Noname manuscript No.
(will be inserted by the editor)

An Alternating Direction and Projection Algorithm for Structure-enforced Matrix Factorization

Lijun Xu · Bo Yu · Yin Zhang

Received: date / Accepted: date

Abstract Structure-enforced matrix factorization (SeMF) represents a large class of mathematical models appearing in various forms of principal component analysis, sparse coding, dictionary learning and other machine learning techniques useful in many applications including neuroscience and signal processing. In this paper, we present a unified algorithm framework, based on the classic alternating direction method of multipliers (ADMM), for solving a wide range of SeMF problems whose constraint sets permit low-complexity projections. We propose a strategy to adaptively adjust the penalty parameters which is the key to achieving good performance for ADMM. We conduct extensive numerical experiments to compare the proposed algorithm with a number of state-of-the-art special-purpose algorithms on test problems including dictionary learning for sparse representation and sparse nonnegative matrix factorization. Results show that our unified SeMF algorithm can solve different types of factorization problems as reliably and as efficiently as special-purpose algorithms. In particular, our SeMF algorithm provides the ability to explicitly enforce various combinatorial sparsity patterns that, to our knowledge, has not been considered in existing approaches.

Keywords Matrix Factorization · Alternating Direction Method · Projection · Adaptive Penalty Parameter · Sparse Optimization · Dictionary Learning

L. Xu
Dalian University of Technology, Dalian, Liaoning 116024, P. R. China. Tel.: +86 13204275065
E-mail: xulijundlut@gmail.com

B. Yu
Dalian University of Technology, Dalian, Liaoning 116024, P. R. China. E-mail: yubo@dlut.edu.cn

Y. Zhang
Rice University, Houston, TX 77005, U.S.A. E-mail: yzhang@rice.edu. Research supported in part by NSF DMS-1115950 and NSF DMS-1418724.

1 Introduction

1.1 A matrix factorization model

Matrix factorization has a long history as a fundamental mathematical tool in matrix analysis. Traditionally, the term *matrix factorization* is used in an exact sense in reference to decomposing a matrix into an equivalent product form. Well-known examples of such exact factorization include, but not limited to, LU, QR, SVD and Cholesky factorizations. More recently, the term matrix factorization has also been widely used in an inexact sense in reference to approximating a matrix by a product of two factors where certain structures are either encouraged or imposed on one or both of the factors, reflecting prior knowledge about the desired factorization. Such approximate and structured matrix factorizations have found great utility in various data-related applications, such as in signal and image processing and in machine learning tasks, primarily because they often help reveal latent features in a dataset.

To have a precise mathematical description, we let $M \in \mathbb{R}^{m\times n}$ be a given data matrix. For example, M may represent a sequence of n images each having m pixels. We now aim at approximating M by a product of two factors, i.e., $M \approx XY$, where some prior knowledge on $X \in \mathbb{R}^{m\times p}$ or $Y \in \mathbb{R}^{p\times n}$, or both, are available. To perform an approximate and structured matrix factorization, we consider the following general optimization model

$$\min_{X,Y} \ \|M - XY\|_F^2 \ \text{ s.t. } \ X \in \mathcal{X},\ Y \in \mathcal{Y}. \tag{1}$$

where $\|\cdot\|_F$ is Frobenius norm, and $\mathcal{X}$ and $\mathcal{Y}$ are subsets of $\mathbb{R}^{m\times p}$ and $\mathbb{R}^{p\times n}$, respectively. In model (1), the objective function measures data fidelity in a least-square sense, which is the most popular metric for data fidelity although other measures are frequently used as well. In this model, prior knowledge are explicitly enforced as two constraint sets $\mathcal{X}$ and $\mathcal{Y}$ whose members possess desirable matrix structures. The most useful structures of this kind include, for example, nonnegativity and various sparsity patterns.

We note that in the literature unconstrained optimization models are widely used where prior knowledge are handled through penalty or regularization functions added to the data fidelity term so that a weighted sum of the two is to be simultaneously minimized. In unconstrained optimization models, it is generally the case that desired structures are encouraged or promoted, but not exactly enforced as in a constrained optimization model like (1). Obviously, both types of formulations have their distinct advantages and disadvantages under different circumstances.

The focus of this work is exclusively on studying a specific algorithmic approach to solving model (1). This approach is applicable to a range of constraint sets $\mathcal{X}$ and $\mathcal{Y}$ that are *easily projectable* (more details will follow later). Fortunately, as we will demonstrate, in many practically relevant applications constraint sets $\mathcal{X}$ and $\mathcal{Y}$ are indeed *easily projectable*.

We will call model (1) a *structure-enforced matrix factorization* (or SeMF) model, partly to distinguish it from models where structures are only encouraged or promoted like in unconstrained optimization models. Generally speaking, model (1) is a nonconvex optimization problem that could allow a large number of non-global local

minima. In this case, it is well understood that a theoretical guarantee for a global minimum is extremely difficult or impossible to obtain. In this work, our evaluation of algorithm performance is solely based on empirical evidence, i.e., computational results obtained from controlled numerical experiments.

1.2 Related Works

Recently, numerous inexact and structured matrix factorization problems have arisen from various applications, including Matrix Completion, Principal component analysis (PCA), Sparse PCA, nonnegative matrix factorization (NMF), Dictionary Learning, to name a few. Many of these factorizations can be represented by the SeMF model (1) with different structure constraints.

Non-negative Matrix Factorization (NMF) [1] is a common matrix decomposition method for finding meaningful representations of nonnegative data. NMF has been proven useful in dimension reduction of images, text data and signals, for example. The most popular data fidelity function in NMF is Frobenius norm squared, while another widely used function is the Kullback-Leibler divergence. There are many algorithms developed for NMF in the past decades such as multiplicative updates [2–4], alternating least squares [1, 5], and projected gradient type [6]. The work of Lee and Seung [7] demonstrates that NMF models tend to return part-based sparse representations of data, which has popularized the use of and research on NMF-related techniques. In particular, various NMF-inspired formulations add different regularization or penalty terms to promote desired properties, such as sparsity patterns in Y or orthogonality between columns of X, in addition to nonnegativity (see [8–14], for example).

Approximate and structured matrix factorization problems also arise from dictionary learning, a data processing technique widely used in many applications including signal processing, compressive sensing and machine learning. Dictionary learning is to decompose a sampled dataset into a product of two factors, say X and Y where X is called a dictionary and Y a representation of the data (or a coding matrix) under the dictionary. As usual, some desired properties, such as nonnegativity and sparsity, can be imposed on either or both factors. Again, most algorithms in dictionary learning are developed based on minimizing data fidelity functions either with penalty/regularization terms or with explicit constraints (occasionlly with both). For instance, the popular algorithm K-SVD [15], which is widely used to learn dictionaries for sparse data representation, is built on minimizing the Frobenius-norm data fidelity with explicit sparsity constraints on each column of the coding matrix Y. For very large training sets and dynamic (time-dependent) training data, a number of approximate matrix factorization models and so-called online algorithms have been proposed that also either encourage or enforce sparsity structures by various means (see [12, 16–18], for example).

To this date, there is already a vast literature on applications of approximate and structured matrix factorization models to various areas along with special-purpose algorithms developed for solving those models. Many of those models can be formulated as instances of the SeMF model (1) if one uses the squared Frobenius norm

fidelity measure and imposes structures as constraints. In practice, many structures are simple enough to allow "easy projections" that include, but are not limited to, nonnegativity, normality and various sparsity patterns.

1.3 Main Contributions

The structure-enforced matrix factorization (SeMF) model studied in this paper is a general and unifying mathematical model encompassing numerous problem classes arising from diverse application backgrounds. The current state of affair is generally such that special-purpose algorithms are developed for solving individual problem classes. This work is motivated in part by the recent success of the classic alternating direction method of multipliers (ADMM or ADM) [19, 20] applied to compressive sensing (see [21, 22], for example). Based on an extension of the classic ADMM (see more introduction in the next section), we devise a unified algorithm for solving the general SeMF model that can be effectively implemented as long as desired structure sets allow low-complexity projections (or approximate projections). Fortunately, most commonly desired structures, including sparsity and nonnegativity, do possess this *property of easy projection*. Applicable to a wide range of problem classes, a major advantage of the proposed algorithm is its extraordinary versatility unparalleled by most special-purpose algorithms. In addition, as will be discussed later, our algorithm is capable of handling certain sparsity patterns of combinatorial nature that have not been treated, as far as we know, by existing algorithms.

The classic ADMM methodology has theoretical guarantees of convergence for convex programs of two separable variables (see [21, 23–25], for example). However, the general SeMF model (1) is highly nonconvex and non-separable for which even a moderately consistent practical performance is hard to obtain without careful modifications to the classic framework, let alone any theoretical guarantee of performance. In this work, we develop a simple dynamic scheme to adaptively adjust the two most critical parameters in the algorithm. This dynamic scheme enables the resulting algorithm to work quite well in our extensive numerical experiments, in terms of both reliability and efficiency. Numerical results on several problem classes indicate that, on most tested problems, our unified algorithm compares favorably with state-of-the-art, special-purpose algorithms designed for individual classes. We believe that the new algorithm adds a versatile and useful technique to the toolbox of solving structured matrix factorization problems arising from a wide array of applications.

1.4 Organization

This paper is organized as follows. In Section 2, we propose to solve the Structured-enforced Matrix Factorization (SeMF) problem (1) by extending the classic alternating direction method of multipliers (ADMM) for convex optimization to the SeMF case with a strategy for adaptively updating penalty parameters which is critical for the reliability and efficiency of the algorithm. In Section 3, we discuss the issue of projections onto several popular structure sets that need to be performed in the algorithm for solving relevant problems. Section 4 contains several sets of computational

results comparing the proposed algorithm with several state-of-art, special-purpose algorithms. Finally, we conclude this paper in Section 5.

2 Alternating Direction Algorithm for SeMF

2.1 Classic ADMM Method

In a finite-dimensional setting, the classic alternating direction method of multipliers (ADMM or simply ADM) is designed for solving separable convex programs of the form

$$\min_{x\in\mathcal{X},y\in\mathcal{Y}} f(x)+g(y) \ \text{ s.t. } Ax+By=c, \tag{2}$$

where f and g are convex functions defined on closed convex subsets $\mathcal{X}$ and $\mathcal{Y}$ of finite-dimensional spaces, respectively, and A, B and c are matrices and vector of appropriate sizes. The augmented Lagrangian function of (2) is

$$\mathcal{L}_A(x,y,\lambda)=f(x)+g(y)+\lambda^T(Ax+By-c)+\frac{\beta}{2}\|Ax+By-c\|_2^2, \tag{3}$$

where λ represents a Lagrangian multiplier vector and $\beta>0$ is a penalty parameter.

ADMM method [19, 20] is an extension of the classic augmented Lagrangian multiplier method [26–28]. It performs one sweep of alternating minimization with respect to x and y individually, then updates the multiplier λ; that is, at the iteration k an ADMM scheme executes the following three steps: given (x^k, y^k, λ^k),

$$x^{k+1} \leftarrow \operatorname*{argmin}_{x\in\mathcal{X}} \mathcal{L}_A(x,y^k,\lambda^k), \tag{4a}$$

$$y^{k+1} \leftarrow \operatorname*{argmin}_{y\in\mathcal{Y}} \mathcal{L}_A(x^{k+1},y,\lambda^k), \tag{4b}$$

$$\lambda^{k+1} \leftarrow \lambda^k+\gamma\beta(Ax^{k+1}+By^{k+1}-c), \tag{4c}$$

where $\gamma\in(0,1.618)$ is a step length. It is worth noting that (4a) only involves $f(x)$ in the objective and (4b) only $g(y)$, whereas the classic augmented Lagrangian multiplier method requires a joint minimization with respect to both x and y, that is, substituting steps (4a) and (4b) by

$$(x^{k+1},y^{k+1}) \leftarrow \operatorname*{argmin}_{x\in\mathcal{X},y\in\mathcal{Y}} \mathcal{L}_A(x,y,\lambda^k),$$

which involves both $f(x)$ and $g(y)$ and is usally more difficult to solve. A convergence proof for the above ADMM algorithm can be found in [29].

2.2 Extension to SeMF

To facilitate an efficient use of alternating minimization, we introduce two auxiliary variables U and V and consider the following model equivalent to (1),

$$\min_{X,Y,U,V} \frac{1}{2}\|M-XY\|_F^2 \ \text{ s.t. } \ X-U=0,\ Y-V=0,\ U\in\mathcal{X},\ V\in\mathcal{Y}, \tag{5}$$

where $U \in \mathbb{R}^{m\times p}$ and $V \in \mathbb{R}^{p\times n}$. The augmented Lagrangian function of (5) is

$$\begin{aligned}\mathcal{L}_A(X,Y,U,V,\Lambda,\Pi) = \tfrac{1}{2}\|M-XY\|_F^2 + \\ \Lambda\bullet(X-U)+\Pi\bullet(Y-V)+\tfrac{\alpha}{2}\|X-U\|_F^2+\tfrac{\beta}{2}\|Y-V\|_F^2,\end{aligned} \tag{6}$$

where $\Lambda \in \mathbb{R}^{m\times p}$, $\Pi \in \mathbb{R}^{p\times n}$ are Lagrangian multipliers and $\alpha, \beta > 0$ are penalty parameters for the constraints $X - U = 0$ and $Y - V = 0$, respectively, and the scalar product "$\bullet$" of two equal-size matrices A and B is the sum of all element-wise products, i.e., $A \bullet B = \sum_{i,j} a_{ij}b_{ij}$.

The alternating direction method of multipliers (ADMM) for (5) is derived by successively minimizing the augmented Lagrangian function $\mathcal{L}_A$ with respect to X, Y and (U, V), one at a time while fixing others at their most recent values, and then updating the multipliers after each sweep of such alternating minimization. The introduction of the two auxiliary variables U and V makes it easy to carry out each of the alternating minimization steps. Specifically, these steps can be written in the following form,

$$X_+ \approx \operatorname*{argmin}_X \mathcal{L}_A(X,Y,U,V,\Lambda,\Pi), \tag{7a}$$
$$Y_+ \approx \operatorname*{argmin}_Y \mathcal{L}_A(X_+,Y,U,V,\Lambda,\Pi), \tag{7b}$$
$$U_+ = \mathcal{P}_{\mathcal{X}}(X_+ + \Lambda/\alpha), \tag{7c}$$
$$V_+ = \mathcal{P}_{\mathcal{Y}}(Y_+ + \Pi/\beta), \tag{7d}$$
$$\Lambda_+ = \Lambda + \alpha(X_+ - U_+), \tag{7e}$$
$$\Pi_+ = \Pi + \beta(Y_+ - V_+), \tag{7f}$$

where $\mathcal{P}_{\mathcal{X}}$ ($\mathcal{P}_{\mathcal{Y}}$) stands for the projection onto the set $\mathcal{X}$ ($\mathcal{Y}$) in Frobenius norm, and the subscript "+" is used to denote iterative values at the new iteration. Actually, we can write (7a) and (7b) exactly in closed forms,

$$X_+ = \left(MY^T + \alpha U - \Lambda\right)\left(YY^T + \alpha I\right)^{-1}, \tag{8a}$$
$$Y_+ = \left(X_+^TX_+ + \beta I\right)^{-1}\left(X_+^TM + \beta V - \Pi\right). \tag{8b}$$

Since the involved inverse matrices are both $p \times p$, the corresponding linear systems are relatively inexpensive for small p, especially for $p \ll \max(m, n)$. In this case, the dominant computational tasks at each iteration are the matrix multiplications MY^T and X^TM, together requiring about $4kmn$ arithmetic operations (scalar additions and multiplications). On the other hand, when p is relatively large, instead of using the inversions in (8a) and (8b), it will be more efficient to employ suitable iterative procedures, such as the conjugate gradient method, to approximately solve the two convex quadratic minimization problems in (7a) and (7b).

Based on the formulas in (7), we can implement the following ADMM algorithmic framework so long as we can compute the projections in steps (7c) and (7d). An update scheme for α and β, stated as Algorithm 2, will be described in the next subsection.

Algorithm 1: ADMM Framework for SeMF

Input: $M \in \mathbb{R}^{m\times n}$, integers $p, maxiter > 0$ and $tol > 0$
Output: $X \in \mathbb{R}^{m\times p}$ and $Y \in \mathbb{R}^{p\times n}$
Set $\alpha, \beta > 0$.
Set U, V, Λ, Π to zero matrices of appropriate sizes, and Y to a random matrix.
for $k = 1$, *maxiter* **do**
  Update $(X, Y, U, V, \Lambda, \Pi)$ by the formulas in (7).
  if *stopping criterion (9) is met* **then**
    output X, Y, and exit.
  end
  Update penalty parameters α and β by Algorithm 2.
end

We use the following practical stopping criterion: for given tolerance tol,

$$\min\left\{\frac{|f_k - f_{k+1}|}{|f_k|}, \max\left(\frac{\|X_k - X_{k+1}\|_F}{\|X_k\|_F}, \frac{\|Y_k - Y_{k+1}\|_F}{\|Y_k\|_F}\right)\right\} \le tol \tag{9}$$

where $f_k = \|M - X_k Y_k\|_F$ and X_k is the k-th iterate for the variable X, and so on. For the sake of robustness, in our implementation we require that the above condition be satisfied at three consecutive iterations. In other words, we stop the algorithm either when data fidelity does not change meaningfully in three consecutive iterations or both variables X and Y do not change meaningfully in three consecutive iterations.

We note that Algorithm 1, with fixed penalty parameter values, has been studied in [30] for a special case of the SeMF model – the nonnegative matrix factorization (NMF) problem where the structure sets $\mathcal{X}$ and $\mathcal{Y}$ contain element-wise nonnegative matrices of appropriate sizes. The current work provides several meaningful and nontrivial extensions beyond the work in [30].

2.3 Adaptive Penalty Parameter Update

It is well known that the penalty parameters α and β are the most important algorithmic parameters in the ADMM framework. Even in the classic case of separable convex programming problem (2) where f and g are convex functions and $\mathcal{X}$ and $\mathcal{Y}$ are convex sets, the value of the penalty parameter β can still greatly affect the speed of convergence in practice, even though global convergence is guaranteed for any $\beta > 0$ in theory. Recently, He *et al.* [31] proposed some self-adaptive rules for adjusting the penalty parameter in ADMM method for monotone variational inequalities, which attempted to balance the errors in different parts of optimality conditions. Wen *et al.* [32] presented an alternating direction dual augmented Lagrangian method for solving semidefinite programming (SDP) where tuning the penalty parameter by balancing the primal and dual infeasibilities. Moreover, Lin *et al.* [33] presented an update rule for penalty parameter in linearized alternating direction method (LADM), which is aimed at accelerating convergence by increasing a penalty parameter whenever the associated error has not been improved sufficiently over a number of iterations with the current parameter value. In these works, efficiency and convergence properties of these update rules have been proved under suitable assumptions among

which convexity is the fundamental one. In addition, these papers only study the case where there is only a single penalty parameter. This sensitivity to penalty parameter values, not surprisingly, only becomes much more severe in our extended ADMM framework where the objective function is neither convex nor separable, and the constraint sets are mostly nonconvex as well. In addition, there are three sets of variables, X, Y and (U, V) that are minimized sequentially, as opposed to two sets in the classic case. Indeed, experiments indicate that without getting both α and β in some proper ranges, the algorithm could hardly find any good solution close to a global minimum, either going to a bad local minimum or becoming excessively slow or even stagnate. In general, it is difficult to properly choose fixed values for the penalty parameters α and β for each class of problems due to widely varying characteristics of problem instances. Therefore, we consider developing an adaptive scheme to update the penalty parameters during iterations. Short of solid theoretical guidance, we have developed a set of heuristic rules and validated them by extensive numerical experiments.

Firstly, we note that both U and V are feasible since $U \in \mathcal{X}$ and $V \in \mathcal{Y}$ always hold in our algorithm after the steps (7c) and (7d). If $\|M - UV\|_F$ is small, then $XY = UV$ should represent a desired structured factorization since $M \approx XY$. Now, if the quantity $\|M - UV\|_F$ has been sufficiently decreased after every iteration (or after every 3 iterations, say, for that matter), then we consider that the current values for both α and β are appropriate and will leave them unchanged. Specifically, we skip updating whenever

$$\|M - U_+V_+\|_F < (1 - \varepsilon)\|M - UV\|_F \tag{10}$$

where $\varepsilon \in (0, 1)$ is a small tolerance value. If the above test fails, then we update (α, β) according to three different scenarios.

Case 1. Near feasibility of X and Y:

$$\left| \frac{\|M - UV\|_F}{\|M - XY\|_F} - 1 \right| \leq \varepsilon. \tag{11}$$

The above inequality indicates that, relatively speaking, there is little difference between the two terms $\|M - XY\|_F$ and $\|M - UV\|_F$, implying that UV is almost the same as XY after projections (7c) and (7d). Thus, most likely, (X, Y) is already nearly feasible due to large enough penalty parameters α and β. In this case, we increase the weight of the fidelity violation term $\|XY - M\|_F^2$ in the augmented Lagrangian function (6) by reducing both α and β in order to facilitate a significant decrease in fidelity violation at the next iteration (or next a few iterations).

Case 2. Non-improved feasibility for at least one variable:

$$\|X_+ - U_+\|_F \geq \|X - U\|_F \ \text{ or } \ \|Y_+ - V_+\|_F \geq \|Y - V\|_F. \tag{12}$$

In this case, we consider increasing α and/or β independently according to the changes in $\|X - U\|_F$ and in $\|Y - V\|_F$. Take the former as an example. If $\|X - U\|_F$ does not decrease during the past several iterations, then we increase its corresponding penalty parameter α in order to facilitate its decrease in future steps to make X more feasible. The same argument applies to the term $\|Y - V\|_F$. We can expect that if

we keep increasing one or both penalty parameters the iterates will eventually reach **Case 1** since (X, Y) will be getting closer and closer to (U, V).

Case 3. Now both conditions (11) and (12) have failed. We consider the condition

$$\left| \frac{\|M - X_+Y_+\|_F}{\|M - XY\|_F} - 1 \right| \leq \varepsilon. \tag{13}$$

which implies that fidelity has not been improved sufficiently. In this case, we choose to decrease both penalty parameters α and β to allow a faster reduction in fidelity violation. On the other hand, if condition (13) does not hold, then both feasibility and fidelity are improving, but there is still a considerable gap between $\|M - XY\|_F$ and $\|M - UV\|_F$ since condition (11) does not hold. In this case, we choose to increase α and β in order to accelerate feasibility satisfaction and to narrow the gap between $\|M - XY\|_F$ and $\|M - UV\|_F$. In general, $\|M - UV\|_F$ is greater than $\|M - XY\|_F$ since (U, V) is from restrictive subsets while (X, Y) is "free". Hence suddenly closing the gap between $\|M - XY\|_F$ and $\|M - UV\|_F$ usually means at least a temporary increase in the value of $\|M - XY\|_F$. An alternative option here is to keep α and β unchanged which would also seem reasonable. We opt for increasing α and β based on empirical observations that this strategy can often speed up convergence and help avoid to be trapped by local minima.

The proposed adaptive penalty parameter update scheme is summarized in Algorithm 2.

Algorithm 2: Adaptive Penalty Parameter Update

Input: α and β
Output: α and β
Set $\mu, \nu > 1$ and select a small $\varepsilon \in (0, 1)$.
If (10) is satisfied, then **exit**. (No update)
If (11) is satisfied, set $\alpha = \alpha/\nu$, $\beta = \beta/\nu$, and **exit**. (Case 1)
If the first inequality in (12) holds, set $\alpha = \mu\alpha$. (Case 2a)
If the second inequality in (12) holds, set $\beta = \mu\beta$, and **exit**. (Case 2b)
If (13) holds, set $\alpha = \alpha/\nu$ and $\beta = \beta/\nu$; (Case 3a)
otherwise, set $\alpha = \mu\alpha$ and $\beta = \mu\beta$. (Case 3b)

For the sake of robustness, we evaluate all the conditions in Algorithm 2 in an average sense at every $q > 1$ iterations rather than at every iteration. Namely, all the quantities involved are the average of q iterations. In this paper, we always fix the number at $q = 5$ throughout our experiments. Specifically, to evaluate condition (13) at iteration 20, for example, we actually evaluate the inequality below:

$$\sum_{k=16}^{20} \|M - X_kY_k\|_F \geq (1 - \varepsilon) \sum_{k=11}^{15} \|M - X_kY_k\|_F$$

which of course requires to save and update the involved average quantities.

Overall, the spirit of the above updating rules is to find a good balance between the progresses of fidelity and feasibility; namely, between the three terms $\|M -$

$XY\|_F$, $\|X - U\|_F$ and $\|Y - V\|_F$ in the augmented Lagrangian function while also taking into account the value of $\|M - UV\|_F$. Although there is no theoretical guarantee about the performance of our algorithm on the highly nonconvex problem (1), our adaptive update strategy does appear to have worked well on numerous test matrices and structure sets. In particular, we present numerical results comparing the dynamical update scheme with fixed-value penalty parameters in Section 4.

3 Projections onto Structure Sets

Our SeMF algorithm, i.e., Algorithm 1, requires to project a point X onto a structure set $\mathcal{X}$, and Y onto $\mathcal{Y}$ as well, at each iteration. Since either $\mathcal{X}$ or $\mathcal{Y}$ can be nonconvex, some clarifications are necessary.

3.1 Definition of Projection

For a given norm $\|\cdot\|$, the projection of $x \in \mathbb{R}^n$ onto a subset $\mathcal{S} \subset \mathbb{R}^n$ is normally defined as

$$\mathcal{P}_{\mathcal{S}}(x) := \arg\min\{\|y - x\| : y \in \mathcal{S}\}. \tag{14}$$

In this paper, we will always use the Euclidean norm for vectors and the Frobenius norm for matrices. Under such norms, it is well-known that when $\mathcal{S}$ is a nonempty, closed and convex subset, then the projection is uniquely defined for any $x \in \mathbb{R}^n$. Short of convexity for $\mathcal{S}$, however, the projection defined by (14) can be non-unique at least for some x. Although this non-uniqueness hardly poses any real problem in practice, we need to extend the definition of projection so that $\mathcal{P}_{\mathcal{S}}(\cdot)$ refers to any one of the minima if multiple minima exist in (14).

3.2 Projections onto Some Simple Sets

We first briefly discuss projections onto several simple sets often appearing in applications that allow easy projections. We will assume that the relevant spaces consist of matrices X. For simplicity, we only list structures imposed on each column X_j of X where X_j is the j-th column of X, but they can be equally imposed on rows (or on other types of blocks) of X.

- **Non-negativity**: $\mathcal{S} = \{X : X_{ij} \geq 0, \forall\, i, j\}$.

$$[\mathcal{P}_{\mathcal{S}}(X)]_{ij} = \max(0, X_{ij}).$$

- **Sparsity**: $\mathcal{S} = \{X : \|X_j\|_0 \leq k, \forall\, j\}$ where $\|\cdot\|_0$ is the number of nonzero elements in a vector.
  In the above, $\|X_j\|_0 \leq k$ means that the j-th column, X_j, contains at most k nonzero elements. Without loss of generality, assume that the absolute-value vector $|X_j|$ is already ordered in a descending order so that $|X_{1j}| \geq |X_{2j}| \geq \cdots$, and so on.

$$[\mathcal{P}_{\mathcal{S}}(X)]_{ij} = \begin{cases} X_{ij}, & i \leq k, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

– **Orthogonality to a fixed column**: $\mathcal{S} = \{X : X_j \perp X_{j'}, \forall\, j \neq j'\}$ where j' is given.

$$[\mathcal{P}_{\mathcal{S}}(X)]_j = \begin{cases} X_j - X_{j'}(X_{j'}^T X_{j'})^{-1} X_{j'}^T X_j, & j \neq j', \\ X_{j'}, & \text{otherwise}, \end{cases} \tag{16}$$

where we assume that $X_{j'} \neq 0$ (otherwise no operation is necessary).

– **Column Normalization**: $\mathcal{S} = \{X : \|X_j\|_2 = 1, \forall\, j\}$.

$$[\mathcal{P}_{\mathcal{S}}(X)]_j = \frac{X_j}{\|X_j\|_2}, \forall\, j, \tag{17}$$

where, without loss of practical generality, we assume that $X_j \neq 0$ for some j (otherwise, a zero column could be replaced by an arbitrary unit vector).

3.3 Projections onto Combinatorial Structure Sets

We demonstrate that some sparsity structure sets of combinatorial nature also permit easy projections. For clarity, we do this by taking a simple, made-up example. Let us say that a fictitious DNA consists of 4 genes, A, B, C and D, each admitting 5 mutations. Therefore, there are totally 5^4 possible combinations, corresponding to 20 basic building elements for this DNA each being a mutation of a distinct gene.

Any given sample of the DNA can be viewed as a linear combination of four elements coming from the four distinct groups of five. As such, each expression has a sparse representation under a basis consisting of the 20 basic elements. In such a sparse representation, each nonzero must be from a distinct group of five. Let M be a given sample set of the DNA. We wish to find X and Y such that $M \approx XY$, where X consists of the 20 basic elements each being expressed as a column of X, and each column of Y is a representation (or signature) of a DNA sample under the basis X. Now we concentrate on the sparsity of Y.

We assign an index, from 1 to 20, to each gene mutation and group them so that $A = \{1, 2, 3, 4, 5\}$, $B = \{6, 7, 8, 9, 10\}$, $C = \{11, 12, 13, 14, 15\}$ and $D = \{16, 17, 18, 19, 20\}$. By partitioning the 20 rows of Y into four equal-size blocks, we write

$$Y = \begin{bmatrix} Y_A \\ Y_B \\ Y_C \\ Y_D \end{bmatrix}$$

By the properties described above, each column of Y_K, $K = A, B, C, D$, can have at most one nonzero component; that is, each block Y_K belongs to the structure set

$$\mathcal{T} = \{Z : \|Z_j\|_0 \leq 1, \forall\, j\}, \tag{18}$$

and the projection onto $\mathcal{T}$, $\mathcal{P}_{\mathcal{T}}(\cdot)$, is defined as in (15). Meanwhile, the matrix Y belongs to the set

$$\mathcal{S} = \{Y : Y_K \in \mathcal{T}, K = A, B, C, D\} \tag{19}$$

and the projection onto $\mathcal{S}$ is

$$\mathcal{P}_{\mathcal{S}}(Y) = \begin{bmatrix} \mathcal{P}_{\mathcal{T}}(Y_A) \\ \mathcal{P}_{\mathcal{T}}(Y_B) \\ \mathcal{P}_{\mathcal{T}}(Y_C) \\ \mathcal{P}_{\mathcal{T}}(Y_D) \end{bmatrix}. \tag{20}$$

Obviously, the above projection can be easily extended to more complex situations. For example, each block of Y may have a different number of rows, and the column sparsity of some blocks may be more than one. To the best of our knowledge, so far there has been no algorithm designed to directly handle such combinatorial properties.

In the above example we may very well demand that Y be nonnegative. This means that Y belongs to the intersection of $\mathcal{S}$ defined in (19) and the set of nonnegative matrices of proper sizes. Projections onto such intersections will be discussed next.

3.4 Projections onto Certain Intersection Sets

We consider two intersection sets that will appear in our experiments later. We prove that the projections onto these intersection sets can be carried out by successively performing one projections after another in a specified order. Assume again that structures are imposed on each column of X.

Proposition 1 ***(Non-negativity + Sparsity)***
Let $\mathcal{S}_1 = \{X : X_{ij} \geq 0, \forall\, i, j\}$, $\mathcal{S}_2 = \{X : \|X_j\|_0 \leq k, \forall\, j\}$, $\mathcal{S} = \mathcal{S}_1 \cap \mathcal{S}_2$, *then*

$$\mathcal{P}_{\mathcal{S}}(\cdot) = \mathcal{P}_{\mathcal{S}_2}\left(\mathcal{P}_{\mathcal{S}_1}(\cdot)\right) \tag{21}$$

Proof Due to separability of Frobenius-norm square, without loss of generality we can assume that X has only a single column. For convenience, we replace X by a vector $x \in \mathbb{R}^m$, and use the following notation:

$$|x| = (|x_1|, \cdots, |x_m|)^T, \quad x_+ = \frac{x + |x|}{2} \quad \text{and} \quad x_- = \frac{x - |x|}{2}$$

so that $x = x_+ + x_-$ and $x_+^T x_- = 0$. There also holds

$$x_+ = \mathcal{P}_{\mathcal{S}_1}(x) \triangleq \arg\min_{z \geq 0} \|z - x\|$$

where the norm is the Euclidean norm by default. Let

$$\tilde{y} \in \mathcal{P}_{\mathcal{S}_2}\left(\mathcal{P}_{\mathcal{S}_1}(x)\right) = \arg\min_{y \in \mathcal{S}_2} \|y - x_+\|. \tag{22}$$

Clearly $\tilde{y} \in \mathcal{S}_1 \cap \mathcal{S}_2$ and

$$(\tilde{y} - x_+)^T x_- = 0, \tag{23}$$

for any $y \in \mathcal{S}_1 \cap \mathcal{S}_2$, since $y \geq 0$ and $x_- \leq 0$,

$$-(y - x_+)^T x_- = -y^T x_- + x_+^T x_- = -y^T x_- + 0 \geq 0. \tag{24}$$

For any $y \in \mathcal{S}_1 \cap \mathcal{S}_2$, in view of the identity $x = x_+ + x_-$,

$$\begin{aligned} \|\tilde{y} - x\|^2 &= \|\tilde{y} - x_+ - x_-\|^2 = \|\tilde{y} - x_+\|^2 + \|x_-\|^2 \\ &\leq \|y - x_+\|^2 + \|x_-\|^2 \leq \|y - x_+ - x_-\|^2 \\ &= \|y - x\|^2, \end{aligned}$$

where the second equality follows from (23), the first inequality from (22), and the second inequality from (24). This proves that $\mathcal{P}_{\mathcal{S}}(x) = \arg\min_{y \in \mathcal{S}} \|y - x\|_2 = \mathcal{P}_{\mathcal{S}_2}(\mathcal{P}_{\mathcal{S}_1}(x))$. □

Proposition 2 ***(Sparsity + Positive Equal Nonzeros)*** *Let $\mathcal{S} = \mathcal{S}_1 \cap \mathcal{S}_2$ where $\mathcal{S}_1 = \{X : \|X_j\|_0 = k, \forall j\} \cup \{0\}$ and $\mathcal{S}_2 = \{X : X_{ij} = \alpha_j > 0, \forall X_{ij} \neq 0\}$. For all j, let X_j be the j-th column of X, and I_j contain the indices of k largest elements of X_j and $\alpha_j^* = \frac{1}{k}\sum_{i \in I_j} X_{ij}$. Then*

$$[\mathcal{P}_{\mathcal{S}}(X)]_{ij} = \begin{cases} \max(0, \alpha_j^*), & i \in I_j \\ 0, & \textit{otherwise.} \end{cases}$$

Proof Again by separability, we replace matrix X by vector x without loss of generality. For any vector $x \in \mathbb{R}^m$, we will explicitly solve the minimization problem $\mathcal{P}_{\mathcal{S}}(x) = \arg\min_{y \in \mathcal{S}} \|y - x\|_2$.

We note that for any $y \in \mathcal{S}$, either y has k positive nonzeros so that $y_I = \alpha > 0$ for some index set I of cardinality k (i.e., $|I| = k$) or $y = 0$ corresponding to $I = \emptyset$, while elements of y outside of I are understood to be all zeros. For an arbitrary I with $|I| = k$ and a corresponding $y \in \mathcal{S}$, consider the problem of minimizing

$$\begin{aligned} \|y - x\|^2 &= \sum_{j \in I} (\alpha - x_j)^2 + \sum_{j \notin I} x_j^2 \\ &= k\alpha^2 - 2\alpha \sum_{j \in I} x_j + \sum_{j=1}^{m} x_j^2 \\ &= k\left[(\alpha - \text{mean}(x_I))^2 - (\text{mean}(x_I))^2\right] + \|x\|^2. \end{aligned}$$

If for all I with $|I| = k$ we have $\text{mean}(x_I) \leq 0$, then the first term above is nonnegative so that $\|y - x\|^2 \geq \|x\|^2$, implying that $y = 0$ is the unique minimizer (thus the projection of x on $\mathcal{S}$). Otherwise, $\|y - x\|^2 \geq \|x\|^2 - k(\text{mean}(x_I))^2$ and the minimum is attained when (i) $\alpha = \text{mean}(x_I)$, and (ii) $\text{mean}(x_I)$ is maximized over all I with $|I| = k$; i.e., when I contains k largest elements of x (the minimizer may not be unique though). To sum up, we conclude that the optimal value of α is $\alpha^* = \max(0, \text{mean}(x_{I^*}))$ where I^* contains k largest elements of x. This completes the proof.

We note that in general projections onto intersections can be difficult even when it is easy to project onto each individual set. In the ADMM framework, one might consider introducing more split variables and doing projections in a sequence. However, it is well know that the more blocks there are in an alternating minimization scheme, the slower the convergence will be in general. Thus, it is desirable to use as few blocks as possible, and to consider computing approximate projections onto intersection sets (such as successive projections onto individual sets involved).

4 Computational Results

This section contains four sets of numerical experiments. In Section 4.1, we evaluate the performance of our SeMF algorithm, implemented in Matlab, with and without the adaptive penalty update scheme to illustrate the advantage of the scheme (i.e., Algorithm 2). In Section 4.2, we apply the SeMF algorithm to dictionary learning for sparse representation and compare it with the well-established algorithm K-SVD [15]. In Section 4.3, we apply the SeMF algorithm to sparse nonnegative matrix factorization using the dataset ORL [34] and compare it with one of the latest algorithms designed for this problem. Finally, in Section 4.4, we illustrate the versatility of our SeMF algorithm by adding various constraints to the factorization of the swimmer dataset [35] to achieve improved quality.

In the four sets of numerical experiments, we will examine the performance of our SeMF algorithm using different measures of accuracy according to different purposes of experiments and different features of problems. Specifically, in Section 4.1 we use the residual error (i.e., $\|M - XY\|_F$ and $\|M - UV\|_F$) to examine the convergence behavior of our adaptive penalty parameter update scheme. In Sections 4.2 and 4.4, we examine the accuracy of recovering "ground-truth" in given datasets in order to compare algorithms' recovery quality. In Section 4.3, the problem is to find base features for face images for which there is no "ground-truth". In this case, we measure the solution quality by SNR which is often used in image processing literatures.

All numerical experiments were run under Matlab version 8.0 (R2012b) on a Thinkpad laptop computer with an Intel Core i5 processor at 2.5GHz with 8GB RAM.

The basic default setting of our SeMF algorithm is as follows. Throughout our experiments, we always use the exact formula (8a) and (8b) in place of (7a) and (7b) to update X and Y in Algorithm 1. We set the maximum number of iterations to $maxiter = 1000$ and the tolerance value to tol =1e-06, unless otherwise specified. In any comparison run, we always use the same random initial guess to start all tested algorithms. In Algorithm 2 (adaptive penalty parameter update scheme), unless otherwise specified we always use the default parameter values

$$\mu = 2, \quad \nu = 5 \quad \text{and} \quad \varepsilon = 5 \times 10^{-4}. \tag{25}$$

We will justify these default values in the next section based on empirical evidence from numerical experiments. In addition, in Algorithm 1 we initialize penalty parameters to the default value $\alpha = \beta = \|M\|_F/100$ except in Section 4.1 below where we vary initial values for these two penalty parameters.

4.1 Validating Adaptive Penalty Update Scheme

To evaluate the effectiveness of our adaptive penalty parameter update scheme given in Algorithm 2, we conduct a set of tests using synthetic data to compare the behavior of our SeMF algorithm with and without the adaptive scheme.

For each test instance, we randomly generate a matrix $X \in \mathbb{R}^{40\times 60}$ using the Matlab commend `randn` while each column of X is normalized to unit ℓ_2-norm. We then construct a sparse matrix $Y \in \mathbb{R}^{60\times 1500}$ so that each column of Y has 3 non-zeros in random values (using `randn`) and at random locations. Then we synthesize the 40×1500 "exact" data matrix as the product $M = XY$. In this case, the structure sets are $\mathcal{X} = \{X : \|X_j\|_2 = 1, \forall j\}$ and $\mathcal{Y} = \{Y : \|Y_j\|_0 \leq 3, \forall j\}$. We will conduct several tests based on this synthetic data to demonstrate the effectiveness of the proposed adaptive scheme and discuss how the parameters μ, ν, and ε in the penalty update scheme affect the ADMM algorithm's performance.

First, we focus on the behavior of our SeMF algorithm with and without the adaptive scheme when started from different initial values for α and β. In this test, μ, ν, and ε are set to their default values as in (25).

We test on 3 pairs of initial penalty parameter values

$$(\alpha,\, \beta) = 10^{1-2k} \times \|M\|_F \times (1,\, 0.1), \quad k = 1, 2, 3.$$

For example, $(\alpha,\, \beta) = (3.873 \times 10^{-3}, 3.873 \times 10^{-4})$ for $k = 1$. Based on empirical evidence, we set $\alpha = 10 \times \beta$ since such a choice tends to give better convergence results for fixed penalty values in a proper range.

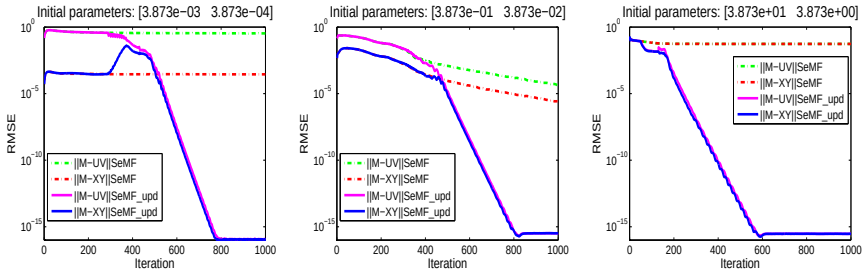


Fig. 1 Convergence history of $\|M - UV\|_F$ and $\|M - XY\|_F$ produced by Algorithm 1 with fixed and adaptive penalty parameters for 3 pairs of different penalty parameter values. In each plot, the two dashed lines are with fixed penalty parameters and the two solid lines are with the adaptive penalty parameter scheme.

It should be evident from Fig. 1 that the adaptive updating strategy indeed dramatically improves the robustness of Algorithm 1 with respect to the choices of penalty parameters. With the adaptive scheme it succeeded in all 3 cases whose parameter values span a wide range in magnitude. We believe that this robustness represents a major advantage for our adaptive strategy.

We mention that all the conditions in Algorithm 2 can be encountered, but the frequencies of their occurrences are situation-dependent. Take the left-most plot of

Fig. 1 as an example. Because of the small initial parameter values, Cases 1 and 3a in Algorithm 2 never occur. Instead, Cases 2a or 2b occasionally occur when at least one inequality of (12) is satisfied. Conversely, for the case in the right-most plot of Fig. 1 where initial parameters are large, Case 1 occurs early at the 50th iteration since large penalties result in nearly equal values of $\|M-UV\|_F$ and $\|M-XY\|_F$ which make (11) to be satisfied. Later Case 1 occurs again at around the 150th iteration because by then the penalties are still relatively large. Afterwards, the algorithm starts converging at a steady and fast linear rate.

Next we focus on examining how μ, ν, and ε affect the algorithm's performance, and conduct numerical experiments to help find appropriate default values.

We first consider the parameter $\mu > 1$ which is for increasing the penalty parameter α (or β) to $\mu\alpha$ (or $\mu\beta$) when appropriate conditions are met. We start the SeMF algorithm with the relatively small initial penalty parameter values $(\alpha,\ \beta) = 10^{-3} \times \|M\|_F \times (1,\ 1)$ and observe how different values of μ will impact the algorithm's convergence, while fixing other two parameters at their default values $\nu = 5$ and $\varepsilon = 5 \times 10^{-4}$, respectively. Over many randomized runs with different random starting points, we present a set of typical convergence history results with 3 different μ values ($\mu = 1.2, 2.0$ and 4.0) in Fig. 2.

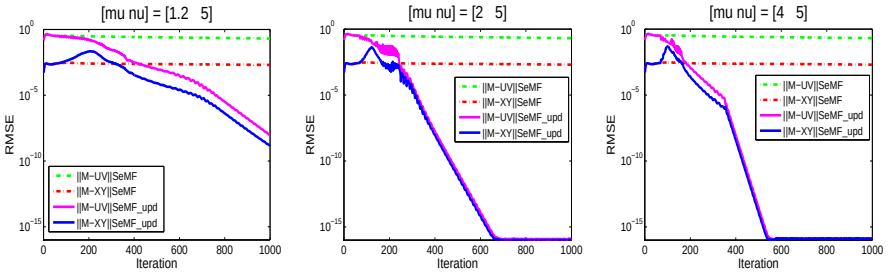


Fig. 2 Convergence history of $\|M-UV\|_F$ and $\|M-XY\|_F$ produced by Algorithm 1 with different μ values in Algorithm 2 (the adaptive penalty parameter scheme). In each plot, the two dashed lines are with fixed penalty parameters and the two solid lines are with Algorithm 2.

We observe from Fig. 2 that convergence was slower for the smaller value of $\mu = 1.2$, which makes a smaller amount of change in penalty parameters for each update. On the other hand, we have found that $\mu > 3$ seems to lead to a loss of robustness. That is, as μ becomes larger than 3, over many randomized runs the observed instances where the algorithm does not reach an "exact" solution increases notably. To strike a balance between speed and robustness, we consider $\mu \in (1, 3]$ to be appropriate, and opt for $\mu = 2$ as the default value. In this set of experiments, we observe that Cases 2 and 3b in Algorithm 2 play a large role in achieving good convergence. Specifically, condition (12) is satisfied frequently and the two penalty parameters are increased again and again.

For the parameters ν and ε, we also designed and conducted experiments similar to those for the parameter μ. After experimenting on wide ranges of values, we have settled at the default values of $\nu = 5$ and $\varepsilon = 5 \times 10^{-4}$ to be used in Algorithm 2.

4.2 Learning Dictionary for Sparse Representation

In recent years, there is a growing interest in the study of dictionary learning for sparse representations of signals. Roughly speaking, we say that a signal $y \in \mathbb{R}^m$ admits a sparse representation under a *dictionary* $D \in \mathbb{R}^{m\times n}$ if one can find a linear combination of "a few" columns (atoms) from D that is "close" to the signal y. Sparse representations serve useful purposes in many signal processing tasks, and a key to success is to have a sufficiently good dictionary. In many situations, a good dictionary, such as a wavelet basis, is known *a priori*. Most earlier works in this field have been done based on this premise, and algorithms have been developed to reconstruct signals from a given dictionary and associated measurements. Such algorithms include, but not limited to, *Matching Pursuit* (MP) [36], *Orthogonal Matching Pursuit* (OMP) [37–40], *Basis Pursuit* (BP) [41] and the *Focal Underdetermined System Solver* (FOCUSS) which use different sparsity measure ℓ_p-norm ($0 \le p \le 1$) [42–45]. More recently, there is a growing body of works without assuming that a dictionary is known. Instead, a dictionary is constructed for training data using learned techniques such as *K-means* [46], *Maximum Likelihood Methods* (ML) [47–50], *Method of Optimal Directions* (MOD) [51–53], *Maximum A-posteriori Probability* (MAP) [54–56], K-SVD [15], or *Online Dictionary Learning* (ODL) [16], for example. Under favorable conditions, learning the dictionary instead of using off-the-shelf bases has been shown to dramatically improve signal reconstruction.

Dictionary learning for sparse representation can be formulated as a Structured-enforced Matrix Factorization (SeMF) problem. Here we will change our notion to the one more popular in the literature of dictionary learning. We denote a training dataset by Y (in place of M), a dictionary by D (in place of X) and a sparse representation by X (in place of Y). The corresponding SeMF model (1) takes the following form,

$$\min_{D,X} ||Y - DX||_F^2 \ \text{ s.t. } \ D \in \mathcal{D},\ X \in \mathcal{X}, \tag{26}$$

where,

$$\begin{aligned} \mathcal{D} &= \{[d_1, \cdots, d_p] \in \mathbb{R}^{m\times p} : \|d_i\|_2 = 1, \forall i = 1, \cdots, p\}, \\ \mathcal{X} &= \{[x_1, \cdots, x_n] \in \mathbb{R}^{p\times n} : \|x_i\|_0 \le k, \forall i = 1, \cdots, n\}. \end{aligned} \tag{27}$$

Both $\mathcal{D}$ and $\mathcal{X}$ are easily projectable sets so that we can apply our SeMF algorithm to (26) without difficulty.

We compare our SeMF algorithm with the well-established K-SVD algorithm [15] on synthetic signals constructed as in the experiments in [15]. The operations of K-SVD consist of two alternating stages: an SVD stage and a sparse-coding stage. The main computational cost is with the latter stage which is performed by using an orthogonal marching pursuit (OMP) algorithm. In our comparison, we use the Matlab code KSVDBox (v13)[1] that calls the package OMPBox (v10) [40] for doing OMP operations.

Generation of the data: A random matrix D (referred to later on as the *generating dictionary*) of size $m \times p$ is generated, consisting of normally distributed iid

[1] Available at http://www.cs.technion.ac.il/~ronrubin/software.html.

entries with mean zero and variance one, each column of which is normalized to a unit ℓ_2-norm. Then n signal samples of dimension m are produced, each a linear combination of k different generating dictionary atoms, with uniformly distributed iid coefficients in random and independent locations. White Gaussian noise is added to the resulting data signal samples. We denote the generated signal samples as Y.

Evaluation of computed dictionaries: The quality of a computed dictionary, $\tilde{D}$, is evaluated against the generating dictionary D. This comparison is done by sweeping through columns of the generating dictionary and finding the closest one (in ℓ_2-norm) in $\tilde{D}$, measuring the distance via the formula

$$dist(d_j, \tilde{D}) = \min_{i=1,\cdots,p} \left(1 - |d_j^T \tilde{d}_i|\right). \tag{28}$$

Then define the distance between the two dictionaries by the mean

$$dist(D, \tilde{D}) = \frac{1}{p} \sum_{i=1}^{p} dist(d_j, \tilde{D}), \tag{29}$$

As is defined in [15], we say that the atom d_j is successfully recovered if the distance $dist(d_j, \tilde{D}) \leq 0.01$.

Setting of the tests: In SeMF algorithm we use the default setting except using $maxiter = 500$. In K-SVD algorithm, we set $maxiter = 200$ as was used in the paper [15] (most tests terminate within 100-150 iterations). In this experiment, we perform three sets of tests where the dictionary size is always set to $(m, p) = (20, 50)$. Unless specified otherwise, we set sparsity $k = 3$, and add white Gaussian noise to generated sample so that the signal-noise ratio $\mathrm{SNR} = 20$.

In the first test, we vary the sample size n from 200 to 1000 with increment 50, compute the percentage of recovered atoms of the generating dictionary and the distance between the learned and generating dictionaries defined in (29), and record computing time used by SeMF and K-SVD algorithm, respectively. For each quantity, we report the average of 20 runs starting from the same random initial points for the two methods. The results are the three plots in Fig. 3.

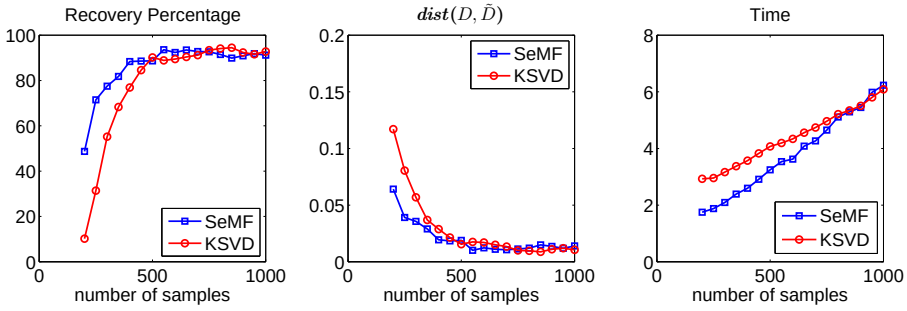


Fig. 3 Performance of SeMF and K-SVD on dictionary learning with different sample sizes

The essential observation from Fig. 3 is that in this test set SeMF tends to perform better than K-SVD in a consistent manner when the number of sample is relatively

small. As the number of samples increases, the performances of the two eventually become indistinguishable in terms of quality of recovery. In terms of computing time, the tendency appears to be that SeMF would eventually become more expensive than K-SVD as the sample size continues to increase. We do mention that the dominant computational task (sparse coding) in K-SVD is coded in C language, while SeMF is coded entirely in Matlab. In addition, we note that in this test there is really no need to use sample sizes much larger than $n = 500$ at which level SeMF is still faster than K-SVD is.

In the second test, we vary the sparsity level from $k = 1$ to 11 and find the smallest number of samples needed to recover at least 90% of the generating dictionary. At each sparsity level k, we start from sample size $n = 200$ and run SeMF and K-SVD each 10 times with different random starting points, and then record the average recovery rate in percentage. If the average is less than 90%, we increase the number of samples by 50 and repeat the process, until both SeMF and K-SVD reach the average recovery rate of 90%. The results from this test are given in Fig. 4, which show that, to learn those tested dictionaries, SeMF tends to require considerably less samples than K-SVD does. Since it needs less samples, SeMF is faster than K-SVD when sparsity level k becomes relatively large. We present results for varying sparsity level from $k = 1$ to 11 because for $k \geq 12$, both algorithms fail to recover 90% of the dictionary atoms. no matter how many samples are taken and how large the number of maximum iterations is set. In fact, K-SVD already fails at $k = 11$ and Fig. 4 contains no statistics for K-SVD at $k = 11$.

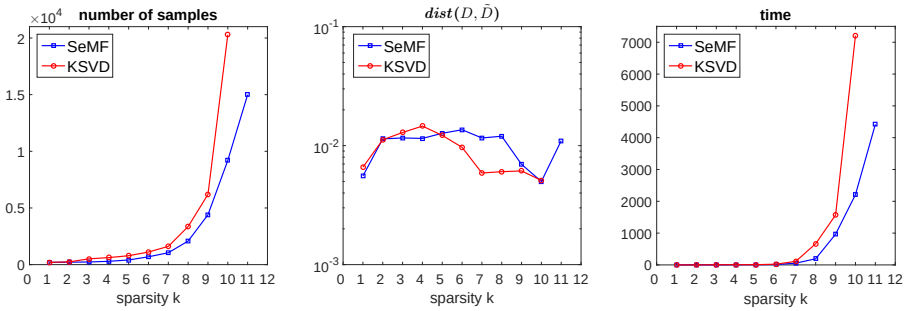


Fig. 4 SeMF and K-SVD: Sample size needed for 90% recovery rate vs sparsity

In the third test, we generate sample datasets with SNR level varying in the range $(10, 20, 30, \infty)$ in order test the performance of SeMF and K-SVD with respect to noise in data. For each instance we do 100 random trials and record the number of recovered atoms in each trial. These 100 numbers are then sorted into 5 groups of 20 and the average values are taken for all five group. Following what is done in [15], we plot these five average numbers of successfully recovered atoms for both SeMF and K-SVD in Fig. 5. Recall that there are 50 atoms in total for all the generating dictionaries. Hence, the number 40, for example, implies a 80% recovery rate.

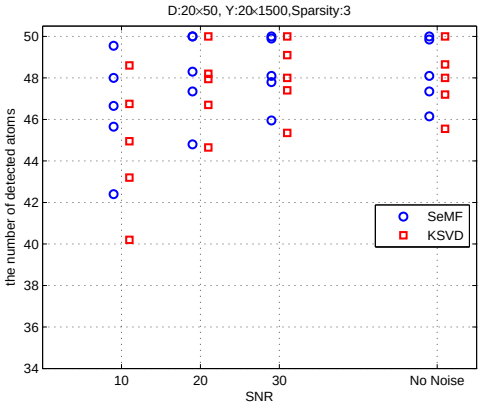


Fig. 5 Number of recovered atoms by SeMF and K-SVD with noisy data

The plot in Fig. 5 suggests that in this test SeMF demonstrate a slightly higher degree of robustness than K-SVD do when there is noise in data, especially when the noise level is relatively high (SNR = 10).

4.3 Factorizing Face Images from ORL database

Lee and Seung [7] find that nonnegative matrix factorization (NMF) produces part-based representations when applied to the CBCL face image database[2]. However, it is noted by Hoyer [10] that when applied to the face images in the ORL database [34], which are not well aligned, the resulting representations appear to be rather global and holistic. Numerous sparsity constraints have been proposed to be added to NMF in order to promote part-based representations.

The ORL database contains 400 face images of the resolution $92 \times 112 = 10304$ pixels. These face images form the training data matrix M of size 10304 by 400. In the computational experiments of [10, 13], the number of basis vectors in X is set to 25. Hence the sizes of X and Y are 10304×25 and 25×400, respectively. We use the same setting to test our SeMF algorithm on the ORL database via solving the following problem,

$$\min_{X,Y} ||M - XY||_F^2 \text{ s.t. } X \in \mathcal{X}, Y \in \mathcal{Y}, \tag{30}$$

where,

$$\begin{aligned} \mathcal{X} &= \{X : X_{ij} \geq 0, \|X_j\|_0 \leq k, \forall\, i, j\} \\ \mathcal{Y} &= \{Y : Y_{ij} \geq 0, \forall\, i, j\}. \end{aligned} \tag{31}$$

As in [10], we run tests for three values of sparsity k (the number of nonzero pixels), corresponding to 33%, 25% and 10% of the total number of pixels per image (10304).

[2] http://cbcl.mit.edu/projects/cbcl/software-datasets/FaceData1Readme.html

In SeMF, we set the maximum number of iterations to $maxiter = 500$ and choose initial penalty parameter value $\alpha = \beta = 0.3\|M\|_F$.

We do a comparison between our SeMF algorithm and a recent algorithm by Peharz and Pernkopf [13], implemented in a Matlab package called NMFℓ^0 (in which only the function NMFℓ^0-W is relevant to our tests)[3], which has been reported to produce good results and to be faster than the earlier algorithm of Hoyer [10]. In our experiment, we use the default setting of the code NMFℓ^0 without any change.

Fig. 6 shows the computed basis vectors in a particular run, reshaped into 92×112 images, returned by SeMF and by NMFℓ^0 in a typical run, with the three columns corresponding to the three sparsity levels at 35%, 25% and 10%. For a better visual effect, we have reversed the pixel values so that dark pixels indicate high values and white pixels indicate low ones.

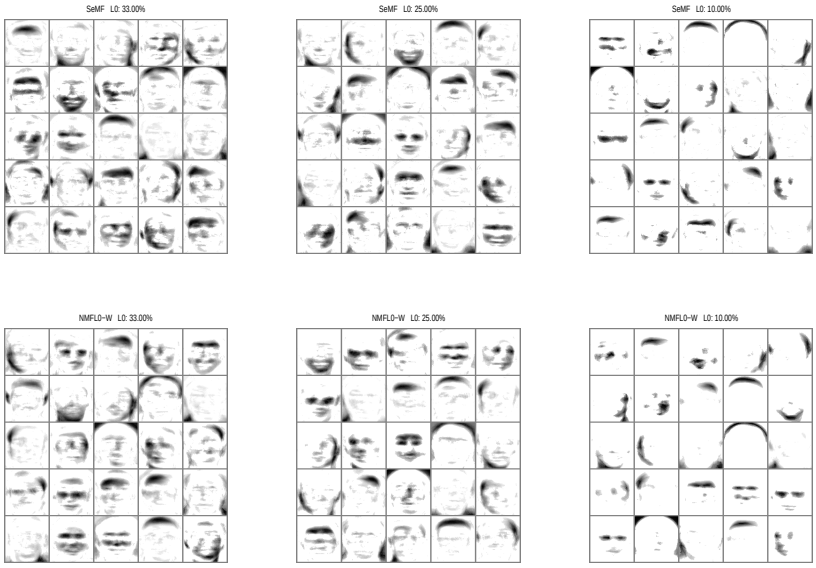


Fig. 6 Basis images computed by SeMF (row 1) and NMFℓ^0 (row 2) at the 3 sparsity levels.

We see from Fig. 6 that the images from the two methods are visually similar. As the basis images becomes sparser, they naturally also become more part-based. To quantify the solution quality and running time, we make 10 random runs and compute the average running time and average SNR, as is done in [13], where SNR is defined in *db* as

$$\text{SNR} = 20 \log_{10} \frac{\|M\|_F}{\|M - XY\|_F}.$$

[3] Available at: http://www.spsc.tugraz.at/tools/nmf-l0-sparseness-constraints.

In Table 1, we tabulate the average SNR (in db) and average running time (in second). While the SNR values are close (with a slight advantage towards SeMF), we observe that SeMF takes considerably less time than NMFℓ^0 does on these test problems.

Table 1 Comparison of reconstruction quality in terms of SNR, and running time for SeMF and NMFℓ^0 [13]when the same ℓ^0-sparseness is enforced.

Method	ℓ^0	SNR	Time	ℓ^0	SNR	Time	ℓ^0	SNR	Time
NMFℓ^0	33%	14.945	399.88	25%	14.832	294.35	10%	14.237	128.94
SeMF	33%	14.973	**76.59**	25%	14.858	**74.42**	10%	14.291	**75.65**

It is interesting to note that the gap in running time widens as the number of nonzero entries in basis images increases. The running time of NMFℓ^0 is roughly linearly proportional to the number k since a major operation in the algorithm is component-wise multiplication of sparse matrices. On the other hand, the running time of SeMF is independent of k. However, we note that the update formulas (8a) and (8b) both require solving linear systems of size $p \times p$ where p is the number of basis vectors in X. In the current tests $p = 25$ which is negligible relative to $m = 10304$. When p becomes relatively large, the speed advantage of SeMF should diminish to some extent.

4.4 Factorizing the Swimmer Dataset

The Swimmer Dataset [35] consists of 256 images of resolution 32×32, representing a swimmer built by an invariant torso and 4 limbs. Each of the 4 limbs can be in one of 4 positions and the dataset is formed of all combinations. Some images are shown in Fig. 7 depicting eight stick figures with four limbs. Hence, the ground truth decomposition is known for this dataset, i.e. each image is comprised by 5 of the 17 distinct non-overlapping basis images (or parts), as are shown in Fig. 8.

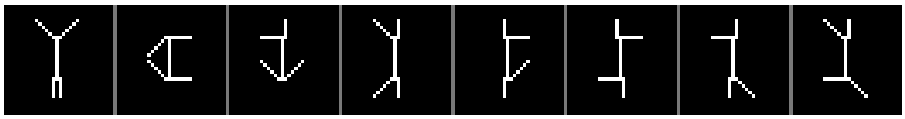


Fig. 7 Sample images from the Swimmer database; illustrating different articulations of limbs.

A central question here is that given enough samples like those shown in Fig. 7, can and how one recover the "ground truth" basis images given in Fig. 8 in an exact order? By "an exact order", we mean that each row should include all four possible positions of a limb. To see this clearly, we look at Fig. 9 where the 17 basis images are grouped into five natural groups so that each swimmer sample consists of five images each coming from one of the five groups. This is precisely a combinatorial property discussed on Section 3.2.

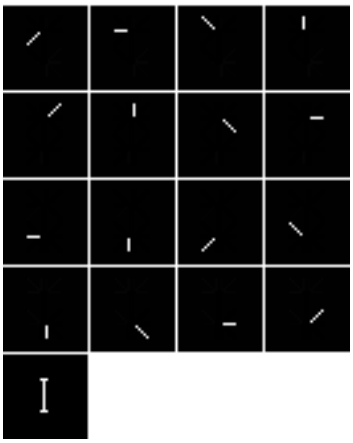
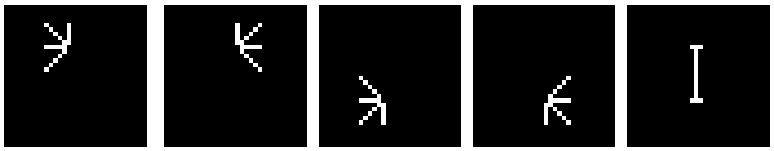


Fig. 8 The seventeen basis images of the Swimmer Dataset



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

Fig. 9 Five natural groups of the 17 basis images

So far, numerous papers have tried their algorithms on the Swimmer Dataset with a varying degree of success (or failure). Like the plain NMF model [7], the code called Nonnegative Sparse Coding (NNSC) [8] typically recovers most of 16 limbs together with a shadow torso attached, and has trouble to recover a clean torso without some limb attached. An algorithm called Localized NMF (`LNMF`) [9,57] imposes penalization on the encoding matrix vectors (columns of Y in our notation) and locality constraints on the basis matrix (X in our notation) aiming to extract binary-like, quasi-orthogonal basis images. Algorithm `NMFsc` (NMF with sparseness constraint) [10] adds sparsity constraints to classical NMF using a particular sparseness measure. Algorithm `nsNMF` (Non-smooth NMF) [11] also adds a non-smooth cost function to promote sparseness which then is smoothed with a parameter controlling the degree of smoothness. These algorithms, `LNMF`, `NMFsc` and `nsNMF`, can extract a cleaner torso, but not completely eliminate torso ghosts in limbs. In [58], a rather general framework called Constrained Sparse Matrix Factorization (CSMF) is proposed including a special case of CSMF, called `CSMFnc` that handles nonnegativity. The model Structured Sparse Principal Component Analysis (SSPCA) [12] utilizes a regularization function introduced in [17], which influences not only the number of nonzeros (in X) but also the whereabouts of them. In a recent work [14], the authors propose models called spatial NMF and spatial nonnegative component analysis (Spatial NCA) using so-called pixel dispersion penalty to favor spatially lo-

calized basis images. Utilizing enough properties of the ground truth images, these latest algorithms, `CSMFnc`, `SSPCA`, `Spatial NMF` and `Spatial NCA`, can successfully recover the ground truth basis images (more details of some of these results can be found in [14,58]). Furthermore, the Swimmer Dataset satisfies the *separability assumption* [35]. There exist numerous algorithms for separable and near separable NMF (including Hottopixx [59], SPA [60], XRAY [61] and the new LP model [62], for example). These models can extract most ground truth limbs, but they have difficulty to recover a clean torso because the nonnegative rank of M is 16 instead of 17 (an analysis and more details can be found in [62]). However, it is worth emphasizing that these recovered ground truth basis images are usually not grouped in "an exact order" as is defined above.

For the Swimmer Dataset, the SeMF problem takes the form:

$$\min_{X,Y} \|M - XY\|_F^2 \text{ s.t. } X \in \mathcal{X} \subseteq \mathbb{R}^{1024\times 17},\ Y \in \mathcal{Y} \subseteq \mathbb{R}^{17\times 256}, \tag{32}$$

where definitions of structure sets $\mathcal{X}$ and $\mathcal{Y}$ will depend on what prior information we impose. We mention that, as in all of the previous tests on Swimmer Dataset, all 256 distinct samples are included in the sample matrix M which is 1024×256.

We test our SeMF algorithm on the Swimmer Dataset with several choices of $\mathcal{X}$ and $\mathcal{Y}$. Throughout the tests, we set $maxiter = 2000$, $tol = 10^{-6}$, and initialize penalty parameters to $(\alpha,\, \beta) = (1,\, 1) \times \|M\|_F/100$, while all other parameters are in default setting.

4.4.1 Sparse NMF

We first try the standard sparse NMF: nonnegativity on X and Y plus sparsity on Y, all column-wise. It is known that the number of nonzeros of each Y_j should be 5 (choosing 5 parts from 17 basis). In this case,

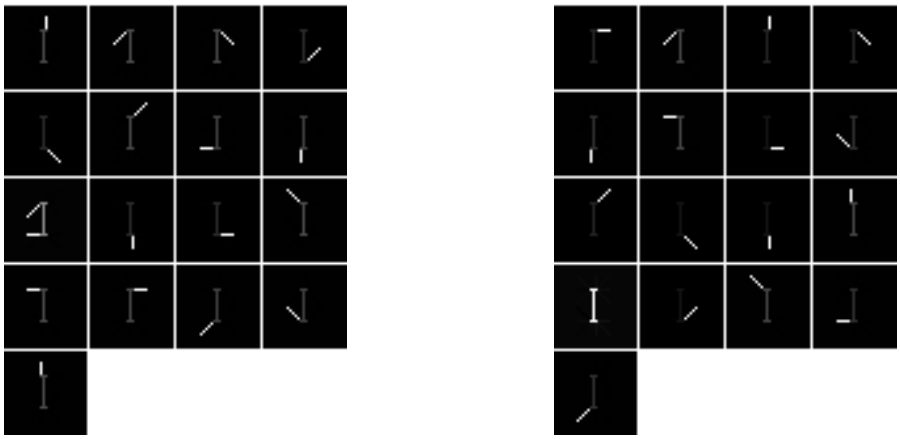
$$\begin{aligned} \mathcal{X} &= \{X : X_{ij} \geq 0, \forall\, i, j\}, \\ \mathcal{Y} &= \{Y : Y_{ij} \geq 0, \|Y_j\|_0 \leq 5, \forall\, j\} \end{aligned} \tag{33}$$

Clearly, both $\mathcal{X}$ and $\mathcal{Y}$ are easily projectable sets (see Section 3). A typical output from our SeMF algorithm is shown in Fig. 10(a). We observe that the outputs are similar to results of any other algorithms when only imposing nonnegativity on X and sparsity or nonnegativity on Y; that is, the recovered torso is not clean and the limbs have ghost torsos attached.

4.4.2 Sparse NMF with equal nonzeros

In the next experiment, we incorporate the information that since the samples are simple sums of the basis images the nonzeros in Y should be equal. Thus we try the following structure sets:

$$\begin{aligned} \mathcal{X} &= \{X : X_{ij} \geq 0, \forall\, i, j\}, \\ \mathcal{Y} &= \{Y : \|Y_j\|_0 = 5, \forall\, j\} \cap (\{Y : Y_{ij} = \alpha_j > 0, \forall\, Y_{ij} \neq 0\} \cup \{0\}) \end{aligned} \tag{34}$$



(a) Sparse NMF (b) Sparse NMF with equal nonzeros

Fig. 10 SeMF Results for structure sets (33) and (34)

which permit easy projections, as is shown in Section 3. With these structure constraints we solve problem (32) by our SeMF algorithm, and plot a typical result in Fig. 10(b). We observe that now a clean torso is recovered, but some limbs still have ghost torso, similar to those results obtained by algorithms like `LNMF` [9, 57], `NMFsc` [10] and `nsNMF` [11]. In addition, we mention that by imposing sparsity on X and requiring the nonzeros in X to be equal, we can also obtain results similar to Fig. 10(b).

4.4.3 Sparse NMF with orthogonality

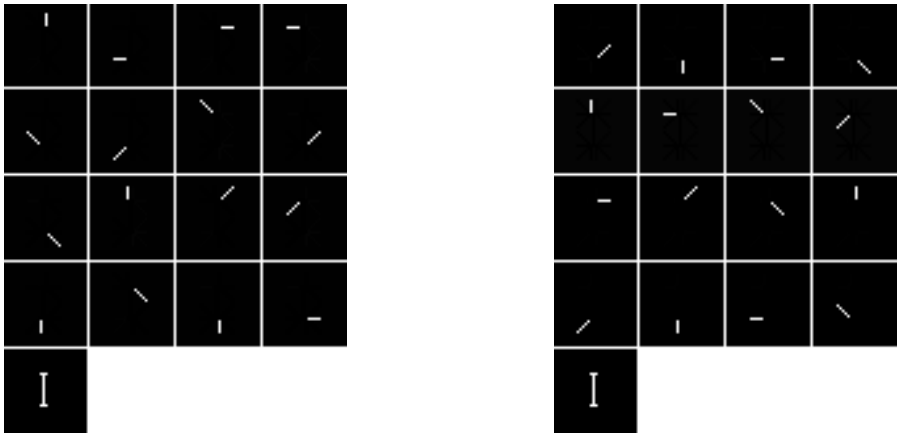
Since the 17 basis images in Swimmer Dataset are non-overlapping, they are mutually orthogonal. We utilize just one piece of such orthogonality information to help improve recoverability. Assuming that the torso is the 17th part, we require that all limbs be orthogonal to it. In addition, we impose a known upper bound on the number of nonzeros in the torso image which happens to be 17 as well. Together, the resulting structure sets are:

$$\begin{aligned} \mathcal{X} &= \{X : X_{ij} \geq 0, \|X_{17}\|_0 \leq 17, X_{1,\cdots,16} \perp X_{17}\}, \\ \mathcal{Y} &= \{Y : Y_{ij} \geq 0, \|Y_j\|_0 \leq 5,\ \forall j\}. \end{aligned} \tag{35}$$

In this case, a closed form of $\mathcal{P}_{\mathcal{X}}$, the projection onto $\mathcal{X}$, is still unknown to us. To approximate $\mathcal{P}_{\mathcal{X}}$, we do a round of successive projections as follows: first nonnegativity projections for all, then sparsity projection for X_{17}, then orthogonality projections for $X_{1,\cdots,16}$, and finally nonnegativity projections for $X_{1,\cdots,16}$ to eliminate any possible violation in nonnegativity. It turns out, as numerical results show, that this approximation to $\mathcal{P}_{\mathcal{X}}$ works adequately well for Swimmer Dataset. A typical output of basis images from our SeMF algorithm is plotted in Fig. 11(a), showing that all 17 basis images are successfully recovered. The quality of the solution is similar to that

obtained by `CSMFnc` [58], `SSPCA` [12], `Spatial NMF` and `Spatial NCA` [14]. (For some reason still unknown to us, the central torso is typically extracted as the 17th part as is planned, even though it is not the only part satisfying the specified structures.)

It is noticeable that the basis images in Fig. 11(a) are not in an exact order, hence there is no group information recovered.



(a) Sparse NMF with structure (35)

(b) Sparse NMF with structure (36)

Fig. 11 SeMF Results for structure sets (35) and (36)

4.4.4 Sparse NMF with combinatorial patterns

Finally we consider the combinatorial structure of Swimmer Dataset discussed earlier in this section. We denote X_{17} as the central torso like in (35), and each 4 columns of $\{X_1, \cdots, X_{16}\}$ as one limb group. Correspondingly, we divide the rows of Y into five groups as well. It is known the 5 non-zeros in every column Y_j are distributed over the 5 groups with one nonzero in each group. As such we have the structure sets:

$$\begin{aligned} \mathcal{X} &= \{X : X_{ij} \geq 0, \|X_{17}\|_0 \leq 17, X_{1,\cdots,16} \perp X_{17}\}, \\ \mathcal{Y} &= \{Y : Y_{ij} \geq 0, \|(Y_j)_{G_t}\|_0 = 1, t = 1, \cdots, 5, \forall\, i, j\} \end{aligned} \tag{36}$$

where $G_t = 4(t-1) + \{1,2,3,4\}, t = 1,2,3,4$, and $G_5 = \{17\}$. We compute the projection $\mathcal{P}_{\mathcal{X}}$ by the same approximation as for (35). The projection $\mathcal{P}_{\mathcal{Y}}$ can be exactly computed as is described in Section 3.3. A set of typical basis images computed by solving (32) with (36) is presented in Fig. 11(b).

It is clear from Fig. 11(b), the recovered basis images are in an exact order, meaning that for each limb the four possible positions appear in the same row. For example, the third row in Fig. 11(b) consists of the 4 different positions for the right-top limb corresponding to Fig. 9(b). To the best of our knowledge, so far there is no other

algorithm that is designed to exploit combinatorial sparsity like our SeMF algorithm does.

As we have purposely alluded to, the results presented are typical but not deterministic. Since we always start from random initial points and the problems are nonconvex, a global minimum is by no means guaranteed in theory, even with out dynamic penalty scheme which often helps escape from local minima in practice. In our repeated testing, we do sometimes obtain less favorable results than those presented in Fig. 10 and Fig. 11. By our estimate, the overall success rate of our SeMF algorithm in all the Swimmer dataset tests is about 90% or higher. In general, the success rate goes up as we add more structural information. For example, we have tried to add the equal nonzero constraint: $Y_{ij} = \alpha_j > 0, \forall\, Y_{ij} \neq 0$ to the structure set $\mathcal{Y}$ in (36) and do a succesive projection approximation. Out of a large number of trials, we have not encountered any failure in getting the results in Fig. 11(b).

5 Concluding Remarks

In summary, we have accomplished the following tasks in this paper.

1. We have devised a versatile algorithmic framework, via the approach of variable-splitting and ADMM, for solving a large class of structure-enforced matrix factorization (SeMF) problems. To apply the algorithm, a user is only required to supply one or two projection functions, either exact or approximate. To tackle the critical issue of penalty parameter selection in ADMM, we have developed an adaptive penalty parameter update scheme that frees a user, at least partially, from the difficult task of selecting and tuning penalty parameters.
2. We have extensively tested our algorithm on several classes of problems. Empirical evidence shows, rather convincingly, that the algorithm is quite effective in solving all the tested problems. Its performance has been found to be competitive with, often favorable to, some existing special-purpose algorithms representing the state of the art.

Structured matrix factorization problems are generally highly nonconvex, and problems in real-world applications can be much more complex and more difficult to solve than the test problems used in this paper. Even though we have not intention to claim that the algorithm presented in this paper can be taken as a out-off-shelf solver for any particular application without further careful work, we do believe that it adds to the toolbox of structured matrix factorization a versatile and useful technique.

Acknowledgements The authors would like to thank two anonymous referees for their many valuable and constructive comments and suggestions that have greatly helped improve the quality and presentation of the paper.

References

1. Paatero, P., Tapper, U.: Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. Environmetrics **5**(2),111–126 (1994)

2. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. Conference on Neural Information Processing Systems (NIPS), pp. 556–562. MIT Press, MIT (2001)
3. Gonzalez, E., Zhang, Y.: Accelerating the lee-seung algorithm for nonnegative matrix factorization. Technical Report TR05-02, CAAM, Rice University (2005)
4. Lin, C.J.: On the convergence of multiplicative update algorithms for nonnegative matrix factorization. IEEE Trans. Neural Netw. **18**(6), 1589–1596 (2007)
5. Albright, R., Cox, J., Duling, D., Langville, A.N., Meyer, C.D.: Algorithms, initializations, and convergence for the nonnegative matrix factorization. NCSU Technical Report, Math 81706 (2006)
6. Lin, C.J.: Projected gradient methods for non-negative matrix factorization. Neural Comput. **19**(10), 2756–2779 (2007)
7. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401**, 788–791 (1999)
8. Hoyer, P.O.: Non-negative sparse coding. Neural Networks for Signal Processing XII (Proc. IEEE Workshop on Neural Networks for Signal Processing), pp. 557–565. Martigny, Switzerland (2002)
9. Feng, T., Li, S.Z., Shum, H.Y., Zhang, H.J.: Local non-negative matrix factorization as a visual representation. In: Proceedings of the 2nd International Conference on Development and Learning, pp. 178–183 (2002)
10. Hoyer, P.O., Dayan, P.: Non-negative matrix factorization with sparseness constraints. J. Mach. Learn. Res. **5**, 1457–1469 (2004)
11. Montano, A.P., Carazo, J.M., Kochi, K., Lehmann, D., Pascual-Marqui, R.D.: Nonsmooth nonnegative matrix factorization (nsnmf). IEEE Trans. Pattern Anal. **28**(3), 403–415 (2006)
12. Jenatton, R., Obozinski, G., Bach, F.: Structured sparse principal component analysis. Int. Conf. Artif. Intell. and Statistics (AISTATS) (2010)
13. Peharz, R., Pernkopf, F.: Sparse nonnegative matrix factorization with ℓ_0-constraints. Neurocomputing **80**, 38–46 (2012)
14. Zheng, W.S., Lai, J.H., Liao, S.C., He, R.: Extracting non-negative basis images using pixel dispersion penalty. Pattern Recogn. **45**(8), 2912–2926 (2012)
15. Aharon, M., Elad, M., Bruckstein, A.: K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Trans. Signal Process. **54**(11), 4311–4322 (2006)
16. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online dictionary learning for sparse coding. In: Proceedings of the 26th Annual International Conference on Machine Learning **8**, 689–696 (2009)
17. Jenatton, R., Audibert, J.Y., Bach, F.: Structured variable selection with sparsity-inducing norms. J. Mach. Learn. Res. **12**, 2777–2824 (2011)
18. Szabo, Z., Poczos, B., Lorincz, A.: Online group-structured dictionary learning. In: Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR) **11**, 2865–2872, Washington, DC (2011)
19. Glowinski, R., Marroco, A.: Sur lapproximation, par elements finis dordre un, et la resolution, par penalisation-dualite dune classe de problemes de dirichlet non lineaires. Revue francaise dautomatique, informatique, recherche operationnelle. Analyse numerique **9**(2), 41–76 (1975)
20. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximation. Comput. Math. Appl. **2**(1), 17–40 (1976)
21. Yang, J., Zhang, Y.: Alternating direction algorithms for L1-problems in compressive sensing. SIAM J. Sci. Comput. **33**(1), 250-278 (2011)
22. Goldstein, T., Osher, S.: The split Bregman method for L1 regularized problems. SIAM J. Imag. Sci. **2**(2), 323–343 (2009)
23. Glowinski, R., Le Tallec, P.: Augmented Lagrangian and operator splitting methods in nonlinear mechanics. SIAM (1989)
24. He, B., Liao, L., Han, D., Yang, H.: A new inexact alternating directions method for monotone variational inequalities. Math. Program. **A**(92), 103-118 (2002)
25. Deng, W., Yin, W.: On the global and linear convergence of the generalized alternating direction method of multipliers. Technical Report TR12-14, CAAM, Rice University (2012)
26. Powell, M.J.D.: A method for nonlinear constraints in minimization problems. Optimization, pp. 283–298. Academic Press, London (1969)
27. Hestenes, M.R.: Multiplier and gradient methods. J. Optimiz. Theory App. **4**, 303–320 (1969)
28. Rockafellar, R.T.: The multiplier method of hestenes and powell applied to convex programming. J. Optimiz. Theory App. **12**, 555–562 (1973)
29. Fortin, M., Glowinski, R.: Augmented Lagrangian methods. North-Holland, Amsterdam (1983)
30. Zhang, Y.: An alternating direction algorithm for nonnegative matrix factorization. Technical Report TR10-03, CAAM, Rice University (2010)

31. He, B.S., Yang, H., Wang, S.L.: Alternating direction method with self adaptive penalty parameters for monotone variational inequalities. J. Optimiz. Theory App. **106**(2), 337-356 (2000)
32. Wen, Z., Goldfarb, D., Yin, W.: Alternating direction augmented Lagrangian methods for semidefinite programming. Math. Prog. Comp., **2**, 203-230 (2010)
33. Lin, Z., Liu, R., Su, Z.: Linearized alternating direction method with adaptive penalty for low rank representation. Advances in Neural Information Processing Systems(NIPS) 24 (2011)
34. Samaria, F.S., Harter, A.C.: Parameterisation of a stochastic model for human face identification. In: Proceedings of the 2nd IEEE Workshop on Applications of Computer Vision, pp. 138–142 (1994)
35. Donoho, D., Stodden, V.: When does non-negative matrix factorization give a correct decomposition into Parts? In: Proceedings of 17th Ann. Conf. Neural Information Processing Systems NIPS (2003)
36. Mallat, S., Zhang, Z.F.: Matching pursuit with time-frequency dictionaries. IEEE Trans. Signal Process. **41**, 3397–3415 (1993)
37. Chen, S., Billings, S.A., Luo, W.: Orthogonal least squares methods and their application to non-linear system identification. Int. J. Control **50**, 1873–1896 (1989)
38. Davis, G.M., Mallat, S.G., Zhang, Z.F.: Adaptive time-frequency decompositions. Opt. Eng. **33**(7), 2183–2191 (1994)
39. Pati, Y.C., Rezaiifar, R., Krishnaprasad, P.S.: Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers **1**, 40–44 (1993)
40. Tropp, J.A.: Greed is good: algorithmic results for sparse approximation. IEEE Trans. Inform. Theory **50**(10), 2231–2242 (2004)
41. Chen, S., Donoho, D., Saunders, M.: Atomic decomposition by basis pursuit. SIAM J. Sci. Comput. **20**, 33–61 (1998)
42. Gorodnitsky, I.F., Rao, B.D.: Sparse signal reconstruction from limited data using focuss: A re-weighted norm minimization algorithm. IEEE Trans. Signal Process. **45**, 600–616 (1997)
43. Rao, B.D., Kreutz-Delgado, K.: An affine scaling methodology for best basis selection. IEEE Trans. Signal Process. **47**(1), 187–200 (1999)
44. Rao, B.D., Kreutz-Delgado, K.: Deriving algorithms for computing sparse solutions to linear inverse problems. IEEE Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers **1**, 955–959 (1998)
45. Rao, B.D., Engan, K., Cotter, S.F., Palmer, J., Kreutz-Delgado, K.: Subset selection in noise based on diversity measure minimization. IEEE Trans. Signal Process. **51**(3), 760–770 (2003)
46. Gersho, A., Gray, R.M.: Vector Quantization and Signal Compression. Kluwer Academic Publishers, Norwell, MA (1991)
47. Lewicki, M.S., Olshausen, B.A.: A probabilistic framework for the adaptation and comparison of image codes. J. Opt. Soc. Am. A: Optics, Image Science and Vision **16**(7), 1587–1601 (1999)
48. Olshausen, B.A., Field, D.J.: Natural image statistics and efficient coding. Network-Comput. Neural **7**(2), 333–339 (1996)
49. Olshausen, B.A., Field, B.J.: Sparse coding with an overcomplete basis set: A strategy employed by v1? Vision Res. **37**, 3311–3325 (1997)
50. Lewicki, M.S., Sejnowski, T.J.: Learning overcomplete representations. Neural Comput. **12**, 337–365 (2000)
51. Engan, K., Aase, S.O., Husϕy, J.H.: Multi-frame compression: Theory and design. EURASIP Signal Processing **80**(10), 2121–2140 (2000)
52. Engan, K., Aase, S.O., Hakon-Husoy, J.H.: Method of optimal directions for frame design. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing **5**, 2443–2446 (1999)
53. Engan, K., Rao, B.D., Kreutz-Delgado, K.: Frame design using focuss with method of optimal directions (MOD). Norwegian Signal Processing Symposium, pp. 65-69 (1999)
54. Kreutz-Delgado, K., Murray, J.F., Rao, B.D., Engan, K., Lee, T., Sejnowski, T.J.: Dictionary learning algorithms for sparse representation. Neural Comput. **15**(2), 349–396 (2003)
55. Murray, J.F., Kreutz-Delgado, K.: An improved focuss-based learning algorithm for solving sparse linear inverse problems. IEEE Int, Conf. on Signals, Systems and Computers **1**, 347–351 (2001)
56. Kreutz-Delgado, K., Rao, B.D.: Focuss-based dictionary learning algorithms. Wavelet Applications in Signal and Image Processing VIII **4119**, 1-53 (2000)
57. Li, S.Z., Hou, X.W., Zhang, H.J., Cheng, Q.S.: Learning spatially localized, parts-based representation. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR2001) **1**, 207–212 (2001)
58. Zheng, W.S., Li, S.Z., Lai, J.H., Liao, S.C.: On constrained sparse matrix factorization. IEEE 11th Int. Conf. Comp. Vis. (ICCV2007), pp. 1–8 (2007)

59. Bittorf, V., Recht, B., Ré, E., Tropp, J.A.: Factoring nonnegative matrices with linear programs. In Advances in Neural Information Processing Systems (NIPS '12), pp. 1223–1231 (2012)
60. Araújo, U.M.C., Saldanha, B.T.C., Galv̄ao, R.K.H., Yoneyama, T., Chame, H.C., Visani, V.: The successive projections algorithm for variable selection in spectroscopic multicomponent analysis. Chemometrics and Intelligent Laboratory Systems **57**(2), 65–73 (2001)
61. Kumar, A., Sindhwani, V., Kambadur, P.: Fast conical hull algorithms for near-separable non-negative matrix factorization. In Int. Conf. on Machine Learning (ICML '13) **28**, 231–239 (2013)
62. Gillis, N., Luce, R.: Robust near-separable nonnegative matrix factorization using linear optimization. Journal of Machine Learning Research **15**(Apr), 1249–1280 (2014)