algorithms

MDPI

Article

Adaptive Clustering via Symmetric Nonnegative Matrix Factorization of the Similarity Matrix

Paola Favati [1,*], Grazia Lotti [2], Ornella Menchi [3] and Francesco Romani [3]

[1] Istituto di Informatica e Telematica-Consiglio Nazionale delle Ricerche (IIT-CNR), Via G. Moruzzi 1, 56124 Pisa, Italy

[2] Dipartimento di Matematica, University of Parma, Parco Area delle Scienze 53/A, 43124 Parma, Italy; grazia.lotti@unipr.it

[3] Dipartimento di Informatica, University of Pisa, Largo Pontecorvo 3, 56127 Pisa, Italy; menchi@di.unipi.it (O.M.); romani@di.unipi.it (F.R.)

* Correspondence: paola.favati@iit.cnr.it

Received: 12 September 2019; Accepted: 15 October 2019 ; Published: 17 October 2019

Abstract: The problem of clustering, that is, the partitioning of data into groups of similar objects, is a key step for many data-mining problems. The algorithm we propose for clustering is based on the symmetric nonnegative matrix factorization (SymNMF) of a similarity matrix. The algorithm is first presented for the case of a prescribed number k of clusters, then it is extended to the case of a not a priori given k. A heuristic approach improving the standard multistart strategy is proposed and validated by the experimentation.

Keywords: clustering; nonnegative matrix factorization; adaptive strategy

1. Introduction

Clustering can be applied to a variety of different kinds of documents as long as a distance measure can be assigned among the data objects. Generally, a similarity function assigns to pairs of closely related objects with a higher similarity than to objects which are only weakly related. A clustering technique classifies the data into groups, called clusters, in such a way that objects belonging to a same cluster are more similar to each other than to objects belonging to different clusters.

In this paper we assume that the data objects are n distinct points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ in $\mathbb{R}^d$ and denote by k, with $1 < k \ll n$, the number of clusters. As usual in this context, we consider the similarity expressed through the Gaussian kernel [1]

$$e_{i,r} = \exp\Big(-\frac{\|\boldsymbol{p}_i - \boldsymbol{p}_r\|_2^2}{\sigma\mu}\Big), \quad \text{where} \quad \mu = \max_{j,s} \|\boldsymbol{p}_j - \boldsymbol{p}_s\|_2^2, \tag{1}$$

and $\sigma > 0$ is a parameter based on the scale of the points.

The nonnegative symmetric matrix $A \in \mathbb{R}_+^{n\times n}$, whose elements are

$$a_{i,r} = d_i^{-1/2} e_{i,r} d_r^{-1/2}, \quad \text{where } d_i = \sum_{s=1}^{n} e_{i,s}, \quad \text{for} \quad i,r = 1,\ldots,n, \tag{2}$$

is called the *similarity matrix*.

Among the many different techniques devised to solve the clustering problem, we chose a method which performs a dimensionality reduction through the *Nonnegative Matrix Factorization* (NMF) of matrix A. NMF finds an approximate factorization of A by means of two low-rank, nonnegative matrices. In its basic form, the problem is set as follows: given a matrix $A \in \mathbb{R}_+^{m\times n}$

and an integer $k \ll n$, we look for two low-rank matrices $W \in \mathbb{R}_+^{m \times k}$ (the *basis* matrix) and $H \in \mathbb{R}_+^{n \times k}$ (the *coefficient* matrix), such that the product WH^T approximates A according to

$$\min_{W,H \geq O} f(W,H), \text{ with } f(W,H) = \tfrac{1}{2} \|A - W H^T\|_F^2, \tag{3}$$

where the subscript F denotes the Frobenius norm. Actually, if the nonnegativity condition was not imposed, the minimum of $f(W,H)$ could be obtained by truncating the Singular Value Decomposition $A = U\Sigma V^T$. Unfortunately, some entries of U and V would be negative, and this would prevent interpretation of the factorization in terms of clustering [1]. In fact, since each column of A can be approximately represented by a linear combination with coefficients in H of columns of W, thanks to the nonnegativity feature, the elements of H, properly normalized, could be seen as the probabilities of assignment of the points to the different clusters. The clustering can be obtained by assigning the ith point $\boldsymbol{p}_i$ to the jth cluster corresponding to the largest coefficient $h_{i,j}$ in the ith row of H.

The problem of computing NMF was first proposed in [2] for data analysis and, from then on, it has received much attention. The first algorithm to compute NMF was proposed under the name of multiplicative updating rule in [3], but its slow convergence suggested looking for more efficient algorithms. Many of them belong to the class of Block Coordinate Descent (BCD) methods [4–8], but other iterative methods for nonlinear optimization have also been suggested, based on some gradient descent procedure. To improve the convergence rate, a quasi-Newton strategy has also been considered, coupled with nonnegativity (see, for example, the Newton-like algorithm of [9]). Open source libraries for computing NMF are available—see, for example, [10], where several methods proposed in related works are implemented. Over the years, many variants of the basic problem (3) have been considered. The case of a symmetric matrix A has been deeply investigated, but other features have also been taken into consideration. For example, sparsity and/or regularizing constraints can be easily embedded [5], classification labels can be incorporated through a supervisor [11], and extensions to tensor nonnegative factorization have been studied [12].

Since our goal was to obtain clustering through the symmetric matrix A, we restricted our analysis to the symmetric NMF problem.

When a clustering problem is proposed, the dimension of the clustering, that is, the number k of the clusters, can be either given a priori or looked for. In this paper we consider both cases:

- problem $\mathcal{P}_{kfix}$, with an a priori assigned rank k of matrix W. The matrix W is computed using NMF and the clustering is obtained.
- $\mathcal{P}_{kvar}$, where an interval $\mathcal{J}_k = [k_{\min}, k_{\max}]$ is assigned. The number k varies in $\mathcal{J}_k$ and the best k, according to a given criterium, has to be found.

The algorithm solving $\mathcal{P}_{kfix}$ is a constitutive brick of a larger procedure for $\mathcal{P}_{kvar}$. When dealing with $\mathcal{P}_{kvar}$, the factorization of A must be performed for each k in $\mathcal{J}_k$, and a validity index must be provided to compare the clusterings obtained with different k to decide which is the best one.

Our contribution consists in devising an algorithm which implements a heuristic approach by improving a multistart policy. The algorithm, especially suitable when working in a multitasking/multiprocessor environment, lets the computation efforts be concentrated on the more promising runs, without any a priori knowledge. Each run applies a BCD technique known as alternating nonnegative least squares, which replaces the BPP method used in [13] by the GCD method of [7]. This choice, as shown in a previous paper [14], leads to a saving in the computational cost of the single run. The suggested heuristics enjoys the same effectiveness of the standard implementation of a multistart approach, with a considerable reduction of the cost. Its only drawback is more complicated coding.

The paper is organized as follows: Section 2 is devoted to recall the alternating nonnegative least squares procedure for NMF with both general matrices and symmetric matrices, constituting the basis of our proposed algorithm. Section 3 describes the validity indices used for comparison criteria.

Next, in Section 4, the algorithm for problem $\mathcal{P}_{kfix}$ is presented, with a priority queue implementation to reduce the computational time. The extension of the algorithm to problem $\mathcal{P}_{kvar}$, described in Section 5, requires particular attention to the scale parameter σ chosen for the construction of A by using (1) and (2). The experiments are described and commented on in Section 6.

Notations: the (i,j)th element of a matrix M is denoted by $(M)_{i,j}$ or by the lowercase $m_{i,j}$. The notation $M \geq O$ means that all the elements of M are nonnegative. Several parameters enter in the description of the algorithm. In the text, where they are introduced, the notation "(Section 6)" appears, meaning that the values assigned to them in the experiments are listed at the beginning of Section 6.

2. The Alternating Nonnegative Least Squares for NMF

Problem (3) is nonconvex and finding its global minimum is NP-hard [15]. In general, nonconvex optimization algorithms guarantee only the stationarity of the limit points, and only local minima can be detected. Most iterative algorithms for solving (3) are based on a simple block nonlinear Gauss–Seidel scheme [4,5]. An initial $W_0 \in \mathbb{R}_+^{m\times k}$ is chosen, and the sequences

$$\begin{cases} H_\nu = \underset{H\geq 0}{\operatorname{argmin}} \ \frac{1}{2}\ \|A - W_{\nu-1}\, H^T\|_F^2, \\ W_\nu = \underset{W\geq 0}{\operatorname{argmin}} \ \frac{1}{2}\ \|A^T - H_\nu\, W^T\|_F^2, \end{cases} \tag{4}$$

are computed for $\nu = 1, 2, \ldots$, until a suitable stopping condition is satisfied, such as by checking whether a local minimum of the objective function $f(W,H)$ has been sufficiently well-approximated. The convergence of this scheme, called Alternating Nonnegative Least Squares (`ANLS`), follows from [16].

Although the original problem (3) is nonconvex, the two subproblems of (4) are convex. Let us denote by $\varphi(X)$ the objective function to be minimized in both cases,

$$\varphi(X) = f(Y,X) = \tfrac{1}{2}\ \|B - Y\,X^T\|_F^2, \tag{5}$$

where $B = A$ and $Y = W_{\nu-1}$ for the first subproblem in (4), and $B = A^T$ and $Y = H_\nu$ for the second subproblem in (4). Problem (5) can be dealt with by applying aprocedure for nonnegatively constrained least squares optimization, such as an Active-Set-like method [13,17–19] or an iterative inexact method as a gradient descent method modified to satisfy the nonnegativity constraints. We have performed in [14] a preliminary ad hoc experimentation, comparing an Active-Set-like method (the one called `BPP` and coded as Algorithm 2 in [13]) with the Greedy Coordinate Descent method, called `GCD` in [7]. The results have shown that in our context, `GCD` is faster and equally reliable. `GCD`, specifically designed for nonnegative constrained least squares problems, implements a selection policy of the elements updated during the iteration, based on the largest decrease of the objective function. We give here a brief description of `GCD` (the corresponding code can be found as Algorithm 1 in [7]).

The gradient of $\varphi(X)$ is

$$G(X) = \left(X\,Y^T - B^T\right)Y = XQ - B^TY, \quad \text{where} \quad Q = Y^TY.$$

`GCD` computes a sequence of matrices X_j, $j = 1, 2, \ldots$, until termination. Initially, X_0 is set equal to $H_{\nu-1}$ for the first subproblem in (4) and to $W_{\nu-1}$ for the second subproblem in (4) (with $H_0 = O$, which ensures that $G(X_0) \leq O$). At the jth iteration, matrix X_j is obtained by applying a single coordinate correction to the previous matrix X_{j-1} according to the rule $X_j = X_{j-1} + s\,\boldsymbol{e}_r\boldsymbol{e}_i^T$, where r and i are suitably selected indices and $\boldsymbol{e}_r$ and $\boldsymbol{e}_i$ are the rth and ith canonical vectors of compatible lengths. The scalar s is determined by imposing that $\varphi(X_j)$, as a function of s, is the minimum on the set

$$S = \{s \text{ such that } (X_{j-1})_{r,i} + s \geq 0\}.$$

Denoting $g_{r,i} = \big(G(X_{j-1})\big)_{r,i}$ and $q_{i,i}$ the ith principal element of Q, the value which realizes the minimum of $\varphi(X_j)$ on S is

$$\widehat{s} = -\frac{g_{r,i}}{q_{i,i}} \quad \text{if} \quad \frac{g_{r,i}}{q_{i,i}} \leq (X_{j-1})_{r,i} \quad \text{and} \quad \widehat{s} = -(X_{j-1})_{r,i} \quad \text{otherwise.}$$

In correspondence, the objective function is decreased by

$$\theta_{r,i}^{(j)} = \varphi(X_{j-1}) - \varphi(X_j) = -g_{r,i}\,\widehat{s} - \frac{1}{2}\,q_{i,i}\,\widehat{s}^2.$$

A natural choice for indices (r,i) would be the one that maximizes $\theta_{r,i}^{(j)}$ on all the pairs (r,i), but it would be too expensive. So it is suggested to proceed by rows. For a fixed $r = 1,\ldots,n$, the index i is chosen as the one that maximizes $\theta_{r,i}^{(j)}$ on $i \in [1,k]$ and the (r,i)th element of X_{j-1} is updated. Then a new index i is detected, and so on, until a stopping condition is met, such as

$$\max_i \theta_{r,i}^{(j)} < \eta_1\,\theta_{\max},$$

where η_1 is a preassigned tolerance (Section 6) and $\theta_{\max} = \max_{r,i}\theta_{r,i}^{(1)}$ is the largest possible reduction of the objective function that can be expected when a single element is modified in X_0.

The Symmetric Case

In our case, the similarity matrix $A \in \mathbb{R}_+^{n\times n}$ is symmetric; hence, the general NMF problem (3) is replaced by the symmetric NMF (SymNMF) problem

$$\min_{W\geq O} f(W), \text{ with } f(W) = \tfrac{1}{2}\,\|A - W\,W^T\|_F^2, \quad \text{with} \quad W \in \mathbb{R}_+^{n\times k}. \tag{6}$$

Since A may generally be indefinite, while WW^T is positive semidefinite, WW^T represents a good factorization of A if A has enough nonnegative eigenvalues. When this happens, we are led to believe that W naturally captures the cluster structure hidden in A [1,9]. Thanks to the nonnegativity, the largest entry in the ith row of W is assumed as the clustering assignment of the ith point—that is, $\boldsymbol{p}_i$ belongs to the jth cluster if $w_{ij} = \max_{r=1,\ldots,k} w_{ir}$. This interpretation furnishes a clustering Π induced by the matrix W.

In [9], this approach, which efficiently solves the clustering problem, is shown to be competitive with the widely used spectral clustering techniques, which perform a dimensionality reduction by means of the eigenvectors corresponding to the leading eigenvalues of A.

Problem (6) has a fourth-order nonconvex objective function. Following [1], we suggest solving it through a nonsymmetric penalty problem of the form

$$\min_{W,H\geq O} f_\alpha(W,H), \text{ with } f_\alpha(W,H) = \tfrac{1}{2}\,\Big(\|A - WH^T\|_F^2 + \alpha\|W - H\|_F^2\Big), \tag{7}$$

α being a positive parameter which acts on the violation of the symmetry. Problem (7) can be tackled by using any of the techniques proposed in the literature for solving the general NMF problem (3). For example, algorithm `ANLS` can be applied by alternating the solution of the two subproblems

$$\begin{cases} H_\nu = \underset{H\geq 0}{\operatorname{argmin}}\ \frac{1}{2}\left\| \begin{bmatrix} A \\ \sqrt{\alpha_\nu}\,W_{\nu-1}^T \end{bmatrix} - \begin{bmatrix} W_{\nu-1} \\ \sqrt{\alpha_\nu}\,I_k \end{bmatrix} H^T \right\|_F^2, \\ W_\nu = \underset{W\geq 0}{\operatorname{argmin}}\ \frac{1}{2}\left\| \begin{bmatrix} A \\ \sqrt{\alpha_\nu}\,H_\nu^T \end{bmatrix} - \begin{bmatrix} H_\nu \\ \sqrt{\alpha_\nu}\,I_k \end{bmatrix} W^T \right\|_F^2. \end{cases} \tag{8}$$

In [1], the sequence of penalizing parameters is constructed by setting

$$\alpha_\nu = \beta_\nu \max A, \quad \text{with} \quad \beta_0 = 1,$$

and β_ν modified according to the geometric progression $\beta_\nu = \zeta^\nu$ with the fixed ratio $\zeta = 1.01$. We suggest, instead, to let β_ν be modified adaptively, as shown in [14].

Both problems (8) have a form similar to (5), with k rows involving the $\sqrt{\alpha_\nu}$, appended at the bottom. Hence, each problem is solved by applying `GCD` with gradient

$$G(X) = XY^TY - B^TY + \alpha(X - Y) = XQ - (B + \alpha I_n)^TY, \text{ where } Q = \alpha I_k + Y^TY.$$

The function which computes a single iteration of `ANLS` by applying `GCD` and modifying α_ν as described in [14], is thus called

$$\left(W_\nu, H_\nu, \alpha_\nu\right) = \texttt{Sym\_ANLS}\left(A, W_{\nu-1}, H_{\nu-1}, \alpha_{\nu-1}\right).$$

The whole iterative procedure is organized in the following function, `Sym_NMF`. The stopping condition tests the stabilization of the residuals $\epsilon_\nu = \|A - W_\nu W_\nu^T\|_F^2$ by using two parameters: a tolerance η_2 and a maximum number $\nu_{\max}$ of allowed iterations (Section 6).

`function` `Sym_NMF` $\left(A, W_0, H_0, \alpha_0, \nu_{\max}\right)$

$\nu = 0$; $\epsilon_0 = \|A - W_0 W_0^T\|_F^2$;
`repeat`
  $\nu = \nu + 1$;
  $\left\{W_\nu, H_\nu, \alpha_\nu\right\} = \texttt{Sym\_ANLS}\left(A, W_{\nu-1}, H_{\nu-1}, \alpha_{\nu-1}\right)$;
  $\epsilon_\nu = \|A - W_\nu W_\nu^T\|_F^2$;
  $cond = |\epsilon_\nu - \epsilon_{\nu-1}|/\epsilon_\nu \leq \eta_2$;
`until` $cond$ || $(\nu \geq \nu_{\max})$;
`return` $\left(\nu,\ W_\nu,\ H_\nu,\ \alpha_\nu,\ cond\right)$;

The effective dimension k_e of Π can be smaller than the expected k, because void clusters may turn out (see the following function `Partition`, where $\{\ \}$ denotes the void set).

`function` `Partition` $\left(W\right)$

$\pi_r = \{\ \}$ `for` $r = 1, \ldots, k$;
`for` $i = 1, \ldots, n$
  `let` j be such that $w_{i,j} = \max_{r=1,\ldots,k} w_{i,r}$;
  `append` i `to` π_j
`end for`;
$\Pi = [\pi_1, \ldots, \pi_k]$;
$k_e = k$;
`for` $r = 1, \ldots, k$
  `if` $\pi_r = \{\ \}$ `then` discard π_r; $k_e = k_e - 1$
`end for`;
`return` $\left(\Pi,\ k_e\right)$;

When applying `Sym_NMF` we expect that, together with the stabilization of the sequence of the residuals $\epsilon_\nu = \|A - W_\nu W_\nu^T\|_F^2$, a form of stabilization of the corresponding clustering should

also take place. Actually, we have verified experimentally that, in general, the stabilization of the clusterings proceeds irregularly, before the residuals ϵ_ν show a substantial reduction. This is the reason why `Sym_NMF` does not rely on early clustering stabilization, which could produce incorrect results, and exploits the decrease of the residuals.

3. The Validity Indices

Since the solution W of problem (6), computed by starting with different initial approximations W_0, may be only a local minimum, tools to evaluate the validity of the clustering Π induced by W are needed. Denoting by π_j the jth cluster, a clustering $\Pi = \{\pi_1, \ldots, \pi_k\}$, with $k \geq 2$, corresponds to a partitioning of the indices $\{1, \ldots, n\}$ (actually, the number k_e of clusters effectively obtained by applying `Partition`(W) can be smaller than the required k). A clustering algorithm generally aims at minimizing some function of the distances of the points of the jth cluster from its centroid defined by

$$\boldsymbol{c}_j = \frac{1}{n_j} \sum_{i \in \pi_j} \boldsymbol{p}_i, \quad \text{with} \quad n_j = \#\,\pi_j,$$

where the symbol # denotes the cardinality. Over the years, dozens of *validity* indices have been proposed (see [20] and its references). Most of them take into account the *compactness*, which measures the *intra-cluster* distances, and the *separability*, which measures the *inter-cluster* distances. A good clustering algorithm should produce small intra-cluster distances and large inter-cluster distances.

3.1. The DB Index

Both compactness and separability are combined by the following Davies-Bouldin index (DB) [21]:

$$\chi = \frac{1}{k_e} \sum_{j=1}^{k_e} \zeta_j, \quad \text{where} \quad \zeta_j = \max_{r \neq j} \frac{\gamma_j + \gamma_r}{d_{j,r}}, \quad \gamma_j = \frac{1}{n_j} \sum_{i \in \pi_j} \|\boldsymbol{p}_i - \boldsymbol{c}_j\|_2 \quad \text{and} \quad d_{j,r} = \|\boldsymbol{c}_j - \boldsymbol{c}_r\|_2. \tag{9}$$

A good clustering is associated to a small value of the DB index. In the codes, the function which uses (9) to compute the DB index χ of the clustering Π obtained by `Partition` (W), is called

$$\chi = \texttt{DBindex}\,(\Pi).$$

3.2. The DB** Index

When the validity of different clusterings corresponding to different dimensions k must be established, the DB index can still be used, but if the given points can be organized in different subclusters, it may be difficult to establish which one results in being the best one. In [22] another index, called DB**, is proposed to deal with this case. Unlike index DB, index DB** takes into account the compactness behavior of the clustering corresponding to different values of k in order to penalize unnecessary merging and to detect finer partitions.

We assume that a sequence of clusterings $\Pi^{(h)}$ of dimensions $k^{(h)}$ is given, with $h = 1, \ldots, h_{\max}$, such that $k^{(h)} < k^{(h+1)}$. Let $\boldsymbol{c}^{(h)}$ denote the vector of the centroids of $\Pi^{(h)}$, and $\boldsymbol{\gamma}^{(h)}$ the vector of $k^{(h)}$ components whose jth component is

$$\gamma_j^{(h)} = \frac{1}{n_j^{(h)}} \sum_{i \in \pi_j^{(h)}} \|\boldsymbol{p}_i - \boldsymbol{c}_j^{(h)}\|_2 \quad \text{where} \quad n_j^{(h)} = \#\,\pi_j^{(h)}.$$

For $h = 1, \ldots, h_{\max} - 1$, let $\boldsymbol{u}^{(h)}$ and $\boldsymbol{v}^{(h)}$ be the vectors of $k^{(h)}$ components whose jth component is

$$u_j^{(h)} = \max_{r \neq j} \left(\gamma_j^{(h)} + \gamma_r^{(h)}\right) - \max_{r \neq j} \left(\gamma_j^{(h+1)} + \gamma_r^{(h+1)}\right), \quad v_j^{(h)} = \max_{r=h,\ldots,h_{\max}-1} u_j^{(r)}.$$

DB** returns a vector χ^{**} of length $h_{\max}-1$ whose hth component is

$$\chi_h^{**} = \frac{1}{k^{(h)}} \sum_{j=1}^{k^{(h)}} \zeta_j^{(h)}, \quad \text{where} \quad \zeta_j^{(h)} = \frac{\max\limits_{r \neq j} \left(\gamma_j^{(h)} + \gamma_r^{(h)}\right) + v_j^{(h)}}{\min\limits_{r \neq j} d_{j,r}^{(h)}}.$$

The lowest component of χ^{**} detects the best clustering among the clusterings $\Pi^{(h)}$ with $h \in \{1, \ldots, h_{\max}-1\}$.

3.3. The CL Index

Both the DB index and the DB** index base their valuation of the clustering separability on the distances between centroids, according to the rule: the farther the centroids, the more separated the clusters. This rule does not appear profitable in the presence of nonconvex clusters or of clusters that nearly touch each other, or when the densities of the points differ much from cluster to cluster. This can be particularly troublesome when the points are affected by noise or when there are outliers. To check whether the rth and sth clusters nearly touch each other, one should compute their distance; that is, the minimum distance between each pair of points $\boldsymbol{p}_i \in \pi_r$ and $\boldsymbol{p}_j \in \pi_s$, with a not negligible increase in computational time. For example, this minimum distance between two clusters is used by the Dunn index [20].

To reduce the burden of this control, we propose a new index called CL, which provides a measure of closeness. First, a small integer p (Section 6) is selected, and for any point $\boldsymbol{p}_i$, the indices of its p nearest points are listed in ℓ_i. Using the same notations of the previous subsection, we assume that a sequence of clusterings $\Pi^{(h)}$ of dimensions $k^{(h)}$ is given, with $h = 1, \ldots, h_{\max}$. Let $\pi_r^{(h)}$ be the cluster to which $\boldsymbol{p}_i$ belongs. We consider the quantity

$$x_i^{(h)} = \sum_{j \in \ell_i,\ j \notin \pi_r^{(h)}} \exp\Big(-c\,\frac{\|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2^2}{\mu}\Big),$$

where c is a constant (Section 6) large enough to make negligible in the sum the contribution of far-away points not belonging to $\pi_r^{(h)}$. Set

$$y_h = \sum_{i=1}^{n} x_i^{(h)}, \quad h = 1, \ldots, h_{\max}.$$

The sequence y_h can be very oscillating. For this reason, we defined as the CL index the vector ψ_h obtained by computing the exponential moving average [23] with smoothing constant 0.1, of the vector y_h

$$\psi_1 = y_1, \quad \psi_{h+1} = \psi_h + 0.1(y_{h+1} - \psi_h), \quad h = 1, \ldots, h_{\max}-1.$$

This index is large when at least one point $\boldsymbol{p}_i$ exists, having some sufficiently close points not belonging to the same cluster of $\boldsymbol{p}_i$ in $\Pi^{(h)}$. Typically, the CL index has the following behavior: a zero ψ_h means that the clusters of $\Pi^{(h)}$ are well-separated, and this occurs only in the absence of noise. The increasing rate, which is bounded for small values of k, tends towards infinity when k exceeds values for which separated clusters exist. This behavior, together with DB** values, suggests when stopping the increase of k if $k_{\max}$ is not assigned.

4. The Clustering Algorithm for Problem $\mathcal{P}_{kfix}$

Given a fixed value of k, the algorithm `Kfix` we propose for problem $\mathcal{P}_{kfix}$ searches the minimum of (6). This algorithm will be used in Section 5 for solving problem $\mathcal{P}_{kvar}$. Before describing it, it is

necessary to deal with the important issue of how to choose a suitable value of the scale parameter σ for the construction of the similarity matrix A according to (1) and (2).

Values of σ that are too small or too large would reduce the variability of A, making all the points equally distant, resulting in badly mixed-up clusters. Since we must discriminate between close and far points, an intermediate value between the smallest and the largest distance of the points should be chosen. Clearly, there is a strong correlation between the value of σ and the number k of clusters that the algorithm is expected to find: more clusters may have a smaller size and may require a smaller σ to separate them. To this aim, a triplet $\boldsymbol{\sigma}^{(k)}$ of suitable values (Section 6) is detected and tested for any k.

Besides the choice of σ, the choice of the initial matrix W_0 may also be critical, due to the fact that only a local minimum of problem (6) can be expected. It is common practice to tackle this issue by comparing the results obtained by applying `Sym_NMF` to several matrices W_0, randomly generated with uniform distribution between 0 and 1. The number of the starting matrices is denoted q (Section 6).

For any pair $\{\sigma, W_0\}$ with $\sigma \in \boldsymbol{\sigma}^{(k)}$, matrix $A(\sigma)$ is constructed using (1) and (2) and the function `Sym_NMF` is applied starting with W_0. Then, the DB index χ of the clustering so obtained is computed. A trivial multistart implementation could consist in carrying out, up to convergence, function `Sym_NMF` for any pair $\{\sigma, W_0\}$ and choosing, at the end, the best χ.

In the following we present a more sophisticated version, suitable for large problems, which uses a heuristic approach. The run for any pair $\{\sigma, W_0\}$ is split in segments with a fixed number λ of iterations, and the execution scheduling is driven by the values χ already found. The possibility of discarding the less promising runs is considered. The proposed strategy, which is accomplished by using a priority queue, specializes to the present clustering problem the procedure given in [24], where the global minimization of a polynomial function is dealt with. In [24], the heuristic approach is shown to outperform the multistart one for what regards the computational cost, maintaining the same efficacy.

The items of the priority queue $\mathcal{Q}$ have the form

$$Y = \{r, t, W, H, \alpha, \xi\},$$

where

- r is the index which identifies the pair $\{\sigma, W_0\}$.
- $t = t^{(r)}$ is the number of iterations computed until now by `Sym_NMF` on the item with index r. Initially $t = 0$.
- $W = W^{(r)}$ and $H = H^{(r)}$ are the NMF factors of $A(\sigma)$ computed for the rth item. Initially $W = W_0$, $H = O$.
- $\alpha = \alpha^{(r)}$ is the penalizing parameter of the rth item. Initially $\alpha = \max A(\sigma)$.
- $\xi = \chi^{(r)} + t^{(r)}/t_{\max}$, where $\chi^{(r)}$ is the DB index defined in (9) for the rth item, and $t_{\max}$ (Section 6) is the maximum number of allowed iterations for the rth item. Initially $\xi = 0$.

The queue is ruled, according to the minimum policy, by the value of ξ, allowing to compute first the more promising items, keeping into account also the age of the computation. Initially, the priority queue contains $3\,q$ items. Also referred in the procedure, but not belonging to the queue, are

- an array $\boldsymbol{m}$ of length k, whose ith element contains the minimum of the DB indices computed so far for all the items returning $k_e = i$. Initially, the elements of $\boldsymbol{m}$ are set equal to ∞.
- an array M, containing in position i the partition Π corresponding to m_i. Initially, the elements of M are empty sets.

The belonging of an element to an item Y is denoted by using the subscript (Y). Moreover, $A_{(Y)}$ denotes the similarity matrix $A(\sigma)$ generated with the scale parameter σ associated with $r_{(Y)}$.

After the initialization of the queue, the adaptive process evolves as follows: an item Y is extracted from the queue according to its priority $\xi_{(Y)}$, and the function `Sym_NMF` is applied. A new item is so built and, if it is recognized as promising by a specific function `Control`, it is inserted back

into $\mathcal{Q}$. Otherwise, if Y is not promising, no new item is inserted back into $\mathcal{Q}$—that is, Y is discarded. Two bounds, $t_{\min}$ and $t_{\max}$ (Section 6), are used to control the execution flow for each item.

`function Kfix` $\left(k, \boldsymbol{\sigma}^{(k)}, q\right)$;

---

inizialize $\boldsymbol{m}$, M, $\mathcal{Q} = \{\ \}$;
compute matrices $A(\sigma_i^{(k)})$, for $i = 1, \ldots, 3$, using (1) and (2);
generate randomly $W_0(j)$, for $j = 1, \ldots, q$;
`for` $i = 1$ `to` 3
    $A = A(\sigma_i)$; $\alpha = \max A$;
    `for` $j = 1$ `to` q
        $W_0 = W_0(j)$;
        $r = i + 3(j-1)$;
        `Enqueue` $\left(\mathcal{Q}, \{r, 0, W_0, O, \alpha, 0\}\right)$;
    `end for`
`end for`
$\chi_{\min} = \infty$;
`while Length` $(\mathcal{Q}) \geq 1$
    $Y =$ `Dequeue` $(\mathcal{Q})$;
    $\{\nu, W, H, \alpha, cond\} =$ `Sym_NMF` $\left(A_{(Y)}, W_{(Y)}, H_{(Y)}, \alpha_{(Y)}, \lambda\right)$;
    $\left(\Pi,\ k_e\right) =$ `Partition` (W);
    $\chi =$ `DBindex`(Π);
    $t = t_{(Y)} + \nu$; $\xi = \chi + t/t_{\max}$;
    `if Control` $(t, \chi, \Pi, k_e, cond)$ `then Enqueue` $\left(\mathcal{Q}, \{r_{(Y)}, t, W, H, \alpha, \xi\}\right)$
`end while`
`return` $(M,\ m)$;

At the end, the kth partition in M, if not empty, is the solution of the problem for the given k. Other partitions possibly present in M can be of interest, and will be used when `Kfix` is called for solving $\mathcal{P}_{kvar}$.

The core of the whole procedure is the following Boolean function `Control`, which verifies whether an item is promising (i.e., must be further processed) or not.

`function Control` $(t, \chi, \Pi, k_e, cond)$;

`if` $\chi < m_{k_e}$ `then` $m_{k_e} = \chi$; $M_{k_e} = \Pi$;
`if` $k_e < k$ `then return` $t < 2\, t_{\min}$;
`if` $cond\ ||\ (t > t_{\max})$ `then return False`;
`if` $t < t_{\min}$ `then return True`;
`return` $\chi < m_{k_e}(1 + \exp(1 - t/t_{\min}))$

If `True` is returned, the selected item is enqueued again, otherwise it is discarded. When the number k_e of clusters effectively found is equal to the expected k, and the convergence of `Sym_NMF` has not been yet reached, the item is enqueued both when $t < t_{\min}$ and when $t_{\min} < t < t_{\max}$, provided that χ satisfies a bound, which becomes tighter as t increases. When, instead, $k_e < k$, the item is enqueued only if less than $2\, t_{\min}$ iterations have been performed. In any case, the best value of the DB index and the corresponding clustering are maintained for each value of $k_e \leq k$.

Remark 1.

(a) The use of the priority queue allows for an easy parallelization of the computation by simultaneously keeping active a number of items most equal to the number of the available processors. The numerical examples of Section 6 have been carried out both with eight active processors and one active processor in order to evaluate the scalability of the algorithm.

(b) The use of an internal validity index is fundamental in our heuristics. Because of its high impact in the algorithm, the choice of the DB index among the many indices listed in [20] has been driven by its low computational time complexity. The DB index, which turned out to be reliable in our experimentation, tends to detect clusters with separated convex envelopes.

5. The Clustering Algorithm for Problem $\mathcal{P}_{kvar}$

For problem $\mathcal{P}_{kvar}$, the dimension k of the clustering is not a priori assigned, but an interval $\mathcal{J}_k = [k_{\min}, k_{\max}]$ is only given where k has to be looked for, according to some performance criterium. The problem can be tackled by applying the algorithm designed for problem $\mathcal{P}_{kfix}$ with different values of k in the interval, and then by choosing the one which gives the better result. The strategy we propose in the following algorithm `Kvar` selects the consecutive integers $k = k_{\min}, \ldots, k_{\max}$. For each k, the triple $\boldsymbol{\sigma}^{(k)}$ of scale parameters is constructed using formula (10), given in the next section, and function `Kfix` is applied. The array $\boldsymbol{m}$ of the optimal DB values and the array M of the corresponding clusterings Π are returned. The indices DB** and CL, described in Section 3, help in choosing the best k and the corresponding clustering. Moreover, if a proper value of $k_{\max}$ is not assigned, plots of DB** and CL can indicate when the computation may be stopped.

`function Kvar` $\left(k_{\min}, k_{\max}\right)$

---

initialize $\boldsymbol{m}$ and M;
`for` $k = k_{\min}$ `to` $k_{\max}$
  compute triple $\boldsymbol{\sigma}^{(k)}$;
  $(M,\ m) =$ `Kfix` $\left(k, \boldsymbol{\sigma}^{(k)}, q\right)$;
`end for`;
`return` $(M,\ m)$

From (10) we see that the same σ appears repeatedly for different values of k. Hence, it is not worthwhile to construct each time the matrix $A(\sigma)$. Whenever possible, it is better to construct and store beforehand the matrices $A(\sigma)$ for all the different scale parameters σ that are expected to be used.

Another shortcut that immediately comes to mind is to exploit the already computed NMF factorization of an $A(\sigma)$ for a given dimension k to obtain the NMF factorization of the same matrix for a larger k, instead of computing it from scratch as suggested in `Kvar`. In [5], these updating algorithms are proposed. We have tried them in our experimentation and have found that, in our clustering context, the improvements on the computational cost are negligible in the face of a more complex code. Hence we do not recommend them.

6. Experimentation

The experimentation has been performed in JAVA on a Macintosh with a 4 GHz Intel 8 core Xeon W processor running MAC OS MOJAVE 10.14.4 on synthetic datasets of points in $\mathbb{R}^2$. It intends to prove that the proposed heuristics is reliable. In fact, with regard to a multistart strategy, the proposed heuristics allows for time-saving, but one could expect that during the elimination of the less promising roads, roads leading to final good solutions can also be lost. To verify that this does not happen, the experimentation aims to test the effectiveness of the algorithm.

Initially, an extensive computation has been carried out on selected datasets of points, to determine suitable values for the parameters to be used in the subsequent computation. Namely:

- the tolerances for `GCD` and `Sym_NMF` are set $\eta_1 = 10^{-3}$ and $\eta_2 = 10^{-4}$, respectively,
- the constants for CL index are set $p = 4$ and $c = 100$,
- the number of the starting random matrices for `Kfix` is set $q = 8$,
- the number of iterations for a single segment of `Kfix` is set $\lambda = 10$, hence $\nu_{\max} = 10$,
- the bounds for `Control` are set $t_{\min} = 3\,\lambda$ and $t_{\max} = 20\,\lambda$,
- for the definition of $\boldsymbol{\sigma}^{(k)}$ we have set $\widehat{\sigma} = 0.04$ and

$$\sigma_0^{(k)} = \widehat{\sigma} \text{ for } k \leq 5,\ \sigma_0^{(k)} = \widehat{\sigma}/2 \text{ for } 5 < k \leq 10,\ \sigma_0^{(k)} = \widehat{\sigma}/4 \text{ for } 10 < k \leq 20,$$
$$\sigma_0^{(k)} = \widehat{\sigma}/8 \text{ for } 20 < k \leq 40,\ \sigma_0^{(k)} = \widehat{\sigma}/16 \text{ for } k > 40.$$

The triplets considered in the experiments, varying k, are

$$\boldsymbol{\sigma}^{(k)} = \{\sigma_i^{(k)} = \sigma_0^{(k)}/2^i,\ i = 0,\ 1,\ 2\}. \tag{10}$$

Regarding the value $\widehat{\sigma}$, we have taken into account that the quantities $\|\boldsymbol{p}_i - \boldsymbol{p}_r\|_2^2/\mu$ in (1) are not greater than 1. The choice $\widehat{\sigma} = 0.04$ makes the lowest value $\exp(-1/\sigma)$ of $e_{i,r}$ comparable with the machine precision.

The datasets used in the first two experiments are suggested in [25]. The points are normally distributed with different variances around a given number of centers.

The first experiment concerns the clustering of the three datasets of Figure 1: dataset *Subcluster* (SC) has five clusters, and four of them are subclusters forming two pairs of clusters, respectively; dataset *Skewdistribution* (SD) has three clusters with skewed dispersions; dataset *Differentdensity* (DD) has clusters with different densities. Each dataset is generated with 1000 points.

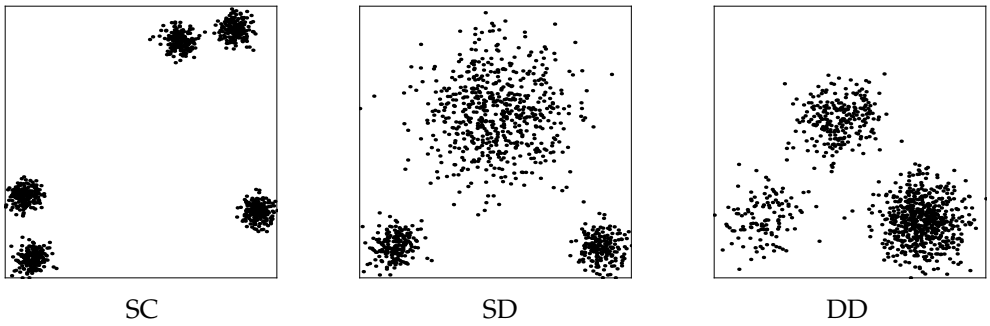

Figure 1. Synthetic datasets SC, SD, and DD of points in $\mathbb{R}^2$.

We applied `Kvar` to each dataset with k varying in the interval $[2,\ 15]$. Figure 2 shows the piecewise linear plots versus k of the DB index (dashed line), DB** index (solid line), and CL index (dotted line) for the three datasets. The CL plot is normalized in order to fit the size of the figure.

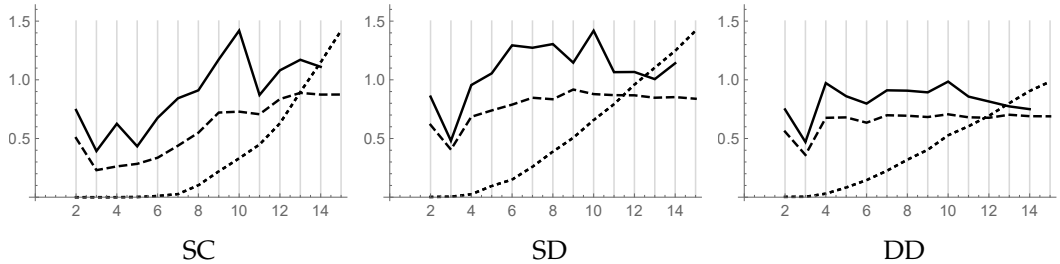

Figure 2. Plots versus k of DB (dashed line), DB** (solid line), and CL (dotted line) for SC, SD, and DD.

For the dataset SC, the DB index suggests choosing $k = 3$, while the DB** index suggests both $k = 3$ and $k = 5$, with a weak preference for the former. The plot of the CL index, which is zero for $k \leq 5$, shows that the clusters in this range of k are well-separated, while for $k \geq 6$ the increase of the plot suggests that some clusters are improperly subdivided.

For both datasets SD and DD, the DB and DB** indexes indicate the same value, 3, for k. The CL plot, which is not zero even for small values of k, reveals the presence of close points in different clusters. The difficulty encountered in the assignment of these points to the clusters suggests that such points could be considered as noise. In any case, using the information gathered from the CL plot, the computation should stop before k becomes too large.

The purpose of the second test is precisely to see how the procedure behaves in the presence of noise. To this aim, the two datasets of five points *Well-separated* (WS) and *Well-separated noise* (WSN) of Figure 3 are generated, the latter one with 5% noise added to the former.

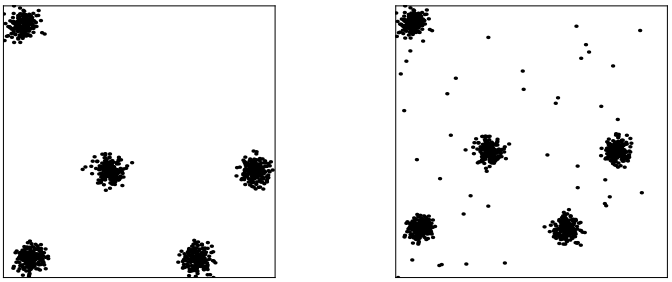

Figure 3. Synthetic datasets, well-separated (WS) and well-separated noise (WSN) of points in $\mathbb{R}^2$.

We applied `Kvar`, with k varying in the interval $[2, 20]$ and obtained the piecewise linear plots versus k of Figure 4. For dataset WSN, according to the DB index, both $k = 5$ and $k = 6$ seemed to be acceptable. This uncertainty is explained by the presence of noise which causes spurious clusters to appear. The DB** index correctly suggests $k = 5$. The presence of noise is confirmed by the CL index which is not zero for small values of k.

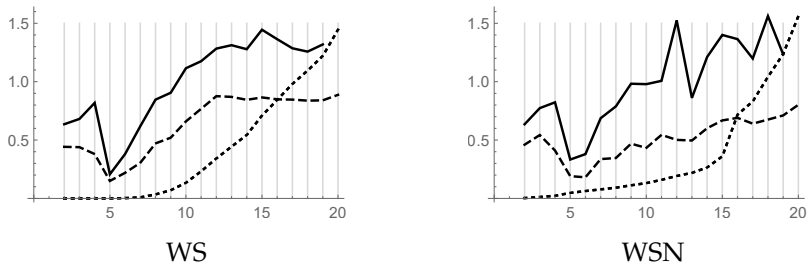

Figure 4. Plots versus k of DB (dashed line), DB** (solid line) and CL (dotted line) for WS and WSN.

To analyze more thoroughly the performance of our algorithm when subclusters are present, a third experiment was carried out concerning the three datasets of Figure 5. Dataset C17 consists of 1700 points generated around 17 randomly chosen centers; dataset C23, consisting of 2300 points, has 23 clusters of 100 points which can be grouped in several ways in a lower number of clusters; dataset C27, consisting of 2700 points, has 27 clusters of 100 points which can be seen as 9 clusters of 300 points or three clusters of 900 points, depending on the granularity level of interest.

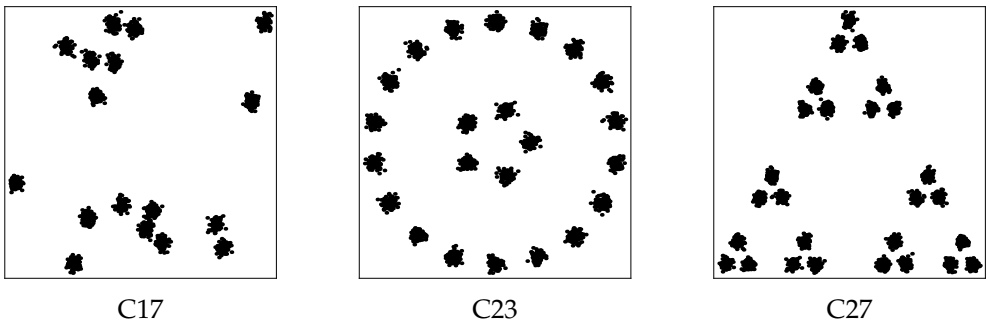

Figure 5. Synthetic datasets C17, C23, and C27 of points in $\mathbb{R}^2$.

Problems C17, C23, and C27 have been tested with values of k ranging in the intervals [2, 25], [2, 30], and [2, 35], respectively. The resulting piecewise linear plots versus k of DB, DB**, and CL indices are shown in Figure 6. By using the DB** indices, it clearly appears that our algorithm is able to find good clusterings for all three problems. The inspection of the plot of the DB** index gives further information. For example, for problem C17, a local minimum of the DB** index for $k = 4$ evidences the clustering shown in Figure 7 on the left. For problem C23, a local minimum of the DB** index for $k = 7$ evidences the possible clustering shown in Figure 7 on the right. For problem C27, two local minima in the DB** plot for $k = 3$ and $k = 9$ evidence the two other expected clusterings.

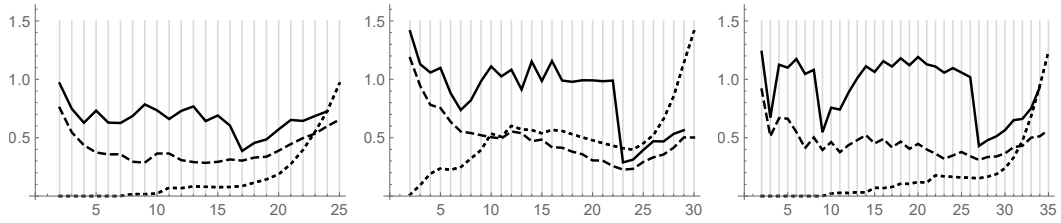

Figure 6. Plots versus k of DB (dashed line), DB** (solid line), and CL (dotted line) for C17, C23, and C27.

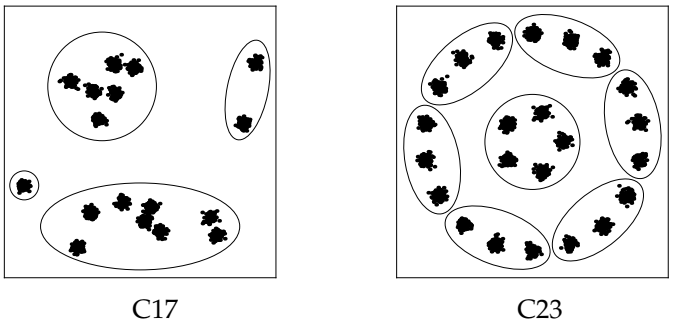

Figure 7. Further clusterings evidenced by the DB** plots for C17 and C23.

In all the experiments shown above, the measure CL helps decide the value of $k_{\max}$ if not assigned.

The fourth test regards the computational cost of algorithm `Kvar` when run on one or eight processors. Due to its iterative and heuristic structure, our algorithm is not suitable for a theoretical analysis of the complexity. In order to analyze how the required time increases as a function of the number of the points, the datasets SC, SD, DD, WS, and WSN have been generated with n varying in $\{1000, 2000, 4000, 8000\}$ and k varying in the interval $[2, 15]$. Figure 8 shows the plots versus n of the costs in seconds, averaged on all the datasets. The times corresponding to runs with one processor are fitted with a solid line, where those corresponding to runs with eight processors are fitted with a dashed line. The resulting polynomial fits are $1.5 \times 10^{-5}\, n^{2.16}$ for one processor and $1.4 \times 10^{-6}\, n^{2.23}$ for eight processors. This shows that for the considered problems, the time grows slightly more than quadratically as a function of n, but clearly less than cubically, as one could expect. It also shows that the whole procedure has a good degree of scalability.

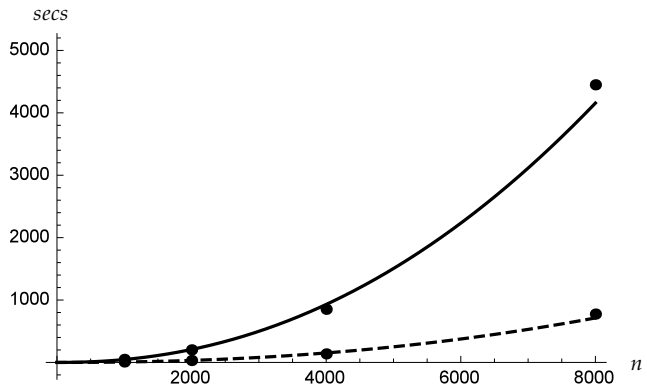

Figure 8. Averaged computational costs in seconds for one (solid line) and eight (dashed line) processors.

7. Discussion and Conclusions

In this paper, we considered the symmetric modeling of the clustering problem and proposed an algorithm which applies NMF techniques to a penalized nonsymmetric minimization problem. The proposed algorithm makes use of a similarity matrix A, which depends on the Gaussian kernel function and a scale parameter σ, varying with k. The Gaussian kernel function was chosen because it is widely used in the clustering problems of points in $\mathbb{R}^d$. Of course, other kernels can be used in different environments, with different procedures. We think that our heuristic approach could also be applied to treat other kernels.

As shown in [1,13], the ANLS algorithm with the `BPP` method handles large datasets more efficiently than other algorithms frequently used for the same problem. Moreover, in [14], it is shown that ANLS with `GCD` allows for further time reduction with respect to ANLS with `BPP`.

The proposed heuristic approach, which uses ANLS with `GCD`, processes multiple instances of the problem and focuses the computation only on the more promising ones, making use of the DB index. In this way, an additional time reduction is achieved with respect to the standard multistart approach. The experimentation, conducted on synthetic problems with different characteristics, evidences the reliability of the algorithm.

The same framework is repeatedly applied with different values of k varying in an assigned interval, allowing us to deal also with the clustering problem when the number of clusters is not a priori fixed. The combined use of the different considered validity indices allows to point out the correct solutions.

Author Contributions: All the authors have contributed substantially and in equal measure to all the phases of the work reported.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kuang, D.; Yun, S.; Park, H. SymNMF: Nonnegative low-rank approximation of a similarity matrix for graph clustering. *J. Glob. Optim.* **2015**, *62*, 545–574. [CrossRef]
2. Paatero, P.; Tappert, U. Positive matrix factorization: A non-negative factor model with optimal solution of error estimates of data values. *Environmetrics* **1994**, *5*, 111–126. [CrossRef]
3. Lee, D.D.; Seung, H.S. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 2001, Proceedings of the 2000 Conference (NIPS 2000), Denver, CO, USA, 1 January 2001;* Neural Information Processing Systems Foundation: San Diego, CA, USA; pp. 535–541.
4. Kim, H.; Park, H. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM J. Matrix Anal. Appl.* **2008**, *30*, 713–730. [CrossRef]
5. 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.* **2014**, *58*, 285–319. [CrossRef]

6. Cichocki, A.; Phan, A.H. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Trans. Fund. Electron. Commun. Comput. Sci.* **2009**, *92*, 708–721. [CrossRef]
7. 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, San Diego, CA, USA, 21–24 August 2011; pp. 1064–1072.
8. Belachew, M.T. Efficient algorithm for sparse symmetric nonnegative matrix factorization. *Pattern Recogn. Lett.* **2019**, *125*, 735–741. [CrossRef]
9. Kuang, D.; Ding, C.; Park, H. Symmetric nonnegative matrix factorization for graph clustering. In Proceedings of the 2012 SIAM International Conference on Data-Mining (SDM 2012), Anaheim, CA, USA, 26–28 April 2012; pp. 106–117.
10. Janecek, A.; Grotthoff, S.S.; Gansterer, W.N. LIBNMF—A library for nonnegative matrix factorization. *Comput. Inform.* **2011**, *30*, 205–224.
11. Wu, W.; Jia, Y.; Kwong, S.; Hou, J. Pairwise constraint propagation-induced symmetric nonnegative matrix factorization. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 6348–6361. [CrossRef] [PubMed]
12. Cichocki, A.; Zdunek, R.; Phan, A.H.; Amari, S. *Nonnegative Matrix and Tensor Factorizations: Applications to Explanatory Multi-Way Data Analysis and Blind Source Separation*; Wiley: New York, NY, USA, 2009.
13. Kim, J.; Park, H. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM J. Sci. Comput.* **2011**, *33*, 3261–3281. [CrossRef]
14. Favati, P.; Lotti, G.; Menchi, O.; Romani, F. Adaptive computation of the Symmetric Nonnegative Matrix Factorization (NMF). *arXiv* **2019**, arXiv:1903.01321. [CrossRef]
15. Vavasis, S.A. On the complexity of nonnegative matrix factorization. *SIAM J. Optim.* **2009**, *20*, 1364–1377. [CrossRef]
16. Grippo, L.; Sciandrone, M. On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Oper. Res. Lett.* **2000**, *26*, 127–136. [CrossRef]
17. Lawson, C.L.; Hanson, R.J. *Solving Least Squares Problems*; Prentice-Hall: Englewood Cliffs, NY, USA, 1974.
18. Björck, Å. *Numerical Methods for Least Squares Problems*; SIAM: Philadelphia, PE, USA, 1996.
19. Kim, H.; Park, H. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In Proceedings of the 8th IEEE International Conference on Data-Mining (ICDM), Pisa, Italy, 15–19 December 2008; pp. 353–362.
20. Desgraupes, B. Clustering Indices. Available online: https://cran.r-project.org/web/packages/clusterCrit/vignettes/clusterCrit.pdf (accessed on 15 October 2019).
21. Davis, D.L.; Bouldin, D.W. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell. PAMI* **1979**, *1*, 224–227. [CrossRef]
22. Kim, M.; Ramakrishma, R.S. New indices for cluster validity assessment. *Pattern Recogn. Lett.* **2005**, *26*, 2353–2363. [CrossRef]
23. NIST/SEMATECH e-Handbook of Statistical Methods. Available online: http://www.itl.nist.gov/div898/handbook (accessed on 15 October 2019).
24. Favati, P.; Lotti, G.; Menchi, O.; Romani, F. An adaptive procedure for the global minimization of a class of polynomial functions. *Algorithms* **2019**, *12*, 109. [CrossRef]
25. Liu, Y.; Li, Z.; Xiong, H.; Gao, X.; Wu, J. Understanding of internal clustering validation measures. In Proceedings of the IEEE International Conference on Data-Mining, Sydney, Australia, 13–17 December 2010; pp. 911–916.