Image Classification with Nonnegative Matrix Factorization Based on Spectral Projected Gradient

Rafał Zdunek, Anh Huy Phan, and Andrzej Cichocki

Abstract. Nonnegative Matrix Factorization (NMF) is a key tool for model dimensionality reduction in supervised classification. Several NMF algorithms have been developed for this purpose. In a majority of them, the training process is improved by using discriminant or nearest-neighbor graph-based constraints that are obtained from the knowledge on class labels of training samples. The constraints are usually incorporated to NMF algorithms by l_2-weighted penalty terms that involve formulating a large-size weighting matrix. Using the Newton method for updating the latent factors, the optimization problems in NMF become large-scale. However, the computational problem can be considerably alleviated if the modified Spectral Projected Gradient (SPG) that belongs to a class of quasi-Newton methods is used. The simulation results presented for the selected classification problems demonstrate the high efficiency of the proposed method.

1 Introduction

Nonnegative Matrix Factorization (NMF) [20] decomposes a nonnegative matrix into lower-rank factor matrices that have nonnegative entries and usually some physical meaning. When NMF is applied to the matrix of training samples, we obtain sparse nonnegative feature vectors and coefficients of their nonnegative combinations. The vectors of the coefficients, which are here referred to as encoding vectors, lie in a low-dimensional latent component space. Hence, NMF is often

Rafał Zdunek
Department of Electronics, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
e-mail: rafal.zdunek@pwr.wroc.pl

Anh Huy Phan · Andrzej Cichocki
Laboratory for Advanced Brain Signal Processing
RIKEN BSI, Wako-shi, Japan
e-mail: {phan,cia}@brain.riken.jp

Warsaw University of Technology, Poland and Systems Research Institute,
Polish Academy of Science (PAN), Poland

© Springer International Publishing Switzerland 2015
P. Koprinkova-Hristova et al. (eds.), *Artificial Neural Networks*,
Springer Series in Bio-/Neuroinformatics 4, DOI: 10.1007/978-3-319-09903-3_2

regarded as a dimensionality reduction technique, and it has been widely applied for classification of various objects [2, 6, 12, 29, 33].

As reported in [8], the factor matrices obtained with NMF are generally non-unique. Several attempts have been done to impose additional constraints on the estimated factors. The sparsity constraints are probably the most frequently used. It is well-known that the factors should be somehow sparse to expect the uniqueness [16]. Hoyer [14] demonstrated that the sparsity can be readily controlled in NMF by the l_1-norm based penalty term in an objective function. When l_2-norm is used instead, the smoothness is enforced in one factor, and intrinsically the sparsity level increases in the other [26]. The sparse NMF that minimizes the l_p-diversity measure was proposed in [37]. This approach was also developed by Kim and Park in [17] using the active-set minimization strategy. The sparsity constraints were also analyzed in the context of supervised classification. Li *et al* [21] proposed the Local NMF (LNMF) that combines several kinds of constraints. It enforces orthogonality between basis vectors and selects the most meaningful encoding vectors. As a result, the basis vectors contain more localized features, which usually leads to better classification results.

The training process can be also improved by using the Fisher's discriminant information. The examples include hybrid methods, such as FNMF (*Fisher NMF*) or DNMF (*Discriminant NMF*), which combine NMF with FDA (*Fisher's Discriminant Analysis*) or LDA (*Linear Discriminant Analysis*). The first was proposed by Wang *et al.* [31], and the second developed by Zafeiriou *et al.* [33]. In these hybrids, the penalty terms are constructed in such a way to minimize the inner-class scattering and simultaneously to maximize intra-class scattering. The inner-class scatter models the dispersion of vectors that belong to the same class around their mean, while the intra-class scatter expresses the distances of the local class means from the global mean. As a result, the well-known Fisher discriminant criterion is maximized. Both FNMF and DNMF are based on the multiplicative algorithm that minimizes the penalized Kullback-Leibler (KL) divergence.

Another group contains the NMF algorithms which explore a geometrical structure of observed data. Cai *et al.* [4, 5] noticed that samples in a latent space lie on a low-dimensional manifold embedded in a high-dimensional ambient space. Thus, the projection from a high-dimensional observation space to a low-dimensional latent space should preserve a data geometrical structure. The neighboring samples should belong to the same class in both spaces. Thus, they proposed Graph regularized NMF (GNMF) [5] that constrains one of the factor matrices with the information on the data geometric structure encoded in a nearest-neighbor graph of training samples. This constraint was imposed to NMF by a specifically designed regularization term in an objective function that was then minimized with the standard multiplicative algorithm. This approach combines NMF with the Laplacian eigenmaps [1] and locality-preserving projections [13]. Yang *et al.* [32] divided the coefficients of the encoding vectors into two parts to which the discriminant information is incorporated accordingly. The first part is enforced by the intrinsic graph that characterizes the favorite relationships among the training data. The other is affected by the penalty graph that expresses the unfavorable relationships. This approach was then computationally improved and extended to tensor decompositions by Wang *et al.* [30], and to the projective NMF by Liu *et al.* [24].

An interesting property of nearly all the above mentioned learning methods is the common way of introducing the prior knowledge to the training process. The penalty terms in the regularized objective function can be expressed in terms of the weighted Frobenius norm. The weighting matrix reflects the group sparsity, discriminant information or graph-based embedding. Its size is usually equal to the number of training samples, so it is large and usually sparse matrix.

The penalty term expressed by the the weighted Frobenius norm can be easily considered in an optimization process when the multiplicative algorithm is used. Hence, this approach is explored in a large number of research papers on NMF. However, multiplicative algorithms are slowly convergent, and in the basic version they do not guarantee convergence to a stationary point. Using the simple numerical improvements, as proposed in [22], the algorithms can be numerically stable but the slow convergence is still an open problem.

To tackle the convergence problems, several other computational strategies have been proposed in the literature. Guan *et al.* [10] considerably accelerated the convergence of GNMF by using additive quasi-Newton gradient descent updates. In the next paper, Guan *et al.* [11] proposed the NeNMF that is based on the Nesterovs optimal gradient approach. This method can be used for minimization of the above-mentioned penalized objective functions, provided that the Lipschitz constant can be easily calculated. A simple version of the Projected Gradient (PG) algorithm was also used for classification problems in [18].

The optimization problems that include the weighted Frobenius norm-based penalty terms can be also efficiently solved using the Hierarchical Alternating Least Squares (HALS) algorithm [6]. Phan *et al.* [27, 28] applied the HALS algorithm to obtain multi-way array decomposition with higher-order discriminant analysis. A similar computational strategy, known as the Sequential Coordinate-Wise (SCW) [9], was used in [38] for updating lateral factors in DNMF. As a result, the SCW-DNMF substantially outperforms the basic DNMF and LNMF algorithms but at the cost of an increased computational complexity.

GNMF can be also efficiently obtained with the Spectral Gradient Projection (SPG) method. This approach was proposed in [39] in the context of facial image classification. The SPG [25] belongs to a class of quasi-Newton methods. It approximates the Hessian matrix with the scalar that is estimated from the secant equation, separately for each column vector of the encoding matrix. Such computations can be easily parallelized, which leads to a high performance with a low computational cost. Unfortunately, the steplengths in gradient descent updates cannot be so easily determined for the penalty terms used in GNMF. Hence, the Armijo rule was used in [39], similarly as in the PG NMF algorithm proposed by Lin [23].

In this chapter, we extend the SPG algorithm discussed in [39] in several aspects: (1) we propose a better computational strategy for estimating the steplengths, separately for each column vector of the encoding matrix; (2) we adapt this algorithm for solving generalized weighted Frobenius norm-based penalty terms, including sparsity and discriminant information; (3) we present the results for more classification problems and demonstrate the efficiency of using preprocessing based on the wavelet transform.

The chapter is organized in the following way. The next section discusses the penalty terms in NMF that are expressed by the weighted Frobenius norm. Section 3 is concerned with the optimization algorithms. The numerical experiments for image classification problems are presented in Section 4. Finally, the conclusions are drawn in Section 5.

2 Penalty Terms

Let $Y = [y_1, \dots, y_T] \in \mathbb{R}_+^{I \times T}$, where $y_t \in \mathbb{R}_+^I$ is the t-th training sample. Applying NMF to Y, we get $Y \cong AX$, where the columns of the matrix $A \in \mathbb{R}_+^{I \times J}$ represent the feature or basis vectors, and the columns of the matrix $X \in \mathbb{R}_+^{J \times T}$ are encoding vectors. The parameter J is the rank of factorization.

In several variants of NMF, the objective function can be expressed by the quadratic function:

$$\begin{aligned}\Psi(A,X) &= \frac{1}{2}||Y - AX||_F^2 + \frac{\alpha_A}{2}\,\mathrm{tr}(A^T L_A A) + \frac{\alpha_X}{2}\,\mathrm{tr}(X L_X X^T) \\ &= \frac{1}{2}||Y - AX||_F^2 + \frac{\alpha_A}{2}||L_A^{\frac{1}{2}} A||_F^2 + \frac{\alpha_X}{2}||X L_X^{\frac{1}{2}}||_F^2, \qquad (1)\end{aligned}$$

where $\frac{\alpha_A}{2}||L_A^{\frac{1}{2}} A||_F^2$ and $\frac{\alpha_X}{2}||X L_X^{\frac{1}{2}}||_F^2$ are *penalty terms*, expressed in terms of the weighted Frobenius norm, and $\alpha_A, \alpha_X \geq 0$ are penalty parameters that control the amounts of introduced *a priori* information. The weighting matrices $L_A \in \mathbb{R}^{I \times I}$ and $L_X \in \mathbb{R}^{T \times T}$ are symmetric and nonnegative definite. Both matrices are determined on the basis of the prior knowledge on the estimated factors. The matrix L_A enforces column profiles in the matrix A, and L_X – row profiles in X. Below we present a short survey of the typical penalty terms.

2.1 Sparse NMF

The sparsity in the factor X can be modeled in many ways, e.g. by the l_p-diversity measure [7]: $J^{(p,q)} = \sum_{j=1}^J (||\underline{x}_j||_q)^p$ for $p \geq 0, q \geq 1$, where $\underline{x}_j$ is the j-th row vector of X. In [37], we assumed $q = 1$ and $p = 2$ to enforce sparsity in the columns of X. Note that the sparsity can be also enforced in the rows, if $q = 2$ and $p = 1$. This leads to the term: $J^{(1,2)} = \sum_{j=1}^J (||\underline{x}_j||_1)^2 = \mathrm{tr}\{X E_T X^T\}$, where $E_T \in \mathbb{R}^{T \times T}$ is a matrix of all ones. Considering (1), we have $L_X = E_T$. The sparsity can be also modeled in a similar way in the column vectors of A, which leads to $L_A = E_I \in \mathbb{R}^{I \times I}$. Such sparsity measures were also used in the SNMF/L and SNMF/R algorithms [17].

Applying the Hoyer's sparsity measure [15] to the rows of X, we have:

$$\sigma = \frac{\sqrt{T} - \frac{||\underline{x}_j||_1}{||\underline{x}_j||_2}}{\sqrt{T} - 1}. \tag{2}$$

Following [19], this measure can be approximated by the additive form:

$$J = \frac{1}{2} \sum_{j=1}^{J} ||\underline{x}_j||_1^2 - ||\underline{x}_j||_2^2 = \mathrm{tr}\{X L_X X^T\}, \tag{3}$$

where $L_X = E_T - I_T$, where $I_T \in \mathbb{R}^{T \times T}$ is an identity matrix. Similarly, $L_A = E_I - I_I$. In this case, $\mathrm{rank}(L_X) = T$, which leads to better numerical properties than for the previous case ($\mathrm{rank}(E_T) = 1$).

2.2 DNMF

The objective function in DNMF [31, 33] has the following form:

$$\Psi(A, X) = D_{KL}(Y||AX) + \gamma \mathrm{tr}\{S_X\} - \delta\, \mathrm{tr}\{S_{\bar{X}}\}, \tag{4}$$

where $D_{KL}(Y||AX)$ is the KL divergence between Y and AX. The matrices S_X and $S_{\bar{X}}$ represent the inner- and intra-scattering, respectively. They are defined in the following way:

$$S_X = \sum_{k=1}^{K} \sum_{t_k=1}^{|\mathcal{N}_k|} (x_{t_k} - \bar{x}_k)(x_{t_k} - \bar{x}_k)^T, \tag{5}$$

$$S_{\bar{X}} = \sum_{k=1}^{K} |\mathcal{N}_k| (\bar{x}_k - \bar{x})(\bar{x}_k - \bar{x})^T, \tag{6}$$

where K is the number of classes, $\mathcal{N}_k$ is the set of indices of the samples x_t that belong to the k-th class, $|\mathcal{N}_k|$ is the number of samples in the k-th class, x_{t_k} is the t_k-th image in the k-th class, $\bar{x}_k$ is the mean vector of the k-th class, and $\bar{x}$ is the mean vector over all the column vectors in X.

Let $C = [c_{st}] \in \mathbb{R}^{T \times T}$, where

$$c_{st} = \begin{cases} \frac{1}{\mathcal{N}_k} & \text{if } (s,t) \in \mathcal{N}_k \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

for $k = 1, \ldots, K$. Considering (7), the matrix S_X in (5) can be rearranged as follows:

$$S_X = (X - XC)(X - XC)^T = X L_X^{(1)} X^T, \tag{8}$$

where $L_X^{(1)} = (I_T - C)(I_T - C)^T$. Similarly, the matrix $S_{\bar{X}}$ in (6) can be rewritten as

$$S_{\bar{X}} = (XC - X\tilde{E}_T)(XC - X\tilde{E}_T)^T = XL_X^{(2)}X^T, \tag{9}$$

where $\tilde{E}_T = \frac{1}{T}11^T$, $1 = [1,1,\ldots,1]^T \in \mathbb{R}^T$, and $L_X^{(2)} = (C - \tilde{E}_T)(C - \tilde{E}_T)^T$.

Thus, the penalty terms in (4) can be reformulated as

$$\gamma \mathrm{tr}\{S_X\} - \delta\, \mathrm{tr}\{S_{\bar{X}}\} = \mathrm{tr}\{XL_XX^T\} = ||XL_X^{\frac{1}{2}}||_F^2, \tag{10}$$

where $L_X = \gamma L_X^{(1)} - \delta L_X^{(2)}$. The penalty term in (10) can be combined not only with the KL divergence but also with other disimilarity measures. In [39], this modeling was used for deriving the SCW-DNMF algorithm.

2.3 GNMF

GNMF [5] integrates NMF with the Laplacian eigenmaps [1] and locality-preserving projections [13]. The matrix L_X in GNMF is expressed by the graph Laplacian matrix that represents a data geometrical structure in the observation space. It takes form: $L_X = D - W$, where $W = [w_{nm}] \in \mathbb{R}_+^{T\times T}$ contains the entries that determine the edges in the nearest neighbor graph of the observed points, and $D = \mathrm{diag}\left(\sum_{m\neq n} w_{nm}\right) \in \mathbb{R}_+^{T\times T}$. The edges can be determined by the hard connections:

$$w_{nm} = \begin{cases} 1, \text{ if } & y_n \in \mathcal{N}_p(y_m), \text{ or } y_m \in \mathcal{N}_p(y_n), \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where $\mathcal{N}_p(y_t)$ is the p nearest neighbor of the sample y_t. We can also use the Heat kernel weighting:

$$w_{nm} = \begin{cases} \exp\left\{-\frac{||y_n - y_m||_2^2}{2\sigma^2}\right\}, \text{ if } & y_n \in \mathcal{N}_p(y_m), \text{ or } y_m \in \mathcal{N}_p(y_n), \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

or the cosine measure:

$$w_{nm} = \begin{cases} \frac{y_n^T y_m}{||y_n||||y_m||}, \text{ if } & y_n \in \mathcal{N}_p(y_m), \text{ or } y_m \in \mathcal{N}_p(y_n), \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

The graph-based regularization helps to preserve the data geometrical structure in the low dimensional latent space that contains the samples $\{x_t\}$. If any two samples y_{t_1} and y_{t_2} belong to one cluster, the corresponding samples x_{t_1} and x_{t_2} should also belong to the same cluster. Thus, this approach seems to be very useful for clustering but not necessarily for classification problems. For the latter, the discriminant information is more relevant.

The discriminant constraints can be also combined with the graph-based ones. In this approach, the graph Laplacian matrices can be defined separately for the inner- and intra-class samples. Guan *et al.* [10] models k_1 nearest-neighbor inner class samples with the graph Laplacian matrix $L_{(inner)}$, and k_2 nearest-neighbor intra class samples with the graph Laplacian matrix $L_{(intra)}$. The Laplacian eigenmaps can be combined in the following way:

$$L_X = \left(\tilde{L}_{(intra)}^{-\frac{1}{2}}\right)^T L_{(inner)} \tilde{L}_{(intra)}^{-\frac{1}{2}}, \tag{14}$$

where $\tilde{L}_{(intra)} = L_{(intra)} + \xi I$. The regularization parameter $\xi > 0$ should be selected in such a way to guarantee the inversion of $L_{(intra)}$. Note that the parameter ξ also controls a ratio of inner-to-intra class information. Hence it has a discriminant property, and can be treated as a penalty parameter.

Note that the matrices $L_X^{(1)}$ and $L_X^{(2)}$ given in Section 2.2 can be also integrated using the formula in (14). In this approach, the parameter ξ refers to the ratio $\frac{\gamma}{\delta}$.

3 Algorithm

The penalty term $||XL_X^{\frac{1}{2}}||_F^2$ in (1) can be reformulated as follows:

$$\Psi_r(X) = ||XL_X^{\frac{1}{2}}||_F^2 = ||(L_X^{\frac{1}{2}} \otimes I_J)x||_2^2 = x^T(L_X \otimes I_J)x, \tag{15}$$

where $x = \text{vec}(X) \in \mathbb{R}^{JT}$ is a vectorized form of X, and $\otimes$ stands for the Kronecker product.

Assuming $L_A = I_I$, the Hessian matrices of $\Psi(A,X)$ in (1) with respect to A and X have the following forms:

$$H_A = \nabla_A^2 \Psi(A,X) = (XX^T + \alpha_A I_J) \otimes I_I \in \mathbb{R}^{IJ \times IJ}, \tag{16}$$

$$H_X = \nabla_X^2 \Psi(A,X) = I_T \otimes A^T A + \alpha_X L_X \otimes I_J \in \mathbb{R}^{JT \times JT}. \tag{17}$$

For $\alpha_A > 0$, the matrix H_A is positive definite. Under the assumption of positive definiteness of the matrix L_X, the matrix H_X is also symmetric and positive definite.

The matrix H_A has a block-diagonal structure, and hence the updates of A might be considerably accelerated by transforming the nonnegative least-squares problem: $\min_{A \geq 0} \frac{1}{2}||Y - AX||_F^2 + \frac{\alpha_A}{2}||A||_F^2$ to the normal equations $XX^TA^T = XY^T$ subject to the nonnegativity constraints $A \geq 0$. Then, the solution can be efficiently searched with any iterative solver for solving this kind of problems, e.g. HALS [6], PG [23], FCNNLS [17], etc.

Remark 1. Assuming $(XX^T)^{-1}$ is calculated with the $O(J^3)$ complexity, then one iterative step of the Newton method for updating the matrix A entails the complexity about $O(IJT) + O(J^3) + O(IJ^2)$. Assuming $J << \min\{I, T\}$, an overall complexity is dominated by $O(IJT)$.

Remark 2. The Hessian H_X is no longer a block-diagonal matrix, hence the Newton updates for X cannot be simplified considerably. The computational complexity of one step of the Newton algorithm is $O(IJT) + O(J^3T^3) + O(JT)$. Thus it is strongly affected by the computational cost of the Hessian inverse.

The updates for X cannot be accelerated in the similar way, however, there is still a possibility of applying some quasi-Newton method without formulating the Hessian H_X directly. Note that the matrix H_X is very large when the number of training samples is large, and it is rather a dense matrix due to the matrix L_X. One of these possibilities is to use the SPG method [3] that combines the standard gradient projection scheme with the nonmonotonic Barzilai-Borwein (BB) method [25]. It is used for minimization of convex functions subject to box-constraints.

In the SPG method, the descent direction $p_t^{(k)}$ for updating the vector x_t in the k-th iteration is defined as follows:

$$p_t^{(k)} = \left[x_t^{(k)} - (\alpha_t^{(k)})^{-1} \nabla_{x_t} \Psi(A, x_t^{(k)})\right]_+ - x_t^{(k)}, \tag{18}$$

for $\alpha_t^{(k)} > 0$ selected in such a way that the matrix $\alpha_t^{(k)} I_J$ approximates the Hessian matrix.

In [6], this method was adopted to parallel processing of all column vectors in X. Using this approach, we have the update rule:

$$X^{(k+1)} = X^{(k)} + P^{(k)} Z^{(k)}, \tag{19}$$

where $Z^{(k)} = \mathrm{diag}\{\eta^{(k)}\}$. The column vectors of $P^{(k)} \in \mathbb{R}^{J \times T}$ and the entries of the vector $\eta^{(k)} \in \mathbb{R}_+^T$ are descent directions and steplengths for updating the vectors $\{x_t\}$, respectively. According to (18), the matrix $P^{(k)}$ has the form:

$$P^{(k)} = \left[X^{(k)} - G_X^{(k)} D^{(k)}\right]_+ - X^{(k)}, \tag{20}$$

where $G_X^{(k)} = \nabla_X \Psi(A, X^{(k)}) \in \mathbb{R}^{J \times T}$ and $D^{(k)} = \mathrm{diag}\{(\alpha_t^{(k)})^{-1}\} \in \mathbb{R}^{T \times T}$.

The coefficients $\{\alpha_t^{(k)}\}$ can be obtained from the secant equation that is given by $S^{(k)} \mathrm{diag}\{\alpha_t^{(k+1)}\} = W^{(k)}$, where $S^{(k)} = X^{(k+1)} - X^{(k)}$ and $W^{(k)} = \nabla_X \Psi(A, X^{(k+1)}) - \nabla_X \Psi(X^{(k)})$. For the minimization of the objective function (1) with respect to X, the matrix $W^{(k)}$ takes the form: $W^{(k)} = A^T A S^{(k)} + \alpha_X S^{(k)} L_X$. From (19) we have: $S^{(k)} = P^{(k)} Z^{(k)}$. In consequence, the secant equation leads to:

$$\begin{aligned}
\alpha^{(k+1)} &= \frac{\operatorname{diag}\left\{(S^{(k)})^T W^{(k)}\right\}}{\operatorname{diag}\left\{(S^{(k)})^T S^{(k)}\right\}} = \frac{\operatorname{diag}\left\{(S^{(k)})^T A^T A S^{(k)} + \alpha_X (S^{(k)})^T S^{(k)} L_X\right\}}{\operatorname{diag}\left\{(S^{(k)})^T S^{(k)}\right\}} \\
&= \frac{\operatorname{diag}\left\{(P^{(k)})^T A^T A P^{(k)} + \alpha_X (P^{(k)})^T P^{(k)} Z^{(k)} L_X (Z^{(k)})^{-1}\right\}}{\operatorname{diag}\left\{(P^{(k)})^T P^{(k)}\right\}} \\
&= \frac{1_J^T \left[P^{(k)} \circledast \left(A^T A P^{(k)} + \alpha_X P^{(k)} Z^{(k)} L_X (Z^{(k)})^{-1}\right)\right]}{1_J^T \left[P^{(k)} \circledast P^{(k)}\right]}, \qquad (21)
\end{aligned}$$

where $\circledast$ stands for the Hadamard product, and the operation $\operatorname{diag}\{M\}$ creates a vector containing the main diagonal entries of a matrix M. Note that the matrix $Z^{(k)}$ is diagonal, so the product $Z^{(k)} L_X (Z^{(k)})^{-1}$ can be readily calculated.

The steplengths can be estimated by solving the minimization problem:

$$\eta_*^{(k)} = \arg\min_{\eta^{(k)}} \Psi\left(A, X^{(k)} + P^{(k)} \operatorname{diag}\{\eta^{(k)}\}\right). \qquad (22)$$

For $\alpha_X = 0$, the objective function $\Psi(A, X^{(k+1)})$ takes the form:

$$\begin{aligned}
\Psi(A, X^{(k+1)}) &= \frac{1}{2}||Y - A(X^{(k)} + P^{(k)} D_\eta^{(k)})||_F^2 = \frac{1}{2}\operatorname{tr}\left\{D_\eta^{(k)} (P^{(k)})^T A^T A P^{(k)} D_\eta^{(k)}\right\} \\
&\quad - \operatorname{tr}\left\{(Y - AX^{(k)})^T A P^{(k)} D_\eta^{(k)}\right\} + \text{const}, \qquad (23)
\end{aligned}$$

where $D_\eta^{(k)} = \operatorname{diag}\{\eta^{(k)}\}$. From the stationarity of $\Psi(A, X)$ with respect to X, we have:

$$\frac{\partial}{\partial \eta}\Psi(A, X^{(k+1)}) = \operatorname{diag}\left\{(P^{(k)})^T A^T A P^{(k)} D_\eta^{(k)}\right\} + \operatorname{diag}\left\{(G_X^{(k)})^T P^{(k)}\right\} \triangleq 0. \quad (24)$$

Hence, the solution to (22) can be presented in a closed-form:

$$\eta_*^{(k)} = -\frac{\operatorname{diag}\left\{(G_X^{(k)})^T P^{(k)}\right\}}{\operatorname{diag}\left\{(P^{(k)})^T A^T A P^{(k)} D_\eta^{(k)}\right\}} = -\frac{1_J^T \left[G_X^{(k)} \circledast P^{(k)}\right]}{1_J^T \left[P^{(k)} \circledast (A^T A P^{(k)})\right]}. \qquad (25)$$

For $\alpha_X > 0$, we have:

$$\begin{aligned}
\Psi(A, X^{(k+1)}) &= \frac{1}{2}||Y - A(X^{(k)} + P^{(k)} D_\eta^{(k)})||_F^2 + \frac{\alpha_X}{2}||(X^{(k)} + P^{(k)} D_\eta^{(k)}) L_X^{\frac{1}{2}}||_F^2 \\
&= \frac{1}{2}\operatorname{tr}\left\{D_\eta^{(k)} (P^{(k)})^T A^T A P^{(k)} D_\eta^{(k)}\right\} - \operatorname{tr}\left\{(Y - AX^{(k)})^T A P^{(k)} D_\eta^{(k)}\right\} \\
&\quad + \operatorname{tr}\left\{X^{(k)} L_X D_\eta^{(k)} (P^{(k)})^T\right\} + \frac{1}{2}\operatorname{tr}\left\{P^{(k)} D_\eta^{(k)} L_X D_\eta^{(k)} (P^{(k)})^T\right\} \\
&\quad + \text{const}. \qquad (26)
\end{aligned}$$

Thus

$$\begin{aligned}\frac{\partial}{\partial \eta}\Psi(A,X^{(k+1)}) &= \operatorname{diag}\left\{(P^{(k)})^T A^T A P^{(k)} D_\eta^{(k)}\right\} + \operatorname{diag}\left\{(G_X^{(k)})^T P^{(k)}\right\} \\ &+ \operatorname{diag}\left\{(P^{(k)})^T X^{(k)} L_X\right\} + \frac{1}{2}\operatorname{tr}\left\{(P^{(k)})^T P^{(k)} D_\eta^{(k)} L_X\right\} \\ &+ \frac{1}{2}\operatorname{tr}\left\{L_X D_\eta^{(k)} (P^{(k)})^T P^{(k)}\right\} \triangleq 0. \qquad (27)\end{aligned}$$

From (27), we have:

$$\begin{aligned}\operatorname{diag}\left\{\operatorname{diag}\{(P^{(k)})^T A^T A P^{(k)}\}\right\}\eta_*^{(k)} &+ \left(\left[(P^{(k)})^T P^{(k)}\right] \circledast L_X\right)\eta_*^{(k)} \\ &= -\operatorname{diag}\left\{(G_X^{(k)})^T P^{(k)} + (P^{(k)})^T X^{(k)} L_X\right\},\end{aligned}$$

which leads to the following system of linear equations:

$$\tilde{T}^{(k)}\eta_*^{(k)} = -b^{(k)}, \qquad (28)$$

where $\tilde{T}^{(k)} = T^{(k)} + \tilde{\xi} I_T$ for $\tilde{\xi} > 0$,

$$T^{(k)} = \left[(P^{(k)})^T P^{(k)}\right] \circledast L_X + \operatorname{diag}\left\{1_I^T (AP^{(k)})^2\right\}, \qquad (29)$$

$$b^{(k)} = 1_J^T\left(P^{(k)} \circledast (P^{(k)} + X^{(k)} L_X)\right). \qquad (30)$$

The matrix $T^{(k)}$ in (28) is symmetric and at least nonnegative definite. Introducing a small positive constant $\tilde{\xi}$, the positive definiteness is enforced, which allows us to use the Cholesky factorization to solve the system (28). Using the Matlab's function $[R,p] = \texttt{chol}(\tilde{T}^{(k)})$, we can easily control the positive definiteness. If $p = 0$, the matrix $\tilde{T}^{(k)}$ is positive definite, and $R^T R = \tilde{T}^{(k)}$. When $p > 0$, the parameter $\tilde{\xi}$ should be set up to a positive value.

The solution $\eta_*^{(k)}$ must satisfy the box constraints: $0 < \eta_*^{(k)} \leq 1$. Hence, the update for η in the k-th iterative step is determined by $\eta^{(k)} = \max\{\varepsilon, \min\{1, \eta_*^{(k)}\}\}$, where $\eta_*^{(k)} = -R\backslash(R^T\backslash b^{(k)})$ for a small positive constant ε. The operator $\backslash$ denotes the back-substitution.

The final form of the modified SPG algorithm is given by Algorithm 1. It is a fundamental part of the NMF algorithm used in the training process (see Algorithm 2).

In the training process, we obtain the nonnegative matrices A and X. To classify the test sample $\tilde{y}$, first we need to project it onto the subspace spanned by the column vectors of the matrix A. As a result, we obtain $\tilde{x} \in \mathbb{R}_+^J$. This step can be carried out with the SPG, assuming $\alpha_X = 0$. Then, the following problem is solved: $t_* =$

Algorithm 1. SPG algorithm

```
   Input : $Y \in \mathbb{R}_+^{I\times T}$, $A \in \mathbb{R}_+^{I\times J}$, $X^{(0)} \in \mathbb{R}_+^{J\times T}$ - initial guess, $L_X \in \mathbb{R}^{T\times T}$, $\alpha_{min} > 0$,
           $\alpha_{max} > 0$, $\forall t : \bar{\alpha}_t^{(0)} = \frac{1}{2}\alpha_{max}$, $\tilde{\xi} = 10^{-12}$, $k = 0$,
   Output: $\hat{X}$ - estimated factor matrices,
1  repeat
2      $k \leftarrow k+1$;
3      $G_X^{(k)} = \nabla_X \Psi(A, X^{(k)}) = A^T(AX^{(k)} - Y) + \alpha_X X^{(k)} L_X$ ;          // Gradient
4      $P^{(k)} = \left[X^{(k)} - G_X^{(k)} \mathrm{diag}\{(\bar{\alpha}_t^{(k)})^{-1}\}\right]_+ - X^{(k)}$ ;    // Descent direction
5      $[R, p] = \mathtt{chol}(T^{(k)})$ ;                    // where $T^{(k)}$ is given by (29)
6      while $p > 0$ do
7          $\tilde{\xi} \leftarrow 2\tilde{\xi}$;
8          $\tilde{T}^{(k)} = T^{(k)} + \tilde{\xi} I_T$;
9          $[R, p] = \mathtt{chol}(\tilde{T}^{(k)})$;
10     $\eta_*^{(k)} = -R \backslash (R^T \backslash b^{(k)})$;                // where $b^{(k)}$ is given by (30)
11     $\bar{\eta}^{(k)} = \max\{\varepsilon, \min\{1, \eta_*^{(k)}\}\}$;                        // Steplengths
12     $X^{(k+1)} = X^{(k)} + P^{(k)} \mathrm{diag}\{\bar{\eta}^{(k)}\}$;
13     $\bar{\alpha}^{(k+1)} = \max\{\alpha_{min}, \min\{\alpha_{max}, \alpha^{(k+1)}\}\}$;    // where $\alpha^{(k+1)}$ is set to
       (21)
14 until Stop criterion is satisfied;
```

$\arg\min_{1\le t\le T} ||\tilde{x} - x_t||_2$, which gives us the index t_* of the class to which the sample $\tilde{y}$ is classified.

Remark 3. The complexity of one iterative step of the SPG algorithm for updating the matrix A is only $O(IJT)$. It increases considerably for updating the matrix X when $\alpha_X > 0$. In this case, it can be roughly estimated as $O(IJT) + O(J^2I) + O(J^2T) + O(JT^2) + O(T^3)$, assuming that the term $X^{(k)}L_X$ needs $O(JT^2)$, and the solution of the system (28) is obtained in $O(T^3)$. Using the Cholesky factorization, the computational cost for calculating $\tilde{T}^{(k)}$ can be diminished to $\frac{T^3}{6}$ elementary operations. Despite this computational cost predominates in the calculations, the overall complexity for the regularized SPG is still significantly lower than for the standard Newton method that needs $O(IJT) + O(J^3T^3) + O(JT)$ (Remark 2).

4 Experiments

In the experiments, the selected NMF algorithms are compared in the context of their usefulness for supervised classification of various images. The algorithms are evaluated with respect to the classification accuracy, normalized residual error and runtime.

Algorithm 2. SPG-NMF Algorithm

```
Input  : $Y \in \mathbb{R}^{I \times T}$, $J$ - lower rank, $L_X \in \mathbb{R}^{T \times T}$ - weighting matrix, $\alpha_X$ - penalty
         parameter,
Output: Factor matrices: $A \in \mathbb{R}_+^{I \times J}$ and $X \in \mathbb{R}_+^{J \times T}$
1  Initialize: $A$ and $X$ with nonnegative random numbers;
2  Replace negative entries (if any) in $Y$ with zero-value, $k = 0$ ;
3  repeat
4      $X^{(k+1)} = \mathtt{SPG}(Y, A^{(k)}, X^{(k)}, L_X, \alpha_X)$;
5      $\bar{d}_j^{(k+1)} = \sum_{t=1}^{T} x_{jt}^{(k+1)}$,
       $X^{(k+1)} \leftarrow \operatorname{diag}\left\{ \left( \bar{d}_j^{(k+1)} \right)^{-1} \right\} X^{(k+1)}, \quad A^{(k)} \leftarrow A^{(k)} \operatorname{diag}\left\{ \bar{d}_j^{(k+1)} \right\}$;
6      $\bar{A}^{(k+1)} = \mathtt{SPG}(Y^T, (X^{(k+1)})^T, (A^{(k)})^T)$;
7      $A^{(k+1)} = (\bar{A}^{(k+1)})^T$;
8      $\bar{\bar{d}}_j^{(k+1)} = \sum_{i=1}^{I} a_{ij}^{(k+1)}$,
       $X^{(k+1)} \leftarrow \operatorname{diag}\left\{ \bar{\bar{d}}_j^{(k+1)} \right\} X^{(k+1)}, \quad A^{(k+1)} \leftarrow A^{(k+1)} \operatorname{diag}\left\{ \left( \bar{\bar{d}}_j^{(k+1)} \right)^{-1} \right\}$;
9      $k \leftarrow k + 1$;
10 until Stop criterion is satisfied;
```

The classification accuracy is statistically evaluated using *n*-fold Cross-Validation (CV). The mean-accuracy averaged over all CV-folds is expressed as the recognition rate. The normalized residual error in the *k*-iterative step is calculated as $r^{(k)} = \frac{||Y - A^{(k)} X^{(k)}||_F}{||Y||_F}$. The runtime is calculated in Matlab as the elapsed time of processing the algorithm until its stop criterion is satisfied.

We used the following data:

- *Dataset A*: It contains log-magnitude spectrograms created from the audio recordings of 6 musical instruments (cello, soprano saxophone, violin, bassoon, flute, and piano). The recordings are taken from the MIS database[1] of the University of Iowa. The sampling frequency is 44.1kHz. The audio samples are created from 4 sec. excerpts that contain meaningful information in the frequency range from 86Hz to 10.9kHz. Then, the spectrogram are downsampled to 64 frequencies $\times$ 128 time intervals. Each class is represented by 12 samples. The samples are divided into the training and testing sets according to the regular 6-fold CV rule.
- *Dataset B*: It is created from images of hand-written digits from 0 to 9. They were prepared by one student from Wroclaw University of Technology, and used in [34]. The images are downsampled to the resolution of 64×64 pixels. Each class contains 10 images. For testing the algorithms with this dataset, we used the regular 5-fold CV rule.

[1] `http://theremin.music.uiowa.edu`

Table 1 Mean recognition rates, standard deviations (in parenthesis), and elapsed time averaged over CV-folds for $J = 30$, and the datasets: A, B1 (dataset B without processing), B2 (dataset B with WT processing) and C.

Algorithm	A		B1		B2		C	
	Rec. rate	Time	Rec. rate	Time	Rec. rate	Time	Rec. rate	Time
MKL	78.6 (12.0)	0.71	70 (11.2)	0.4	96 (4.2)	0.79	69.75 (3.8)	3.95
MUE	91.7 (10.5)	6.19	90 (5.0)	2.18	96 (4.2)	1.86	94.5 (3.0)	14.45
ALS	90.3 (12.3)	1.05	21 (24.6)	1.82	96 (4.2)	0.81	92.5 (1.8)	2.68
LPG	94.4 (8.6)	8.94	91 (7.4)	3.13	93 (2.7)	2.99	95.75 (1.7)	13.44
FCNNLS	92.9 (8.2)	183.3	84 (6.5)	41.1	94 (2.2)	58.3	95.25 (1.6)	201.6
LNMF	94.4 (8.6)	0.71	25 (7.9)	0.39	95 (3.5)	1.32	91 (6.3)	2.49
DNMF	84.1 (5.5)	0.67	68 (2.7)	0.4	94 (4.2)	0.41	81 (4.2)	2.84
GNMF	93.3 (7.0)	4.51	90 (5.0)	2.07	**98** (2.7)	1.68	96 (2.1)	15.06
SPG	95.8 (7.0)	6.66	91 (9.6)	2.89	97 (2.7)	4.24	96.25 (1.8)	9.88
SPG-MD	95.8 (7.0)	14.8	87 (5.7)	2.05	97 (2.7)	2.59	94.75 (3.0)	13.28
SPG-DNMF	**97.2** (6.8)	11.76	**93** (7.6)	1.71	98 (2.7)	4.65	**97 (1.4)**	9.67

- *Dataset C*: This dataset is obtained from facial images taken from the ORL database[2]. It contains 400 frontal facial images of 40 people (10 pictures per person). The resolution of each image is 112×92 pixels. The 5-fold CV rule is used for testing the algorithms using this dataset.

Additionally, the tests on the dataset B are carried out for two cases: (B1) the original downsampled images are used (without preprocessing), (B2) the samples are preprocessed using the 2-D Wavelet Transform (WT) decomposition with the Haar wavelets at the first level. This task was achieved with the `wavedec2` function from Matlab.

We test the following NMF algorithms: MKL and MUE (standard multiplicative Lee-Seung algorithms for minimizing the Euclidean distance and KL divergence, respectively) [20], standard projected ALS [6], LPG (Lin's Projected Gradient) [23], regularized FCNNLS [35] (improved version of the l_2-norm regularized NMF algorithm proposed in [17]), LNMF [21], DNMF [33], GNMF [5], SPG (Algorithm 2 without the penalty terms), SPG-MD (regularized version with the penalty term determined by (14)), SPG-DNMF (the penalty term modeled by (10)).

Each NMF algorithm was extensively tested for various values of the related parameters. The results presented here are obtained for the parameters that give the highest performance. In the SPG, we set up: $\alpha_{min} = 10^{-8}$, $\alpha_{max} = 10^4$, $\varepsilon = 10^{-8}$. The parameters, such as α_X, γ and δ (DNMF), were tuned up individually to each dataset. For the GNMF, the matrix L_X is determined using the hard connection

[2] `http://people.cs.uchicago.edu/~dinoj/vis/orl/`

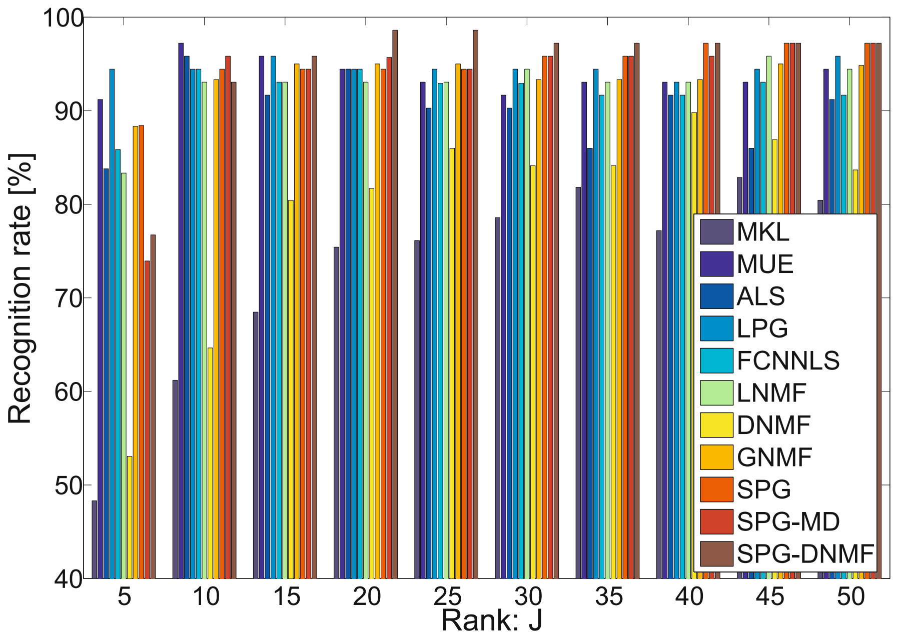


Fig. 1 Recognition rates obtained for dataset A using various NMF algorithms versus the rank J

criterion given by (11). For the SPG-MD, the Heat kernel weighting in (12) was used with $\sigma^2 = 10^7$, $k_1 = 5$ and $k_2 = 20$. For many test cases: $\alpha_X = 10^{-2}$ and $\xi = 10^2$. For the SPG-DNMF: $\alpha_X = 1$, $\gamma \approx 10^{-5}$ and $\delta \leq 10^{-6}$.

All the NMF algorithms were initialized by the SimplexMax algorithm (for $p = 1$) that was proposed in [36]. The stop criterion in all the tested algorithms was the same. The inner iterations (i.e. for updating one factor at the other fixed) were determined on the basis of the projected gradient criterion that was used in the LPG algorithm [23]. The maximum number of inner iterations was set to 10. The alternating steps (outer iterations) were terminated in all the tested algorithms if the normalized residual error drops below 10^{-4}.

The mean recognition rates versus the rank of factorization (J) are illustrated in Figs. 1–4 for the datasets A, B1, B2 and C, respectively. The same results but at the rank fixed to 30 are presented in Table 1. Additionally, the table contains the standard deviations of recognitions rates across CV-folds, and the runtime for processing each algorithm. Note that it is not the runtime of executing a given number of iterations but the elapsed time measured until the stop criterion holds.

The normalized residual errors versus the number of iterations for the selected NMF algorithms are plotted in Fig. 5.

The averaged classification results obtained with the MKL, LNMF and SPG-DNMF algorithms are also illustrated in Fig. 6 in the form of the Hinton graph of

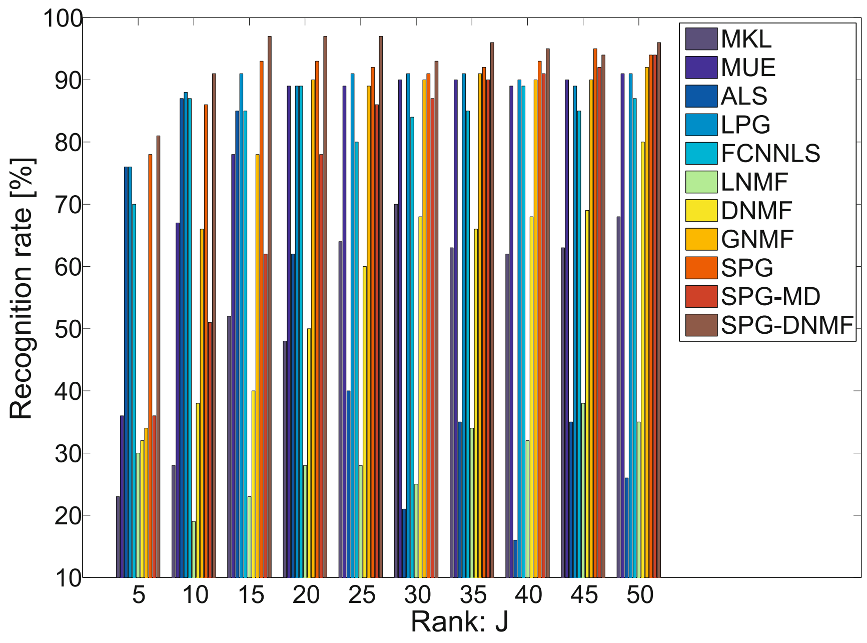


Fig. 2 Recognition rates obtained for dataset B without preprocessing using various NMF algorithms versus the rank J

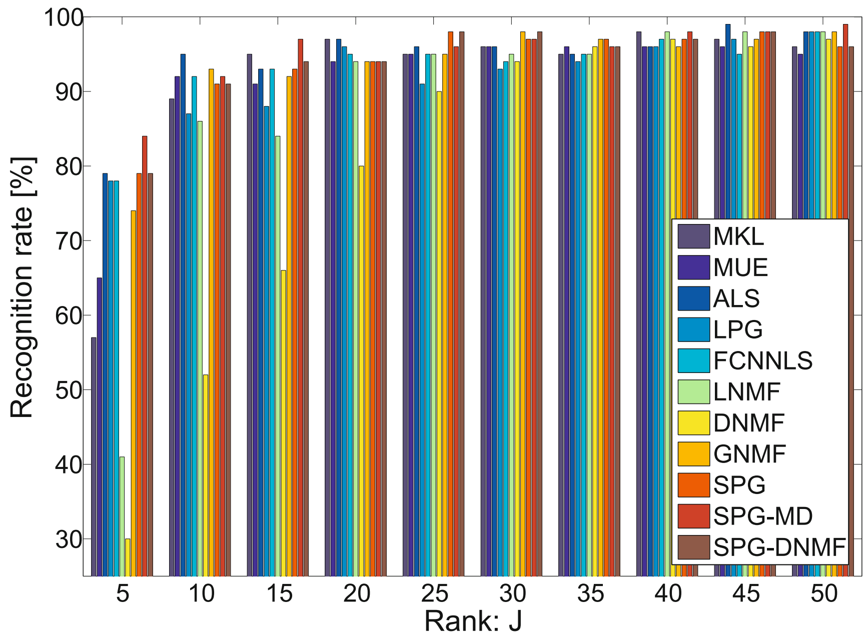


Fig. 3 Recognition rates obtained for dataset B with WT preprocessing using various NMF algorithms versus the rank J

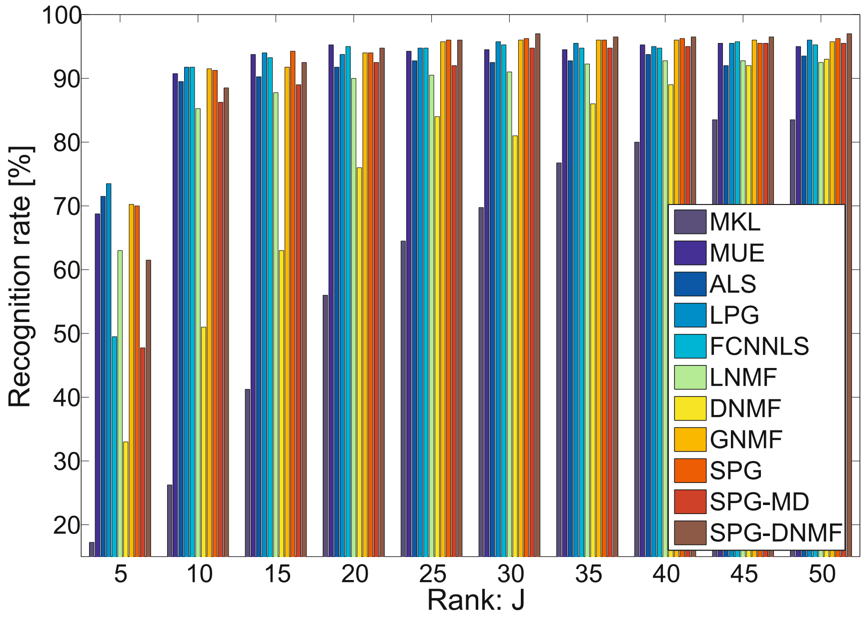


Fig. 4 Recognition rates obtained for dataset C using various NMF algorithms versus the rank J

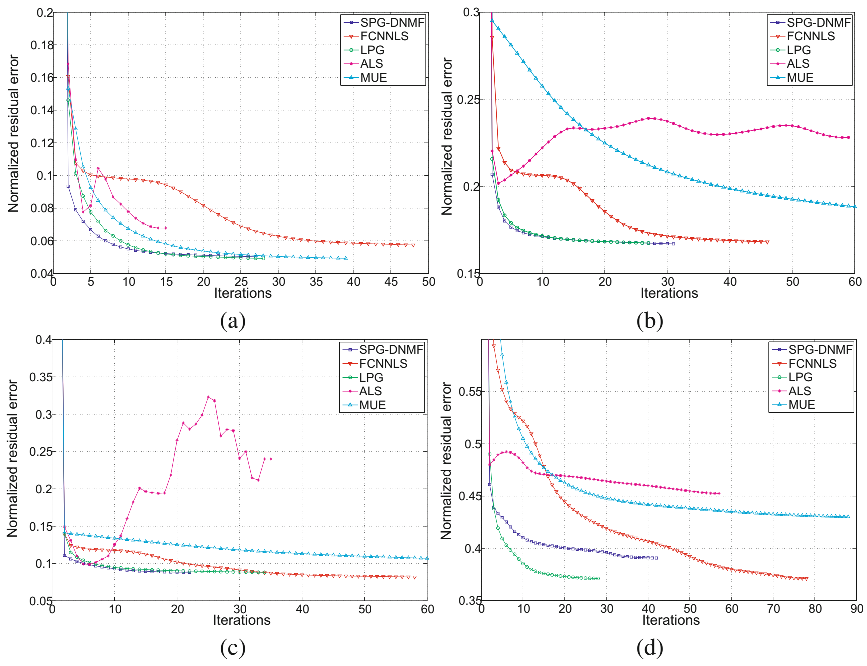
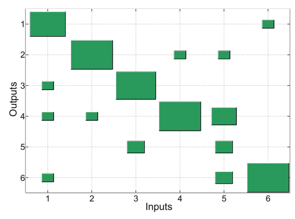
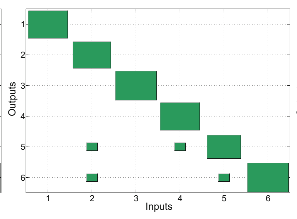


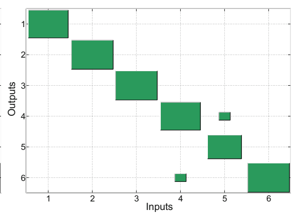
Fig. 5 Normalized residual errors versus alternating iterations: (a) dataset A; (b) dataset B without preprocessing; (c) dataset B with WT preprocessing; (d) dataset C



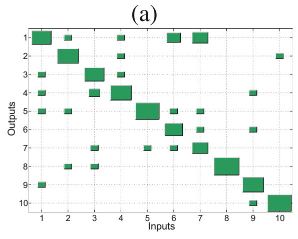
(a)



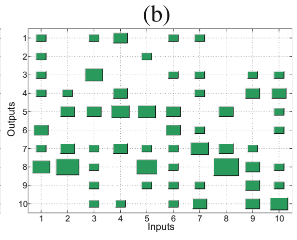
(b)



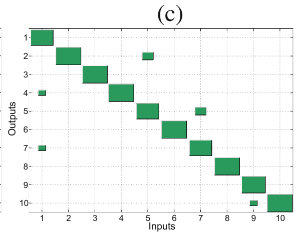
(c)



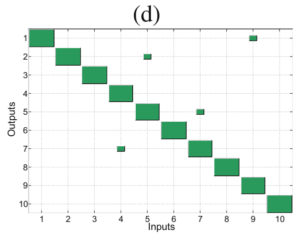
(d)



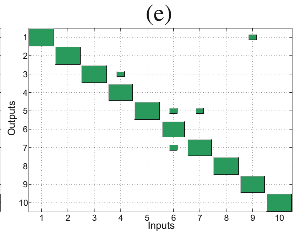
(e)



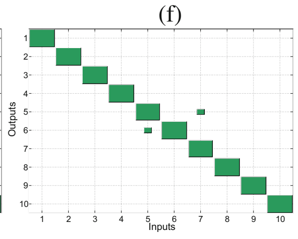
(f)



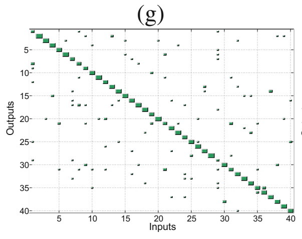
(g)



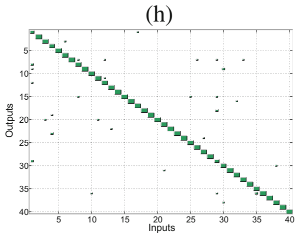
(h)



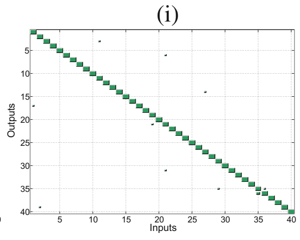
(i)



(j)



(k)



(l)

Fig. 6 Confusion matrices: (a) MKL, Dataset A; (b) LNMF: Dataset A; (c) SPG-DNMF, Dataset A; (d) MKL, Dataset B without preprocessing; (e) LNMF, Dataset B without preprocessing; (f) SPG-DNMF, Dataset B without preprocessing; (g) MKL, Dataset B with WT preprocessing; (h) LNMF, Dataset B with WT preprocessing; (i) SPG-DNMF, Dataset B with WT preprocessing; (j) MKL, Dataset C; (k) LNMF: Dataset C; (l) SPG-DNMF, Dataset C

confusion matrices. Both the Hinton graph and the confusion matrix are obtained with the Matlab functions from the *Neural Network Toolbox*.

5 Conclusions

In the chapter, we compared several NMF algorithms for various classification problems. We observed that the classification results are more affected by the dataset or the preprocessing than the NMF algorithm. For the dataset A, the SPG-DNMF outperforms the other algorithms for $J \geq 15$ (see Fig. 1). This algorithm behaves similarly for the dataset B1 (see Fig. 2), even for a lower rank ($J = 5$). After applying the preprocessing, nearly all the tested algorithms give the similar performance (see Fig. 3). Only the DNMF works noticeably worse for smaller ranks. For $J = 30$, the GNMF is slightly better than the SPG-DNMF (see Table 1). For classification of facial images, the SPG-DNMF is ranked first for $J > 20$. Usually an increase in the factorization rank leads to a higher recognition rate.

Comparing the results obtained for the datasets B1 and B2, one can easily notice that the WT preprocessing is crucial for this kind of images. For the other datasets, it does not lead to such a big difference.

The tests show that the Euclidean distance-based algorithms better classify the images in the datasets A, B1 and C than the KL divergence-based ones. Despite, the MKL and MUE have a similar convergence rate, the difference in performance is large. The LNMF and DNMF significantly outperform the MKL but are inferior to the SPG-DNMF.

Fig. 5 proves that the SPG-DNMF algorithm has a good convergence behavior. It converges monotonically and usually faster (especially in initial iterations) than the others. In each alternating step, the updates for A and X are optimal according to the first-order KKT optimality conditions. For example, the monotonic convergence is not observed for the ALS algorithm. Additionally, Fig. 5 shows the number of iterations performed to satisfy the stop criterion. For the threshold of the normalized residual error at the level of 10^{-4}, this number for the SPG-DNMF is usually lower than 50 iterations, while the MUE and FCNNLS require much more iterations. Hence, the runtime for MUE in Table 1 is sometimes longer than for the SPG-based algorithms. In this comparison, the FCNNLS is the slowest.

Summing up, the experiments demonstrate that the SPG-based algorithms work very efficiently in NMF-based classification of various images. It can be also easily extended to other applications, such as multi-linear discriminant analysis or multi-way array decompositions.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems 14, pp. 585–591. MIT Press (2001)
2. Benetos, E., Kotti, M., Kotropoulos, C.: Musical instrument classification using non-negative matrix factorization algorithms and subset feature selection. In: Proc. of 2006 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006), Toulouse, France, p. V (2006)
3. Birgin, E.G., Martnez, J.M., Raydan, M.: Nonmonotone spectral projected gradient methods on convex sets. SIAM Journal on Control and Optimization 10, 1196–1211 (2000)
4. Cai, D., He, X., Han, J., Huang, T.: Graph regularized nonnegative matrix factorization for data representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(8), 1548–1560 (2011)

5. Cai, D., He, X., Wu, X., Han, J.: Nonnegative matrix factorization on manifold. In: Proc. 8th IEEE International Conference on Data Mining (ICDM), pp. 63–72 (2008)
6. Cichocki, A., Zdunek, R., Phan, A.H., Amari, S.I.: Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation. Wiley and Sons (2009)
7. Cotter, S.F., Rao, B.D., Engan, K., Kreutz-Delgado, K.: Sparse solutions to linear inverse problems with multiple measurement vectors. IEEE Transaction on Signal Processing 53(7), 2477–2488 (2005)
8. Donoho, D., Stodden, V.: When does non-negative matrix factorization give a correct decomposition into parts? In: Thrun, S., Saul, L., Schölkopf, B. (eds.) Advances in Neural Information Processing Systems (NIPS), vol. 16. MIT Press, Cambridge (2004)
9. Franc, V., Hlaváč, V., Navara, M.: Sequential coordinate-wise algorithm for the non-negative least squares problem. In: Gagalowicz, A., Philips, W. (eds.) CAIP 2005. LNCS, vol. 3691, pp. 407–414. Springer, Heidelberg (2005)
10. Guan, N., Tao, D., Luo, Z., Yuan, B.: Manifold regularized discriminative nonnegative matrix factorization with fast gradient descent. IEEE Transactions on Image Processing 20(7), 2030–2048 (2011)
11. Guan, N., Tao, D., Luo, Z., Yuan, B.: NeNMF: An optimal gradient method for nonnegative matrix factorization. IEEE Transactions on Signal Processing 60(6), 2882–2898 (2012)
12. Guillamet, D., Vitria, J.: Classifying faces with nonnegative matrix factorization. In: Proc. 5th Catalan Conference for Artificial Intelligence, Castello de la Plana, Spain, pp. 24–31 (2002)
13. He, X., Niyogi, P.: Locality preserving projections. In: Advances in Neural Information Processing Systems 16. MIT Press (2003)
14. Hoyer, P.O.: Non-negative sparse coding. In: Neural Networks for Signal Processing XII (Proc. IEEE Workshop on Neural Networks for Signal Processing), Martigny, Switzerland, vol. 2, pp. 557–565 (2002)
15. Hoyer, P.O.: Non-negative matrix factorization with sparseness constraints. Journal of Machine Learning Research 5, 1457–1469 (2004)
16. Huang, K., Sidiropoulos, N.D., Swami, A.: Nonnegative matrix factorization revised: Uniqness and algorithm for symmetric decomposition, pp. 211–224 (2014)
17. Kim, H., Park, H.: Non-negative matrix factorization based on alternating non-negativity constrained least squares and active set method. SIAM Journal in Matrix Analysis and Applications 30(2), 713–730 (2008)
18. Kotsia, I., Zafeiriou, S., Pitas, I.: Discriminant non-negative matrix factorization and projected gradients for frontal face verification. In: Schouten, B., Juul, N.C., Drygajlo, A., Tistarelli, M. (eds.) BIOID 2008. LNCS, vol. 5372, pp. 82–90. Springer, Heidelberg (2008)
19. Lanteri, H., Theys, C., Richard, C.: Nonnegative matrix factorization with regularization and sparsity-enforcing terms. In: 4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), pp. 97–100 (2011)
20. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401, 788–791 (1999)
21. Li, S.Z., Hou, X.W., Zhang, H.J., Cheng, Q.S.: Learning spatially localized, parts-based representation. In: Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), vol. 1, pp. I–207–I–212 (2001)
22. Lin, C.J.: On the convergence of multiplicative update algorithms for non-negative matrix factorization. IEEE Transactions on Neural Networks 18(6), 1589–1596 (2007)

23. Lin, C.J.: Projected gradient methods for non-negative matrix factorization. Neural Computation 19(10), 2756–2779 (2007)
24. Liu, X., Yan, S., Jin, H.: Projective nonnegative graph embedding. IEEE Transactions on Image Processing 19(5), 1126–1137 (2010)
25. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer Series in Operations Research. Springer, New York (1999)
26. Pascual-Montano, A., Carazo, J.M., Kochi, K., Lehmean, D., Pacual-Marqui, R.: Nonsmooth nonnegative matrix factorization (nsNMF). IEEE Transaction Pattern Analysis and Machine Intelligence 28(3), 403–415 (2006)
27. Phan, A.H., Cichocki, A.: Tensor decompositions for feature extraction and classification of high dimensional datasets. IEICE Nonlinear Theory and Its Applications 1(1), 37–68 (2010)
28. Phan, A.H., Cichocki, A.: Extended HALS algorithm for nonnegative Tucker decomposition and its applications for multiway analysis and classification. Neurocomputing 74(11), 1956–1969 (2011); Adaptive Incremental Learning in Neural Networks; Learning Algorithm and Mathematic Modelling. Selected paper from the International Conference on Neural Information Processing (ICONIP 2009)
29. Qin, L., Zheng, Q., Jiang, S., Huang, Q., Gao, W.: Unsupervised texture classification: Automatically discover and classify texture patterns. Image and Vision Computing 26(5), 647–656 (2008)
30. Wang, C., Song, Z., Yan, S., Lei, Z., Zhang, H.J.: Multiplicative nonnegative graph embedding. In: CVPR, pp. 389–396. IEEE (2009)
31. Wang, Y., Jia, Y., Hu, C., Turk, M.: Fisher nonnegative matrix factorization for learning local features. In: Proc. 6th Asian Conf. on Computer Vision, Jeju Island, Korea, pp. 27–30 (2004)
32. Yang, J., Yan, S., Fu, Y., Li, X., Huang, T.S.: Non-negative graph embedding. In: CVPR 2008, vol. 4, pp. 1–8 (2008)
33. Zafeiriou, S., Tefas, A., Buciu, I., Pitas, I.: Exploiting discriminant information in nonnegative matrix factorization with application to frontal face verification. IEEE Transactions on Neural Networks 17(3), 683–695 (2006)
34. Zak, M.: Recognition of hand-written digits with nonnegative matrix factorization. Master's thesis, Wroclaw University of Technology, Wroclaw (2011); Supervisor: R. Zdunek
35. Zdunek, R.: Regularized active set least squares algorithm for nonnegative matrix factorization in application to Raman spectra separation. In: Cabestany, J., Rojas, I., Joya, G. (eds.) IWANN 2011, Part II. LNCS, vol. 6692, pp. 492–499. Springer, Heidelberg (2011)
36. Zdunek, R.: Initialization of nonnegative matrix factorization with vertices of convex polytope. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2012, Part I. LNCS, vol. 7267, pp. 448–455. Springer, Heidelberg (2012)
37. Zdunek, R., Cichocki, A.: Nonnegative matrix factorization with constrained second-order optimization. Signal Processing 87, 1904–1916 (2007)
38. Zdunek, R., Cichocki, A.: Sequential coordinate-wise DNMF for face recognition. In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010, Part I. LNCS, vol. 6113, pp. 563–570. Springer, Heidelberg (2010)
39. Zdunek, R., Phan, A.-H., Cichocki, A.: GNMF with Newton-based methods. In: Mladenov, V., Koprinkova-Hristova, P., Palm, G., Villa, A.E.P., Appollini, B., Kasabov, N. (eds.) ICANN 2013. LNCS, vol. 8131, pp. 90–97. Springer, Heidelberg (2013)