SeMA Journal (2020) 77:203–217
https://doi.org/10.1007/s40324-019-00211-z

Adaptive computation of the Symmetric Nonnegative Matrix Factorization (SymNMF)

P. Favati[1] · G. Lotti[2] · O. Menchi[3] · F. Romani[3]

Received: 4 June 2018 / Accepted: 19 December 2019 / Published online: 1 January 2020

Abstract
Symmetric Nonnegative Matrix Factorization (SymNMF) provides a symmetric nonnegative low rank decomposition of symmetric matrices. It has found application in several domains such as data mining, bioinformatics and graph clustering. In this paper SymNMF is obtained by solving a penalized nonsymmetric minimization problem. Instead of letting the penalizing parameter increase according to an a priori fixed rule, as suggested in literature, we propose a heuristic approach based on an adaptive technique. The factorization is performed by applying, as inner-outer procedure, the Alternating Nonnegative Least Squares (ANLS). The inner problems are solved by two nonnegative quadratic optimization methods: an Active-Set-like method (`BPP`) and the Greedy Coordinate Descent method (`GCD`). Extensive experimentation shows that the proposed heuristic approach is effective with both the inner methods, `GCD` outperforming `BPP` in terms of computational time complexity.

Keywords Symmetric Nonnegative Matrix Factorization · Graph Clustering

Mathematics Subject Classification 65F30 · 65Y20 · 68Q25

✉ P. Favati
paola.favati@iit.cnr.it

G. Lotti
grazia.lotti@unipr.it

O. Menchi
menchi@di.unipi.it

F. Romani
romani@di.unipi.it

[1] IIT-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy

[2] Dip. di Scienze Matematiche, Fisiche e Informatiche, University of Parma, Parco Area delle Scienze 53/A, 43124 Parma, Italy

[3] Dip. di Informatica, University of Pisa, Largo Pontecorvo 3, 56127 Pisa, Italy

1 Introduction

Dimensional reduction problems are of fundamental relevance for data compression and classification. An important problem of this kind is represented by the nonnegative matrix factorization, which was first proposed in [14] for data analysis and afterwards widely applied (see [6] for an extensive bibliography).

Let $\mathbf{R}_+^m$ be the m-dimensional space of vectors with nonnegative components and M a matrix of n columns $\boldsymbol{m}_i \in \mathbf{R}_+^m$, for $i = 1, \ldots, n$. Given an integer $k < \min(m, n)$, the problem of finding two low-rank matrices $W \in \mathbf{R}_+^{m \times k}$ (the *basis* matrix) and $H \in \mathbf{R}_+^{n \times k}$ (the *coefficient* matrix) such that the product WH^T approximates M, is known as *Nonnegative Matrix Factorization* (NMF). In this way the n objects $\boldsymbol{m}_i$ result represented by linear combinations with nonnegative coefficients of few nonnegative basis vectors.

A specific formulation of the problem requires that a metric is assigned to measure the distance between M and WH^T. The nonnegativity of the involved items would suggest to minimize a divergence, as the likelihood *Kullback–Leibler* divergence, but some computational difficulties and the slow convergence rate of common iterative procedures used to tackle the problem, suggest the more flexible metric of the F-norm (Frobenius norm). With this norm the general NMF problem takes the form

$$\min_{W,\, H \geq O} \Phi(M, W, H), \quad \text{where } \Phi(M, W, H) = \tfrac{1}{2} \|M - W H^T\|_F^2. \tag{1}$$

The best factorization of a matrix in terms of the F-norm is achieved by the Singular Value Decomposition $M = U \Sigma V^T$. Then the best k-rank approximation of M is $U_k \Sigma_k V_k^T$, where U_k and V_k are the truncated submatrices of U and V to k columns and Σ_k is the $k \times k$ leading submatrix of Σ. Unfortunately, nothing can be said about the sign of the entries of U_k and V_k, which could be negative. It follows that nonnegativity must be imposed as a constraint. Other constraints could appear in (1) in order to satisfy additional requirements [15]. For example, adding the terms $\rho_1 \|W\| + \rho_2 \|H\|$ to the function Φ to be minimized, where $\|\cdot\|$ is a suitable norm and ρ_1 and ρ_2 are positive parameters, would give a regularized solution and possibly control the sparsity of the factors W and H.

In this paper we consider the additional requirement of symmetry: problem (1) is then replaced by the symmetric NMF (*SymNMF*) problem

$$\min_{W \geq O} \Psi(A, W), \quad \text{where } \Psi(A, W) = \tfrac{1}{2} \|A - W W^T\|_F^2, \tag{2}$$

where A is a symmetric $n \times n$ matrix of nonnegative entries and $W \in \mathbf{R}_+^{n \times k}$. The nonnegativity requirement for the entries of A is not a necessary condition [9]. For example in the case of clustering problems, the matrix A could be a similarity matrix obtained by applying on the data $\boldsymbol{m}_i$ a Kernel which produces elements of mixed sign. The similarity kernels used in our experimentation give nonnegative matrices A, but the proposed heuristic can be used also in the general case. Note that the required approximation given by the positive semidefinite matrix WW^T can be very poor if A does not have enough nonnegative eigenvalues.

Problem (2) has a fourth-order nonconvex objective function and optimization algorithms guarantee only the stationarity of the limit points, so one only looks for a local minimum. Standard gradient algorithms lead to stationary solutions, but suffer from slow convergence. Newton-like algorithms, which have a better rate of convergence, are computationally expensive. In [10] a nonsymmetric formulation of (2) is suggested by considering the following penalized problem

$$\min_{W,\,H\geq O} \Phi_\alpha(A,W,H), \quad \text{where } \Phi_\alpha(A,W,H) = \tfrac{1}{2}\left(\|A - WH^T\|_F^2 + \alpha\|W - H\|_F^2\right), \tag{3}$$

α being a positive parameter which acts on the violation of the symmetry. Choosing α aligned with the magnitude of A, makes the penalized problem invariant with the scale of matrix A. In this paper we propose an algorithm to approximate the solution of (2) by solving iteratively problem (3) and dealing adaptively with the penalizing parameter α.

To this aim in Sect. 2 we recall the ANLS framework, a standard approach for tackling a general (possibly not symmetric) NMF problem of form (1) by addressing alternatively two convex subproblems, together with the two methods (`BPP` [7] and `GCD` [5]) which will be used to solve each subproblem. Section 3 deals with the heuristic for the choice of the parameter α to solve (3). In Sect. 4 the results of an extensive experimentation are presented to validate the proposed adaptive strategy and to compare the performance of the two chosen methods when applied in our context.

2 The ANLS framework for the general NMF

Problem (1) is nonconvex and finding its global minimum is NP-hard. Most nonconvex optimization algorithms guarantee only the stationarity of the limit points, so one looks for a local minimum. There is a further source of nonunicity, since $WH^T = W'H'^T$ with $W' = WS$, $H' = HS^{-T}$, where $S \in \mathbf{R}_+^{k\times k}$ is a nonsingular scaling matrix. This can be fixed by choosing for example S in such a way to normalize the columns of W to unit 2-norm.

The *alternating nonnegative least squares* (ANLS) method, which belongs to the *block coordinate descent* (BCD) framework of nonlinear optimization [8], solves iteratively problem (1). First, one of the factors, say W, is initialized to $W^{(0)}$ with nonnegative entries and the matrix $H^{(1)} \in \mathbf{R}_+^{n\times k}$ realizing the minimum of $\Phi(M, W^{(0)}, H)$ on $H \geq 0$ is computed. Then a new matrix $W^{(1)} \in \mathbf{R}_+^{m\times k}$ realizing the minimum of $\Phi(M, W, H^{(1)})$ on $W \geq 0$ is computed, and so on, updating W and H alternatively. In practice the following *inner-outer* scheme is applied

$$H^{(\nu)} = \operatorname*{argmin}_{H\geq 0} \Phi(M, W^{(\nu-1)}, H), \tag{4}$$

$$W^{(\nu)} = \operatorname*{argmin}_{W\geq 0} \Phi(M^T, H^{(\nu)}, W), \tag{5}$$

for $\nu = 1, 2, \ldots$, where each subproblem is solved by applying a chosen inner method. At the νth outer iteration a suitable stopping condition should check whether a local minimum of the object function $\Phi(M, W, H)$ of (1) has been sufficiently well approximated, for example by monitoring the *error*, i.e. the distance of $W^{(\nu)}H^{(\nu)T}$ from M

$$e^{(\nu)} = \|M - W^{(\nu)}H^{(\nu)T}\|_F^2.$$

The choice of the initial matrix $W^{(0)}$ may be critical, due to the fact that only a local minimum is expected, which obviously depends on this choice and, typically, the algorithm is run several times with different initial matrices.

Although the original problem (1) is nonconvex, subproblems (4) and (5) are convex and nearly any procedure for constrained quadratic optimization can be chosen as inner method (for example an Active-Set-like method [1,7,11]). The requirement that the inner problems are exactly solved at each outer step is necessary for convergence [4] but makes the overall

algorithm rather slow at large dimensions. Faster approaches have been devised by computing iteratively approximate solutions with inexact methods like modified gradient descent methods or projected Newton-type methods [8]. In this paper we take into consideration, as inner methods, an Active-Set-like method with block principal pivoting (the BPP method, coded as Algorithm 2 in [7]) and a coordinate descent method, called Greedy Coordinate Descent (GCD) in [5]). Their main difference lies in the termination: exact for BPP and approximated for GCD.

When the ANLS method is applied, at each outer step, say the νth outer step, the inner method computes the solution of two problems of the form

$$\min_{X \geq 0} \Phi(B, C, X) = \min_{X \geq 0} \tfrac{1}{2} \| B - C\, X^T \|_F^2, \tag{6}$$

where $B = M$, $C = W^{(\nu-1)}$, $X = H$ for problem (4) and $B = M^T$, $C = H^{(\nu)}$, $X = W$ for problem (5). We assume matrix C to have full rank.

Let $r \times s$ be the dimensions of B ($r = m$, $s = n$ in the first case and $r = n$, $s = m$ in the second case). Denoting by $\boldsymbol{b} \in \mathbf{R}_+^r$ and $\boldsymbol{x} \in \mathbf{R}^k$ the hth columns of B and X^T respectively, for $h = 1, \ldots, s$, problem (6) can be decomposed into s independent least squares nonnegatively constrained problems

$$\min_{\boldsymbol{x} \geq 0} \varphi(\boldsymbol{x}), \quad \text{where} \quad \varphi(\boldsymbol{x}) = \tfrac{1}{2} \| \boldsymbol{b} - C\boldsymbol{x} \|_2^2. \tag{7}$$

The gradient of the objective function $\varphi(\boldsymbol{x})$ is $\boldsymbol{g}(\boldsymbol{x}) = C^T(C\boldsymbol{x} - \boldsymbol{b})$.

The s problems (7) are solved in sequence, using either BPP or GCD. Before proceeding, we give a brief description of the two considered methods. The corresponding codes can be found in the cited papers.

2.1 The BPP method

BPP method derives from the classical active set method for linearly constrained optimization. For a point $\boldsymbol{x} \in \mathbf{R}^k$ consider the active and passive index sets at $\boldsymbol{x}$

$$\mathcal{A}(\boldsymbol{x}) = \{1 \leq i \leq k, \text{ such that } x_i = 0\}, \quad \mathcal{P}(\boldsymbol{x}) = \mathcal{K} - \mathcal{A}(\boldsymbol{x}),$$

where $\mathcal{K} = \{1, \ldots, k\}$ is the complete index set. Let $C_{\mathcal{A}}$ and $C_{\mathcal{P}}$ be the restrictions of the matrix C to $\mathcal{A}(\boldsymbol{x})$ and $\mathcal{P}(\boldsymbol{x})$ respectively. Since $C_{\mathcal{P}}$ has full column rank, the solution of the unconstrained least squares problem

$$\boldsymbol{x}_{\mathcal{P}} = \operatorname*{argmin}_{z} \tfrac{1}{2} \| \boldsymbol{b} - C_{\mathcal{P}}\, \boldsymbol{z} \|_2^2 \tag{8}$$

is given by the solution of the system

$$C_{\mathcal{P}}^T C_{\mathcal{P}}\, \boldsymbol{z} = C_{\mathcal{P}}^T \boldsymbol{b}, \tag{9}$$

which has size less than or equal to k. If the size is not too large, the system is solved by applying the Cholesky factorization (otherwise, one can resort to the conjugate gradient). Let $\boldsymbol{x}^*$ be the vector which coincides with $\boldsymbol{x}_{\mathcal{P}}$ on $\mathcal{P}(\boldsymbol{x})$ and has zero components on $\mathcal{A}(\boldsymbol{x})$. Denote by

$$\boldsymbol{g}_{\mathcal{A}}(\boldsymbol{x}^*) = C_{\mathcal{A}}^T (C_{\mathcal{P}} \boldsymbol{x}_{\mathcal{P}} - \boldsymbol{b}) \tag{10}$$

the gradient restricted to $\mathcal{A}(\boldsymbol{x})$. According to Karush–Kuhn–Tucker (KKT) optimality conditions, the vector $\boldsymbol{x}^*$ is a solution of (7) if and only if $\boldsymbol{x}_{\mathcal{P}} \geq \mathbf{0}$ and $\boldsymbol{g}_{\mathcal{A}}(\boldsymbol{x}^*) \geq \mathbf{0}$.

If the active and passive index sets of $\boldsymbol{x}^*$ were known in advance, problem (7) could be solved by simply solving (9). Since the two index sets are initially unknown, a sequence of unconstrained subproblems is solved with the two index sets in turn predicted and exchanged. The computation starts with index sets associated to an initial point supplied by the outer iteration and goes on until all the constraints become passive or the gradient has nonnegative components corresponding to all the active constraints, indicating that the objective function cannot be reduced any more.

In the classical Active-Set method [11], only an index moves from $\mathcal{A}(\boldsymbol{x})$ to $\mathcal{P}(\boldsymbol{x})$ at a time. This makes the number of iterations to grow considerably with the size of the problem. An overcome to this drawback consists in exchanging more indices between $\mathcal{A}(\boldsymbol{x})$ and $\mathcal{P}(\boldsymbol{x})$ at each iteration, as suggested in [7]. The number of iterations results reduced, but the generated vectors $\boldsymbol{x}$ may fail to maintain nonnegativity and the monotonic decrease of the objective function is not guaranteed. A finite termination is achieved by a backup rule which implements the standard one index exchange when necessary.

When the procedure described above for a single column $\boldsymbol{b}$ of B is applied to all the columns of B, the following improvement, proposed in [7], reduces the computational cost. Since each problem (7) shares the same matrix C, and the main cost depends on solving system (9) and on computing vector (10) with matrices $C_{\mathcal{P}}^T C_{\mathcal{P}}$, $C_{\mathcal{A}}^T C_{\mathcal{P}}$ and vectors $C_{\mathcal{P}}^T \boldsymbol{b}$, $C_{\mathcal{A}}^T \boldsymbol{b}$, it is suggested to extract these matrices and vectors from the complete matrices $C^T C$ and $C^T B$ computed once at the beginning. Another improvement consists in grouping the right-hand side vectors which share the same index set $\mathcal{P}$ in order to avoid repeated computation of the Cholesky factorization in solving systems (9).

2.2 The GCD method

GCD derives from FastHals [2], an iterative method which performs a cyclic coordinate descent scheme. GCD, instead, at each step selectively replaces the element whose update leads to the largest decrease of the objective function.

In [5] GCD works on the entire matrix X, but in practice the method is applied to solve in sequence problems of type (7). For each problem (7), starting from a $\boldsymbol{x}^{(0)} \in \mathbf{R}_+^k$ chosen according to the outer iteration, GCD computes a sequence $\boldsymbol{x}^{(j)}$, $j = 1, 2, \ldots$, until suitably stopped. A global stopping condition based on the entire matrix B, suggested in [5], is described at the end of the paragraph.

At the jth iteration the vector $\boldsymbol{x}^{(j)}$ is obtained by applying a single coordinate correction according to the rule

$$\boldsymbol{x}^{(j)} = \boldsymbol{x}^{(j-1)} + \widehat{\lambda}\, \boldsymbol{e}_i,$$

where i is an index to be selected in $\{1, \ldots, k\}$, $\boldsymbol{e}_i$ is the ith canonical k-vector and the scalar $\widehat{\lambda}$ is determined by imposing that $\varphi(\boldsymbol{x}^{(j)})$, as a function of λ, is the minimum on the set $S = \{\lambda \text{ such that } x_i^{(j-1)} + \lambda \geq 0\}$. This scalar is

$$\widehat{\lambda} = -\frac{g_i^{(j-1)}}{q_{i,i}} \quad \text{if} \quad \frac{g_i^{(j-1)}}{q_{i,i}} \leq x_i^{(j-1)} \quad \text{and} \quad \widehat{\lambda} = -x_i^{(j-1)} \quad \text{otherwise,}$$

where $Q = C^T C$ is the Hessian of φ and $\boldsymbol{g}^{(j-1)} = \boldsymbol{g}\big(\boldsymbol{x}^{(j-1)}\big)$. In correspondence, the objective function is decreased by

$$d_i^{(j)} = \varphi\big(\boldsymbol{x}^{(j-1)}\big) - \varphi\big(\boldsymbol{x}^{(j)}\big) = -\, g_i^{(j-1)}\, \widehat{\lambda} - \frac{1}{2}\, q_{i,i}\, \widehat{\lambda}^2.$$

A natural choice for index i is the one that maximizes $d_i^{(j)}$ varying i. As a consequence, $x_i^{(j-1)}$ is updated by adding $\widehat{\lambda}$ and the elements of the gradient become

$$g_t^{(j)} = g_t^{(j-1)} + \widehat{\lambda}\, q_{t,i}, \quad \text{for} \quad t = 1, \ldots, k.$$

Then a new iteration begins, where a new index i is detected, and so on, until a stopping condition is met. In [5] the following condition is suggested

$$\max_i d_i^{(j)} < \eta\, \mu, \tag{11}$$

where the quantity μ is the largest possible reduction of all the objective functions φ of problems (7) varying $\boldsymbol{b}$, that can be expected when a single element is modified at the first iteration and η is a preassigned tolerance. Of course, the value of η influences the convergence of the outer method, hence the overall computational cost. In [5] $\eta = 10^{-3}$ is suggested. We will examine this question in Sect. 4.2.2.

3 The SymNMF problem

We turn now to the SymNMF problem (2). As anticipated, its solution is computed through the nonsymmetric formulation (3), applying ANLS as the inner-outer algorithm, i.e. by alternating the solution of the two subproblems,

$$H^{(\nu)} = \underset{H \geq 0}{\operatorname{argmin}}\ \Phi\left(\begin{bmatrix} A \\ \sqrt{\alpha}\, W^{(\nu-1)T} \end{bmatrix}, \begin{bmatrix} W^{(\nu-1)} \\ \sqrt{\alpha}\, I_k \end{bmatrix}, H \right), \tag{12}$$

$$W^{(\nu)} = \underset{W \geq 0}{\operatorname{argmin}}\ \Phi\left(\begin{bmatrix} A \\ \sqrt{\alpha}\, H^{(\nu)T} \end{bmatrix}, \begin{bmatrix} H^{(\nu)} \\ \sqrt{\alpha}\, I_k \end{bmatrix}, W \right). \tag{13}$$

The corresponding function `Sym_ANLS` is shown in Fig. 1, where we denote by `Inner_solve` $\left(B, C, X_0\right)$ the function used to solve (6) employing a method which starts

---

`function Sym_ANLS` $\left(W, H, \alpha, \nu_{max}\right)$

computes recursively the solution of (2) by solving (3) given initial W, H, β *and the number of allowed iterations* ν_{max}.

$W^{(0)} = W$; $H^{(0)} = H$; $\beta^{(0)} = 1$; $\nu = 0$; *cond* = `True`;
`while` *cond*
  $\alpha^{(\nu)} = \beta^{(\nu)} \max A$;
  $\nu = \nu + 1$;
  $\xi = \sqrt{\alpha^{(\nu-1)}}$;
  $H^{(\nu)} =$ `Inner_solve` $\left(\begin{bmatrix} A \\ \xi\, W^{(\nu-1)T} \end{bmatrix}, \begin{bmatrix} W^{(\nu-1)} \\ \xi\, I_k \end{bmatrix}, H^{(\nu-1)} \right)$;
  $W^{(\nu)} =$ `Inner_solve` $\left(\begin{bmatrix} A \\ \xi\, H^{(\nu)T} \end{bmatrix}, \begin{bmatrix} H^{(\nu)} \\ \xi\, I_k \end{bmatrix}, W^{(\nu-1)} \right)$;
  compute $\epsilon_S^{(\nu)}$ and $\delta^{(\nu)}$, according to (14) and (15);
  stop = $\left|\epsilon_S^{(\nu)} - \epsilon_S^{(\nu-1)}\right| \leq \tau_1\, \epsilon_S^{(\nu)}$ `and` $\delta^{(\nu)} \leq \tau_2$;
  cond = `not` *stop* `and` $\nu < \nu_{max}$;
  $\beta^{(\nu)} =$ `Update` $\left(\beta^{(\nu-1)}\right)$;
`end while`;
`return` $W^{(\nu)}$;

Fig. 1 Algorithm to solve problem (2). The function `Inner_solve` solves problems (12) and (13). In all the experiments we have assumed $\tau_1 = 10^{-3}$ and $\tau_2 = 0.1$

with initial iterate $X_0 \geq 0$. For problems (12) and (13) both BPP and GCD, used as inner methods, can be implemented without explicitly forming the four block matrices.

At the νth outer iteration the stopping condition checks whether a local minimum of the object function $\Phi_\alpha(A, W, H)$ of (3) has been sufficiently well approximated by monitoring $\epsilon_S^{(\nu)}$ and $\delta^{(\nu)}$, where

$$\epsilon_S^{(\nu)} = \frac{\|A - W^{(\nu)} W^{(\nu)T}\|_F}{\|A\|_F} \tag{14}$$

measures the objective function of problem (2) and

$$\delta^{(\nu)} = \frac{\|W^{(\nu)} - H^{(\nu)}\|_F}{\min(\|W^{(\nu)}\|_F, \|H^{(\nu)}\|_F)} \tag{15}$$

measures the *degree of symmetry*. For the stopping condition two parameters τ_1 and τ_2 are used as tolerances. Their values (see the caption of Fig. 1) have been tuned through a preliminary experimentation on a reduced set of test problems.

The starting points $W^{(0)}$ and $H^{(0)}$ are required by the first call of the inner method. Both BPP and GCD can start with $H^{(0)} = O$, because the gradient of the objective function in (12) is

$$G(H) = H(W^{(\nu-1)T} W^{(\nu-1)} + \alpha I_k) - (A + \alpha I_n) W^{(\nu-1)}$$

and evaluated in the starting point is

$$G\big(H^{(0)}\big) = -(A + \alpha I_n) W^{(0)} \leq O.$$

As $W^{(0)}$ we suggest a matrix of the form $R\sqrt{\|A\|_F}/\|R\|_F$, where R is a random matrix with entries in $[0, 1]$. Moreover, the function Sym_ANLS needs a procedure for updating the value of α.

Let (W_α, H_α) be the solution of (3). The value of α influences the symmetry of the solution: the largest α, the smallest $\|W_\alpha - H_\alpha\|$, but a too large α, with respect to the magnitude of A, could lead to a poor solution. If $W_\alpha = H_\alpha$ for some α, then W_α is also a solution of (2) and we call it a *symmetric* solution of (3). We call *quasi-symmetric* solution of (3) a solution with $W_\alpha \sim H_\alpha$ and $\|A - W_\alpha H_\alpha^T\|^2$ dominating over $\alpha\|W_\alpha - H_\alpha\|^2$. In this case W_α is assumed as a good approximation of the solution of (2).

If a quasi-symmetric solution exists, it is possible that the convergence to it is achieved even for a small α, provided that the starting point $W^{(0)}$ is sufficiently close to the quasi-symmetric solution. On the other hand, a too large value of α should be avoided because it tends to move the solution of (3) away from a minimum point of (2).

The sequence of penalizing parameters is constructed by setting

$$\alpha^{(\nu)} = \beta^{(\nu)} \max A, \quad \text{with} \quad \beta^{(0)} = 1, \tag{16}$$

where ν is the step index of the outer iteration. The starting value $\alpha^{(0)} = \max A$ is tuned according to the scale of A.

In [10] the parameter β is modified according to a geometric progression of the form $\beta^{(\nu)} = \zeta^\nu$ where the fixed ratio $\zeta = 1.01$ is suggested. Actually, the value of ζ is critical, as we will see in the experimentation, where the strategy proposed in [10] has been tested also with different values of ζ.

Instead of considering an increasing sequence $\beta^{(\nu)}$, we suggest to modify the parameter $\beta^{(\nu)}$ in (16) using an adaptive strategy, called ADA, which takes into account the following quantities

function ADA $\left(W, H, \delta, \epsilon_S, \beta\right)$
computes the new value of β.

compute ϵ_N and ρ according to (17) and (18);
if $\rho < 1$ and $\beta > 8$ and ($\delta < 0.01$ or $\rho < 0.8$) then $\beta = \beta/8$,
else if $\rho < 1$ and $\beta > 4$ and ($\delta < 0.1$ or $\rho < 0.9$) then $\beta = \beta/4$,
else if $\rho < 1$ and $\beta > 2$ then $\beta = \beta/2$,
else $\beta = \beta \min\left(8, \rho^2\right)$;
return β;

Fig. 2 The function ADA implements the proposed adaptive strategy

$$\epsilon_N^{(\nu)} = \frac{\|A - W^{(\nu)} H^{(\nu)T}\|_F}{\|A\|_F} \tag{17}$$

which measures the first component of the objective function of problem (3), and the ratio

$$\rho^{(\nu)} = \frac{\epsilon_S^{(\nu)}}{\epsilon_N^{(\nu)}}, \tag{18}$$

where $\epsilon_S^{(\nu)}$ is defined in (14). One might think that $\epsilon_S^{(\nu)} \geq \epsilon_N^{(\nu)}$, but this is not always true. In fact, if the stationary point to which the outer method converges is a symmetric solution of problem (3), substituting $H^{(\nu)}$ with $W^{(\nu)}$ can be seen as a sort of extrapolation, that may even decrease the error.

When $\epsilon_S^{(\nu)}$ is smaller than $\epsilon_N^{(\nu)}$, the value of $\beta^{(\nu)}$ can be safely reduced without risking an increase of the distance from the symmetric solution. Otherwise the value of β is increased depending on the value of $\rho^{(\nu)}$. More precisely, when $\rho^{(\nu)} < 1$ the decreasing rate of β is tuned by the value of $\rho^{(\nu)}$, the degree of symmetry $\delta^{(\nu)}$ and the magnitude of $\beta^{(\nu)}$. Since in this case the outer iteration is well directed towards a quasi-symmetric solution, the penalty condition can be relaxed without any risk. When $\rho^{(\nu)} > 1$, the value of β is updated by means of multiplication by $\rho^{(\nu)2}$, paying attention to avoid a too large increase. This adaptive strategy is implemented by function ADA, whose policy has been chosen among several possible alternatives through an ad hoc preliminary experimentation (see Fig. 2).

In Fig. 1, function Update denotes the function used to update β. When the geometrical updating is chosen,

$$\beta^{(\nu)} = \zeta\, \beta^{(\nu-1)}, \quad \text{for a fixed } \zeta.$$

When the adaptive updating is chosen,

$$\beta^{(\nu)} = \text{ADA}\left(W^{(\nu)}, H^{(\nu)}, \delta^{(\nu)}, \epsilon_S^{(\nu)}, \beta^{(\nu-1)}\right).$$

4 The experimentation

The experimentation has been performed with a 3.2GHz 8-core Intel Xeon W processor machine. It was carried out on three classes of problems: Class 1 and Class 3 deal with artificially generated datasets, Class 2 deals with real-world problems. Class 2 and Class 3 refer to graph clustering, where matrix A is the similarity matrix generated by a suitable nonnegative kernel function.

4.1 The datasets

Class 1: It consists of matrices of the form $A = VV^T$, where $V \in \mathbf{R}_+^{n\times p}$ has random elements uniformly distributed over [0, 1]. Namely, three matrices $R1$, $R2$ and $R3$ are generated with $n = 2000$ and $p = 20,\ 40,\ 80$.
Class 2: The matrices of this class are obtained starting from undirected weighted graphs associated to three real-world datasets of documents $\boldsymbol{m}_i$, $i = 1, \ldots, n$. The considered datasets are

(1) MC: a collection of $n = 1033$ medical abstracts from Medline (Medical Literature Analysis and Retrieval System Online). The vector $\boldsymbol{m}_i$ contains the frequencies in the ith abstract of the words belonging to a given dictionary of size 5735.
(2) F575: UMist Faces collection of $n = 575$ gray-scale 112×92 images of 20 different people [3]. The vector $\boldsymbol{m}_i$ is obtained by columnwise vectorizing the pixel matrix representing the ith image.
(3) F400: Olivetti Faces collection of $n = 400$ gray-scale 64×64 images of several different people, from the Olivetti database at ATT. As above, the vector $\boldsymbol{m}_i$ is obtained by columnwise vectorizing the pixel matrix representing the ith image.

Collection (1) was downloaded from
http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/
Collections (2) and (3) were downloaded from
http://www.cs.nyu.edu/~roweis/data.html
For collection (1) the similarity matrix A is constructed through the usually considered weights for text data, i.e. the cosine similarity between two documents

$$a_{i,j} = \frac{\boldsymbol{m}_i^T \boldsymbol{m}_j}{\|\boldsymbol{m}_i\|_2 \|\boldsymbol{m}_j\|_2}, \quad \text{for} \quad i \neq j, \quad \text{and} \quad a_{i,i} = 0. \tag{19}$$

For collections (2) and (3) the similarity matrix A is constructed using weights $e_{i,j}$ suitable for image data, followed by the normalized cut

$$a_{i,j} = d_i^{-1/2} e_{i,j} d_j^{-1/2} \text{ where } \quad d_i = \sum_{r=1}^{n} e_{i,r}, \quad \text{for} \quad i = 1, \ldots, n.$$

The weights are expressed through a Gaussian kernel of the form

$$e_{i,j} = \exp\left(- \frac{\|\boldsymbol{m}_i - \boldsymbol{m}_j\|_2^2}{\sigma^2} \right), \quad \text{for} \quad i \neq j, \quad \text{and} \quad e_{i,i} = 0, \tag{20}$$

where σ is a global scaling parameter, chosen as the mean value of the distances σ_i of the ith point $\boldsymbol{m}_i$ from its 7th nearest neighbor [16].
Class 3: The matrices of this class are obtained starting from undirected weighted graphs associated to four synthetic data sets of points $\boldsymbol{m}_i$, $i = 1, \ldots, n$ in $\mathbf{R}^2$ suggested in [13] (see Fig. 3): dataset WellSeparatedNoise (WSN) consists of five clusters generated with the same variance and noise points in the amount of 5% ; dataset SubClusters (SC) has three clusters, and two of them can be divided into subclusters; dataset SkewDistribution (SK) has three clusters with different dispersion; dataset DifferentDensity (DD) has clusters with different cardinality.

All the datasets are generated with $n = 1000$, 2000, 4000 and 8000 points, in order to assess the sensitivity of the algorithm to the increase of the dimensions. For each dataset the similarity matrix A is constructed as in Class 2 with the choice $\sigma^2 = 0.02 \max_{i,j} \|\boldsymbol{m}_i - \boldsymbol{m}_j\|_2^2$.

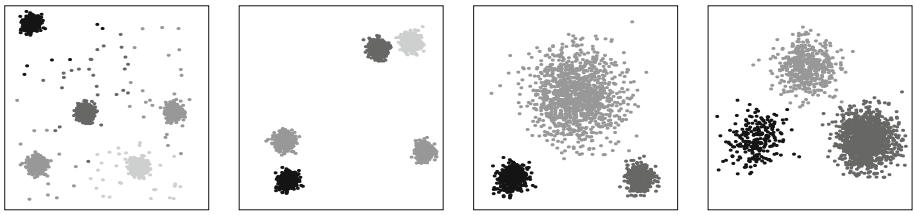

Fig. 3 Synthetic datasets of points in $\mathbf{R}^2$: from left to right WSN, SC, SK and DD

For the matrices of Classes 1 and 2 we look for factors W with ranks $k = 5$, 10, 20, 40 and 80. For the matrices of Class 3 we look for factors W with ranks $k = 3$, 5 and 10. By the term "problem" we mean a pair (A_n, k) where A_n is the given symmetric matrix of size n and k is the rank of the sought matrix W. Classes 1 and 2 consist of 15 problems each, while Class 3 consists of 48 problems.

4.2 The tests

Function `Sym_ANLS` calls four updating functions: function `ADA` and, for comparison, three geometrical updatings with ratio $\zeta = 1.01$ (proposed in [10], here denoted `G1.01`, which gives a slow progression), $\zeta = 1.1$ (here denoted `G1.1`, which gives a mid-level progression) and $\zeta = 1.4$ (here denoted `G1.4`, which gives the fastest progression). `BPP` and `GCD` methods are called as `Inner_solve`. `GCD` is called with different values of the tolerance η used in the stopping condition, namely $\eta = 10^{-\ell}$, with $\ell = 1, \ldots, 5$. Due to the fact that in general only approximations of a local minimum of problem (2) can be expected, for each problem and each instance of function `Sym_ANLS`, five randomly generated matrices $W^{(0)}$ have been considered as starting points and five runs have been performed in parallel. Since the final error is always available to the user, the solution with the best final error will be considered as the approximation of the global minimum.

The number of outer iterations and the final error of this solution are indicated as ν_{tot} and ϵ_S, while the running time in seconds of the five runs, indicated as T, is considered in order to estimate the cost of the whole processing.

The results presented in Tables 1 and 2 are obtained by averaging on the different problems of each class, including Class 2 despite the fact that its problems are inhomogeneous.

4.2.1 Testing the performance of `ADA`

The first set of experiments has a twofold aim: comparing the performances of `BPP` and `GCD`, and evaluating the strategy for updating β. Table 1 shows the values of ϵ_S, ν_{tot} and T, averaged on the problems of each class, for the inner methods `BPP` and `GCD` with $\eta = 10^{-3}$. The averaged results obtained with the other values of η are not listed since they, in comparison with the results of `BPP`, are pretty much the same of those shown for $\eta = 10^{-3}$.

While for each class all the methods appear to be quite equivalent from the point of view of the error, remarkable differences appear from the point of view of the number of outer iterations and of the required time. In general, `BPP` has a smaller number of outer iterations than `GCD`, but a much larger T, indicating that a single outer iteration of `BPP` costs much more than a single outer iteration of `GCD`. The time comparison shows that, at least in our experimentation, the exact local termination of `BPP` does not pay over the approximated

Table 1 Behavior of Sym_ANLS applied with the inner methods BPP and GCD with $\eta = 10^{-3}$, averaged on the problems of each class

Updating	Inner	Class 1			Class 2			Class 3		
		ϵ_S	ν_{tot}	T	ϵ_S	ν_{tot}	T	ϵ_S	ν_{tot}	T
ADA	BPP	0.010	6.	17.71	0.416	18.67	8.04	0.313	13.02	93.01
	GCD	0.010	16.73	16.96	0.415	17.20	1.62	0.309	14.90	13.68
G1.01	BPP	0.010	298.3	775.8	0.416	18.80	7.36	0.314	58.94	471.5
	GCD	0.010	309.6	195.8	0.415	18.40	1.62	0.309	60.67	61.13
G1.1	BPP	0.010	57.80	164.9	0.419	17.27	7.10	0.314	22.94	145.57
	GCD	0.010	58.73	37.47	0.418	17.27	1.51	0.309	23.77	21.55
G1.4	BPP	0.010	21.80	68.60	0.436	11.73	4.16	0.316	12.27	79.13
	GCD	0.010	21.80	14.05	0.435	11.87	1.06	0.309	12.06	9.92

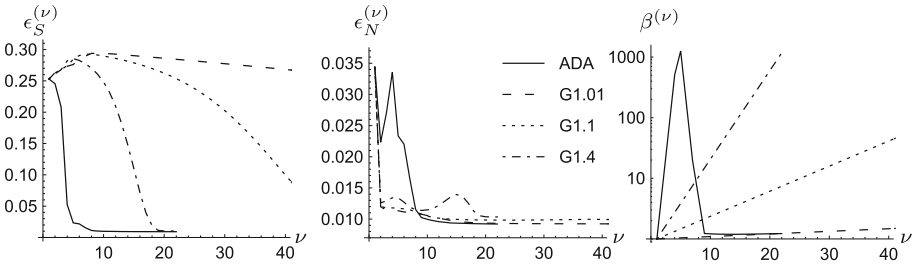

Fig. 4 Plots of $\epsilon_S^{(\nu)}$ (left), $\epsilon_N^{(\nu)}$ (center) and $\beta^{(\nu)}$ (right) versus the outer iteration number for problem R3 with $k = 80$. Sym_ANLS is applied with updating functions ADA, G1.01, G1.1 and G1.4. The inner method GCD is stopped with $\eta = 10^{-3}$

termination of the iterative method. For this reason in the following we do not consider BPP anymore.

Regarding the behavior of the strategies which implement a geometrical updating of $\beta^{(\nu)}$, it appears that it deeply depends on the chosen ratio ζ. In general a slower progression requires more time than a faster progression, with a possible advantage of the error. As a consequence, it can be very difficult to determine a suitable ratio of the updating which combines a low computational time with an acceptable error level. On the contrary, ADA adaptively produces a dynamical evolution of $\beta^{(\nu)}$ which on average leads to reasonable computational times and acceptable errors. As can be seen in Table 1, the geometrical updating G1.01 produces errors comparable with those of ADA but possibly larger computational times. On the other hand, the geometrical updating G1.4 produces computational times lower than those of ADA with the presence of larger errors.

The two following examples present typical situations where the choice of the ratio of the geometrical updating is critical. The first example (see Fig. 4) shows how a low rate geometrical updating is outperformed by ADA by the point of view of the cost. The second example (see Fig. 5) shows how a fast rate geometrical updating is outperformed by ADA by the point of view of the error.

Figure 4 shows the behaviors of $\epsilon_S^{(\nu)}$ (left plot), $\epsilon_N^{(\nu)}$ (middle plot) and $\beta^{(\nu)}$ (right plot) for problem R3 with $k = 80$ when Sym_ANLS is applied with inner method GCD with $\eta = 10^{-3}$. The plots are obtained with the updating functions ADA, G1.01, G1.1 and G1.4.

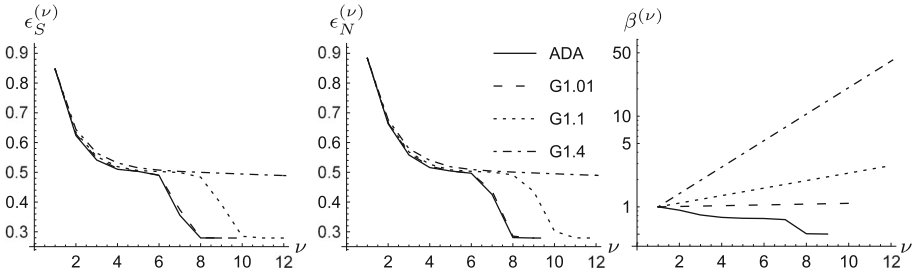

Fig. 5 Plots of $\epsilon_S^{(\nu)}$ (left), $\epsilon_N^{(\nu)}$ (center) and $\beta^{(\nu)}$ (right) versus the outer iteration number for problem WSN with $n = 1000$ and $k = 5$. `Sym_ANLS` is applied with updating functions `ADA`, `G1.01`, `G1.1` and `G1.4`. The inner method `GCD` is stopped with $\eta = 10^{-2}$

By inspection of the $\epsilon_S^{(\nu)}$ and $\epsilon_N^{(\nu)}$ plots, it appears that the adaptive strategy (solid line) produces a transition from a local minimum of the nonsymmetric error to another local minimum of both the nonsymmetric and the symmetric errors. This transition is obtained through a fast increase followed by a fast decrease of $\beta^{(\nu)}$. The same final error ϵ_S is obtained by `ADA` and all the geometric updatings but with very different numbers of iterations. For example, when `Sym_ANLS` is combined with the low rate geometrical updating `G1.01` the final error is obtained with 317 outer iterations (not shown in the plot).

Figure 5 shows the behaviors of $\epsilon_S^{(\nu)}$ (left plot), $\epsilon_N^{(\nu)}$ (middle plot) and $\beta^{(\nu)}$ (right plot) for problem WSN with $n = 1000$ and $k = 5$, when `Sym_ANLS` is applied with inner method `GCD` with $\eta = 10^{-2}$. The plots are obtained with the updating functions `ADA`, `G1.01`, `G1.1` and `G1.4`. For this problem the $\epsilon_S^{(\nu)}$ and $\epsilon_N^{(\nu)}$ plots are very similar from the beginning, so $\beta^{(\nu)}$ is monotonically decreased by `ADA`. It appears that the adaptive strategy produces a transition from a local minimum of the symmetric error with $\epsilon_S^{(\nu)} = 0.5$ to another local minimum, assumed as final, with $\epsilon_S = 0.28$ with 9 iterations. This transition is obtained with a decrease of $\beta^{(\nu)}$ which reaches and maintains the value 0.5. When `Sym_ANLS` is combined with the fast rate geometrical updating `G1.4`, it obtains the final error $\epsilon_S = 0.48$ comparable with the one of the first minimum obtained by using function `ADA`, i.e. a worst error is obtained with comparable time.

In conclusion, when compared to geometrical updatings, `ADA` produces a dynamical evolution of $\beta^{(\nu)}$ which leads to comparable computational times and errors, without the need of any a priori choice of the critical parameter ζ.

4.2.2 Analyzing the performance of `GCD` in the `Sym_ANLS` schema

The second set of experiments is aimed at examining how the choice of the tolerance η used in the stopping condition of `GCD` influences the performance. Table 2 shows the values of ϵ_S, ν_{tot} and T, averaged on the problems of each class, for the inner method `GCD` with $\eta = 10^{-\ell}$, $\ell = 1, \ldots, 5$.

It is evident that the influence of η on ϵ_S is negligible, while η affects the time. In general, a smaller η entails a smaller number of outer iterations, but the consequences on the computational time are not immediate and require a deeper analysis.

At each outer iteration ν of `Sym_ANLS` the two matrices $H^{(\nu)}$ and $W^{(\nu)}$ are computed by using `GCD`, which has an initialization phase where the gradient and the Hessian of the objective function and the quantity μ of (11) are computed. After the initialization phase, each inner iteration performs a single coordinate correction. In a standard implementation,

Table 2 Behavior of Sym_ANLS applied with updating ADA and inner method GCD with different η, averaged on the problems of each class

η	Class 1			Class 2			Class 3		
	ϵ_S	ν_{tot}	T	ϵ_S	ν_{tot}	T	ϵ_S	ν_{tot}	T
10^{-1}	0.010	40.8	29.98	0.419	38.27	4.02	0.308	18.54	18.49
10^{-2}	0.010	19.60	15.95	0.416	20.13	1.95	0.309	14.58	13.71
10^{-3}	0.010	16.73	16.96	0.415	17.20	1.62	0.309	14.90	13.68
10^{-4}	0.010	15.33	17.87	0.416	16.60	1.63	0.308	14.27	14.11
10^{-5}	0.010	13.80	19.81	0.415	17.40	1.72	0.309	14.21	13.64

Table 3 Behavior of Sym_ANLS applied with updating ADA and inner method GCD with different η, on problem R3 with $k = 80$

η	ϵ_S	ν_{tot}	cor/1000	T_{tot}	T_{inn}	T
10^{-1}	0.009	64	8216	105.4	2.46	122.1
10^{-2}	0.009	24	31666	47.28	9.09	50.75
10^{-3}	0.009	22	50020	50.20	14.55	51.98
10^{-4}	0.009	22	62609	53.38	18.03	53.38
10^{-5}	0.009	22	91755	61.67	26.28	62.45

Table 4 Behavior of Sym_ANLS applied with updating ADA and inner method GCD with different η, on problem MC with $k = 80$

η	ϵ_S	ν_{tot}	cor/1000	T_{tot}	T_{inn}	T
10^{-1}	0.360	58	243	24.69	0.221	37.64
10^{-2}	0.358	27	582	11.55	0.339	11.57
10^{-3}	0.356	23	1238	10.13	0.595	10.98
10^{-4}	0.356	23	1979	10.38	0.859	11.71
10^{-5}	0.355	23	2856	10.66	1.167	13.42

the initialization phase requires a number of floating point operations $\Gamma(k, n)$ of order $k\,n^2$ and each coordinate correction requires a number of floating point operations $\gamma(k)$ of order k. Hence for each problem, besides the number of outer iterations ν_{tot}, also the total number cor of single coordinate corrections on both $H^{(\nu)}$ and $W^{(\nu)}$, $\nu = 1, \ldots, \nu_{tot}$, has to be considered. Of course, both ν_{tot} and cor depend on η, then the overall cost of a run of Sym_ANLS can be expressed as

$$c_{tot}(\eta) = c_{out}(\eta) + c_{inn}(\eta), \text{ where } c_{out}(\eta) = 2\,\nu_{tot}\,\Gamma(k, n),\ c_{inn}(\eta) = cor\,\gamma(k). \quad (21)$$

In order to analyze how $c_{tot}(\eta)$ depends on the choice of η, a specific experimentation is made on three problems, one for each class. Namely, the considered problems are R3 of Class 1, MC of Class 2 and WSN with $n = 8000$ of Class 3. The values chosen for k are $k = 80$ for the first two cases and $k = 10$ for the third case.

The following tables show the behavior of Sym_ANLS applied with updating ADA and inner method GCD with different η. For each problem the solution with the best final error has been selected among the performed five runs. In Tables 3, 4 and 5 the total and inner running times in seconds spent to obtain this selected solution are denoted by T_{tot} and T_{inn} (they correspond to $c_{tot}(\eta)$ and $c_{inn}(\eta)$).

Table 5 Behavior of Sym_ANLS applied with updating ADA and inner method GCD with different η, on problem WSN with $n = 8000$ and $k = 10$

η	ϵ_S	ν_{tot}	cor/1000	T_{tot}	T_{inn}	T
10^{-1}	0.115	32	132	103.2	0.050	103.2
10^{-2}	0.115	25	368	82.82	0.051	141.1
10^{-3}	0.115	19	621	60.45	0.062	73.97
10^{-4}	0.116	13	958	42.14	0.074	54.29
10^{-5}	0.115	17	1520	51.04	0.105	64.26

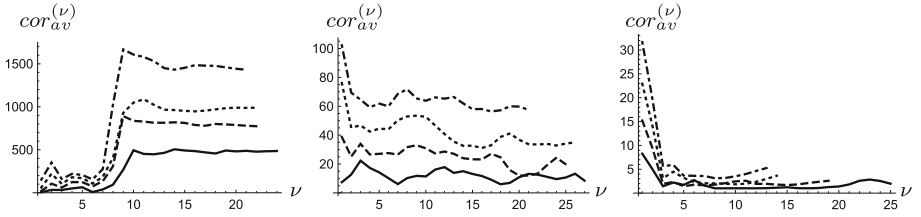

Fig. 6 Plots of $cor_{av}^{(\nu)}$ for problem R3 with $k = 80$ (left), problem MC with $k = 80$ (middle), problem WSN with $n = 8000$ and $k = 10$ (right). The inner tolerance is $\eta = 10^{-2}$ (continue line), $\eta = 10^{-3}$ (dashed line), $\eta = 10^{-4}$ (dotted line), $\eta = 10^{-5}$ (dashed-dotted line)

In the three tables the values of the T_{inn} column are significantly smaller than the corresponding values of the T_{tot} column and, when k is small, the T_{inn} column is negligible compared to the T_{tot} column. This result agrees with the theoretical estimate (21), taking into account the values of ν_{tot} and cor, and it shows that the inner phase contributes to the cost less than the outer phase. From Tables 3 and 4 it appears that decreasing values of η induce a nonincreasing number of outer iterations ν_{tot} and an increasing number of total corrections cor. As a consequence, also T_{inn} increases. The initial decrease of η leads to a decrease of T_{tot}, since the decrease of the outer cost prevails on the increase of the inner cost. For smaller values of η, the number of outer iterations stabilizes leading to an increasing T_{tot}. This behavior is less evident in Table 5 which refers to a problem where a small k is coupled with a much larger n.

An analysis of T_{tot} would suggest that an intermediate value for η appears to be a good choice. However, the algorithm is called with five different starting points $W^{(0)}$, and the T column, showing the running time cost, represents the effective cost in our parallel environment. Of course T is greater than T_{tot}, but typically shares the same behavior of T_{tot}, and gives the same suggestion for the choice of η, confirming what was already shown in Table 2 on average for the problems of each class.

To have a better insight on the error behavior, for each selected problem and each η, the error ϵ_{av} averaged on five runs has been computed. In any case the difference between ϵ_{av} and ϵ_S is less than 1%.

A better understanding of the cost behavior of the inner phase varying η can be obtained through Fig. 6, where $cor_{av}^{(\nu)}$ is the number of the coordinate corrections of W and H performed at the νth outer iteration, divided by $2n$. For each figure, these average behaviors corresponding to $\eta = 10^{-\ell}$ with $\ell = 2, \ldots, 5$ and starting with the same initial $W^{(0)}$ are shown. The figures show in a greater details how the results of the previous tables are formed. In any case we can see that a larger number of outer iterations corresponds to a lower number of average corrections. The steep increase of $cor_{av}^{(\nu)}$ in the first figure happens at the same

time of the analogous increase of $\beta^{(\nu)}$ and corresponds to a change of the local minimum point (see Fig. 4).

5 Conclusions

In this paper an adaptive strategy, called ADA, has been introduced for the updating of the parameter α in the penalized nonsymmetric minimization problem (3), when such problem is solved by applying an ANLS method. An extensive experimentation has shown that, when compared to geometrical updatings, ADA produces a dynamical evolution of the penalizing parameter $\alpha^{(\nu)}$ which leads to comparable computational times and errors, without the need of any a priori choice of the critical parameter ζ. Moreover, both BPP and GCD have been tested as inner solvers in the Sym_ANLS schema, concluding that the latter outperforms the former from the point of view of the computational cost and that an intermediate value of the internal tolerance η, i.e $\eta = 10^{-2}$ or 10^{-3}, should be preferred with GCD, especially when k is not very small in comparison with n.

References

1. Björck, Å.: Numerical Methods for Least Squares Problems. SIAM, Philadelphia (1996)
2. Cichocki, A., Phan, A.H.: Fast local algorithms for large scale nonnegative matrix and tensor factorizations IEICE. Trans. Fundam. E92-A(3), 708–721 (2009)
3. Graham, D.B., Allinson, N.M.: Characterizing virtual eigensignatures for general purpose face recognition. In: Wechsler, H., Phillips, P.J., Bruce, V., Fogelman-Soulie, F., Huang, T.S. (eds.) Face Recognition: From Theory to Applications; NATO ASI Series F, Computer and Systems Sciences, vol. 163, pp. 446–456. Springer, Berlin (1998)
4. Grippo, L., Sciandrone, M.: On the convergence of the block nonlinear Gauss–Seidel method under convex constraints. Oper. Res. Lett. **26**, 127–136 (2000)
5. Hsieh, C.J., Dhillon, I.S.: Fast coordinate descent methods with variable selection for non-negative matrix factorization. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1064–1072 (2011)
6. Kim, H., Park, H.: Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. SIAM J. Matrix Anal. Appl. **30**, 713–730 (2008)
7. Kim, H., Park, H.: Fast nonnegative matrix factorization: an active-set-like method and comparisons. SIAM J. Sci. Comput. **33**, 3261–3281 (2011)
8. Kim, J., He, Y., Park, H.: Algorithms for nonnegative matrix and tensor factorization: an unified view based on block coordinate descent framework. J. Glob. Optim. **58**, 285–319 (2014)
9. Kuang, D., Ding, C., Park, H.: Symmetric nonnegative matrix factorization for graph clustering. In: The 12th SIAM International Conference on Data Mining (SDM '12), pp. 106–117 (2012)
10. Kuang, D., Yun, S., Park, H.: SymNMF: nonnegative low-rank approximation of a similarity matrix for graph clustering. J. Glob. Optim. **62**, 545–574 (2015)
11. Lawson, C.L., Hanson, R.J.: Solving Least Squares Problems. Prentice-Hall, Englewood Cliffs (1974)
12. Lin, C.J.: Projected gradient methods for nonnegative matrix factorization. Neural Comput. **19**, 2756–2779 (2007)
13. Liu, Y., Li, Z., Xiong, H.: Understanding and enhancement of internal clustering validation measures. IEEE Trans. Cybern. **43**, 982–993 (2013)
14. Paatero, P., Tappert, U.: Positive matrix factorization: a non-negative factor model with optimal solution of error estimates of data values. Environmetrics **5**, 111–126 (1994)
15. Pauca, V.P., Piper, J., Plemmons, R.J.: Nonnegative matrix factorization for spectral data analysis. Linear Algebra Appl. **416**, 29–47 (2006)
16. Zelnik-Manor, L., Perona, P.: Self-tuning spectral clustering. Adv. Neural Inf. Process. Syst. **17**, 1601–1608 (2004)