathscr{W}$ will be denoted then by $(\mathcal{A}, \boldsymbol{B}, c)$ where we use different letters to emphasize the different role played by the corresponding elements.

Alternatively we will denote elements of $\mathscr{W}$, i.e., abstract vectors, by lower case letters $(w, v, \ldots)$ like ordinary vectors, i.e., elements of $\mathbb{R}^P$. We use this convention whenever we do not want to specify the structure of $\mathscr{W}$. We note that this is consistent with the fact that elements of $\mathscr{W}$ can always be considered as "long vectors" avoiding involved notation.

Additionally we denote by capital letters $(A, B, C, \ldots)$ general operators between abstract spaces such as $F : \mathscr{W} \to \mathscr{V}$ and use lower case letters $(f, g, \ldots)$ to denote functionals on $\mathscr{W}$ namely operators of the type $f : \mathscr{W} \to \mathbb{R}$. Next we introduce the general family of optimization problems of interest.

3 General problem setting

3.1 Main optimization problem

The learning tasks that we formulate in this paper can be tackled via special instances of the following convex optimization problem on an abstract vector space:

$$\begin{array}{ll} \underset{w \in \mathscr{W}}{\text{minimize}} & \bar{f}(w) + \bar{g}(w) \\ \text{subject to} & w \in \bar{\mathscr{C}}. \end{array} \tag{8}$$

In this problem $\bar{f}$ is a convex and differentiable functional. As we will illustrate by preliminary examples in Sect. 3.2, it plays the role of a (possibly averaged) cost; it is assumed that $\nabla \bar{f}$ is $L_{\bar{f}}$-Lipschitz, namely that:

$$\left\| \nabla \bar{f}(w) - \nabla \bar{f}(v) \right\|_{\mathscr{W}} \leq L_{\bar{f}} \| w - v \|_{\mathscr{W}} \quad \forall w, v \in \mathscr{W}; \tag{9}$$

$\bar{g}$ is a convex but possibly non-differentiable functional playing the role of a penalty. Finally $\bar{\mathscr{C}} \subseteq \mathscr{W}$ is a set which is non-empty, closed and convex; it is used to impose over w a specific structure, which will depend on the specific instance of the learning task of interest.

3.2 Some illustrative examples

Problem (8) is very general and covers a wide range of machine learning formulations where one faces single as well as *composite penalties*, i.e., functions corresponding to the sum of multiple atomic (stand-alone) penalties. To show this and illustrate the formalism introduced above, we begin by the simplest problems that can be cast as (8). Successively, we will move on to the cases of interest, namely tensor-based problems. In the simplest cases, such as in *Ridge Regression* (Hoerl and Kennard 1970), $\bar{f}$ can be set equal to the error functional of interest f. In other cases, such as those that we will deal with in the remainder of the paper, it is more convenient to duplicate optimization variables; in these cases $\bar{f}$ is related to f in a way that we will clarify later.

In the optimization literature the idea of solving optimization problems by duplicating variables has roots in the 1950s and was developed in the 1970s, mainly in connection to control problems, see, e.g., Bertsekas and Tsitsiklis (1989). This general approach underlies the *alternating methods of multipliers* and the related Douglas-Rachford technique that we discuss later in more details. As we will see, duplicating variables allows to decompose the original problem into simpler sub-problems that can be solved efficiently and can be distributed across multiple processing units.

3.2.1 Ridge regression

Unlike the original proposal we considered an additional bias term in the model, as common in machine learning. In this case the ambient space is defined upon two module spaces;

Eqs. (6) and (7) read:

$$\mathscr{W} = \mathbb{R}^D \times \mathbb{R}, \qquad \big\langle (w,b),(v,c)\big\rangle_{\mathscr{W}} := \sum_{d\in\mathbb{N}_D} w_d v_d + bc. \tag{10}$$

The error functional and the penalty term are, respectively,

$$f(w,b) = \frac{1}{2N}\sum_{n\in\mathbb{N}_N}\left(y_n - \sum_{d\in\mathbb{N}_D} w_d x_{dn} - b\right)^2 \quad \text{and} \quad g(w) = \frac{\lambda}{2}\sum_{d\in\mathbb{N}_D} w_d^2 \tag{11}$$

where $\lambda > 0$ is a user-defined parameter. The problem of interest, namely

$$\min_{w\in\mathbb{R}^D\times\mathbb{R}} f(w,b) + g(w) \tag{12}$$

can be solved via problem (8) by letting $\bar{f} := f$, $\bar{g} := g$ and, finally, $\bar{\mathscr{C}} := \mathscr{W}$. The affine model

$$\hat{m}(x) = \langle \hat{w}, x\rangle + \hat{b}\ , \tag{13}$$

corresponding to the unique solution $(\hat{w}, \hat{b})$, is estimated based upon input data collected in the *design matrix* $\boldsymbol{X} = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^{D\times N}$ and a vector of measured responses $y \in \mathbb{R}^N$.

3.2.2 Group lasso

As a second example, consider the more involved situation where the l_2 penalty used in Ridge Regression is replaced by the group lasso penalty with (possibly overlapping) groups, see Zhao et al. (2009), Jacob et al. (2009). Let 2^{N_D} denote the *power set*[3] of N_D and consider some collection of M ordered sets $\{\mathscr{S}_1, \mathscr{S}_2, \ldots, \mathscr{S}_M\} \subseteq 2^{N_D}$. For any $w \in \mathbb{R}^D$ let $w|_{\mathscr{S}}$ be defined entry-wise by

$$(w|_{\mathscr{S}})_s = \begin{cases} w_s, & \text{if } s \in \mathscr{S} \\ 0, & \text{otherwise.}\end{cases}$$

The group lasso problem with overlapping groups and an unpenalized bias term can be expressed as

$$\min_{w\in\mathbb{R}^D\times\mathbb{R}} f(w,b) + g(w) \tag{14}$$

in which we have for $\lambda > 0$:

$$g(w) := \lambda \sum_{m\in\mathbb{N}_M} g_m(w), \quad \text{where } g_m(w) := \|w|_{\mathscr{S}_m}\| \text{ for any } m \in \mathbb{N}_M. \tag{15}$$

The latter is a first example of *composite penalty*. In this case, grouped selection occurs for non-overlapping groups; hierarchical variable selection is reached by defining groups with particular overlapping patterns (Zhao et al. 2009). Consider now the abstract vector space

$$\mathscr{W} = \underbrace{\mathbb{R}^D \times \mathbb{R}^D \times \cdots \times \mathbb{R}^D}_{M \text{ times}} \times \mathbb{R} \tag{16}$$

[3] The power set of a set $\mathscr{A}$, denoted as $2^{\mathscr{A}}$, is the set of all subsets of $\mathscr{A}$, including the empty set, denoted as $\emptyset$, and $\mathscr{A}$ itself.

endowed with the canonical inner product

$$\left\langle (w_{[1]}, w_{[2]}, \ldots, w_{[M]}, b), (v_{[1]}, v_{[2]}, \ldots, v_{[M]}, c) \right\rangle_{\mathscr{W}} := \sum_{m \in \mathbb{N}_M} \sum_{d \in \mathbb{N}_D} (w_{[m]})_d (v_{[m]})_d + bc. \quad (17)$$

Note that the original variable w is duplicated into M copies, namely, $w_{[1]}, w_{[2]}, \ldots, w_{[M]}$. Once defined the set

$$\bar{\mathscr{C}} := \left\{ (w_{[1]}, w_{[2]}, \ldots, w_{[M]}, b) \in \mathscr{W} : w_{[1]} = w_{[2]} = \cdots = w_{[M]} \right\} \quad (18)$$

we can solve (14) by means of the problem

$$\begin{array}{ll} \underset{(w_{[1]}, w_{[2]}, \ldots, w_{[M]}, b) \in \mathscr{W}}{\text{minimize}} & \bar{f}(w_{[1]}, w_{[2]}, \ldots, w_{[M]}, b) + \bar{g}(w_{[1]}, w_{[2]}, \ldots, w_{[M]}) \\ \text{subject to} & (w_{[1]}, w_{[2]}, \ldots, w_{[M]}, b) \in \bar{\mathscr{C}} \end{array} \quad (19)$$

where

$$\bar{f}(w_{[1]}, w_{[2]}, \ldots, w_{[M]}, b) = \frac{1}{M} \sum_{m \in \mathbb{N}_M} f(w_{[m]}, b)$$

and

$$\bar{g}(w_{[1]}, w_{[2]}, \ldots, w_{[M]}) = \sum_{m \in \mathbb{N}_m} g_m(w_{[m]}).$$

Indeed it is clear that if $(\hat{w}_{[1]}, \hat{w}_{[2]}, \ldots, \hat{w}_{[M]}, \hat{b})$ is a solution of (19) then, for any $m \in \mathbb{N}_M$, $(\hat{w}_{[m]}, \hat{b})$ is a solution of (14).

3.3 Learning with tensors

In the next sections we will deal with both inductive and transductive tensor-based learning problems. Regularization will be based upon *composite spectral penalties* that we introduce in Sect. 5. Multiple module spaces will be used to account for tensor unknowns of different orders. We will tackle multiple tasks simultaneously and assume input feature are collected within higher order tensors. A strategy similar to the one considered above for the group lasso will be used to conveniently recast our learning problems in term of (8).

3.3.1 Transductive Learning

In the transductive case one has an input data tensor with missing features and, possibly, a partially observed matrix of labels. The goal is to both infer the missing entries in the data tensors as well as predict the missing labels. Notably, the special case when there is no labeling information, corresponds to tensor completion that was considered for the first time in Liu et al. (2009) and can be regarded as a single learning task. For the case where input patterns are represented as vectors our approach boils down to the formulation in Goldberg et al. (2010). In this sense the transductive formulation that we propose can be regarded as a generalization to the case when input data admit a higher order representation. In this case the essential idea consists of regularizing the collection of input features and labels directly without learning a model.

Table 1 The learning tasks that we deal with via the optimization problem in (8)

transductive learning with tensors		inductive learning with tensors	
	soft-completion		
data:	partially specified input data tensor and matrix of target labels	**data:**	pairs of fully specified input features and vectors of target labels
output:	latent features and missing labels	**output:**	models for out-of-sample evaluations of multiple tasks
	hard-completion		
data:	pairs of fully specified input features and vectors of target labels		
output:	missing input data		

3.3.2 Inductive learning

For the second family of problems we consider, within the setting of inductive learning, the goal is to determine a model for each learning task to be used for out of sample prediction. For the inductive case the model corresponding to a single task will be

$$\hat{m}(\mathcal{X}) = \langle \hat{\mathcal{W}}, \mathcal{X} \rangle + \hat{b}, \tag{20}$$

where $\mathcal{X} \in \mathbb{R}^{D_1 \times D_2 \times \cdots \times D_M}$ represents here a generic data-tensor, and $(\hat{W}, \hat{b})$ are the estimated parameters, see (13) for comparison.

Each training pair consists of an input tensor data observation and a vector of labels that corresponds to related but distinct tasks. This setting extends the standard penalized empirical risk minimization problem to allow for both multiple tasks and higher order observational data.

3.3.3 Common algorithmic framework

The full taxonomy of learning formulations we deal with is illustrated in Table 1. The fact that these distinct classes of problems can be seen as special instances of (8) allows us to develop a unified algorithmical strategy to find their solutions. In particular, a central tool is given by the fact that $\mathscr{W}$ is a metric space (with the metric induced by an inner product as in (10) and (17)). Next we describe a provably convergent algorithm that is suitable for the situation where $\mathscr{W}$ is high dimensional. In the next sections we will show how this general approach can be adapted to our different purposes.

4 Unifying algorithmical approach

For certain closed forms of $\bar{f}$ and $\bar{g}$, (8) can be restated as a semi-definite programming (SDP) problem (Vandenberghe and Boyd 1996) and solved via SDP solvers such as SeDuMi (Sturm 1999), or SDPT3 (Tütüncü et al. 2003). However there is an increasing interest in the case where $\mathscr{W}$ is high dimensional in which case this approach is not satisfactory.

Alternative scalable techniques that can be adapted to the solution of (3) consist of proximal point algorithms designed to find a zero of the sum of maximal monotone operators. Classical references include Rockafellar (1976), Lions and Mercier (1979) and Spingarn (1983). A modern and comprehensive review with application to signal processing is found in Combettes and Pesquet (2009). These algorithms include as special cases the Alternating Direction Method of Multipliers (ADMMs), see Boyd et al. (2011) for a recent review. Here we propose an algorithm in the family of the Douglas-Rachford splitting methods. Notably, ADMMs can be seen as a special instance of the Douglas-Rachford splitting method, see Eckstein and Bertsekas (1992) and references therein. Our general approach can be regarded as a variant of the proximal decomposition method proposed in Combettes and Pesquet (2008) and Combettes (2009) by which it was inspired. As the main advantage, the approach does not solve the original problem directly; rather, it duplicates some of the optimization variables and solve simpler problems (proximal problems) in a distributed fashion. As we will show later, the simplicity of proximal problems lies on the fact that they can be solved exactly in terms of the SVD. Notably, as Sect. (3.2) shows, the algorithm we develop is not relevant for our tensor-based framework only.

4.1 Proximal point algorithms and operator splitting techniques

4.1.1 Problem restatement

In order to illustrate our scalable solution strategy we begin by equivalently restating (8) as the unconstrained problem:

$$\underset{w\in\mathscr{W}}{\text{minimize}} \quad \bar{h}(w)=\bar{f}(w)+\bar{g}(w)+\delta_{\bar{\mathscr{C}}}(w) \tag{21}$$

where $\delta_{\bar{\mathscr{C}}}$ is defined as:

$$\delta_{\bar{\mathscr{C}}} : \bar{w} \mapsto \begin{cases} 0, & \text{if } \bar{w}\in\bar{\mathscr{C}} \\ \infty, & \text{otherwise.} \end{cases}$$

Note that $\hat{w}$ is a solution to (21) if and only if (Rockafellar 1970a)

$$0\in\nabla\bar{f}(\hat{w})+\partial\bar{g}(\hat{w})+N_{\bar{\mathscr{C}}}(\hat{w}) \tag{22}$$

where $\nabla\bar{f}$ denotes the gradient of $\bar{f}$, $\partial\bar{g}$ is the subdifferential of $\bar{g}$ and $N_{\bar{\mathscr{C}}}$ is the subdifferential of $\delta_{\bar{\mathscr{C}}}$, i.e., the *normal cone* (Bauschke and Combettes 2011) of $\bar{\mathscr{C}}$:

$$N_{\bar{\mathscr{C}}}(w) := \begin{cases} \{x\in\mathscr{W} : \langle x, y-w\rangle_{\mathscr{W}} \le 0,\ \forall y\in\bar{\mathscr{C}}\}, & \text{if } w\in\bar{\mathscr{C}} \\ \emptyset, & \text{otherwise.} \end{cases}$$

Letting now

$$A := \nabla\bar{f}+N_{\bar{\mathscr{C}}}, \qquad B := \partial\bar{g} \quad \text{and} \quad T := A+B \tag{23}$$

Eq. (22) can be restated as

$$0\in T(\hat{w}) = A(\hat{w})+B(\hat{w}) \tag{24}$$

where A and B, as well as their sum $T=A+B$, are set-valued operators (for each w their image is a subset of $\mathscr{W}$) and they all qualify as *maximal monotone*. Maximal monotone operators, of which subdifferentials are a special instance, have been extensively studied in the literature, see e.g. Minty (1962), Rockafellar (1970b), Brézis (1973) and Phelps (1993). A recent account on the argument can be found in Bauschke and Combettes (2011).

4.1.2 Resolvent and proximal point algorithms

It is well known that, for any $\tau > 0$ and a given maximal monotone operator T on $\mathscr{W}$, $\hat{x} \in T^{-1}(0)$ if and only if $\hat{x}$ satisfies $\hat{x} \in R_{\tau T}\hat{x}$, i.e., if $\hat{x}$ is a fixed point of the single-valued *resolvent* of τT, defined as

$$R_{\tau T} := (I + \tau T)^{-1} \tag{25}$$

see e.g. Bauschke and Combettes (2011). Proximal point algorithms are based on this fundamental fact and consist of variations of the basic proximal iteration:

$$x^{(t+1)} = (I + \tau T)^{-1} x^{(t)}. \tag{26}$$

In the problem of interest T is a special monotone operator; indeed it corresponds to the subdifferential of the convex function $\bar{h} = \bar{f} + \bar{g} + \delta_{\bar{\mathscr{C}}}(w)$. In case of a subdifferential, (26) can be restated as $x^{(t)} \in x^{(t+1)} + \tau \partial \bar{h}(x^{(t+1)})$. This, in turn, is equivalent to:

$$0 \in \partial \left(\tau \bar{h}(x) + \frac{1}{2} \| x - x^{(t)} \|^2_{\mathscr{W}} \right) \Bigg|_{x = x^{(t+1)}}. \tag{27}$$

4.1.3 Proximity operator

Equation (27) represents the optimality condition for the optimization problem:

$$x^{(t+1)} = \arg\min_x \tau \bar{h}(x) + \frac{1}{2} \| x - x^{(t)} \|^2_{\mathscr{W}}. \tag{28}$$

In light of this, one can restate the proximal iteration (26), as:

$$x^{(t+1)} = \text{prox}_{\tau \bar{h}}\left(x^{(t)}\right), \tag{29}$$

where prox_g is the *proximity operator* (Moreau 1962) of g:

$$\text{prox}_g : x \mapsto \arg \min_{w \in \mathscr{W}} g(w) + \frac{1}{2} \| w - x \|^2. \tag{30}$$

4.1.4 Operator splitting approaches

The proximal iteration (29) is numerically viable only in those cases in which it is easy to solve the optimization problem (28). When $\bar{h}$ is a quadratic function, for instance, (28) corresponds to the solution of a system of linear equations that can be approached by reliable and well studied routines. In general, however, it is non trivial to tackle problem (28) directly. A viable alternative to the proximal iteration (29) rely on an *operator splitting* approach, see Bauschke and Combettes (2011) for a modern review. In the present context, the use of a splitting technique arises quite naturally from separating the objective function $\bar{h} = \bar{f} + \bar{g} + \delta_{\bar{\mathscr{C}}}$ into (1) $\bar{f} + \delta_{\bar{\mathscr{C}}}$ (corresponding to the operator A) and (2) the (generally) nonsmooth term $\bar{g}$ (corresponding to the operator B). As we will see, this decomposition leads to a tractable algorithm, in which the operators A and B are employed in separate subproblems that are easy to solve. In particular, a classical method to solve (24) is the Douglas-Rachford splitting technique that was initially developed in Lions and Mercier (1979) based upon an idea found in Douglas and Rachford (1956).

4.2 Douglas-Rachford splitting technique

The Douglas-Rachford splitting technique allows one to solve the inclusion problem (24) when A and B are maximal monotone operators. The main iteration G_{DR} consists of the following steps:

$$G_{DR}\big(w^{(k)}; A, B, \gamma^{(k)}, \tau\big) \begin{cases} y^{(k)} = R_{\tau A}\big(w^{(k)}\big), & \text{(31a)} \\ r^{(k)} = R_{\tau B}\big(2y^{(k)} - w^{(k)}\big), & \text{(31b)} \\ w^{(k+1)} = w^{(k)} + \gamma^{(k)}\big(r^{(k)} - y^{(k)}\big). & \text{(31c)} \end{cases}$$

In the latter τ is a positive proximity parameter and $(\gamma^{(k)})_k$ is a sequence of parameters that, once chosen appropriately, ensures convergence. With reference to (23), Eq. (31a) reads in our context

$$y^{(k)} = \arg\min_{x \in \bar{\mathscr{C}}} \bar{q}(x) := \bar{f}(x) + \frac{1}{2\tau}\big\|x - w^{(k)}\big\|^2_{\mathscr{W}} \tag{32}$$

whereas (31b) reads

$$r^{(k)} = \operatorname{prox}_{\tau\bar{g}}\big(2y^{(k)} - w^{(k)}\big). \tag{33}$$

4.3 Modelling workflow within the Douglas-Rachford algorithmic framework

The use of a splitting technique arises quite naturally in our context from separating the objective function (with constraints embedded via the indicator function) into (1) a part that can be approached by gradient projection and (2) a non-smooth term that can be conveniently tackled via a proximal problem. On the other hand, the Douglas-Rachford algorithmic framework, together with the abstract vector space machinery introduced above, naturally results into the following mathematical engineering workflow.

Optimization modelling Specification of the target problem: definition of the cost f and of the composite penalty g depending on the learning task of interest.

Problem casting Specification of the auxiliary problem: definition of the abstract vector space $\mathscr{W}$; $\bar{f}$, $\bar{g}$ and $\bar{C}$ are specified so that a solution of the auxiliary problem can be mapped into a solution of the target problem.

Sect. 3.2 already provided an illustration of these steps in connection to learning problems involving a parameter vector and a bias term. In general, a key ingredient in doing the problem casting is to ensure that $\bar{g}$ is an *additive separable function*. In this case, in fact, computing $\operatorname{prox}_{\tau\bar{g}}$ in (33) involves subproblems on each module space that can be distributed. We formally state this result in the following proposition. The simple proof can be found in the literature, see e.g. Bauschke and Combettes (2011, Proposition 23.30).

Proposition 1 *For $i \in \mathbb{N}_I$ let $\mathscr{W}_i$ be some vector space with inner product $\langle\cdot,\cdot\rangle_i$. Let $\mathscr{W}$ be the space obtained endowing the Cartesian product $\mathscr{W}_1 \times \mathscr{W}_2 \times \cdots \times \mathscr{W}_I$ with the inner product $\langle x, y\rangle = \sum_{i\in\mathbb{N}_I}\langle x_i, y_i\rangle_i$. Assume a function $\bar{g} : \mathscr{W} \to \mathbb{R}$ defined by*

$$\bar{g} : (x_1, x_2, \ldots, x_I) \mapsto \sum_{i\in\mathbb{N}_I} g_i(x_i)$$

where for any $i \in \mathbb{N}_I$, $g_i : \mathscr{W}_i \to \mathbb{R}$ is convex. Then we have:

$$\operatorname{prox}_{\bar{g}}(x) = \big(\operatorname{prox}_{g_1}(x_1), \operatorname{prox}_{g_2}(x_2), \ldots, \operatorname{prox}_{g_I}(x_I)\big).$$

4.4 Limits of two-level strategies

Next we present our algorithm based on an *inexact* variant of the Douglas-Rachford iteration. Our interest is in those situations where (33) can be computed exactly whereas the inner problem (32) requires an iterative procedure. As it turns out, in fact, in many situations one can cast the learning problem of interest in such a way that (33) can be computed easily and with high precision. Nonetheless, for general $\bar{f}$ in the inner problem (32), using the Douglas-Rachford iteration to solve (8) requires a procedure consisting of two nested iterative schemes. In general, the convergence of such a two-level strategy is ensured only upon exact solution of the inner problem. On the other hand, practical implementations require to specify a termination criterion and a corresponding accuracy. Notably Gandy et al. (2011) proposes different algorithms for an instance of the general problem in (8) similar to the formulations we will consider in Sect. 6. In particular, in Sect. 5.4 they also devise an inexact algorithm but they do not provide any convergence guarantee. Motivated by this we propose an adaptive termination criterion for the inner problem and prove the convergence of the outer scheme to a solution of (8).

4.5 Template based on inexact splitting technique

The approach that we propose here for solving (8), termed *Inexact Splitting Method* (ISM), is presented in Algorithm 1 in which we denoted by $P_{\bar{\mathscr{C}}}$ the projection onto $\bar{\mathscr{C}}$:

$$P_{\bar{\mathscr{C}}} : x \mapsto \arg\min_{w \in \bar{\mathscr{C}}} \|x - w\|^2_{\mathscr{W}}. \tag{34}$$

The idea is sketched as follows.

1. We apply an inexact version of G_{DR} to solve problem (8), where we only require to compute $y^{(k)}$ in (32) up to a given precision $\epsilon(k)$. Since, in our setting, (31b) can be computed in a closed form, we do not require any inexactness at this step.
2. Problem (32) is strongly convex for any $\tau > 0$ and convex and differentiable function $\bar{f}$. One can apply a gradient method that converges in this situation at a linear rate (Nesterov 2003, Theorem 2.2.8, p. 88).

Notice that step 2 in the Main procedure consists of solving the optimization subproblem (32) with a precision $\epsilon(k)$ that depends upon the iteration index k. In practice this is achieved via the Goldstein-Levitin-Polyak gradient projection method, see Bertsekas (1976, 1995). In the first main iterations a solution for (32) is found with low accuracy (from which the term *inexact*); as the estimate is refined along the iterations of MAIN the precision within the inner problem is increased; this ensures that the sequence $(y^{(k)})_k$ produced by Algorithm 1 converges to a solution of problem (8), as the following result shows.

Theorem 1 *Assume the solution set* $\hat{\mathscr{S}} := \arg\min\{\bar{f}(w) + \bar{g}(w) : w \in \bar{\mathscr{C}}\}$ *of problem* (8) *is non-empty*; *In Algorithm* 1 *let* $\epsilon_0 > 0,\ \sigma > 1$ *and* $\tau > 0$ *be arbitrarily fixed parameters. Then* $\{y^{(k)}\}_k$ *converges to* $\hat{w} \in \hat{\mathscr{S}}$.

Proof See Appendix B. □

Remark 5 (Unknown Lipschitz constant) Notice that in the procedure that computes the proximity operator with adaptive precision we assumed known $L_{\bar{f}}$ as defined in (9); based

Algorithm 1 ISM()

```
procedure MAIN()
  comment: $\epsilon_0 > 0, \sigma > 1, \tau > 0$ arbitrarily fixed.
  1. $w^{(0)} \in \mathcal{W}$
  2. $\kappa \leftarrow \sqrt{\tau L_{\bar f}(\tau L_{\bar f} + 1)} + \tau L_{\bar f}$
  repeat
    3. $\epsilon^{(k)} \leftarrow \epsilon_0/(\kappa(k+1)^{\sigma})$
    4. $y^{(k)} \leftarrow$ INEXACTPROXY$(w^{(k)}, \epsilon^{(k)}, \tau, L_{\bar f})$
    5. $r^{(k)} \leftarrow \mathrm{prox}_{\tau \bar g}(2y^{(k)} - w^{(k)})$
    6. $w^{(k+1)} \leftarrow w^{(k)} + r^{(k)} - y^{(k)}$
  until convergence criterion met
  return $(y^{(k)})$

procedure INEXACTPROXY$(z, \epsilon, \tau, L_{\bar f})$
  comment: $z \in \mathcal{W}$
  1. $L_{\bar q} \leftarrow L_{\bar f} + 1/\tau$
  2. $w^{(0)} \leftarrow z$
  repeat
    3. $w^{(t+1)} \leftarrow P_{\bar{\mathcal{C}}}(w^{(t)} - \frac{1}{L_{\bar q}}(\nabla \bar f(w^{(t)}) + \frac{1}{\tau}(w^{(t)} - z)))$
  until $\|w^{(t)} - w^{(t-1)}\|_{\mathcal{W}} \leq \epsilon$
  return $(w^{(t)})$
```

upon the latter, $L_{\bar q}$ is immediately computed since $L_{\bar q} = L_{\bar f} + 1/\tau$, see Lemma 2 in Appendix B. In practical application, however, $L_{\bar f}$ is often unknown or hard to compute. In this situation an upper bound for $L_{\bar q}$ can be found according to a backtracking strategy, see Beck and Teboulle (2009), Nesterov (2007) for details. The constant step-size $L_{\bar q}$ in step 3 of INEXACTPROXY is replaced by an adaptive step-size $h \in (0, \frac{1}{L_{\bar q}}]$ as appropriately chosen by the backtracking procedure.

Remark 6 (Termination of the outer loop) Since, as we proved, the sequence $\{y^{(k)}\}_k$ converges to the solution of problem (8), one can use the condition

$$\frac{\|y^{(k+1)} - y^{(k)}\|_{\mathcal{W}}}{\|y^{(k)}\|_{\mathcal{W}}} \leq \eta \tag{35}$$

to terminate the loop in the procedure MAIN, where $\eta > 0$ is a desired accuracy. However, for the specific form of the learning problems considered in this paper, we prefer to use the objective value. Typically, we terminate the outer loop if[4]

$$\frac{|\bar h(y^{(k+1)}) - \bar h(y^{(k)})|}{|\bar h(y^{(k)})|} \leq \eta. \tag{36}$$

[4] Note that in (36) and before in (35) we implicitly assumed that the denominator in the left hand-side is never exactly zero. In all the problems we will consider later, in particular, one can verify that this is always the case unless $0 \in \mathcal{W}$ is a solution, which never occurs in practical applications. For those specifications of $\bar h$ where this condition might arise one can replace $|\bar h(y^{(k)})|$ by $|\bar h(y^{(k)})| + 1$.

The reason for this choice is as follows: generally the termination condition (36) finds solution close to optimal (with respect to the optimization problem). When it does not, the algorithm is normally stuck in a plateau which means that the optimization is typically going to require a lot of time, with no significant improvement in the estimate. In this setting the termination condition achieves a shorter computational time by accepting the estimate we got so far and exiting the loop.

5 Spectral regularization and multilinear ranks

So far we have elaborated on the general formulation in (8); in this section we specify the nature of the penalty functions that we are concerned with in our tensor-based framework. We begin by focusing on the case where $\mathscr{W}$ corresponds to $\mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$; we then consider multiple module spaces in line with (6) and (7).

5.1 Spectral penalties for higher order tensors

We recall that a *symmetric gauge function* $h:\mathbb{R}^P\to\mathbb{R}$ is a norm which is both absolute and invariant under permutations[5] (von Neumann 1937), see also Horn and Johnson (1994, Definition 3.5.17). Symmetric gauge functions are for instance all the l_p norms. The following definition generalizes to higher order tensors the concept of spectral regularizer studied in Abernethy et al. (2009) and Argyriou et al. (2010).

Definition 1 (n-mode spectral penalty for higher order tensors) For $n\in\mathbb{N}_N$ a function $\Omega:\mathbb{R}^{I_1\times I_2\times\cdots\times I_N}\to\mathbb{R}$ is called an n-mode spectral penalty if it can be written as:

$$\Omega(\mathcal{W})=h\big(\sigma(\mathcal{W}_{\langle n\rangle})\big)$$

where, for $R=\min\{I_n,\prod_{j\in\mathbb{N}_N\setminus\{n\}}I_j\}$, $h:\mathbb{R}^R\to\mathbb{R}$ is some symmetric gauge function and $\sigma(\mathcal{W}_{\langle n\rangle})\in[0,\infty)^R$ is the vector of singular values of the matrix $\mathcal{W}_{\langle n\rangle}$ in non-increasing order.

We are especially interested in composite spectral regularizers corresponding to the (weighted) sum of different n-mode spectral penalties. The earliest example of such a situation is found in Liu et al. (2009). Denoting by $\|\cdot\|_*$ the nuclear norm for matrices, Liu et al. (2009) considers the penalty

$$g(\mathcal{W})=\sum_{n\in\mathbb{N}_N}\frac{1}{N}\|\mathcal{W}_{\langle n\rangle}\|_* \tag{37}$$

with the purpose of performing completion of a partially observed tensor. It is clear that since $\|\mathcal{W}_{\langle n\rangle}\|_*=\|\sigma(\mathcal{W}_{\langle n\rangle})\|_1$ and $\|\cdot\|_1$ is a symmetric gauge function, (37) qualifies as a composite spectral regularizer.

The nuclear norm has been used to devise convex relaxation for rank constrained matrix problems (Recht et al. 2007; Candès and Recht 2009; Candes et al. 2011); this parallels the use of the l_1-norm in sparse approximation and cardinality minimization (Tibshirani 1996; Chen et al. 2001; Donoho 2006). Likewise, minimizing (37) can be seen as a convex proxy for the minimization of the multilinear ranks.

[5] The reason for restricting to the class of symmetric gauge functions will become apparent in Proposition 2, in which their properties are used for the derivation of proximity operators.

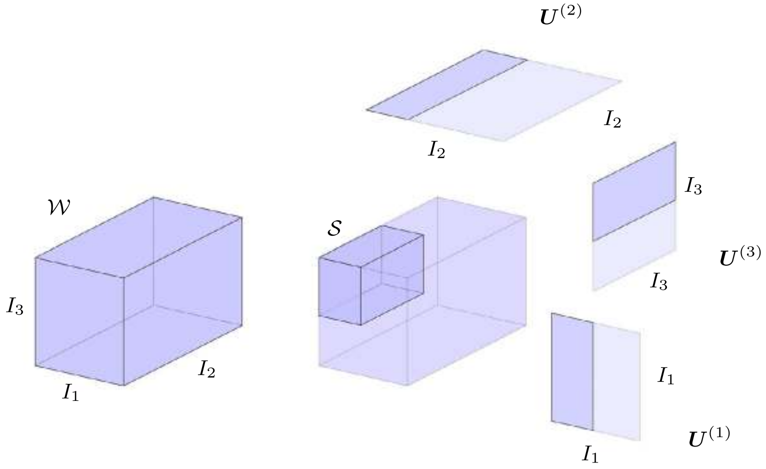

Fig. 2 An illustration of the (truncated) MLSVD

5.2 Relation with multilinear rank

A tensor $\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ can be written as Tucker (1964)

$$\mathcal{W} = \mathcal{S} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \times_3 \cdots \times_N \boldsymbol{U}^{(N)} \tag{38}$$

where $\mathcal{S} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is called the *core tensor* and for any $n \in \mathbb{N}_N$, $\boldsymbol{U}^{(n)} \in \mathbb{R}^{I_n \times I_n}$ is a matrix of n-mode singular vectors, i.e., the left singular vectors of the n-mode unfolding $\mathcal{W}_{\langle n \rangle}$ with SVD[6]

$$\mathcal{W}_{\langle n \rangle} = \boldsymbol{U}^{(n)} \operatorname{diag}\bigl(\sigma(\mathcal{W}_{\langle n \rangle})\bigr) \boldsymbol{V}^{(n)\top}. \tag{39}$$

Equation (38) is also known as the Multilinear Singular Value (MLSVD) decomposition. It has some striking similarities with the matrix SVD, see De Lathauwer et al. (2000). In particular, a good approximation of $\mathcal{W}$ can often be achieved by disregarding the n-mode singular vectors corresponding to the smallest singular values $\sigma(\mathcal{W}_{\langle n \rangle})$. See Fig. 2 for an illustration. Since penalizing the nuclear norm of $\mathcal{W}_{\langle n \rangle}$ enforces the sparsity of $\sigma(\mathcal{W}_{\langle n \rangle})$, (37) favors low multilinear rank tensors. Notably for $N = 2$ (second order case) it is easy to see that (37) is consistent with the definition of nuclear norm for matrices.

The nuclear norm is the convex envelope of the rank function on the spectral-norm unit ball (Fazel 2002); as such it represents the best convex approximation for a number of non-convex matrix problems involving the rank function. Additionally it has been established that under certain probabilistic assumptions it allows one to recover with high probability a low rank matrix from a random subset of its entries (Candès and Recht 2009; Koltchinskii et al. 2010). Similar results do not exist for (37) when $N > 2$, no matter what definition of tensorial rank one considers (see Sect. 2.1). It is therefore arguable whether or not it is appropriate to call it *nuclear norm for tensors*, as done in Liu et al. (2009). Nonetheless

[6] Assume the unfolding is performed according to the ordering rule in De Lathauwer et al. (2000). Then one has $\boldsymbol{V}^{(n)} = (\boldsymbol{U}^{(n+1)} \otimes \cdots \otimes \boldsymbol{U}^{(N-1)} \otimes \boldsymbol{U}^{(N)} \otimes \boldsymbol{U}^{(1)} \otimes \cdots \otimes \boldsymbol{U}^{(n-1)})^\top$ where $\otimes$ denotes here the matrix Kronecker product.

this penalty provides a viable way to compute low complexity estimates in the spirit of the Tucker decomposition. By contrast problems stated in terms of the tensorial rank (2) are notoriously intractable (Hillar and Lim 2010; Hastad 1990). To the best of our knowledge it remains an open problem to devise an appropriate convexification for this type of rank function.

5.3 Proximity operators

The numerical feasibility of proximal point algorithms largely depends upon the simplicity of computing the proximal operator introduced in (30). For the class of n-mode spectral penalties we can establish the following.

Proposition 2 (Proximity operator of an n-mode spectral penalty) *Assume* $\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *and let* (39) *be the SVD of its n-mode unfolding* $\mathcal{W}_{\langle n \rangle}$. *Then the evaluation at* $\mathcal{W}$ *of the proximity operator of* $\Omega(\mathcal{W}) = h(\sigma(\mathcal{W}_{\langle n \rangle}))$ *is*

$$\operatorname{prox}_{\Omega}(\mathcal{W}) = \left(\boldsymbol{U}^{(n)} \operatorname{diag}\left(\operatorname{prox}_h\left(\sigma(\mathcal{W}_{\langle n \rangle})\right)\right) \boldsymbol{V}^{(n)^\top}\right)^{\langle n \rangle}. \tag{40}$$

Proof For a matrix $\boldsymbol{A}$ with SVD $\boldsymbol{A} = \boldsymbol{U} \operatorname{diag}(\sigma(\boldsymbol{A}))\boldsymbol{V}^\top$, Argyriou et al. (2011, Proposition 3.1) established that

$$\operatorname{prox}_{h \circ \sigma}(\boldsymbol{A}) = \boldsymbol{U} \operatorname{diag}\left(\operatorname{prox}_h\left(\sigma(\boldsymbol{A})\right)\right) \boldsymbol{V}^\top.$$

It remains to show that $\operatorname{prox}_{\Omega}(\mathcal{W}) = (\operatorname{prox}_{h \circ \sigma}(\mathcal{W}_{\langle n \rangle}))^{\langle n \rangle}$. Note that $\cdot_{\langle n \rangle}$ is a linear one-to-one (invertible) operator and that $(\mathcal{W}_{\langle n \rangle})^{\langle n \rangle} = \mathcal{W}$ namely, the composition between the folding operator and its adjoint yields the identity[7] on $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. Additionally by the chain rule for the subdifferential (see e.g. Nesterov 2003, Lemma 3.18) and by definition of Ω one has $\partial \Omega(\mathcal{V}) = (\partial(h \circ \sigma)(\mathcal{V}_{\langle n \rangle}))^{\langle n \rangle}$. We now have:

$$\begin{aligned}
\mathcal{V} = \operatorname{prox}_{\Omega}(\mathcal{W}) \quad &\Leftrightarrow \quad \mathcal{V} - \mathcal{W} \in \partial \Omega(\mathcal{V}) = \left(\partial(h \circ \sigma)(\mathcal{V}_{\langle n \rangle})\right)^{\langle n \rangle} \\
&\Leftrightarrow \quad (\mathcal{V} - \mathcal{W})_{\langle n \rangle} \in \left(\left(\partial(h \circ \sigma)(\mathcal{V}_{\langle n \rangle})\right)^{\langle n \rangle}\right)_{\langle n \rangle} \\
&\Leftrightarrow \quad \mathcal{V}_{\langle n \rangle} = \operatorname{prox}_{h \circ \sigma}(\mathcal{W}_{\langle n \rangle}) \\
&\Leftrightarrow \quad \mathcal{V} = \left(\operatorname{prox}_{h \circ \sigma}(\mathcal{W}_{\langle n \rangle})\right)^{\langle n \rangle}.
\end{aligned} \tag{41}$$

□

In particular for the case where $\Omega(\mathcal{W}) = \lambda \|\sigma(\mathcal{W}_{\langle n \rangle})\|_1$ one has

$$\operatorname{prox}_{\lambda \|\sigma(\cdot_{\langle n \rangle})\|_1}(\mathcal{W}) = \left(\boldsymbol{U}^{(n)} \operatorname{diag}(d_\lambda) \boldsymbol{V}^{(n)^\top}\right)^{\langle n \rangle} \tag{42}$$

where $(d_\lambda)_i := \max(\sigma_i(\mathcal{W}_{\langle n \rangle}) - \lambda, 0)$. Note that (42) corresponds to refolding the matrix obtained applying to $\mathcal{W}_{\langle n \rangle}$ the *matrix shrinkage operator* as introduced in Cai et al. (2010).

5.4 Multiple module spaces

So far we considered the case where $\mathscr{W}$ consisted solely of the module space $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. Next we focus on the case where $\mathscr{W}$ is given by 2 modules, see Sect. 2.3. The following

[7] Equivalently, $\cdot_{\langle n \rangle}$ is unitary.

definition will turn out useful in the next section where we deal with two distinct type of unknowns that are jointly regularized.

Definition 2 ((n_1, n_2)-mode spectral penalty) Assume a vector space $\mathscr{W}$ obtained endowing $(\mathbb{R}^{I_1\times I_2\times\cdots\times I_{N_1}})\times(\mathbb{R}^{J_1\times J_2\times\cdots\times J_{N_2}})$ with the canonical inner product

$$\langle\mathcal{W},\mathcal{V}\rangle_{\mathscr{W}}=\langle\mathcal{W}_1,\mathcal{V}_1\rangle+\langle\mathcal{W}_2,\mathcal{V}_2\rangle \tag{43}$$

and norm $\|\mathcal{W}\|_{\mathscr{W}}=\sqrt{\langle\mathcal{W},\mathcal{W}\rangle_{\mathscr{W}}}$. Suppose that for $n_1\in\mathbb{N}_{N_1}$ and $n_2\in\mathbb{N}_{N_2}$ $I_{n_1}=J_{n_2}=K$ and let $S_1:=\prod_{p\in\mathbb{N}_{N_1}\setminus\{n_1\}}I_p,\ S_2:=\prod_{p\in\mathbb{N}_{N_2}\setminus\{n_2\}}J_p$. A function $\Omega:\mathscr{W}\to\mathbb{R}$ is called an (n_1,n_2)-mode spectral penalty if it can be written as:

$$\Omega(\mathcal{W})=h\big(\sigma\big([\mathcal{W}_{1\langle n_1\rangle},\mathcal{W}_{2\langle n_2\rangle}]\big)\big) \tag{44}$$

where, for $R=\min\{K,S_1S_2\}$, $h:\mathbb{R}^R\to\mathbb{R}$ is some symmetric gauge function and $\sigma([\mathcal{W}_{1\langle n_1\rangle},\mathcal{W}_{2\langle n_2\rangle}])\in[0,\infty)^R$ is the vector of singular values of the matrix $[\mathcal{W}_{\langle n_1\rangle},\mathcal{W}_{\langle n_2\rangle}]$ in non-increasing order.

Note that we required that $I_{n_1}=J_{n_2}=K$ since otherwise $\mathcal{W}_{1\langle n_1\rangle}$ and $\mathcal{W}_{2\langle n_2\rangle}$ cannot be concatenated.

Proposition 3 (Proximity operator of an (n_1,n_2)-mode spectral penalty) *Let $\mathscr{W}$, Ω, S_1 and S_2 be defined as in Definition 2 and assume the SVD*:

$$[\mathcal{W}_{1\langle n_1\rangle},\mathcal{W}_{2\langle n_2\rangle}]=\boldsymbol{U}\sigma\big([\mathcal{W}_{1\langle n_1\rangle},\mathcal{W}_{2\langle n_2\rangle}]\big)\boldsymbol{V}^\top.$$

Then we have

$$\operatorname{prox}_\Omega(\mathcal{W})=\big(\boldsymbol{Z}_1^{\langle n_1\rangle},\boldsymbol{Z}_2^{\langle n_2\rangle}\big) \tag{45}$$

where

$$\boldsymbol{Z}=\boldsymbol{U}\operatorname{diag}\big(\operatorname{prox}_h\big(\sigma\big([\mathcal{W}_{1\langle n_1\rangle},\mathcal{W}_{2\langle n_2\rangle}]\big)\big)\big)\boldsymbol{V}^\top \tag{46}$$

is partitioned into $[\boldsymbol{Z}_1,\boldsymbol{Z}_2]$ where $\boldsymbol{Z}_1$ is a $(K\times S_1)$-matrix and $\boldsymbol{Z}_2$ is a $(K\times S_2)$-matrix.

Proof Consider the unfolding operator on $\mathscr{W}$, $\cdot_{\langle n_1n_2\rangle}:(\mathcal{W}_1,\mathcal{W}_1)\mapsto[\mathcal{W}_{1\langle n_1\rangle},\mathcal{W}_{2\langle n_2\rangle}]$. Based on (43) it is not difficult to see that its adjoint corresponds to the operator $\cdot^{\langle n_1n_2\rangle}:\mathbb{R}^{K\times(S_1+S_2)}\to\mathscr{W}$ given by

$$\cdot^{\langle n_1n_2\rangle}:[\boldsymbol{W}_1,\boldsymbol{W}_2]\mapsto\big(\boldsymbol{W}_1^{\langle n_1\rangle},\boldsymbol{W}_2^{\langle n_2\rangle}\big).$$

The chain rule for the subdifferential reads now $\partial\Omega(\mathcal{V})=(\partial(h\circ\sigma)(\mathcal{V}_{\langle n_1n_2\rangle}))^{\langle n_1n_2\rangle}$. In the same fashion as in (41) we now have:

$$\begin{aligned}\mathcal{V}=\operatorname{prox}_\Omega(\mathcal{W})\quad&\Leftrightarrow\quad\mathcal{V}-\mathcal{W}\in\partial\Omega(\mathcal{V})=\big(\partial(h\circ\sigma)(\mathcal{V}_{\langle n_1n_2\rangle})\big)^{\langle n_1n_2\rangle}\\&\Leftrightarrow\quad(\mathcal{V}-\mathcal{W})_{\langle n_1n_2\rangle}\in\Big(\big(\partial(h\circ\sigma)(\mathcal{V}_{\langle n_1n_2\rangle})\big)^{\langle n_1n_2\rangle}\Big)_{\langle n_1n_2\rangle}\\&\Leftrightarrow\quad\mathcal{V}_{\langle n_1n_2\rangle}=\operatorname{prox}_{h\circ\sigma}(\mathcal{W}_{\langle n_1n_2\rangle})\\&\Leftrightarrow\quad\mathcal{V}=\big(\operatorname{prox}_{h\circ\sigma}(\mathcal{W}_{\langle n_1n_2\rangle})\big)^{\langle n_1n_2\rangle}.\end{aligned} \tag{47}$$

□

We note that Definition 2 and the result above can be easily generalized to more than two module spaces at the price of a more involved notation.

6 Transductive learning with higher order data

In this section we specialize problem (8) in order to perform transductive learning with partially observed higher order data.[8] It is assumed one has a set of N items with higher order representation $\mathcal{X}^{(n)} \in \mathbb{R}^{D_1 \times D_2 \times \cdots \times D_M}$, $n \in \mathbb{N}_N$. These items are gathered in the input dataset $\mathcal{X} \in \mathbb{R}^{D_1 \times D_2 \times \cdots \times D_M \times N}$ defined entry-wise by

$$x_{d_1 d_2 \cdots d_M n} = x^{(n)}_{d_1 d_2 \cdots d_M}.$$

Associated to the n-th item there is a target vector $y^{(n)} \in \mathcal{Y}^T$. In particular we shall focus on the case where $\mathcal{Y} = \{-1, 1\}$ so that $\boldsymbol{Y} = [y^{(1)}, y^{(2)}, \ldots, y^{(N)}]$ is a $(T \times N)$-matrix of binary labels. Entries of $\mathcal{X}$ and $\boldsymbol{Y}$ can be missing with

$$\mathcal{S}_{\mathcal{X}} = \left\{ \left(d_1^p, \ldots, d_M^p, n^p\right) \in \mathbb{N}_{D_1} \times \mathbb{N}_{D_2} \times \cdots \times \mathbb{N}_{D_M} \times \mathbb{N}_N : p \in \mathbb{N}_P \right\} \quad (48)$$

$$\mathcal{S}_{Y} = \left\{ \left(t^q, n^q\right) \in \mathbb{N}_T \times \mathbb{N}_N : q \in \mathbb{N}_Q \right\} \quad (49)$$

being the index set of the observed entries in, respectively, $\mathcal{X}$ and $\boldsymbol{Y}$. The goal is to infer the missing entries in $\mathcal{X}$ and $\boldsymbol{Y}$ simultaneously, see Fig. 3. We refer to this task as *heterogeneous* data completion to emphasize that the nature of $\mathcal{X}$ and $\boldsymbol{Y}$ is different. Note that this reduces to standard transductive learning as soon as $T = 1$, $M = 1$ and finally $\mathcal{S}_{\mathcal{X}} = \emptyset$ (no missing entries in the input dataset). Goldberg et al. (2010) considers the more general situation where $T \geq 1$ and $\mathcal{S}_{\mathcal{X}} \neq \emptyset$. Here we further generalize this to the case where $M \geq 1$, that is, items admit a higher order representation. We also point out that the special case where $T = 1$ and there is no labeling task defined (in particular, $\mathcal{S}_Y = \emptyset$) corresponds to tensor completion as considered for the first time in Liu et al. (2009). Next we clarify our modelling assumptions.

6.1 Modelling assumptions

The heterogeneous data completion task is ill-posed in the sense that there are infinitely many ways to fully specify the entries of $\mathcal{X}$ and $\boldsymbol{Y}$.[9] Making the inference process feasible requires to formulate assumptions for both the input dataset as well as for the matrix of labels.

In this section we consider the following generative model. It is assumed that the input dataset $\mathcal{X} \in \mathbb{R}^{D_1 \times D_2 \times \cdots \times D_M \times N}$ can be decomposed into

$$\mathcal{X} = \tilde{\mathcal{X}} + \mathcal{E} \quad (50)$$

where $\tilde{\mathcal{X}}$ is a rank-$(r_1, r_2, \ldots, r_M, r_{M+1})$ tensor and $\mathcal{E}$ is a remainder. In our setting the assumption considered in Goldberg et al. (2010) is solely that

$$r_{M+1} \ll \min(N, J) \quad (51)$$

where

$$J = \prod_{j \in \mathbb{N}_M} D_j.$$

[8] The code of some routines can be found at https://securehomes.esat.kuleuven.be/~msignore/.

[9] This is the case since entries of $\mathcal{X}$ are in $\mathbb{R}$.

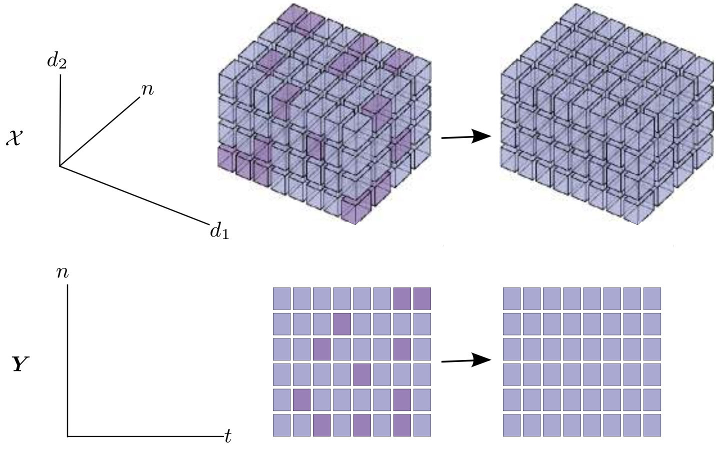

Fig. 3 An illustration of transductive learning with higher order data and multiple tasks. In this case each observation consists of a $D_1 \times D_2$ matrix and a T-dimensional target vector. Input observations are stacked along the third-mode of the input dataset $\mathcal{X}$ (*top*), whereas target observations are gathered in the matrix $\boldsymbol{Y}$ (*bottom*). Missing entries of $\mathcal{X}$ and $\boldsymbol{Y}$, indexed by $\mathscr{S}_{\mathcal{X}}$ and $\mathscr{S}_{\boldsymbol{Y}}$ respectively, are indicated with *purple tones* (Color figure online)

This amounts at regarding items as elements of $\mathbb{R}^J$ hereby neglecting their multimodal structure. By contrast we further assume that

$$r_m \ll \min(D_m, NJ/D_m) \quad \text{for some } m \in \mathbb{N}_M. \tag{52}$$

This type of assumption is generally fulfilled in a number of cases where multimodal dependence arises; this occurs for instance when dealing with spectral images (Signoretto et al. 2011b). Additionally we suppose that $y_t^{(n)}$, the label of the n-th pattern for the t-th task, is linked to $\tilde{\mathcal{X}}^{(n)}$ via a latent variable model. More specifically, we let

$$\tilde{y}_t^{(n)} = \langle \tilde{\mathcal{X}}^{(n)}, \mathcal{W}^{(t)} \rangle \tag{53}$$

where $\mathcal{W}^{(t)}$ is the parameter tensor corresponding to the t-th task; we assume that $y_t^{(n)}$ is produced by assigning at random each binary entry with alphabet $\{-1, 1\}$ following the probability model

$$p(y_{tn}|\tilde{y}_{tn}, b_t) = 1/\big(1 + \exp\big(-y_{tn}(\tilde{y}_{tn} + b_t)\big)\big). \tag{54}$$

Note that, in the latter, we considered explicitly a bias term b_t. Let $\mathcal{W}$ be that element of $\mathbb{R}^{D_1 \times D_2 \times \cdots \times D_M \times T}$ defined as $w_{d_1 d_2 \cdots d_M t} := w_{d_1 d_2 \cdots d_M}^{(t)}$. Note that $\mathcal{W}$ gathers the representers of the linear functionals associated to the T tasks. We now have that

$$\tilde{\boldsymbol{Y}} = \mathcal{W}_{\langle M+1 \rangle} \tilde{\mathcal{X}}_{\langle M+1 \rangle}^{\top} \tag{55}$$

and it follows from (51) that

$$\operatorname{rank}\big(\big[\tilde{\mathcal{X}}_{\langle M+1 \rangle}, \tilde{\boldsymbol{Y}}^{\top}\big]\big) \le r_{M+1} \ll \min(N, J + T). \tag{56}$$

Remark 7 Notice that we have deliberately refrained from specifying the nature of $\mathcal{E}$ in (50). Central to our approach is the way input features and target labels are linked together; this is specified by the functional relation (53) and by (54). One could interpret $\mathcal{E}$ as noise in which case $\tilde{\mathcal{X}}$ can be regarded as the underlying true representation of the input observation. This is in line with error-in-variables models (Golub and Van Loan 1980; Van Huffel and Vandewalle 1991). Alternatively one might regard $\mathcal{X}$ as the true representation and assume that the target variable depends only upon the latent tensor $\tilde{\mathcal{X}}$, having low multilinear rank $(r_1, r_2, \dots, r_M, r_{M+1})$.

6.2 Multi-task learning via soft-completion of heterogeneous data

We denote by $S_{\mathscr{S}_\mathcal{X}}$ and $S_{\mathscr{S}_Y}$ the sampling operators (Sect. 2.2) defined, respectively, upon (48) and upon (49) and let $z^x \in \mathbb{R}^P$ and $z^y \in \mathbb{R}^Q$ be the corresponding measurement vectors. Let $l_x, l_y : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$ be some predefined convex loss functions respectively for the input data and the target labels. The empirical error functional we consider, namely

$$f_{\lambda_0}(\tilde{\mathcal{X}}, \tilde{\boldsymbol{Y}}, b) := f^x(\tilde{\mathcal{X}}) + \lambda_0 f^y(\tilde{\boldsymbol{Y}}, b) \tag{57}$$

is composed by an error for the inputs,

$$f^x : \tilde{\mathcal{X}} \mapsto \sum_{p \in \mathbb{N}_P} l_x\big((\Omega_{\mathcal{S}_\mathcal{X}} \tilde{\mathcal{X}})_p, z^x_p\big) \tag{58}$$

and one for the latent variables and bias terms,

$$f^y : (\tilde{\boldsymbol{Y}}, b) \mapsto \sum_{q \in \mathbb{N}_Q} l_y\big(\big(\Omega_{\mathcal{S}_Y}(\tilde{\boldsymbol{Y}} + b \otimes 1_J)\big)_q, z^y_q\big). \tag{59}$$

The heterogeneous completion task is then solved by means of the optimization problem

$$(\hat{\tilde{\mathcal{X}}}, \hat{\tilde{\boldsymbol{Y}}}, \hat{b}) = \arg \min_{(\tilde{\mathcal{X}}, \tilde{\boldsymbol{Y}}, b) \in \mathscr{V}} f_{\lambda_0}(\tilde{\mathcal{X}}, \tilde{\boldsymbol{Y}}, b) + \sum_{m \in \mathbb{N}_M} \lambda_m \|\tilde{\mathcal{X}}_{\langle m \rangle}\|_* + \lambda_{M+1} \big\| \big[\tilde{\mathcal{X}}_{\langle M+1 \rangle}, \tilde{\boldsymbol{Y}}^\top\big] \big\|_* \tag{60}$$

where $\mathscr{V}$ is obtained endowing the Cartesian product:

$$\big(\mathbb{R}^{D_1 \times D_2 \times \dots \times D_M \times N}\big) \times \big(\mathbb{R}^{T \times N}\big) \times \mathbb{R}^T \tag{61}$$

with the inner product

$$\big\langle (\tilde{\mathcal{X}}_1, \tilde{\boldsymbol{Y}}_1, b_1), (\tilde{\mathcal{X}}_2, \tilde{\boldsymbol{Y}}_2, b_2) \big\rangle_{\mathscr{V}} = \langle \tilde{\mathcal{X}}_1, \tilde{\mathcal{X}}_2 \rangle + \langle \tilde{\boldsymbol{Y}}_1, \tilde{\boldsymbol{Y}}_2 \rangle + \langle b_1, b_2 \rangle; \tag{62}$$

for any $m \in \{0\} \cup \mathbb{N}_{M+1}$, $\lambda_m > 0$ is a user-defined parameter and $\sum_{m \in \mathbb{N}_{M+1}} \lambda_m = 1$. Problem (60) is convex since its objective is the sum of convex functions. It is a form of penalized empirical risk minimization with a composite penalty. The first M penalty terms

$$\Omega_m : \tilde{\mathcal{X}} \mapsto \lambda_m \|\tilde{\mathcal{X}}_{\langle m \rangle}\|_*, \ m \in \mathbb{N}_M \tag{63}$$

reflect the modelling assumption (52). The $(M+1)$-th penalty

$$\Gamma : (\tilde{\mathcal{X}}, \tilde{\boldsymbol{Y}}) \mapsto \lambda_{M+1} \big\| \big[\tilde{\mathcal{X}}_{\langle M+1 \rangle}, \tilde{\boldsymbol{Y}}^\top\big] \big\|_*$$

ensures that the recovered matrix $[\hat{\tilde{\mathcal{X}}}_{\langle M+1 \rangle}, \hat{\tilde{\boldsymbol{Y}}}^\top]$ is approximately low rank, in line with Eq. (56). The contribution of the different terms is trimmed by the associated regularization parameters which are either preselected or chosen according to some model selection criterion. In principle any meaningful pair of convex penalties can be used to define the error functional. Here we follow (Goldberg et al. 2010) and consider in (58) and (59)

$$l_x : (u, v) \mapsto \frac{1}{2}(u - v)^2 \qquad \text{(quadratic loss)}, \tag{64a}$$

$$l_y : (u, v) \mapsto \log\bigl(1 + \exp(-uv)\bigr) \qquad \text{(logistic loss)}. \tag{64b}$$

Note that (64a) is fully justified by assuming that $\mathcal{E}$ in (50) has Gaussian entries. These losses ensure that the overall error functional (57) is smooth. This fact, along with the tools developed in Sect. 5, allows us to use Algorithm 1 as a template to devise a solution strategy.

6.3 Algorithm for soft-completion

In order to rely on Algorithm 1, we need to suitably design $\mathscr{W}$, $\bar{f}$, $\bar{g}$ as well as $\bar{\mathscr{C}}$. That is, we have to cast (60) into the prototypical formulation in (8). Consider the abstract vector space $\mathscr{W}$ obtained endowing[10]

$$\bigl(\times_{m\in\mathbb{N}_{M+1}}\bigl\{\mathbb{R}^{D_1\times D_2\times\cdots\times D_M\times N}\bigr\}\bigr) \times \bigl(\mathbb{R}^{T\times N}\bigr) \times \mathbb{R}^T \tag{65}$$

with the canonical inner product

$$\bigl\langle(\tilde{\mathcal{X}}_{[1]}, \ldots, \tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}}, b), (\mathcal{W}_{[1]}, \ldots, \mathcal{W}_{[M+1]}, \boldsymbol{U}, c)\bigr\rangle_{\mathscr{W}}$$
$$:= \sum_{m\in\mathbb{N}_{M+1}} \langle\tilde{\mathcal{X}}_{[m]}, \mathcal{W}_{[m]}\rangle + \langle\tilde{\boldsymbol{Y}}, \boldsymbol{U}\rangle + \langle b, c\rangle. \tag{66}$$

Once defined the set

$$\bar{\mathscr{C}} := \bigl\{(\tilde{\mathcal{X}}_{[1]}, \tilde{\mathcal{X}}_{[2]}, \ldots, \tilde{\mathcal{X}}_{[M]}, \tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}}, b) \in \mathscr{W} : \tilde{\mathcal{X}}_{[1]} = \tilde{\mathcal{X}}_{[2]} = \cdots = \tilde{\mathcal{X}}_{[M+1]}\bigr\} \tag{67}$$

we can solve (60) by means of the problem

$$\begin{array}{ll} \underset{(\tilde{\mathcal{X}}_{[1]},\tilde{\mathcal{X}}_{[2]},\ldots,\tilde{\mathcal{X}}_{[M]},\tilde{\mathcal{X}}_{[M+1]},\tilde{\boldsymbol{Y}},b)\in\mathscr{W}}{\text{minimize}} & \bar{f}(\tilde{\mathcal{X}}_{[1]}, \ldots, \tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}}, b) + \bar{g}(\tilde{\mathcal{X}}_{[1]}, \ldots, \tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}}) \\ \text{subject to} & (\tilde{\mathcal{X}}_{[1]}, \ldots, \tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}}, b) \in \bar{\mathscr{C}} \end{array} \tag{68}$$

where

$$\bar{f}(\tilde{\mathcal{X}}_{[1]}, \ldots, \tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}}, b) := \frac{1}{M+1}\sum_{m\in\mathbb{N}_{M+1}} f^x(\tilde{\mathcal{X}}_{[m]}) + \lambda_0 f^y(\tilde{\boldsymbol{Y}}, b), \tag{69}$$

$$\bar{g}(\tilde{\mathcal{X}}_{[1]}, \ldots, \tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}}) := \sum_{m\in\mathbb{N}_M} \varOmega_m(\tilde{\mathcal{X}}_{[m]}) + \varGamma(\tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}}). \tag{70}$$

Application of Propositions 2, 3 and 1 shows now that $\operatorname{prox}_{\tau\bar{g}}$ in Step 3 of MAIN (Algorithm 1) reads:

$$\operatorname{prox}_{\tau\bar{g}}(\tilde{\mathcal{X}}_{[1]}, \ldots, \tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}})$$

[10] We adopt the short-hand $\times_{m\in\mathbb{N}_M}\mathscr{A}$ to indicate the iterated Cartesian product: $\underbrace{\mathscr{A}\times\mathscr{A}\times\cdots\times\mathscr{A}}_{M\text{ times}}$.

$$= \big(\mathrm{prox}_{\tau\lambda_1\|\sigma(\cdot_{\langle 1\rangle})\|_1}(\tilde{\mathcal{X}}_{[1]}), \ldots, \mathrm{prox}_{\tau\lambda_M\|\sigma(\cdot_{\langle M\rangle})\|_1}(\tilde{\mathcal{X}}_{[M]}), \boldsymbol{Z}_1, \boldsymbol{Z}_2\big) \tag{71}$$

where $[\boldsymbol{Z}_1(\tilde{\mathcal{X}}, \tilde{\boldsymbol{Y}}), \boldsymbol{Z}_2(\tilde{\mathcal{X}}, \tilde{\boldsymbol{Y}})]$ is a partitioning of

$$\boldsymbol{Z}(\tilde{\mathcal{X}}, \tilde{\boldsymbol{Y}}) = \boldsymbol{U}\,\mathrm{diag}\big(\mathrm{prox}_{\tau\lambda_{M+1}\|\sigma(\cdot)\|_1}\big(\big[\tilde{\mathcal{X}}_{\langle M+1\rangle}, \tilde{\boldsymbol{Y}}^\top\big]\big)\big)\boldsymbol{V}^\top \tag{72}$$

consistent with the dimensions of $\tilde{\mathcal{X}}_{\langle M+1\rangle}$ and $\tilde{\boldsymbol{Y}}^\top$, the operator $\mathrm{prox}_{\lambda\|\sigma(\cdot)\|_1}$ is defined as in (42) and finally $\boldsymbol{U}$ and $\boldsymbol{V}$ are respectively left and right singular vectors of the matrix $[\tilde{\mathcal{X}}_{\langle M+1\rangle}, \tilde{\boldsymbol{Y}}^\top]$. Note that (34) reads here:

$$\begin{aligned} &P_{\bar{\mathscr{C}}}(\tilde{\mathcal{X}}_{[1]}, \ldots, \tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}}, b) \\ &\quad= \left(\frac{1}{M+1}\sum_{m\in\mathbb{N}_{M+1}} \tilde{\mathcal{X}}_{[m]}, \ldots, \frac{1}{M+1}\sum_{m\in\mathbb{N}_{M+1}} \tilde{\mathcal{X}}_{[m]}, \tilde{\boldsymbol{Y}}, b\right). \end{aligned} \tag{73}$$

For completeness we reported in Appendix C the closed form of $\nabla\bar{f}$. We summarize in Algorithm 2 the steps required to compute a solution. We stress that these steps are obtained by adapting the steps of our template procedure given in Algorithm 1.

6.4 Hard-completion without target labels

The problem of missing or unknown values in multi-way arrays is frequently encountered in practice. Missing values due to data acquisition, transmission, or storage problems are for instance encountered in face image modelling by multilinear subspace analysis (Geng et al. 2011). Generally speaking, missing data due to faulty sensors are widespread in biosignal processing; Acar et al. (2011), in particular, considers an EEG (electroencephalogram) application where data are missing due to disconnections of electrodes. Another problem in Acar et al. (2011) arises from modelling time-evolving computer network traffic where cost-sensitivity imposes that only a subset of edges in the network are sampled.

Problem (60) assumes that data consist of both input and target measurements. In turn, the situation where we do not consider target measurements can be dealt with by the following special instance of (60):

$$\hat{\tilde{\mathcal{X}}} = \arg \min_{\tilde{\mathcal{X}}\in\mathbb{R}^{D_1\times D_2\times\cdots\times D_M\times N}} f^x(\tilde{\mathcal{X}}) + \sum_{m\in\mathbb{N}_{M+1}} \Omega_m(\tilde{\mathcal{X}}). \tag{74}$$

In the latter, f^x penalizes the misfit of $\tilde{\mathcal{X}}$ to the partially observed input data tensor; the composite penalty term favors solution with small multilinear rank. The solution strategy illustrated in the previous section can be easily adjusted to deal with this situation. For certain practical problems, however, it is more desirable to complete the missing entries while requiring the exact adherence to the data. Let us use $\mathscr{V}$ as a shorthand notation for $\mathbb{R}^{D_1\times D_2\times\cdots\times D_M\times N}$. Strict adherence to observables can be accomplished by means of the following constrained formulation of *tensor completion* (Gandy et al. 2011; Tomioka et al. 2011; Liu et al. 2009; Signoretto et al. 2011b):

$$\begin{array}{ll} \underset{\tilde{\mathcal{X}}\in\mathbb{R}^{D_1\times D_2\times\cdots\times D_M\times N}}{\text{minimize}} & \displaystyle\sum_{m\in\mathbb{N}_{M+1}} \Omega_m(\tilde{\mathcal{X}}) \\ \text{subject to} & S_{\mathscr{S}_{\mathcal{X}}}\tilde{\mathcal{X}} = z^x \end{array} \tag{75}$$

where $\mathscr{S}_{\mathcal{X}}$ is the sampling set (48).

Algorithm 2 SoftCompletion()

```
input   index sets $\mathscr{S}_{\mathcal{X}}$ and $\mathscr{S}_{\boldsymbol{Y}}$, vectors of measurements $z^x$ and $z^y$
output  estimate $(\tilde{\mathcal{X}}, \tilde{\boldsymbol{Y}}, b)$

procedure MAIN()
  comment $\epsilon_0 > 0,\ \sigma > 1,\ \tau > 0$ arbitrarily fixed.
  comment procedure INEXACTPROXY() is found in Algorithm 1
  1. $(\mathcal{V}^{(k)}_{[1]}, \cdots, \mathcal{V}^{(k)}_{[M+1]}, \boldsymbol{M}^{(k)}, c^{(k)}) \in \mathscr{W}$
  2. $\kappa \leftarrow \sqrt{\tau L_{\bar{f}}(\tau L_{\bar{f}} + 1)} + \tau L_{\bar{f}}$

  repeat
      3. $\epsilon^{(k)} \leftarrow \epsilon_0/(\kappa(k+1)^{\sigma})$
      4. $(\tilde{\mathcal{X}}^{(k)}_{[1]}, \cdots, \tilde{\mathcal{X}}^{(k)}_{[M+1]}, \tilde{\boldsymbol{Y}}^{(k)}, b^{(k)}) \leftarrow$
         INEXACTPROXY($(\mathcal{V}^{(k)}_{[1]}, \cdots, \mathcal{V}^{(k)}_{[M+1]}, \boldsymbol{M}^{(k)}, c^{(k)}), \epsilon^{(k)}, \tau, L_{\bar{f}}$)
         comment  $P_{\bar{\mathscr{C}}}$ in INEXACTPROXY computed according to Eq. (73)
     5a. $\mathcal{R}^{(k)}_{[m]} \leftarrow \mathrm{prox}_{\tau\lambda_1 \|\sigma(\cdot_{\langle m \rangle})\|_1}(2\tilde{\mathcal{X}}^{(k)}_{[m]} - \mathcal{V}^{(k)}_{[m]}),\ m \in \mathbb{N}_M$
     5b. $\mathcal{R}^{(k)}_{[M+1]} = \boldsymbol{Z}_1(2\tilde{\mathcal{X}}^{(k)}_{[M+1]} - \mathcal{V}^{(k)}_{[M+1]}, 2\tilde{\boldsymbol{Y}}^{(k)} - \boldsymbol{M}^{(k)})^{<M+1>}$
                                                                        (see Eq. (72))
         $\boldsymbol{Q}^{(k)} = \boldsymbol{Z}_2(2\tilde{\mathcal{X}}^{(k)}_{[M+1]} - \mathcal{V}^{(k)}_{[M+1]}, 2\tilde{\boldsymbol{Y}}^{(k)} - \boldsymbol{M}^{(k)})^{\top}$
     6a. $\mathcal{V}^{(k+1)}_{[m]} \leftarrow \mathcal{V}^{(k)}_{[m]} + \mathcal{R}^{(k)}_{[m]} - \tilde{\mathcal{X}}^{(k)}_{[m]},\ m \in \mathbb{N}_{M+1}$
     6b. $\boldsymbol{M}^{(k+1)} \leftarrow \boldsymbol{M}^{(k)} + \boldsymbol{Q}^{(k)} - \tilde{\boldsymbol{Y}}^{(k)}$
     6c. $c^{(k+1)} \leftarrow b^{(k)}$
  until convergence criterion met
  $(\tilde{\mathcal{X}}, \tilde{\boldsymbol{Y}}, b) \leftarrow (1/(M+1) \sum_{m \in \mathbb{N}_{M+1}} \tilde{\mathcal{X}}^{(k)}_{[m]}, \tilde{\boldsymbol{Y}}^{(k)}, b^{(k)})$
  return $(\tilde{\mathcal{X}}, \tilde{\boldsymbol{Y}}, b)$
```

6.5 Algorithm for hard-completion

As before in order to devise a solution strategy for problem (75) we accommodate Algorithm 1. Consider the abstract space $\mathscr{W}$ obtained endowing $\mathscr{V}^{M+1}$ with the canonical inner product. Let us introduce the constraint set:

$$\begin{aligned}\bar{\mathscr{C}} := \big\{(\tilde{\mathcal{X}}_{[1]}, \tilde{\mathcal{X}}_{[2]}, \ldots, \tilde{\mathcal{X}}_{[M+1]}) \in \mathscr{W} : \tilde{\mathcal{X}}_{[1]} = \tilde{\mathcal{X}}_{[2]} = \cdots \\ = \tilde{\mathcal{X}}_{[M+1]},\ \Omega_{\mathscr{S}} \tilde{\mathcal{X}}_{[m]} = z^x\ \forall m \in \mathbb{N}_{M+1}\big\}.\end{aligned} \tag{76}$$

It is clear that a solution of (75) is readily obtained from a solution of the following problem:

$$\begin{array}{ll} \underset{(\tilde{\mathcal{X}}_{[1]},\tilde{\mathcal{X}}_{[2]},\ldots,\tilde{\mathcal{X}}_{[M+1]})\in\mathscr{W}}{\text{minimize}} & \sum_{m\in\mathbb{N}_{M+1}} \Omega_m(\tilde{\mathcal{X}}_{[m]}) \\ \text{subject to} & (\tilde{\mathcal{X}}_{[1]},\tilde{\mathcal{X}}_{[2]},\ldots,\tilde{\mathcal{X}}_{[M+1]}) \in \bar{\mathscr{C}}. \end{array} \tag{77}$$

Note that, with respect to the prototypical problem (8), we now have that $\bar{f}$ is identically zero and

$$\bar{g} : (\tilde{\mathcal{X}}_{[1]},\tilde{\mathcal{X}}_{[2]},\ldots,\tilde{\mathcal{X}}_{[M+1]}) \mapsto \sum_{m\in\mathbb{N}_{M+1}} \Omega_m(\tilde{\mathcal{X}}_{[m]}). \tag{78}$$

Additionally, the projection of elements of $\mathscr{W}$ onto $\bar{\mathscr{C}}$ can be computed in closed form. To see this let $\tilde{\mathcal{X}}|_{\mathcal{B}}$ denote the tensor obtained from $\tilde{\mathcal{X}} \in \mathbb{R}^{D_1\times\cdots\times D_M\times N}$ setting to zero those entries that are not indexed by $\mathcal{B} \subset \mathbb{N}_{D_1} \times \mathbb{N}_{D_2} \times \cdots \times \mathbb{N}_{D_M} \times \mathbb{N}_N$:

$$(\tilde{\mathcal{X}}|_{\mathcal{B}})_{b_1 b_2\cdots b_M c} := \begin{cases} 0 & \text{if } (b_1, b_2, \ldots, b_M, c) \notin \mathcal{B} \\ x_{b_1 b_2\cdots b_M c} & \text{otherwise.} \end{cases}$$

We have the following result where we denote by $\mathscr{S}^c_{\mathcal{X}}$ the complement of $\mathscr{S}_{\mathcal{X}}$.

Proposition 4 (Projection onto $\bar{\mathscr{C}}$) *Let $\mathscr{W}$ and $\bar{\mathscr{C}}$ be defined as above. Then for any $(\tilde{\mathcal{X}}_{[1]},\ldots,\tilde{\mathcal{X}}_{[M+1]}) \in \mathscr{W}$, it holds that*

$$P_{\bar{\mathscr{C}}}(\tilde{\mathcal{X}}_{[1]},\tilde{\mathcal{X}}_{[2]},\ldots,\tilde{\mathcal{X}}_{[M+1]}) = (\mathcal{Z},\mathcal{Z},\ldots,\mathcal{Z})$$

where

$$\mathcal{Z} = \left(\frac{1}{M+1}\sum_{m\in\mathbb{N}_{M+1}} \tilde{\mathcal{X}}_{[m]}\right)\Bigg|_{\mathscr{S}^c_{\mathcal{X}}} + S^*_{\mathscr{S}_{\mathcal{X}}} z^x,$$

and we denoted by $S^*_{\mathscr{S}_{\mathcal{X}}}$ the adjoint of the sampling operator $S_{\mathscr{S}_{\mathcal{X}}}$.

Proof See Appendix D. □

Finally by Propositions 2 and 1 it follows that $\text{prox}_{\tau\bar{g}}$ in Step 3 of MAIN (Algorithm 1) reads:

$$\begin{aligned} &\text{prox}_{\tau\bar{g}}(\tilde{\mathcal{X}}_{[1]},\ldots,\tilde{\mathcal{X}}_{[M+1]}) \\ &\quad = \left(\text{prox}_{\tau\lambda_1\|\sigma(\cdot_{\langle 1\rangle})\|_1}(\tilde{\mathcal{X}}_{[1]}),\ldots,\text{prox}_{\tau\lambda_M\|\sigma(\cdot_{\langle M\rangle})\|_1}(\tilde{\mathcal{X}}_{[M+1]})\right). \end{aligned} \tag{79}$$

The steps needed to compute a solution are reported in Algorithm 3, which is obtained adapting Algorithm 1 to the present setting. With reference to the latter, note that INEXACTPROXY is no longer needed. Indeed, since $\bar{f}$ is identically zero, (32) boils down to computing the projection onto $\bar{\mathscr{C}}$. By Proposition 4, this can be done in closed form.

Remark 8 Algorithm 3 is explicitly designed for the higher order case ($M \geq 2$). However it can be easily simplified to perform hard completion of matrices ($M = 1$). In this case it is not difficult to see that one needs to evaluate only one proximity operator; consequently, duplication of the matrix unknown can be avoided.

Algorithm 3 HardCompletion()

```
input   index set $\mathscr{S}_{\mathcal{X}}$, vector of measurements $z^x$
output estimate $\tilde{\mathcal{X}}$

procedure MAIN()
  comment $\tau > 0$ arbitrarily fixed.

  1. $(\mathcal{W}_{[1]}^{(0)}, \dots, \mathcal{W}_{[M+1]}^{(0)}) \in \mathscr{W}$

  repeat

    2.  $\tilde{\mathcal{X}}^{(k)} \leftarrow (\frac{1}{M+1} \sum_{m \in \mathbb{N}_{M+1}} \mathcal{W}_{[m]}^{(k)})|_{\mathscr{S}_{\mathcal{X}}^c} + S_{\mathscr{S}_{\mathcal{X}}}^* z^x$
    3.  $\mathcal{R}_{[m]}^{(k)} \leftarrow \mathrm{prox}_{\tau\lambda_m \|\sigma(\cdot_{\langle m \rangle})\|_1}(2\tilde{\mathcal{X}}^{(k)} - \mathcal{W}_{[m]}^{(k)})$    $\forall m \in \mathbb{N}_{M+1}$
    4.  $\mathcal{W}_{[m]}^{(k+1)} \leftarrow \mathcal{W}_{[m]}^{(k)} + \mathcal{R}_{[m]}^{(k)} - \tilde{\mathcal{X}}^{(k)}$    $\forall m \in \mathbb{N}_{M+1}$

  until convergence criterion met
  $\tilde{\mathcal{X}} \leftarrow \tilde{\mathcal{X}}^{(k)}$
  return $(\tilde{\mathcal{X}})$
```

7 Inductive learning with tensor data

For the inductive case the goal is to learn a predictive model based upon a dataset $\mathscr{D}_N$ of N input-target training pairs

$$\mathscr{D}_N := \left\{ \left(\mathcal{X}^{(n)}, y^{(n)} \right) \in \mathbb{R}^{D_1 \times D_2 \times \cdots \times D_M} \times \mathscr{Y}^T : n \in \mathbb{N}_N \right\}. \tag{80}$$

Each item is represented by an M-th order tensor and is associated with a vector of T labels. As before, we focus on the case where $\mathscr{Y} = \{-1, 1\}$. For ease of notation we assumed that we have the same input data across the tasks; in general, however, this needs not to be the case.

To understand the rationale behind the regularization approach we are about to propose, consider the following generative mechanism.

7.1 Modelling assumptions

For a generic item, represented by the tensor $\mathcal{X}$, assume the decomposition $\mathcal{X} = \tilde{\mathcal{X}} + \mathcal{E}$ where

$$\tilde{\mathcal{X}} = \mathcal{S}_{\tilde{\mathcal{X}}} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times \cdots \times_M \boldsymbol{U}_M \tag{81}$$

where for any $m \in \mathbb{N}_M$ and $R_m < D_m$, $\boldsymbol{U}_m \in \mathbb{R}^{D_m \times R_m}$ is a matrix with orthogonal columns. Note that the core tensor $\mathcal{S}_{\tilde{\mathcal{X}}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_M}$ and $\mathcal{E} \in \mathbb{R}^{D_1 \times D_2 \times \cdots \times D_M}$ are item-specific; on the other hand for any $m \in \mathbb{N}_M$, the full rank matrix $\boldsymbol{U}_m$ spans a latent space relevant to the tasks at hand and common to all the input data. To be precise we assume the target label y_t were generated according to the probability model $p(y_t|\tilde{y}_t) = 1/(1 + \exp(-y_t \tilde{y}_t))$, where $\tilde{y}_t$ depends upon the core tensor $\mathcal{S}_{\tilde{\mathcal{X}}}$:

$$\tilde{y}_t = \langle \mathcal{S}_{\tilde{\mathcal{X}}}, \mathcal{S}_{\mathcal{W}^{(t)}} \rangle + b_t \tag{82}$$

where $\mathcal{S}_{\mathcal{W}^{(t)}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_M}$ and b_t are task-specific unknowns. It is important to remark that, in this scenario, $\mathcal{S}_{\mathcal{W}^{(t)}}$ comprises $R_1 R_2 \cdots R_M \ll D_1 D_2 \cdots D_M$ parameters. In practice the common latent spaces as well as the core tensor $\mathcal{S}_{\tilde{\mathcal{X}}}$ are both unknowns so that $\mathcal{S}_{\mathcal{W}^{(t)}}$ cannot be estimated directly. However if we further assume that

$$\mathscr{R}(\mathcal{E}_{\langle m \rangle}) \perp \mathscr{R}(\boldsymbol{U}_m) \tag{83}$$

for at least one $m \in \mathbb{N}_M$, where we denote by $\mathscr{R}(\boldsymbol{A})$ the range of a matrix $\boldsymbol{A}$, one has the following.

Proposition 5 *Assume* (83) *holds for* $m_1 \in \mathbb{N}_M$. *Then*

$$\tilde{y}_t = \langle \mathcal{X}, \mathcal{W}^{(t)} \rangle + b_t \tag{84}$$

where $\mathcal{W}^{(t)} \in \mathbb{R}^{D_1 \times D_2 \times \cdots \times D_M}$ *is the low multilinear rank tensor*:

$$\mathcal{W}^{(t)} = \mathcal{S}_{\mathcal{W}^{(t)}} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times \cdots \times_M \boldsymbol{U}_M. \tag{85}$$

Proof See Appendix E. □

Note that the right-hand side of (84) is an affine function of the given higher order representation $\mathcal{X}$ of the item, rather than an affine function of the unobserved core tensor $\mathcal{S}_{\tilde{\mathcal{X}}}$, as in (82).

Remark 9 Equation (83) requires that $\mathcal{E}$ does not "overlap" with the discriminative features; in practice this will not hold. One can only hope that $\mathcal{E}$ does not "overlap too much" so that $\tilde{y}_t \approx \langle \mathcal{X}, \mathcal{W}^{(t)} \rangle + b_t$.

Let now $\mathcal{W} \in \mathbb{R}^{D_1 \times D_2 \times \cdots \times D_M \times T}$ be the tensor that gathers all the T tasks:

$$\tilde{w}_{d_1 d_2 \cdots d_M t} := \tilde{w}^{(t)}_{d_1 d_2 \cdots d_M} \quad \text{for any } t \in \mathbb{N}_T. \tag{86}$$

Additionally we consider the case where the tasks are related, as common in the literature of multi-task learning, see e.g. Argyriou et al. (2007c). In our context this is accomplished by assuming that $\mathcal{W}_{\langle M+1 \rangle}$ can be explained by a limited number of factors, namely that $\mathcal{W}_{\langle M+1 \rangle}$ admits thin SVD (Golub and Van Loan 1996):

$$\mathcal{W}_{\langle M+1 \rangle} = \boldsymbol{U}_{M+1} \boldsymbol{S}_{M+1} \boldsymbol{V}_{M+1}^{\top}$$

where for $R_{M+1} \ll D_{M+1}$ one has $\boldsymbol{U}_{M+1} \in \mathbb{R}^{D_{M+1} \times R_{M+1}}$, $\boldsymbol{S}_{M+1} \in \mathbb{R}^{R_{M+1} \times R_{M+1}}$ and finally $\boldsymbol{V}_{M+1} \in \mathbb{R}^{R_{M+1} \times D_1 D_2 \cdots D_M}$. Note that $\mathcal{W}$ can now be equivalently restated as the low multilinear rank tensor:

$$\mathcal{W} = \mathcal{S}_{\mathcal{W}} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \cdots \times_{M+1} \boldsymbol{U}_{M+1} \tag{87}$$

for some core tensor $\mathcal{S}_{\mathcal{W}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_M \times R_{M+1}}$ and latent matrices $\boldsymbol{U}_m$, $m \in \mathbb{N}_{M+1}$ that define subspaces that concentrate the discriminative relationship.

We conclude by pointing out that a supervised learning problem where data observations are represented as matrices, namely second order tensors, is a special case of our setting. Single classification tasks in this situation were studied in Tomioka and Aihara (2007). Similarly to the present setting, the latter proposes a spectral regularization as a principled way to perform complexity control over the space of matrix-shaped models. Before discussing a solution strategy we point out that the method can be easily generalized to regression problems by changing the loss function.

7.2 Model estimation

As for transduction we evaluate misclassification errors via the logistic loss; we therefore measure the empirical risk associated to the different tasks on the dataset (80) via:

$$f_{\mathscr{D}_N} : (\mathcal{W}, b) \mapsto \sum_{n\in\mathbb{N}_N} \sum_{t\in\mathbb{N}_T} \log\big(1 + \exp\big(-y_t^{(n)}\big(\big\langle \mathcal{X}^{(n)}, \mathcal{W}^{(t)}\big\rangle + b_t\big)\big)\big) \tag{88}$$

where

$$w_{d_1 d_2 \cdots d_M t} := w^{(t)}_{d_1 d_2 \cdots d_M} \quad \text{for any } t \in \mathbb{N}_T. \tag{89}$$

The pair $(\mathcal{W}, b)$ is estimated based upon the following penalized empirical risk minimization problem:

$$\underset{(\mathcal{W},b)\in\mathscr{V}}{\text{minimize}}\ f_{\mathscr{D}_N}(\mathcal{W}, b) + \textstyle\sum_{m\in\mathbb{N}_{M+1}} \lambda_m \|\mathcal{W}_{\langle m\rangle}\|_* \tag{90}$$

where $\mathscr{V}$ is formed upon the module spaces $\mathbb{R}^{D_1\times D_2\times\cdots\times D_M\times T}$ and $\mathbb{R}^T$. The composite spectral penalty in (90) is designed to match the assumption that $\mathcal{W}$ has low multilinear rank, as discussed above. Note that, in line with (87), other than performing complexity control, the regularization allows one to determine subspaces that concentrate discriminative information, without any additional feature extraction step.

7.3 Algorithm for inductive learning

Consider the abstract vector space $\mathscr{W}$ obtained endowing

$$\big(\times_{m\in\mathbb{N}_{M+1}}\big\{\mathbb{R}^{D_1\times D_2\times\cdots\times D_M\times T}\big\}\big) \times \mathbb{R}^T \tag{91}$$

with the canonical inner product. Additionally let

$$\bar{\mathscr{C}} := \big\{(\mathcal{W}_{[1]}, \mathcal{W}_{[2]}, \ldots, \mathcal{W}_{[M]}, \mathcal{W}_{[M+1]}, b) \in \mathscr{W} : \mathcal{W}_{[1]} = \mathcal{W}_{[2]} = \cdots = \mathcal{W}_{[M+1]}\big\}. \tag{92}$$

We solve (90) based upon the following problem:

$$\begin{array}{ll} \underset{(\mathcal{W}_{[1]},\mathcal{W}_{[2]},\ldots,\mathcal{W}_{[M]},\mathcal{W}_{[M+1]},b)\in\mathscr{W}}{\text{minimize}} & \bar{f}(\mathcal{W}_{[1]}, \ldots, \mathcal{W}_{[M+1]}, b) + \bar{g}(\mathcal{W}_{[1]}, \ldots, \mathcal{W}_{[M+1]}) \\ \text{subject to} & (\mathcal{W}_{[1]}, \ldots, \mathcal{W}_{[M+1]}, b) \in \bar{\mathscr{C}} \end{array} \tag{93}$$

where

$$\bar{f}(\mathcal{W}_{[1]}, \ldots, \mathcal{W}_{[M+1]}, b) := \frac{1}{M+1} \sum_{m\in\mathbb{N}_{M+1}} f_{\mathscr{D}_N}(\mathcal{W}_{[m]}, b) \tag{94}$$

and $\bar{g}$ is the same as in (78). Its proximity operator is found in (79). The gradient of $\bar{f}$ is given in Appendix C.

Algorithm 4 InductiveLearning()

```
input  data set $\mathscr{D}_N$
output estimate for $\mathcal{W}$ (equation (84)), and vector of bias terms $b$

procedure MAIN()
  comment $\epsilon_0 > 0,\ \sigma > 1,\ \tau > 0$ arbitrarily fixed.
  comment procedure INEXACTPROXY is found in Algorithm 1
  1. $(\mathcal{V}_{[1]}^{(k)}, \ldots, \mathcal{V}_{[M+1]}^{(k)}, c^{(k)}) \in \mathscr{W}$
  2. $\kappa \leftarrow \sqrt{\tau L_{\bar{f}}(\tau L_{\bar{f}} + 1)} + \tau L_{\bar{f}}$
  repeat
      3. $\epsilon^{(k)} \leftarrow \epsilon_0/(\kappa(k+1)^{\sigma})$
      4. $(\mathcal{W}_{[1]}^{(k)}, \ldots, \mathcal{W}_{[M+1]}^{(k)}, b^{(k)}) \leftarrow$
         INEXACTPROXY$((\mathcal{V}_{[1]}^{(k)}, \ldots, \mathcal{V}_{[M+1]}^{(k)}, c^{(k)}), \epsilon^{(k)}, \tau, L_{\bar{f}})$
      5. $\mathcal{R}_{[m]}^{(k)} \leftarrow \mathrm{prox}_{\tau\lambda_m \|\sigma(\cdot_{\langle m \rangle})\|_1}(2\mathcal{W}_{[m]}^{(k)} - \mathcal{V}_{[m]}^{(k)}),\ m \in \mathbb{N}_{M+1}$
     6a. $\mathcal{V}_{[m]}^{(k+1)} \leftarrow \mathcal{V}_{[m]}^{(k)} + \mathcal{R}_{[m]}^{(k)} - \mathcal{W}_{[m]}^{(k)},\ m \in \mathbb{N}_{M+1}$
     6b. $c^{(k+1)} \leftarrow b^{(k)}$
  until convergence criterion met
  $(\mathcal{W}, b) \leftarrow (1/(M+1)\sum_{m \in \mathbb{N}_{M+1}} \mathcal{W}_{[m]}^{(k)}, b^{(k)})$
  return $(\mathcal{W}, b)$
```

8 Experiments

8.1 Transductive learning

We begin by presenting experiments on transductive learning with multiple tasks, see Sect. 6.

8.1.1 Evaluation criterion and choice of parameters

As performance indicators we considered: (1) the capability of each procedure to predict the correct test labels; (2) the capability to interpolate the missing entries in the input data tensor. We measure the latter based upon the *normalized root mean square error* (NRMSE) on the complementary set $\mathscr{S}_{\tilde{\mathcal{X}}}^{c}$:

$$\mathrm{NRMSE}(\tilde{\mathcal{X}}, \hat{\mathcal{X}}) := \frac{\|S_{\mathscr{S}_{\tilde{\mathcal{X}}}^{c}} \tilde{\mathcal{X}} - S_{\mathscr{S}_{\tilde{\mathcal{X}}}^{c}} \hat{\mathcal{X}}\|}{(\max(S_{\mathscr{S}_{\tilde{\mathcal{X}}}^{c}}) - \min(S_{\mathscr{S}_{\tilde{\mathcal{X}}}^{c}}))\sqrt{J^{c}}} \tag{95}$$

where we denoted by J^c the cardinality of $\mathscr{S}_{\tilde{\mathcal{X}}}^{c}$ and $\tilde{\mathcal{X}}$ is as in (50). For both *tensor soft-completion* (tensor-sc) and *matrix soft-completion* (matrix-sc) we solve the optimization problem in (60) via the approach presented in Sect. 6.3. The parameter λ_0 is chosen in the set $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. For tensor-sc we set the parameters in the composite spectral penalty as

$$\lambda_m = \frac{1}{M+1}\bar{\lambda} \quad \text{for any } m \in \mathbb{N}_{M+1} \tag{96}$$

where $M+1$ is the order of the input data tensor and $\bar{\lambda}$ is a varying parameter. For matrix-sc we take

$$\lambda_m = \begin{cases} 0 & \text{if } m \in \mathbb{N}_M \\ \bar{\lambda} & \text{if } m = M+1. \end{cases} \tag{97}$$

Notice that by doing so we essentially recover the matrix-based approach proposed in Goldberg et al. (2010, Formulation 1). We let $\bar{\lambda}$ in both (96) and (97) vary on a wide range. More precisely we follow Goldberg et al. (2010) and Ma et al. (2011); for each value of λ_0 we compute the regularization path with respect to $\bar{\lambda}$ beginning with a large value $\bar{\lambda}^{(0)}$ and solving a sequence of problems with $\bar{\lambda}^{(t)} = \eta_{\bar{\lambda}}\bar{\lambda}^{(t-1)}$ where as in Goldberg et al. (2010) we consider as decay parameter $\eta_{\bar{\lambda}} = 0.25$. At each step t we perform warm-starting, that is, we take as initial point the solution obtained at step $t-1$. We stop when $\bar{\lambda} \leq 10^{-6}$. For both tensor and matrix soft-completion we choose the values of parameters corresponding to the minimum fraction of mis-predicted labels of a hold-out validation set.

8.1.2 Implementation of the optimization algorithm

As termination criterion for the algorithm that finds a solution of (60) we use the relative increment (36) where we set $\eta = 10^{-4}$. With reference to Algorithms 2 and 3 we let $\epsilon_0 = 10^{-2}$ and set $\sigma = 1.1$. We use a backtracking procedure to find an upper bound for the Lipschitz constant $L_{\bar{q}}$ (see Remark 5 and references therein). Finally we let $\tau = 0.02/\tilde{L}_{\bar{f}}$ where $\tilde{L}_{\bar{f}}$ is an upper bound for the Lipschitz constant $L_{\bar{f}}$, also found via backtracking. As explained above we compute the entire path with respect to the penalty parameter and use warm-starting at each step. At step $t=0$ the initialization of the algorithm is performed as follows. For both matrix as well as tensor-sc we set $b^{(0)}$ to be a vector of zeros. For what concerns $\mathcal{X}^{(0)}$ and $\tilde{\boldsymbol{Y}}^{(0)}$ we do as follows. Let $\mathcal{X}^{*}$ and $\boldsymbol{Y}^{*}$ be obtained setting to zero unobserved entries of $\mathcal{X}$ and $\boldsymbol{Y}$ respectively. Consider a partitioning $[\boldsymbol{Z}_{M+1,1}, \boldsymbol{Z}_{M+1,2}]$ of the rank-1 approximation of the matrix $[\mathcal{X}^{*}_{\langle M+1\rangle}, \boldsymbol{Y}^{*\top}]$ consistent with the dimension of $\mathcal{X}^{*}_{\langle M+1\rangle}$ and $\boldsymbol{Y}^{*\top}$. Both matrix-sc and tensor-sc are then initialized according to

$$\mathcal{X}^{(0)} = \boldsymbol{Z}_{M+1,1}^{\langle M+1\rangle} \quad \text{and} \quad \tilde{\boldsymbol{Y}}^{(0)} = \boldsymbol{Z}_{M+1,2}^{\top}.$$

This approach is adapted from the method suggested in Goldberg et al. (2010) for their matrix soft completion procedure.

8.1.3 Alternative approach

We also report results obtained using linear kernel within LS-SVM classifiers applied to vectorized input data, see Suykens and Vandewalle (1999). We find models via the LS-SVMlab toolbox (De Brabanter et al. 2010). A classifier is built for each task independently since these models do not handle vector-valued labels simultaneously. Although the presence of missing values in the context of LS-SVM has been studied (Pelckmans et al. 2005) the toolbox does not implement any strategy to handle this situation. For this reason we considered as input data the vectorized version of $\mathcal{X}^{*} + \boldsymbol{Z}_{M+1,1}^{\langle M+1\rangle}|_{\mathcal{S}^{c}_{\mathcal{X}}}$ where $\mathcal{X}^{*}$ and $\boldsymbol{Z}_{M+1,1}^{\langle M+1\rangle}$ are as in the previous paragraph. We denote this approach as imp+ls-svm where imp is a shorthand for imputation.

Table 2 Fractions of unobserved labels (test data) predicted incorrectly and NRMSEs. tensor-sc exploits low rank assumptions along all the modes; in contrast, matrix-sc works only with the third mode unfolding (hereby ignoring the two-way nature of each data observation). tensor-sc generally performs comparably or better than matrix-sc and imp+ls-svm in terms of misclassification errors; tensor-sc generally outperforms matrix-sc on the reconstruction of the underlying input data tensor $\tilde{\mathcal{X}}$, see (50)

ml-rank	N	ω	tensor-sc		matrix-sc		imp+ls-svm
			label error	NRMSE ($\times 10^{-2}$)	label error	NRMSE ($\times 10^{-2}$)	label error
(3, 3, 3)	30	0.2	**0.28**(0.09)	**6.81**(1.24)	0.31(0.08)	7.63(1.42)	0.32(0.03)
		0.3	**0.20**(0.07)	**3.29**(2.39)	0.21(0.07)	5.98(2.24)	**0.20**(0.05)
		0.4	**0.13**(0.04)	**1.68**(1.29)	**0.13**(0.04)	3.87(1.12)	0.14(0.06)
	90	0.2	**0.11**(0.03)	**2.75**(1.00)	**0.11**(0.02)	3.74(1.26)	0.16(0.03)
		0.3	**0.08**(0.02)	**0.87**(0.36)	0.09(0.02)	1.68(0.67)	0.10(0.02)
		0.4	**0.05**(0.01)	**1.87**(1.56)	0.06(0.01)	2.55(1.67)	0.07(0.02)
(3, 3, 9)	30	0.2	0.44(0.09)	**11.13**(2.52)	0.42(0.09)	11.57(1.96)	**0.41**(0.05)
		0.3	**0.29**(0.04)	**5.57**(2.53)	0.33(0.06)	10.19(2.51)	0.33(0.03)
		0.4	**0.25**(0.04)	**4.91**(5.31)	0.27(0.04)	8.82(2.88)	0.27(0.04)
	90	0.2	**0.17**(0.02)	**4.71**(0.56)	0.18(0.02)	6.63(1.16)	0.27(0.02)
		0.3	**0.13**(0.02)	**3.97**(2.67)	0.14(0.02)	4.07(1.62)	0.17(0.03)
		0.4	**0.10**(0.01)	**2.81**(3.15)	0.11(0.02)	3.44(2.10)	0.14(0.02)

8.1.4 Soft completion: toy problems on multi-labeled data

For the first set of experiments we considered a family of synthetic datasets following the generative mechanism illustrated in Sect. 6.1. For each experiment we generated a core tensor $\mathcal{S}$ in $\mathbb{R}^{r_1 \times r_2 \times r_3}$ with entries i.i.d. from a normal distribution; for $i \in \{1, 2\}$ a matrix $\boldsymbol{U}_i \in \mathbb{R}^{D \times r_i}$ with entries i.i.d. from a normal distribution. Finally $\boldsymbol{U}_3 \in \mathbb{R}^{N \times r_3}$ was generated according to the same distribution. The input data tensor in $\mathbb{R}^{D \times D \times N}$ was taken to be

$$\mathcal{X} = \mathcal{S} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \boldsymbol{U}_3 + \sigma\, \mathcal{E}.$$

Next, for each task $t \in \mathbb{N}_T$ we created a weight tensor $\mathcal{W}_t$ and a bias b_t, again with independent and identically normally distributed entries; successively, we produced $\tilde{\boldsymbol{Y}}$ and $\boldsymbol{Y}$ according to (55) and the probability model (54).[11] Finally, the sampling sets $\mathscr{S}_{\mathcal{X}}$ and $\mathscr{S}_{Y}$ in (48) and (49) were created by picking uniformly at random a fraction ω of entries of the data tensor and the target matrix respectively. For matrix and tensor soft-completion we performed model selection by using 70 % of these entries for training; we measure the performance corresponding to each parameter pair on the hold-out validation set constituted by the remaining entries. We finally use the optimal values of parameters and train with the whole set of labeled data; we then measure performance on the hold-out test set. Model selection for the linear LS-SVM classifiers was based on 10-fold cross-validation.

The procedure above was considered for $D = 30$, $T = 10$, $\sigma = 0.1$ and different values of the multilinear rank (r_1, r_2, r_3), N and ω. Table 2 concerns the fraction of unobserved labels (that is, test data) predicted incorrectly by the different procedures as well as NRMSEs. Note

[11] Note that each input observation possesses multiple (binary) labels. This situation differs from the multi-class paradigm that we consider later on. In here any possible binary vector is admissible; by contrast, in a multi-class setting only those vectors belonging to the codebook are admissible.

that the latter is not reported for the linear LS-SVM models as these approaches do not have an embedded imputation strategy. We report the mean (and standard deviation) over 10 independent trials where each trial deals with independently generated data and sample sets.

Remark 10 According to Table 2 tensor-sc generally performs comparably or better to matrix-sc in terms of misclassification errors; however the experiments show that the former leads to more favorable results for the reconstruction of the underlying input data tensor $\tilde{\mathcal{X}}$, see (50).

8.1.5 Multi-class categorization via soft-completion: Olivetti faces

In this experiment we deal with classification of pictures of faces of different persons; we compared tensor-sc with matrix-sc and imp+ls-svm as before. We considered the first five persons of the Olivetti database.[12] For each person, ten different 56×46 grayscale pictures[13] are available; the input dataset consists therefore of a $(56 \times 46 \times 50)$-tensor of which 65 % of entries were artificially removed. For each input image a vector-valued target label was created with one-vs-one encoding. That is, if $c_i \in \{1, 2, \dots, 5\}$ denotes the class (person) indicator for the i-th image we set $y^{(i)} \in \{-1, 1\}^5$ to be

$$\left(y^{(i)}\right)_j = \begin{cases} -1, & \text{if } j \neq c_i \\ 1, & \text{otherwise.} \end{cases}$$

The same type of encoding was also used within imp+ls-svms. For these classifiers we considered as input the vector unfoldings of the images. For tensor-sc the task is to simultaneously complete the $(56 \times 46 \times 50)$-tensor and the (5×50)-matrix of target vectors. Likewise matrix-sc, obtained from (60) setting all but the last regularization parameter equal to zero, treats each image as a vector by considering only the last matrix unfolding. In all the cases we use 25 images for training and validation and the remaining for testing. As for the toy problems above, the spectral penalties parameters within matrix-sc and tensor-sc were set according to (97) and (96). We compute the regularization path corresponding to the free parameter $\bar{\lambda}$. In all the cases the selection of parameters is driven by the misclassification error. For the LS-SVM models we used ten-fold CV. For matrix-sc and tensor-sc we chose parameters according to a hold-out set. More precisely, of the 25 images we use 17 for actual training and consider the remaining 8 for validation and then use all of the 25 images once the set of optimal parameters has been found. Following this procedure we performed five trials each of which obtained from random splitting of training and test data and random mask of input missing entries. For each method we report in Table 3 the cumulative confusion matrix obtained summing up the confusion matrices associated to the test (unlabeled) data found in the different trials. Table 4 summarizes the performance of the different methods in terms of classification accuracy and feature imputation.

Remark 11 Note that, since the choice of parameters was driven by misclassification errors, the objective of the approach is the correct completion of the labeling. Therefore the estimated input features $\hat{\mathcal{X}}$ in (60) should be interpreted as carriers of latent discriminative information rather than a reconstruction of the underlying images. With reference to Remark 7 we interpret here $\mathcal{X}^{(i)}$ as the true representation of the i-th image, $\tilde{\mathcal{X}}^{(i)}$ as latent discriminative features and $\hat{\mathcal{X}}^{(i)}$ as their estimates.

[12] Publicly available at http://cs.nyu.edu/~roweis/data.html.

[13] Sizes refer to the images obtained after removing borders.

Table 3 Cumulative confusion matrices for the different procedures. tensor-sc leads to better classification accuracy in comparison to the alternative techniques

(a) tensor-sc

		y				
		1	2	3	4	5
$\hat{y}$	1	19	3	0	0	3
	2	0	29	0	0	0
	3	0	0	20	0	0
	4	0	0	1	23	0
	5	0	0	0	0	27

(b) matrix-sc

		y				
		1	2	3	4	5
$\hat{y}$	1	17	3	1	0	4
	2	2	27	0	0	0
	3	0	0	20	0	0
	4	0	0	3	21	0
	5	1	0	2	0	24

(c) imp+ls-svm

		y				
		1	2	3	4	5
$\hat{y}$	1	14	5	0	1	5
	2	0	29	0	0	0
	3	0	0	19	0	1
	4	0	0	2	22	0
	5	0	0	0	0	27

Table 4 Mean and standard deviation of misclassification error rates and NRMSE of features imputation for the Olivetti dataset

tensor-sc		matrix-sc		imp+ls-svm
label err	NRMSE ($\times 10^{-2}$)	label err	NRMSE ($\times 10^{-2}$)	label err
0.05(0.10)	19.90(9.75)	0.11(0.14)	20.90(10.24)	0.09(0.13)

Remark 12 Unlike in the toy problems above, for which $\tilde{\mathcal{X}}$ was available, the NRMSEs in Table 4 are computed upon the actual set of images $\mathcal{X}$.

Figure 4 illustrates the retrieval of latent features for some unlabeled (test) pictures. Notably the latent features obtained by tensor-sc look as over-smoothed images whereas those obtained by matrix-sc generally look more noisy. In particular, the cases for which matrix-sc incorrectly assigns labels often correspond to situations where latent features do not capture person-specific traits (first and second rows). Wrongly assigned labels also correspond to cases where latent features are close to those corresponding to a different person (last two rows).

8.1.6 Hard completion: toy problems

In here we test the capability of hard completion (Sect. 6.4), denoted by tensor-hc, to recover missing entries of a partially specified input data tensor of order 3. We compare to the case where the higher order structure is neglected; in this case the input data tensor is flattened into its third matrix unfolding and one performs matrix completion of the arising second order tensor (matrix-hc). Note that, with reference to (75), this is equivalent to retain the tensor structure and set all but the last of the regularization parameters $\lambda_m,\ m \in \{1, 2, 3\}$

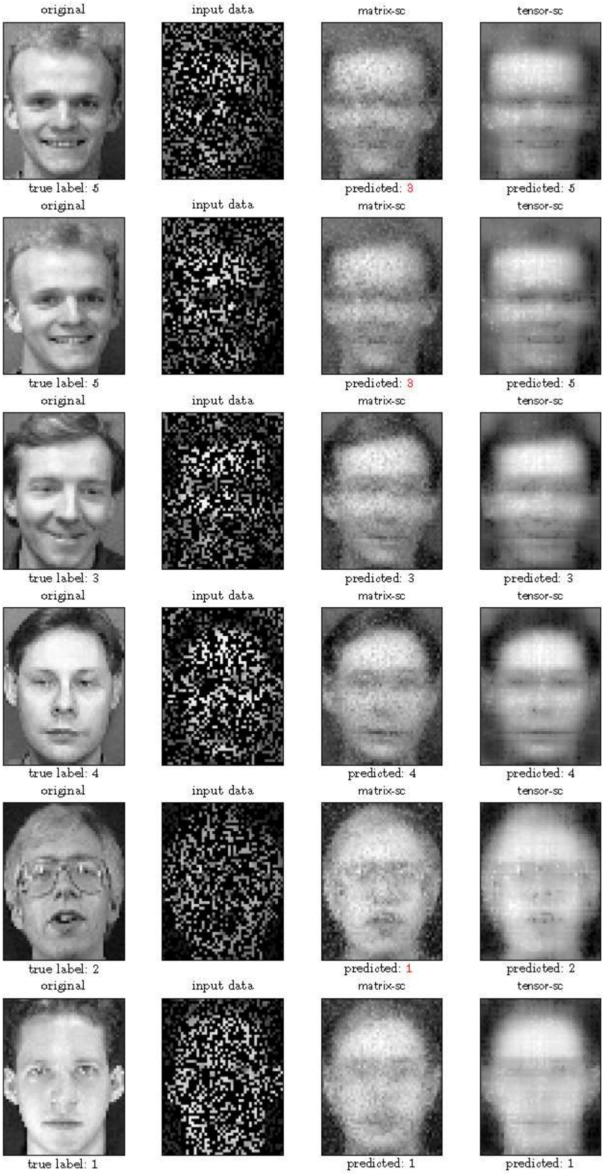

Fig. 4 Olivetti faces: the task is to simultaneously complete the input features, a $(56 \times 46 \times 50)$-tensor (ten 56×46 images for each one of the five persons), and the (5×50)-matrix of five-dimensional target vectors, where five relates to the one-versus-one encoding used for the five classes. Here we reported the reconstructed features and assigned labels for 6 unlabelled 56×46 images; see also Tables 3 and 4. Wrong labels are reported in red. Estimated input features should be interpreted as carriers of latent discriminative information rather than a reconstruction of the underlying images, see Remark 11. In particular, the cases for which matrix-sc incorrectly assigns labels often correspond to situations where latent features do not capture person-specific traits (*first and second rows*). Wrongly assigned labels also correspond to cases where latent features are close to those corresponding to a different person (*last two rows*)

to zero. In either case we compute solutions via Algorithm 3 keeping into account the simplifications that occur in the second order case. For each experiment we generated a core tensor $\mathcal{S}$ in $\mathbb{R}^{r_1 \times r_2 \times r_3}$ with entries i.i.d. according to the uniform distribution on the interval $[-0.5, 0.5]$, denoted as $U([-0.5, 0.5])$; for $i \in \{1, 2, 3\}$ a matrix $\boldsymbol{U}_i \in \mathbb{R}^{D \times r_i}$ was generated also with entries i.i.d. from $U([-0.5, 0.5])$. The input data tensor in $\mathbb{R}^{D \times D \times D}$ was taken to be $\mathcal{X} = \tilde{\mathcal{X}} + \sigma\, \mathcal{E}$ where

$$\tilde{\mathcal{X}} = \mathcal{S} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times_3 \boldsymbol{U}_3$$

Table 5 NRMSEs and execution times for tensor-hc and matrix-hc; the latter completes the given tensor based upon low rank assumption on the third mode only. Note that the computational complexity of tensor-hc is roughly 3 times that of matrix-hc, in line with Remark 14

σ	Multilinear rank		ω	tensor-hc		matrix-hc	
				NMRSE ($\times 10^{-2}$)	time (s)	NMRSE ($\times 10^{-2}$)	time (s)
0.02	(3, 3, 3)	(SNR ≈ 3)	0.3	**4.41**(0.48)	42.04(2.15)	5.51(0.65)	13.93(1.01)
			0.6	**3.10**(0.43)	23.24(1.14)	3.86(0.55)	8.05(1.09)
			0.9	**3.07**(0.60)	16.13(1.26)	3.71(0.73)	5.24(0.58)
	(3, 3, 9)	(SNR ≈ 9.5)	0.3	**3.47**(0.36)	54.65(2.03)	6.15(0.63)	21.09(0.89)
			0.6	**2.36**(0.26)	29.06(1.10)	4.04(0.44)	11.98(1.04)
			0.9	**2.11**(0.22)	18.27(0.96)	3.44(0.36)	5.99(0.58)
	(9, 9, 3)	(SNR ≈ 30)	0.3	3.32(0.45)	104.21(7.00)	**2.26**(0.35)	28.10(1.59)
			0.6	1.96(0.31)	41.01(2.25)	**1.57**(0.24)	13.12(0.67)
			0.9	1.77(0.22)	21.59(1.05)	**1.40**(0.18)	6.64(0.44)
0.04	(3, 3, 3)	(SNR ≈ 0.8)	0.3	**7.88**(0.87)	50.34(3.92)	8.55(0.99)	17.53(1.64)
			0.6	**5.75**(0.82)	35.03(1.51)	6.68(1.00)	11.65(0.94)
			0.9	**5.74**(1.13)	25.67(1.20)	6.68(1.13)	8.17(0.58)
	(9, 9, 3)	(SNR ≈ 2.36)	0.3	**6.28**(0.65)	77.31(3.54)	8.53(0.86)	26.74(1.88)
			0.6	**4.42**(0.50)	46.08(1.58)	6.54(0.70)	15.60(1.02)
			0.9	**3.99**(0.41)	29.97(1.10)	5.90(0.59)	9.71(0.55)
	(9, 9, 3)	(SNR ≈ 7.5)	0.3	5.95(0.78)	118.89(6.74)	**4.55**(0.60)	35.62(2.09)
			0.6	3.72(0.59)	59.66(2.54)	**3.00**(0.47)	19.39(0.88)
			0.9	3.40(0.43)	36.20(1.80)	**2.71**(0.34)	10.61(0.72)

and $\mathcal{E}$ is a $(D \times D \times D)$-tensor with independent normally distributed entries. Finally the sampling sets $\mathcal{S}_{\mathcal{X}}$ were created by picking uniformly at random a fraction ω of entries of the data tensor. We took $D = 50$ and considered different values of σ, the multilinear rank (r_1, r_2, r_3) and ω. In Table 5 we report the mean (and standard deviation) of NRMSEs and execution times over 10 independent trials where each trial deals with independently generated data and sample sets. Note that keeping fixed σ across different values of multilinear rank gives a different noise level. We therefore report in the table the approximate *signal-to-noise ratio* (SNR) obtained on the different trials. The latter is defined as $SNR := \mathrm{var}(\tilde{x})/(\sigma^2)$ where $\tilde{x}$ denotes the generic entry of $\tilde{\mathcal{X}}$ and var denotes the empirical variance.

Remark 13 The experimental evidence suggests that tensor completion performs better than matrix completion (performed along the third mode) when either no n-rank dominates the others or when the 3-rank is higher. This observation holds across different noise levels and fractions of entries used in the reconstruction.

Remark 14 As Table 5 shows, for the third order case, the computational complexity of our implementation of tensor completion is roughly three times that of matrix completion. This is expected since the computational load is determined by Step 5 and 6 which, in turn, involve a number of iterations equal to the order of the tensor ($M+1$). This is no longer needed for $M=1$ (second order case), see Remark 8.

8.1.7 *Impainting of colored images via hard completion*

In here we apply hard-completion to impainting of 8-bit RGB colored images, each of which is represented as a third order tensor. The first two modes span the pixels space; the third mode contains information from the three channels. For each image we remove entries in all the channels simultaneously (first three rows of Fig. 5), or consider the case where entries are missing at random (last two rows of Fig. 5). We then solve the problem in Eq. (75) with the sampling set $\mathscr{S}_{\mathcal{X}}$ indexing non-missing pixels. A solution is found via Algorithm 3. As termination criterion we use the relative increment (36) where we set $\eta = 10^{-7}$. With reference to Algorithm 2 we let $\tau = 10^4$ and $\lambda_m = 1$ for $m \in \{1,2,3\}$. Figure 5 reports the original pictures, the input data tensor and the outcome of our algorithm.

8.2 Inductive learning

We report here experiments on inductive learning with multiple tasks, see Sect. 7. As performance indicator we considered the capability of each procedure to predict the correct test labels. We compared LS-SVM models with linear kernel (lin ls-svm) and naive Bayes classifiers (naive Bayes) (Domingos and Pazzani 1997)[14] with models obtained solving (90). With reference to the latter we set the parameters in the composite spectral penalty as follows. In one case, referred to as log mlrank,[15] we set

$$\lambda_m = \frac{1}{M+1}\bar{\lambda} \quad \text{for any } m \in \mathbb{N}_{M+1} \tag{98}$$

where M is the order of the input data and $\bar{\lambda}$ is a varying parameter. Alternatively we take

$$\lambda_m = \begin{cases} 0 & \text{if } m \in \mathbb{N}_M \\ \bar{\lambda} & \text{if } m = M+1. \end{cases} \tag{99}$$

This latter approach, referred to as log rank, corresponds to leverage only the interdependence between tasks; structural assumptions over the input features are not exploited. In either case we use Algorithm 4 and compute the entire regularization path with respect to $\bar{\lambda}$. In our experiments the choice of this parameter was driven by the misclassification rate. For log mlrank and log rank we use a hold-out validation set. For LS-SVM we found models via the LS-SVMlab toolbox (De Brabanter et al. 2010) and considered ten-fold CV for model selection.

8.2.1 *Multi-labeled data toy problems*

We begin by a set of artificially generated problems. For each trial we considered an ensemble of T models represented by a $(D \times D \times T)$-tensor $\mathcal{W}$ with multilinear rank (r,r,r) and a

[14] Specifically we considered the routine NAIVEBAYES contained in the Statistics Toolbox of MATLAB.

[15] We use log as a short-hand for *logistic* since (90) is based on the logistic loss; we use mlrank as a short-hand for *multilinear rank* since in this case we penalize high multilinear rank of the tensor corresponding to the ensemble of models.

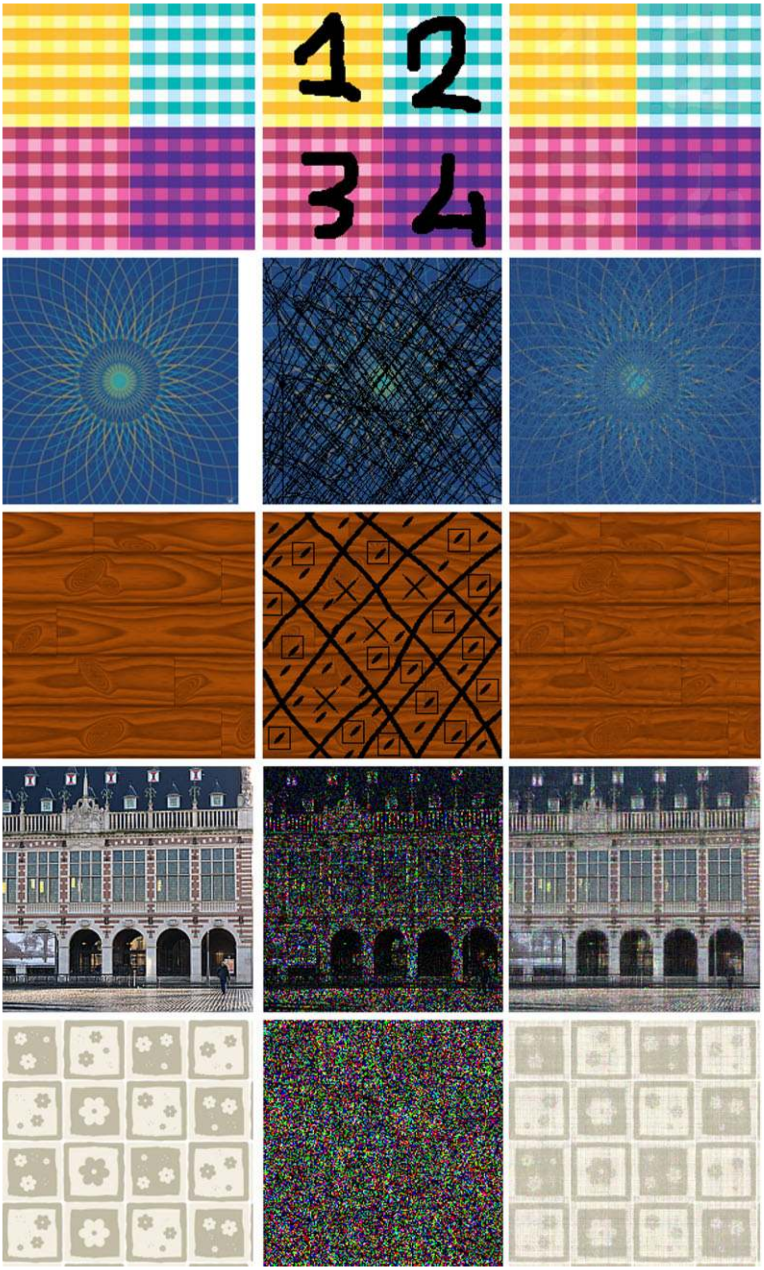

Fig. 5 Hard-completion for impainting of a 8-bit RGB image. Here we report the result of five impainting experiments. Each image is represented as a (300 × 300 × 3)-tensor; the first two modes span the pixels space; the third mode contains information from the three channels. Original images (*left column*); given pixels (*middle column*); reconstruction by Algorithm 3 (*right column*)

Table 6 Fractions of unobserved labels (test data) predicted incorrectly

D	T	N	log mlrank	log rank	lin ls-svm	naive Bayes
5	2	200	**0.08**(0.03)	0.11(0.03)	0.13(0.02)	0.23(0.02)
	4	400	**0.07**(0.02)	0.08(0.01)	0.13(0.01)	0.22(0.01)
	8	800	**0.06**(0.01)	0.07(0.01)	0.14(0.01)	0.22(0.01)
15	2	200	**0.22**(0.03)	0.31(0.02)	0.34(0.01)	0.39(0.01)
	4	400	**0.19**(0.05)	0.28(0.03)	0.35(0.01)	0.39(0.01)
	8	800	**0.17**(0.02)	0.22(0.01)	0.35(0.01)	0.39(0.01)

vector of bias terms $b \in \mathbb{R}^T$ with normal entries; $\mathcal{W}$ was obtained by generating a core tensor $\mathcal{S}_{\mathcal{W}}$ in $\mathbb{R}^{r\times r\times r}$ with entries i.i.d. from a normal distribution; for $i \in \{1,2\}$ a matrix $\boldsymbol{U}_i \in \mathbb{R}^{D\times r}$ and $\boldsymbol{U}_3 \in \mathbb{R}^{T\times r}$ were generated also with entries i.i.d. from a normal distribution. Finally we set

$$\mathcal{W} = \mathcal{S}_{\mathcal{W}} \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 \times \boldsymbol{U}_3. \tag{100}$$

We then created a dataset of N input-output pairs as follows. For $n \in \mathbb{N}_N$ we let $\mathcal{X}^{(n)}$ be a $(D \times D)$-matrix with normal entries; for any $t \in \mathbb{N}_T$ we let the corresponding label $y_t^{(n)}$ be a Bernoulli random variable with alphabet $\{1,-1\}$ and success probability $1/(1+\exp(-\tilde{y}_t^{(n)}))$; the latent variable $\tilde{y}_t^{(n)}$ was taken to be

$$\tilde{y}_t^{(n)} = \langle \mathcal{W}^{(t)}, \mathcal{X}^{(n)} \rangle + b_t$$

where for any $t \in \mathbb{N}_T$, $\mathcal{W}^{(t)}$ is as in (86).

We set $r = 2$ and considered the procedure above for different values of T and D. For a fixed value of T we use $N = 100T$ pairs for training. Note that N refer to the whole set of tasks; in turn, the N pairs were distributed uniformly at random across different tasks. As such, there are on average 100 input-output pairs per task; in this way, the amount of training information is kept constant as T varies. For each setting we perform 10 trials. For log mlrank and log rank we chose $\bar{\lambda}$ based upon a validation set of 30 % pairs selected at random within the N observations. Results in terms of misclassification rate on a test set are reported in Table 6.

Remark 15 The experiments show that leveraging relations between tasks (log mlrank and log rank) significantly improves results; the performance of ls-svm models, which are trained independently, is the same as T is increased. This is to be expected since the amount of training data per task is kept approximately the same across different values of T.

Remark 16 A comparison between log mlrank and log rank reveals that exploiting structural assumptions over the input features is a good idea; this is seen to be the case especially when the number of tasks is small or the features dimensionality is higher.

Table 7 Fractions of misclassified test digits, multiple problems, $T = 2$

(a) "2" vs "7"

log mlrank	log rank	lin ls-svm
0.05(0.04)	0.07(0.04)	0.06(0.04)

(b) "I" vs "J"

log mlrank	log rank	lin ls-svm
0.12(0.05)	0.13(0.05)	0.14(0.04)

(c) "4" vs "L"

log mlrank	log rank	lin ls-svm
0.10(0.05)	0.11(0.04)	**0.10**(0.04)

(d) "R" vs "S"

log mlrank	log rank	lin ls-svm
0.07(0.05)	0.08(0.05)	**0.07**(0.03)

(e) "8" vs "9"

log mlrank	log rank	lin ls-svm
0.06(0.05)	0.07(0.05)	0.07(0.05)

(f) "M" vs "N"

log mlrank	log rank	lin ls-svm
0.15(0.05)	0.16(0.06)	**0.14**(0.07)

8.2.2 Multiclass classification of Binary Alphadigits

In this experiment we considered discrimination between handwritten digits. We focused on the Binary Alphadigits dataset[16] made up of digits from "0" through "9" followed by capital letters from "A" through "Z" (English alphabet). Each digit is represented by 39 examples each of which consists of a binary 20 × 16 matrix. In log mlrank the matrix shape of each digit is retained; log rank and lin ls-svm treat each input pattern as a vector of dimension 320. We consider problems with different numbers of classes. As before we used one-vs-one encoding (see Sect. 8.1.5); correspondingly, the number of tasks T is equal to the number of classes. In each case we train models upon N training examples uniformly distributed across the considered classes; we chose N so that approximately 10 examples per class are used for training (and model selection) whereas the remaining examples are retained for testing. For each setting we average results over 20 trials each of which is obtained from a random splitting of training and test data. Due to the scarcity of training patterns an error occurs when running NAIVEBAYES; therefore we could not obtain results for this approach. Tables 7 and 8 report results for different values of T. For $T = 2$ we considered a subset of arbitrary binary problems; for $T > 2$ we considered classes of digits in their given order. In general, for $T \leq 4$ log mlrank seems to perform slightly better than log rank (Tables 7 and 8). As for the multi-labels example above, there is a strong evidence that enforcing task relationships via the regularization mechanism in log mlrank and log rank improves over the case where tasks are considered independently (Table 8).

9 Concluding remarks

In this paper we have established a mathematical framework for learning with higher order tensors. The transductive approach we considered is especially useful in the presence of missing input features. The inductive formulation, on the other hand, allows one to predict labels associated to input items unavailable during training. Both these approaches work

[16] Publicly available at http://cs.nyu.edu/~roweis/data.html.

Table 8 Fractions of misclassified test digits, $T > 2$

T	N	log mlrank	log rank	lin ls-svm
4	40	**0.07**(0.03)	0.08(0.03)	0.08(0.03)
8	80	**0.17**(0.04)	**0.17**(0.04)	0.22(0.04)
12	120	**0.24**(0.03)	**0.24**(0.03)	0.38(0.04)
16	160	**0.27**(0.02)	**0.27**(0.03)	0.51(0.02)

by simultaneously identifying subspaces of highly predictive features without the need for a preliminary feature extraction step. This is accomplished both leveraging relationships across tasks and within the higher order representation of the (possibly very high dimensional) input data.

A drawback of our methods is their restriction to linear models only. An interesting line of future research concerns the extension to a broader class of models. For certain problem of interest one could perhaps extend results for matrix representer theorems (Argyriou et al. 2009), used within multi-task learning (Argyriou et al. 2008). In the setting of Argyriou et al. (2008), different but related learning problems are associated to task vectors belonging to a feature space associated to a used-defined kernel function. As a special case, i.e. when the feature mapping is the identity, the feature space corresponds to the input space where data are originally represented. In this case one obtains linear models in the data, like in this paper. In general, one can show that these (possibly infinite dimensional) task vectors lie within the span of the mapped data associated to all the tasks (Argyriou et al. 2009). In the context of this paper, we assumed low multilinear ranks hereby leveraging the algebraic structure of data *in the input space*. This is rather crucial, especially when the higher order data entails missing observations and part of the learning problem consists of completing the data. In contrast, Argyriou et al. (2008) exploits the geometry of the feature space rather than that of the original input space. Therefore, although nonlinear extensions are certainly possible (and desirable) they will need working assumptions attached to the geometry of the feature space rather than that of the input space. For some cases, a viable alternative is to conceive a mapping that preserves properties of the higher order data in the input space. This is the spirit of the Grassmanian kernels proposed in Signoretto et al. (2011a).

It is also important to note that in our experiments we either considered a single spectral penalty (as in (97) and (99)) or a composite regularizer where all the penalties were equally enforced ((96) and (98)). Although uniform weights are shown to work in practice, this black and white setting is clearly restrictive: ideally one would perform model selection to search for the optimal combination of parameters. Unfortunately this comes at the price of increasing substantially the computation burden.

Acknowledgements Research supported by: ERC AdG A-DATADRIVE-B, Research Council KUL: GOA/10/09 MaNet, CoE EF/05/006 Optimization in Engineering (OPTEC) en PFV/10/002 (OPTEC), CIF1 and STRT1/08/023, IOF-SCORES4CHEM, several PhD/postdoc and fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects: G0226.06, G.0302.07, G.0320.08, G.0427.10N, G.0558.08, G.0557.08, G.0588.09; IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare. Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007–2011); IBBT; EU: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735).

Appendix A: Inexact Douglas-Rachford iteration

In this appendix we recall a result that we need in order to prove convergence of Algorithm 1. Consider an *inexact* variant of the Douglas-Rachford iteration defined as follows:

$$G_{IDR}\bigl(w^{(k)}; A, B, \gamma^{(k)}, \tau, a^{(k)}, b^{(k)}\bigr)\begin{cases} y^{(k)} = R_{\tau A}\bigl(w^{(k)}\bigr) + a^{(k)}, & (101a)\\ r^{(k)} = R_{\tau B}\bigl(2y^{(k)} - w^{(k)}\bigr), & (101b)\\ w^{(k+1)} = w^{(k)} + \gamma^{(k)}\bigl(r^{(k)} - y^{(k)}\bigr). & (101c)\end{cases}$$

Note that G_{IDR} differs from G_{DR}, introduced in (31a), (31b) and (31c), only in the first step (101a), where we allow to compute $y^{(k)}$ inexactly with an error $a^{(k)}$. The following lemma is a reduced version of Combettes (2009, Theorem 2.1) that deals with the more general case of infinite dimensional problems in a Hilbert space setting and also considers inexactness at step (101b).

Lemma 1 (Asymptotic behavior of the inexact Douglas-Rachford iteration) *Let $\mathcal{W}$ be some finite dimensional space endowed with the norm $\|\cdot\|_{\mathcal{W}}$; assume A and B are maximal monotone operators on $\mathcal{W}$ and that* $\mathrm{zer}(A+B) := \{x : 0 \in A(x) + B(x)\} \neq \emptyset$. *Consider the Douglas-Rachford algorithm with inexact iteration G_{IDR} introduced in* (101a), (101b) *and* (101c); *let $\tau \in]0, +\infty[$ and $\{\gamma^{(k)}\}_k$ be a sequence with elements in* $]0, 2[$. *Then if*

$$\sum_{k\in\mathbb{N}} \gamma^{(k)} \bigl\|a^{(k)}\bigr\|_{\mathcal{W}} < \infty \quad \textit{and} \quad \sum_{k\in\mathbb{N}} \gamma^{(k)}\bigl(2-\gamma^{(k)}\bigr) = \infty \qquad (102)$$

the sequence $\{y^{(k)}\}_k$ converges to an element of $\mathrm{zer}(A+B)$.

Appendix B: Proof of Theorem 1

In order to prove the theorem we rely on the error analysis reported in Appendix A. First, we show in the following lemma that if $\bar{f}$ is Lipschitz continuously differentiable then so is $\bar{q}$ introduced in (32).

Lemma 2 *Suppose that $\bar{f}$ is convex and differentiable and $\nabla\bar{f}$ is $L_{\bar{f}}$-Lipschitz. Then the function $\bar{q}$ defined in* (32) *is strongly convex with parameter $\frac{1}{\tau}$ and its gradient $\nabla\bar{q}$ is $L_{\bar{q}}$-Lipschitz continuous with a Lipschitz constant*

$$L_{\bar{q}} = L_{\bar{f}} + \frac{1}{\tau}. \qquad (103)$$

Proof Since $\bar{f}$ is convex, it is trivial that $\bar{q}$ is strongly convex with a parameter $\frac{1}{\tau} > 0$. Now, we prove the Lipschitz continuity of $\nabla\bar{q}$. We have $\nabla\bar{q}(w) = \nabla\bar{f}(w) + \frac{1}{\tau}(w - w^{(k)})$. Using the Lipschitz continuity of $\nabla\bar{f}$, we have

$$\begin{aligned}\bigl\|\nabla\bar{q}(w) - \nabla\bar{q}\bigl(w'\bigr)\bigr\|_{\mathcal{W}} &= \left\|\nabla\bar{f}(w) - \nabla\bar{f}\bigl(w'\bigr) + \frac{1}{\tau}\bigl(w - w'\bigr)\right\|_{\mathcal{W}}\\ &\le \bigl\|\nabla\bar{f}(w) - \nabla\bar{f}\bigl(w'\bigr)\bigr\|_{\mathcal{W}} + \frac{1}{\tau}\bigl\|w - w'\bigr\|_{\mathcal{W}}\end{aligned}$$

$$\leq \left(L_{\bar{f}} + \frac{1}{\tau}\right) \|w - w'\|_{\mathscr{W}},$$

which shows that $\nabla\bar{q}$ is Lipschitz continuous with a Lipschitz constant $L_{\bar{q}} := L_{\bar{f}} + \frac{1}{\tau}$. □

In order to prove Theorem 1 let us choose now in (101a), (101b) and (101c) a sequence of errors $\{a^{(k)}\}_k$ such that

$$\|a^{(k)}\|_{\mathscr{W}} \leq \frac{\epsilon_0}{(k+1)^\sigma}$$

for any $k \in \mathbb{N}$, where $\epsilon_0 > 0$ and $\sigma > 1$ are given. Additionally let $\gamma^{(k)} = 1$ for any $k \in \mathbb{N}$. By these choices the assumptions of Lemma 1 are both satisfied. In fact the second inequality in (102) is clearly verified. For the first we have:

$$\sum_{k\in\mathbb{N}} \gamma^{(k)} \|a^{(k)}\|_{\mathscr{W}} \leq \epsilon_0 \sum_{k\in\mathbb{N}} \frac{1}{(k+1)^\sigma} < \infty.$$

From (101a), we have

$$\|y^{(k)} - R_{\tau A}(w^{(k)})\|_{\mathscr{W}} = \|a^{(k)}\|_{\mathscr{W}} \leq \frac{\epsilon_0}{(k+1)^\sigma}. \tag{104}$$

Now note that computing the resolvent $R_{\tau A}$ is equivalent to solving the convex optimization problem in (32), namely:

$$\min_{w\in\bar{\mathscr{C}}} \bar{q}(w) := \bar{f}(w) + \frac{1}{2\tau}\|w - w^{(k)}\|^2_{\mathscr{W}}.$$

Since $\bar{q}$ in the latter is strongly convex and $\bar{\mathscr{C}}$ is nonempty, closed and convex, one can apply a projected gradient method (Nesterov 2003) to solve this problem up to accuracy $\frac{\epsilon_0}{(k+1)^\sigma}$. More precisely, let z in INEXACTPROXY correspond to the current estimate $w^{(k)}$ of the outer scheme MAIN and set $w^{(0)} = z$. We then generate a sequence $\{w^{(t)}\}_{t\geq 0}$ as

$$w^{(t+1)} = P_C\big(w^{(t)} - h_t \nabla\bar{q}\big(w^{(t)}\big)\big),$$

where $h_t \in (0, \frac{1}{L_{\bar{q}}}]$ is a given step-size and $L_{\bar{q}}$ is as in (103). Note that since $\nabla\bar{q}(w) = \nabla\bar{f}(w) + \frac{1}{\tau}(w - w^{(k)})$, we can write this iteration as

$$w^{(t+1)} = P_C\left(w^{(t)} - h_t\left(\nabla\bar{f}\big(w^{(t)}\big) + \frac{1}{\tau}\big(w^{(t)} - w^{(k)}\big)\right)\right). \tag{105}$$

Note that in Algorithm 1, we fix the step size h_t at $h_t = \frac{1}{L_{\bar{q}}}$. According to Theorem 2.2.8 in Nesterov (2003), we have

$$\|w^{(t+1)} - R_{\tau A}(w^{(k)})\|_{\mathscr{W}} \leq \left(\frac{\tau L_{\bar{f}}}{\tau L_{\bar{f}} + 1}\right)^{1/2} \|w^{(t)} - R_{\tau A}(w^{(k)})\|_{\mathscr{W}}, \quad \forall t \geq 0. \tag{106}$$

If we set $y^{(k)} = w^{(t+1)}$ then we need to find the iterate t such that $y^{(k)}$ satisfies (104). We have:

$$\|w^{(t+1)} - w^{(t)}\|_{\mathscr{W}} \overset{\text{by triang. inequality}}{\geq} \|w^{(t)} - R_{\tau A}(w^{(k)})\|_{\mathscr{W}} - \|w^{(t+1)} - R_{\tau A}(w^{(k)})\|_{\mathscr{W}} \tag{107}$$

$$\overset{\text{by (106)}}{\geq} \quad \left[\left(\frac{\tau L_{\bar{f}}+1}{\tau L_{\bar{f}}}\right)^{1/2} - 1\right]\left\| w^{(t+1)} - R_{\tau A}\big(w^{(k)}\big)\right\|_{\mathscr{W}} \tag{108}$$

$$\geq \quad \frac{1}{\kappa}\left\| y^{(k)} - R_{\tau A}\big(w^{(k)}\big)\right\|_{\mathscr{W}}, \tag{109}$$

where $\kappa := \sqrt{\tau L_{\bar{f}}(\tau L_{\bar{f}}+1)} + \tau L_{\bar{f}}$. Consequently, if $\|w^{(t+1)} - w^{(t)}\|_{\mathscr{W}} \leq \frac{\epsilon_0}{\kappa(k+1)^\sigma}$ then $\|y^{(k)} - R_{\tau A}(w^{(k)})\|_{\mathscr{W}} \leq \frac{\epsilon_0}{(k+1)^\sigma}$. We use the condition

$$\left\| w^{(t+1)} - w^{(t)}\right\|_{\mathscr{W}} \leq \frac{\epsilon_0}{\kappa(k+1)^\sigma}$$

to terminate the inner loop of Algorithm 1, which guarantees that $y^{(k)} = w^{(t+1)}$ approximates to $R_{\tau A}(w^{(k)})$ up to the accuracy $\frac{\epsilon_0}{(k+1)^\sigma}$. The claim of Theorem 1 now follows from application of Lemma 1.

Appendix C: Closed form of gradients

With reference to Eq. (69), $\nabla \bar{f}$ reads:

$$\nabla_{\tilde{\mathcal{X}}_{[m]}} \bar{f}(\tilde{\mathcal{X}}_{[1]}, \ldots, \tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}}, b) = \begin{cases} \frac{1}{M+1}(\tilde{x}_{[m]d_1^p\cdots d_M^p n^p} - z_p^x), & \text{if } (d_1^p, \ldots, d_M^p, n^p) \in \mathcal{S}_{\mathcal{X}} \\ 0, & \text{otherwise} \end{cases} \tag{110a}$$

$$\nabla_{\tilde{\boldsymbol{Y}}} \bar{f}(\tilde{\mathcal{X}}_{[1]}, \ldots, \tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}}, b) = \begin{cases} -\frac{\lambda_0 z_p^y}{\exp(z_p^y(\tilde{y}_{t^p n^p} + b_{t^p}))+1}, & \text{if } (t^p, n^p) \in \mathcal{S}_{\boldsymbol{Y}} \\ 0, & \text{otherwise} \end{cases} \tag{110b}$$

$$\nabla_b \bar{f}(\tilde{\mathcal{X}}_{[1]}, \ldots, \tilde{\mathcal{X}}_{[M+1]}, \tilde{\boldsymbol{Y}}, b) = -\sum_{p\,:\,(t^p, n^p)\in\mathcal{S}_{\boldsymbol{Y}}} \frac{z_p^y}{\exp(z_p^y(\tilde{y}_{t^p n^p} + b_{t^p}))+1} e_{t^p} \tag{110c}$$

where we denoted by e_{t^p} the t^p-th canonical basis vector of $\mathbb{R}^T$,

$$(e_{t^p})_i := \begin{cases} 1, & \text{if } i = t^p \\ 0, & \text{otherwise.} \end{cases}$$

With reference to equation (94), $\nabla \bar{f}$ reads:

$$\nabla_{\mathcal{W}_{[m]}^{(t)}} \bar{f}(\mathcal{W}_{[1]}, \ldots, \mathcal{W}_{[M+1]}, b) = -\frac{1}{M+1}\sum_{n\in\mathbb{N}_N} \frac{y_t^{(n)} \mathcal{X}^{(n)}}{1+\exp(y_t^{(n)}(\langle \mathcal{W}_{[m]}^t, \mathcal{X}^{(n)}\rangle + b_t))} \tag{110d}$$

$$\nabla_{b_t} \bar{f}(\mathcal{W}_{[1]}, \ldots, \mathcal{W}_{[M+1]}, b) = -\frac{1}{M+1}\sum_{n\in\mathbb{N}_N}\sum_{m\in\mathbb{N}_{M+1}} \frac{y_t^{(n)}}{1+\exp(y_t^{(n)}(\langle \mathcal{W}_{[m]}^t, \mathcal{X}^{(n)}\rangle + b_t))}. \tag{110e}$$

Appendix D: Proof of Proposition 4

Let us use $\mathscr{V}$ as a shorthand notation for $\mathbb{R}^{D_1\times D_2\times\cdots\times D_M\times N}$ and denote by $\ker(S_{\mathscr{S}_{\mathcal{X}}})$ the kernel of $S_{\mathscr{S}_{\mathcal{X}}}$, $\ker(S_{\mathscr{S}_{\mathcal{X}}}) := \{v \in \mathscr{V} : S_{\mathscr{S}_{\mathcal{X}}} v = 0\}$. Observe that its orthogonal complement is $\ker(S_{\mathscr{S}_{\mathcal{X}}})^{\perp} = \ker(S_{\mathscr{S}^c_{\mathcal{X}}}) = \{v \in \mathscr{V} : S_{\mathscr{S}^c_{\mathcal{X}}} w = 0\}$. Recall now the definition of projection onto a convex subset of an abstract inner product space given in (34). With the present specification of $\mathscr{C}$ and $\mathscr{W}$ the latter reads:

$$\begin{aligned}
&\arg \min_{(\mathcal{W}_{[1]},\mathcal{W}_{[2]},\ldots,\mathcal{W}_{[M+1]})\in\bar{\mathscr{C}}} \left\|(\mathcal{X}_{[1]},\mathcal{X}_{[2]},\ldots,\mathcal{X}_{[M+1]}) - (\mathcal{W}_{[1]},\mathcal{W}_{[2]},\ldots,\mathcal{W}_{[M+1]})\right\|^2_{\mathscr{W}} \\
&\overset{\text{(by definition of } \|\cdot\|_{\mathscr{W}})}{=} \arg \min_{(\mathcal{W}_{[1]},\mathcal{W}_{[2]},\ldots,\mathcal{W}_{[M+1]})\in\bar{\mathscr{C}}} \sum_{m\in\mathbb{N}_{M+1}} \|\mathcal{X}_{[m]} - \mathcal{W}_{[m]}\|^2 \\
&\overset{\text{(by definition of } \bar{\mathscr{C}})}{=} \times_{p\in\mathbb{N}_{M+1}} \left\{ \arg \min_{\mathcal{W}\in\{\mathcal{V}\in\mathscr{V}\,:\,S_{\mathscr{S}_{\mathcal{X}}}\mathcal{V}=z^x\}} \sum_{m\in\mathbb{N}_{M+1}} \|\mathcal{X}_{[m]} - \mathcal{W}\|^2 \right\} \\
&\overset{\text{(Remark 1)}}{=} \times_{p\in\mathbb{N}_{M+1}} \left\{ S^*_{\mathscr{S}_{\mathcal{X}}} z^x + \arg \min_{\mathcal{W}\in\ker(S_{\mathscr{S}_{\mathcal{X}}})} \sum_{m\in\mathbb{N}_{M+1}} \|\mathcal{X}_{[m]} - \mathcal{W}\|^2 \right\}. \qquad (111)
\end{aligned}$$

For the problem

$$\min_{\mathcal{W}\in\ker(S_{\mathscr{S}_{\mathcal{X}}})} \sum_{m\in\mathbb{N}_{M+1}} \|\mathcal{X}_{[m]} - \mathcal{W}\|^2 = \min_{\mathcal{W}\in\ker(S_{\mathscr{S}_{\mathcal{X}}})} \sum_{m\in\mathbb{N}_{M+1}} \left\|\mathcal{X}_{[m]}|_{\mathscr{S}^c_{\mathcal{X}}} - \mathcal{W}\right\|^2$$

first order optimality conditions shows that the unique solution can be written as

$$\hat{\mathcal{W}} = \left(\frac{1}{M+1} \sum_{m\in\mathbb{N}_{M+1}} \mathcal{X}_{[m]}\right)\Bigg|_{\mathscr{S}^c_{\mathcal{X}}},$$

which concludes the proof.

Appendix E: Proof of Proposition 5

We have:

$$\begin{aligned}
\langle \mathcal{X}, \mathcal{W}^{(t)} \rangle &= \langle \mathcal{X}_{\langle m_1\rangle}, \mathcal{W}^{(t)}_{\langle m_1\rangle} \rangle = \langle \mathcal{X}_{\langle m_1\rangle}, \boldsymbol{U}_{m_1}\boldsymbol{U}^{\top}_{m_1}\mathcal{W}^{(t)}_{\langle m_1\rangle} \rangle \\
&= \langle \boldsymbol{U}^{\top}_{m_1}(\tilde{\mathcal{X}}_{\langle m_1\rangle} + \mathcal{E}_{\langle m_1\rangle}), \boldsymbol{U}^{\top}_{m_1}\mathcal{W}^{(t)}_{\langle m_1\rangle} \rangle \overset{\text{by (83)}}{=} \langle \boldsymbol{U}^{\top}_{m_1}\tilde{\mathcal{X}}_{\langle m_1\rangle}, \boldsymbol{U}^{\top}_{m_1}\mathcal{W}^{(t)}_{\langle m_1\rangle} \rangle \\
&= \langle \tilde{\mathcal{X}} \times_{m_1} \boldsymbol{U}^{\top}_{m_1}, \mathcal{W}^{(t)} \times_{m_1} \boldsymbol{U}^{\top}_{m_1} \rangle \qquad (112)
\end{aligned}$$

in which we relied on the definition (4), and used the fact that the decomposition (85) implies that $\boldsymbol{U}_{m_1}\boldsymbol{U}^{\top}_{m_1}\mathcal{W}^{(t)}_{\langle m_1\rangle} = \mathcal{W}^{(t)}_{\langle m_1\rangle}$, where $\boldsymbol{U}_{m_1}\boldsymbol{U}^{\top}_{m_1}$ is a projection matrix. Now since by decomposition (85) we actually have $\boldsymbol{U}_{m}\boldsymbol{U}^{\top}_{m}\mathcal{W}^{(t)}_{\langle m\rangle} = \mathcal{W}^{(t)}_{\langle m\rangle}$ for any $m \in \mathbb{N}_M$, we have, for any[17]

[17] Note that m_2 is a generic index; contrary to m_1 it is unrelated to (83).

$m_2 \neq m_1$:

$$\begin{aligned}
&\big\langle \tilde{\mathcal{X}} \times_{m_1} \boldsymbol{U}_{m_1}^\top, \mathcal{W}^{(t)} \times_{m_1} \boldsymbol{U}_{m_1}^\top \big\rangle \\
&\quad = \big\langle \big(\tilde{\mathcal{X}} \times_{m_1} \boldsymbol{U}_{m_1}^\top\big)_{\langle m_2 \rangle}, \big(\mathcal{W}^{(t)} \times_{m_1} \boldsymbol{U}_{m_1}^\top\big)_{\langle m_2 \rangle} \big\rangle \\
&\quad = \big\langle \big(\tilde{\mathcal{X}} \times_{m_1} \boldsymbol{U}_{m_1}^\top\big)_{\langle m_2 \rangle}, \boldsymbol{U}_{m_2} \boldsymbol{U}_{m_2}^\top \big(\mathcal{W}^{(t)} \times_{m_1} \boldsymbol{U}_{m_1}^\top\big)_{\langle m_2 \rangle} \big\rangle \\
&\quad = \big\langle \boldsymbol{U}_{m_2}^\top \big(\tilde{\mathcal{X}} \times_{m_1} \boldsymbol{U}_{m_1}^\top\big)_{\langle m_2 \rangle}, \boldsymbol{U}_{m_2}^\top \big(\mathcal{W}^{(t)} \times_{m_1} \boldsymbol{U}_{m_1}^\top\big)_{\langle m_2 \rangle} \big\rangle \\
&\quad = \big\langle \tilde{\mathcal{X}} \times_{m_1} \boldsymbol{U}_{m_1}^\top \times_{m_2} \boldsymbol{U}_{m_2}^\top, \mathcal{W}^{(t)} \times_{m_1} \boldsymbol{U}_{m_1}^\top \times_{m_2} \boldsymbol{U}_{m_2}^\top \big\rangle. \qquad (113)
\end{aligned}$$

Iterating the same procedure for all $m \in \mathbb{N}_M$ we obtain:

$$\big\langle \mathcal{X}, \mathcal{W}^{(t)} \big\rangle = \big\langle \tilde{\mathcal{X}} \times_1 \boldsymbol{U}_1^\top \times_2 \cdots \times_M \boldsymbol{U}_M^\top, \mathcal{W}^{(t)} \times_1 \boldsymbol{U}_1^\top \times_2 \cdots \times_M \boldsymbol{U}_M^\top \big\rangle. \qquad (114)$$

Recall from De Lathauwer et al. (2000, Property 3) that $(\mathcal{A} \times_m \boldsymbol{F}) \times_m \boldsymbol{G} = \mathcal{A} \times_m (\boldsymbol{GF})$ where $\mathcal{A}$ is a generic tensor and $\boldsymbol{F}$, $\boldsymbol{G}$ are generic matrices with the appropriate sizes. Substituting the expression for $\tilde{\mathcal{X}}$ in (81) and the expression for $\mathcal{W}^{(t)}$ in (85) into (114) and using the aforementioned property we have:

$$\begin{aligned}
&\big\langle \tilde{\mathcal{X}} \times_1 \boldsymbol{U}_1^\top \times_2 \cdots \times_M \boldsymbol{U}_M^\top, \mathcal{W}^{(t)} \times_1 \boldsymbol{U}_1^\top \times_2 \cdots \times_M \boldsymbol{U}_M^\top \big\rangle \\
&\quad = \big\langle (\mathcal{S}_{\tilde{\mathcal{X}}} \times_2 \boldsymbol{U}_2 \times \cdots \times_M \boldsymbol{U}_M) \times_1 \big(\boldsymbol{U}_1^\top \boldsymbol{U}_1\big) \times_2 \cdots \times_M \boldsymbol{U}_M^\top, \\
&\qquad (\mathcal{S}_{\mathcal{W}^{(t)}} \times_2 \boldsymbol{U}_2 \times \cdots \times_M \boldsymbol{U}_M) \times_1 \big(\boldsymbol{U}_1^\top \boldsymbol{U}_1\big) \times_2 \cdots \times_M \boldsymbol{U}_M^\top \big\rangle \\
&\quad = \big\langle (\mathcal{S}_{\tilde{\mathcal{X}}} \times_2 \boldsymbol{U}_2 \times \cdots \times_M \boldsymbol{U}_M) \times_2 \boldsymbol{U}_2^\top \cdots \times_M \boldsymbol{U}_M^\top, \\
&\qquad (\mathcal{S}_{\mathcal{W}^{(t)}} \times_2 \boldsymbol{U}_2 \times \cdots \times_M \boldsymbol{U}_M) \times_2 \boldsymbol{U}_2^\top \cdots \times_M \boldsymbol{U}_M^\top \big\rangle. \qquad (115)
\end{aligned}$$

Iterating the same procedure for all the modes, i.e., for all $m \in \mathbb{N}_M$, we finally obtain:

$$\big\langle \mathcal{X}, \mathcal{W}^{(t)} \big\rangle = \langle \mathcal{S}_{\tilde{\mathcal{X}}}, \mathcal{S}_{\mathcal{W}^{(t)}} \rangle \qquad (116)$$

which implies that the right hand side of (82) and the right hand side of (84) coincide. This proves the statement.

References

Abernethy, J., Bach, F., Evgeniou, T., & Vert, J. (2009). A new approach to collaborative filtering: operator estimation with spectral regularization. *Journal of Machine Learning Research*, *10*, 803–826.

Acar, E., Dunlavy, D., Kolda, T., & Mørup, M. (2011). Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, *106*(1), 41–56.

Argyriou, A., Micchelli, C., Pontil, M., & Ying, Y. (2007a). A spectral regularization framework for multitask structure learning. In *Advances in neural information processing systems*.

Argyriou, A., Micchelli, C. A., Pontil, M., & Ying, Y. (2007b). A spectral regularization framework for multi-task structure learning. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), *Advances in neural information processing systems* (Vol. 20, pp. 25–32). Cambridge: MIT Press.

Argyriou, A., Evgeniou, T., & Pontil, M. (2007c). Multi-task feature learning. In B. Schölkopf, J. Platt, & T. Hoffman (Eds.), *Advances in neural information processing systems* (Vol. 19, pp. 41–48). Cambridge: MIT Press.

Argyriou, A., Evgeniou, T., & Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, *73*(3), 243–272.

Argyriou, A., Micchelli, C., & Pontil, M. (2009). When is there a representer theorem? Vector versus matrix regularizers. *Journal of Machine Learning Research*, *10*, 2507–2529.

Argyriou, A., Micchelli, C., & Pontil, M. (2010). On spectral learning. *Journal of Machine Learning Research*, *11*, 935–953.

Argyriou, A., Micchelli, C., Pontil, M., Shen, L., & Xu, Y. (2011). Efficient first order methods for linear composite regularizers. Arxiv preprint arXiv:1104.1436.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, *68*, 337–404.

Bauschke, H., & Combettes, P. (2011). *Convex analysis and monotone operator theory in Hilbert spaces*. Berlin: Springer.

Beck, A., & Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, *2*(1), 183–202.

Becker, S., Candès, E. J., & Grant, M. (2010). *Templates for convex cone problems with applications to sparse signal recovery*. Tech. rep, Stanford University.

Berlinet, A., & Thomas-Agnan, C. (2004). *Reproducing kernel Hilbert spaces in probability and statistics*. Amsterdam: Kluwer Academic.

Bertsekas, D. (1976). On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control*, *21*(2), 174–184.

Bertsekas, D. P. (1995). *Nonlinear programming*. Belmont: Athena Scientific.

Bertsekas, D. P., & Tsitsiklis, J. N. (1989). *Parallel and distributed computation*. Englewood Cliffs: Prentice-Hall.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, *3*(1), 1–122.

Brézis, H. (1973). *Opérateurs maximaux monotones*. Amsterdam: Elsevier.

Cai, J., Candès, E., & Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, *20*(4), 1956–1982.

Candes, E., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis? *Journal of the ACM*, *58*(3), 11, p. 37.

Candès, E., & Plan, Y. (2010). Matrix completion with noise. *Proceedings of the IEEE*, *98*(6), 925–936.

Candès, E., & Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, *9*(6), 717–772.

Chen, S., Donoho, D., & Saunders, M. (2001). Atomic decomposition by basis pursuit. *SIAM Review*, *43*, 129–159.

Combettes, P. (2009). Iterative construction of the resolvent of a sum of maximal monotone operators. *Journal of Convex Analysis*, *16*(4), 727–748.

Combettes, P., & Pesquet, J. (2008). A proximal decomposition method for solving convex variational inverse problems. *Inverse Problems*, *24*, 065014.

Combettes, P., & Pesquet, J. (2009). *Proximal splitting methods in signal processing*. Arxiv preprint arXiv:0912.3522.

Coppi, R., & Bolasco, S. (1989). *Multiway data analysis*. Amsterdam: North-Holland

De Brabanter, K., Karsmakers, P., Ojeda, F., Alzate, C., De Brabanter, J., Pelckmans, K., De Moor, B., Vandewalle, J., & Suykens, J. A. K. (2010). *LS-SVMlab toolbox user's guide version 1.8*. Internal Report 10-146, ESAT-SISTA, K.U.Leuven (Leuven, Belgium).

De Lathauwer, L., De Moor, B., & Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, *21*(4), 1253–1278.

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, *29*(2), 103–130.

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, *52*(4), 1289–1306.

Douglas, J., & Rachford, H. H. (1956). On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American Mathematical Society*, *82*(2), 421–439.

Eckstein, J., & Bertsekas, D. (1992). On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, *55*(1), 293–318.

Ekeland, I., & Temam, R. (1976). *Convex analysis and variational problems*. Amsterdam: North-Holland

Fazel, M. (2002). *Matrix rank minimization with applications*. Ph.D. thesis, Elec. Eng. Dept., Stanford University.

Gandy, S., Recht, B., & Yamada, I. (2011). Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, *27*(2), 025010.

Geng, X., Smith-Miles, K., Zhou, Z., & Wang, L. (2011). Face image modeling by multilinear subspace analysis with missing values. *IEEE Transactions on Systems, Man and Cybernetics. Part B. Cybernetics*, *41*(3), 881–892.

Goldberg, A., Xiaojin, Z., Recht, B., Xu, J., & Nowak, R. (2010). Transduction with matrix completion: three birds with one stone. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems* (Vol. 23, pp. 757–765).

Golub, G., & Van Loan, C. (1980). An analysis of the total least squares problem. *SIAM Journal on Numerical Analysis*, *17*(6), 883–893.

Golub, G. H., & Van Loan, C. F. (1996). *Matrix Computations* (3rd ed.). Baltimore: Johns Hopkins University Press.

Halmos, P. (1982). *A Hilbert space problem book* (Vol. *19*). Berlin: Springer.

Hastad, J. (1990). Tensor rank is NP-complete. *Journal of Algorithms*, *11*(4), 644–654.

Hillar, C., & Lim, L. (2010). *Most tensor problems are NP hard*. Arxiv preprint arXiv:0911.1393.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67.

Horn, R., & Johnson, C. (1994). *Topics in matrix analysis*. Cambridge: Cambridge University Press.

Jacob, L., Obozinski, G., & Vert, J. (2009). Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*. New York: ACM.

Kolda, T., & Bader, B. (2009). Tensor decompositions and applications. *SIAM Review*, *51*(3), 455–500.

Koltchinskii, V., Tsybakov, A., & Lounici, K. (2010). *Nuclear norm penalization and optimal rates for noisy low rank matrix completion*. Arxiv preprint arXiv:1011.6256.

Kroonenberg, P. (2008). *Applied multiway data analysis*. New York: Wiley-Interscience.

Lions, P., & Mercier, B. (1979). Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, *16*(6), 964–979.

Liu, J., Musialski, P., Wonka, P., & Ye, J. (2009). Tensor completion for estimating missing values in visual data. In *IEEE international conference on computer vision (ICCV)*, Kyoto, Japan (pp. 2114–2121).

Ma, S., Goldfarb, D., & Chen, L. (2011). Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming*, *128*(1), 321–353.

Minty, G. (1962). Monotone (nonlinear) operators in Hilbert space. *Duke Mathematical Journal*, *29*(3), 341–346.

Moreau, J. (1962). Fonctions convexes duales et points proximaux dans un espace hilbertien. *Comptes Rendus Mathematique. Academie Des Sciences Paris, Sér. A Math*, *255*, 2897–2899.

Nesterov, Y. (2003). *Introductory lectures on convex optimization: a basic course*. Norwell: Kluwer Academic.

Nesterov, Y. (2007). *Gradient methods for minimizing composite objective function*. Center for Operations Research and Econometrics (CORE), Université catholique de Louvain, Tech. Rep.

von Neumann, J. (1937). Some matrix inequalities and metrization of matric-space. *Tomsk University Review*, *1*, 286–300.

Pelckmans, K., De Brabanter, J., Suykens, J. A. K., & De Moor, B. (2005). Handling missing values in support vector machine classifiers. *Neural Networks*, *18*, 684–692.

Phelps, R. (1993). *Convex functions, monotone operators, and differentiability*. Berlin: Springer.

Recht, B., Fazel, M., & Parrilo, P. (2007). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, *52*, 471–501.

Rockafellar, R. (1970a). *Convex analysis*. Princeton: Princeton University Press.

Rockafellar, R. (1970b). On the maximal monotonicity of subdifferential mappings. *Pacific Journal of Mathematics*, *33*(1), 209–216.

Rockafellar, R. (1976). Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, *14*, 877.

Signoretto, M., De Lathauwer, L., & Suykens, J. A. K. (2011a). A kernel-based framework to tensorial data analysis. *Neural Networks*, *24*(8), 861–874.

Signoretto, M., Van de Plas, R., De Moor, B., & Suykens, J. A. K. (2011b). Tensor versus matrix completion: a comparison with application to spectral data. *IEEE Signal Processing Letters*, *18*(7), 403–406.

Smale, S., & Zhou, D. (2005). Shannon sampling II: connections to learning theory. *Applied and Computational Harmonic Analysis*, *19*(3), 285–302.

Smilde, A., Bro, R., & Geladi, P. (2004). *Multi-way analysis with applications in the chemical sciences*. New York: Wiley.

Spingarn, J. (1983). Partial inverse of a Monotone Operator. *Applied Mathematics & Optimization*, *10*(1), 247–265.

Srebro, N. (2004). *Learning with matrix factorizations*. Ph.D. thesis, Massachusetts Institute of Technology.

Sturm, J. (1999). Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods & Software*, *11*(1), 625–653.

Suykens, J. A. K., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, *9*(3), 293–300.

Tibshirani, R. (1996). Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society. Series B (Methodological)*, *58*(1), 267–288.

Tomioka, R., & Aihara, K. (2007). Classifying matrices with a spectral regularization. In *Proceedings of the 24th international conference on machine learning* (pp. 895–902). New York: ACM.

Tomioka, R., Hayashi, K., & Kashima, H. (2011). *Estimation of low-rank tensors via convex optimization*. Arxiv preprint arXiv:1010.0789.

Tucker, L. R. (1964). The extension of factor analysis to three-dimensional matrices. In *Contributions to mathematical psychology* (pp. 109–127). New York: Holt, Rinehart & Winston.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, *31*(3), 279–311.

Tütüncü, R., Toh, K., & Todd, M. (2003). Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming*, *95*(2), 189–217.

Van Huffel, S., & Vandewalle, J. (1991). *The total least squares problem: computational aspects and analysis* (Vol. *9*). Philadelphia: Society for Industrial Mathematics.

Vandenberghe, L., & Boyd, S. (1996). Semidefinite programming. *SIAM Review*, *38*(1), 49–95.

Zhao, P., Rocha, G., & Yu, B. (2009). The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, *37*, 3468–3497.