J Glob Optim (2007) 37:593–608
DOI 10.1007/s10898-006-9066-4

ORIGINAL ARTICLE

A new efficient algorithm based on DC programming and DCA for clustering

Le Thi Hoai An · M. Tayeb Belghiti · Pham Dinh Tao

Received: 29 June 2006 / Accepted: 1 July 2006 / Published online: 9 August 2006

Abstract In this paper, a version of K-median problem, one of the most popular and best studied clustering measures, is discussed. The model using squared Euclidean distances terms to which the K-means algorithm has been successfully applied is considered. A fast and robust algorithm based on DC (Difference of Convex functions) programming and DC Algorithms (DCA) is investigated. Preliminary numerical solutions on real-world databases show the efficiency and the superiority of the appropriate DCA with respect to the standard K-means algorithm.

Keywords Clustering · K-median problem · K-means algorithm · DC programming · DCA · Nonsmooth nonconvex programming

AMS subject classifications 65K05 · 65K10 · 90C26 · 90C90 · 15A60 · 90C06

1 Introduction

Clustering is a fundamental problem in unsupervized learning which has many applications in various domains. In recent years, there has been significant interest in

Le Thi Hoai An (✉)
Laboratory of Theoretical and Applied Computer Science, UFR MIM, University of Paul Verlaine - Metz, Ile du Saulcy, 57045 Metz, France
e-mail: lethi@univ-metz.fr

M. Tayeb Belghiti
Laboratory of Modelling, Optimization and Operations Research, National Institute for Applied Sciences - Rouen, BP 08, Place Emile Blondel, 76131 Mont Saint Aignan Cedex, France
e-mail: belghiti-moulay.tayeb@insa-rouen.fr

Pham Dinh Tao
Laboratory of Modelling, Optimization and Operations Research, National Institute for Applied Sciences - Rouen, BP 08, Place Emile Blondel, 76131 Mont Saint Aignan Cedex, France
e-mail: pham@insa-rouen.fr

developing clustering algorithms to the massive data sets ([1–17, 26–28, 34, 36, 40–43] and reference therein). Two main approaches have been studied for clustering: the first one is the statistical and machine learning based on learning mixture models (see, e.g. [1, 2, 28, 34]) and the second is the mathematical programming approach that considers clustering as an optimization problem (see, e.g. [3, 4, 26, 36, 41–43]). The general term "clustering" covers many different types of problems. All consist of subdividing a data set into groups of similar elements, but there are many measures of similarity, many ways of measuring, and various concepts of subdivision.

An instance of the partitional clustering problem consists of a data set $\mathcal{A} := \{a^1, \ldots, a^m\}$ of m points in $\mathbb{R}^n$, a measured distance, and an integer k; we are to choose k members x^ℓ ($\ell = 1, \ldots, k$) (in $\mathcal{A}$ and/or $\mathbb{R}^n$) as "centroid" (or "median") and assign each member of $\mathcal{A}$ to its closest centroid. The assignment distance of a point $a \in \mathcal{A}$ is the distance from a to the centroid to which it is assigned, and the objective function, which is to be minimized, is the sum of assignment distances. If the centroids are not necessarily in $\mathcal{A}$, then the problem can be formulated as a unconstrained optimization problem. In the contrary case, we are faced with a discrete optimization problem. In both cases, different objective functions corresponding to the distance metric being considered are possible. Two models widely studied in the literature are the cases where the points come from a real space $\mathbb{R}^n$, and the assignment distance of a point is defined as the squared Euclidean distance (2-norm), and/or the 1-norm. If the squared Euclidean distance is used and the centroids are not necessarily in $\mathcal{A}$, then the corresponding optimization problem can be expressed as ($\|.\|$ denotes the Euclidean norm)

$$\min \left\{ \sum_{i=1}^{m} \min_{\ell=1,\ldots,k} \left\| x^\ell - a^i \right\|^2 : x^\ell \in \mathbb{R}^n, \ell = 1, \ldots, k \right\}. \tag{1.1}$$

If the 1-norm is considered instead of the squared Euclidean distance, then the problem can be written as

$$\min \left\{ \begin{array}{l} \sum_{i=1}^{m} \min_{\ell=1,\ldots,k} e^T D^{i\ell} : -D^{i\ell} \le x^\ell - a^i \le D^{i\ell}, \\ \qquad\qquad\qquad\qquad D^{i\ell}, x^\ell \in \mathbb{R}^n,\ \ell = 1, \ldots, k,\ i = 1, \ldots, m \end{array} \right\}, \tag{1.2}$$

where $D^{i\ell} \in \mathbb{R}^n$ is a dummy variable that bounds the components of the difference $x^\ell - a^i$ and $e \in \mathbb{R}^n$ denotes the vector of ones. Both (1.1) and (1.2) are nonsmooth nonconvex programs for which there are rarely efficient solution algorithms, especially in the large scale setting.

Two most popular families of partitional clustering algorithms, namely the K-means algorithm (see, e.g. [27, 36]) and the K-median algorithm (see [4, 28] and references therein), have been introduced for solving (1.1) and (1.2), respectively. The K-median algorithm is similar to the K-means algorithm in the description but they actually differ from both theoretical and computational points of view. They are recognized to be the most inexpensive and efficient method, especially for the large-scale setting. Unfortunately neither the K-median algorithm nor the K-means algorithm guarantee the global optimality of computed solutions.

We consider in this work the clustering problem corresponding to the model (1.1). Our approach is based on Difference of Convex functions (DC) programming and DC Algorithms (DCA) that were introduced by Pham Dinh Tao in their preliminary form in 1985. They have been extensively developed since 1994 by Le Thi Hoai An

and Pham Dinh Tao and become now classic and more and more popular (see, e.g. [18–25, 29, 32, 33, 37–39, 44], and references therein). Constituting the backbone of nonconvex programming and global optimization which have known very active developments in the last decade, DC programming and DCA can be considered as a natural and logical extension of Pham's ealier works (1975–1984) on convex maximization programming and its subgradient algorithms (see, e.g. [18, 20, 30, 31] and references therein). It is worth noting that the reference majorization algorithm developed by de Leeuw [7, 8] for solving the Euclidean MDS problem is a special case of the above subgradient method. DCA has been successfully applied to many large-scale (smooth or nonsmooth) nonconvex programs in various domains of applied sciences, in particular in data analysis and data mining [20, 21, 23, 25, 27, 29, 37, 38, 39, 44], for which it provided quite often a global solution and proved to be more robust and efficient than standard methods (see [19–24, 29, 32, 33, 37–39] and references therein).

A so-called DC program is that of minimizing a DC function over a convex set. According to the theory of DC programming, it is easy to show that all the clustering problems models above displayed are DC programs. We then suggested using DC programming approach and DCA to solve the clustering problem (1.1). Preliminary numerical simulations on real-world databases [4, 10, 41–43] show the robustness, the efficiency and the superiority of the appropriate DCA with respect to the K-means algorithm.

The paper is organized as follows. After the introduction, the DC programming and DCA are briefly presented in Sect. 2. Section 3 deals with a special realization of DCA to the underlying clustering problem which is beforehand recast into the well relevant matrix vector space. A combination of DCA with K-means algorithm to find a good starting point for the main algorithm DCA is discussed in Sect. 4. Computational results are reported in the last section.

2 A brief presentation of DC programming and DCA

To give the reader an easy understanding of the theory of DC programming and DCA and our motivation to use them for solving Problem (1.1), we briefly outline these tools in this section. Let $\Gamma_0(\mathbb{R}^n)$ denote the convex cone of all lower semicontinuous proper convex functions on $\mathbb{R}^n$. The vector space of DC functions, $\mathrm{DC}(\mathbb{R}^n) = \Gamma_0(\mathbb{R}^n) - \Gamma_0(\mathbb{R}^n)$, is quite large to contain almost real life objective functions and is closed under all the operations usually considered in optimization.

Consider the general DC program

$$\alpha = \inf\{f(x) := g(x) - h(x) \ : \ x \in \mathbb{R}^n\} \quad (P_{dc})$$

with $g, h \in \Gamma_0(\mathbb{R}^n)$. Such a function f is called DC function, and $g - h$, DC decomposition of f while the convex functions g and h are DC components of f.

If g or h are polyhedral convex functions then (P_{dc}) is called a polyhedral DC program.

It should be noted that a constrained DC program whose feasible set C is convex can always be transformed into an unconstrained DC program by adding the indicator function χ_C of C ($\chi_C(x) = 0$ if $x \in C, +\infty$ otherwise) to the first DC component g.

Let

$$g^*(y) := \sup\{\langle x, y\rangle - g(x) \ : \ x \in \mathbb{R}^n\}$$

be the conjugate function of g. By using the fact that every function $h \in \Gamma_0(\mathbb{R}^n)$ is characterized as a pointwise supremum of a collection of affine functions, say

$$h(x) := \sup\{\langle x, y\rangle - h^*(y) : y \in \mathbb{R}^n\},$$

we have

$$\alpha = \inf\{g(x) - \sup\{\langle x, y\rangle - h^*(y) : y \in \mathbb{R}^n\} : x \in \mathbb{R}^n\} = \inf\{\alpha(y) : y \in \mathbb{R}^n\}$$

with

$$\alpha(y) := \inf\{g(x) - [\langle x, y\rangle - h^*(y)] : x \in \mathbb{R}^n\} \quad (P_y).$$

It is clear that (P_y) is a convex program and

$$\alpha(y) = h^*(y) - g^*(y) \text{ if } y \in \operatorname{dom} h^*, \text{ and } +\infty \text{ otherwise.} \tag{2.1}$$

Finally we state the dual program of (P_{dc})

$$\alpha = \inf\{h^*(y) - g^*(y) : y \in \operatorname{dom} h^*\},$$

that is written, in virtue of the natural convention in DC programming, say $+\infty - (+\infty) = +\infty$:

$$\alpha = \inf\{h^*(y) - g^*(y) : y \in Y\} \quad (D_{dc}).$$

We observe the perfect symmetry between primal and dual DC programs: the dual to (D_{dc}) is exactly (P_{dc}).

Recall that, for $\theta \in \Gamma_0(\mathbb{R}^n)$ and $x_0 \in \operatorname{dom} \theta := \{x \in \mathbb{R}^n : \theta(x_0) < +\infty\}$, $\partial\theta(x_0)$ denotes the subdifferential of θ at x_0, i.e., [15, 35]

$$\partial\theta(x_0) := \{y \in \mathbb{R}^n : \theta(x) \geq \theta(x_0) + \langle x - x_0, y\rangle, \forall x \in \mathbb{R}^n\}. \tag{2.2}$$

The subdifferential $\partial\theta(x_0)$ is a closed convex set in $\mathbb{R}^n$. It generalizes the derivative in the sense that θ is differentiable at x_0 if and only if $\partial\theta(x_0)$ is reduced to a singleton which is exactly $\{\theta'(x_0)\}$.

DC programming investigates the structure of the vector space $DC(\mathbb{R}^n)$, DC duality and optimality conditions for DC programs. The complexity of DC programs resides, of course, in the lack of practical optimal globality conditions. We developed instead the following necessary local optimality conditions for DC programs in their primal part, by symmetry their dual part is trivial (see [19–24, 32, 33] and references therein):

$$\partial h(x^*) \cap \partial g(x^*) \neq \emptyset \tag{2.3}$$

(such a point x^* is called *critical point* of $g - h$ or for (P_{dc})), and

$$\emptyset \neq \partial h(x^*) \subset \partial g(x^*). \tag{2.4}$$

The condition (2.4) is also sufficient for many important classes of DC programs. In particular it is sufficient for the next cases quite often encountered in practice:

- In polyhedral DC programs with h being a polyhedral convex function (see [19–24, 32, 33] and references therein). In this case, if h is differentiable at a critical point x^*, then x^* is actually a local minimizer for (P_{dc}). Since a convex function is differentiable everywhere except for a set of measure zero, one can say that a critical point x^* is almost always a local minimizer for (P_{dc}).
- In case the function f is locally convex at x^* [22, 24].

The transportation of global solutions between (P_{dc}) and (D_{dc}) is expressed by:

$$[\cup_{y^* \in \mathcal{D}} \partial g^*(y^*)] \subset \mathcal{P}, \ [\cup_{x^* \in \mathcal{P}} \partial h(x^*)] \subset \mathcal{D} \tag{2.5}$$

where $\mathcal{P}$ and $\mathcal{D}$ denote the solution sets of (P_{dc}) and (D_{dc}), respectively. Under technical conditions, this transportation holds also for local solutions of (P_{dc}) and (D_{dc}) [22, 24, 32].

Based on local optimality conditions and duality in DC programming, the DCA consists in the construction of two sequences $\{x^k\}$ and $\{y^k\}$, candidates to be optimal solutions of primal and dual programs, respectively, such that the sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing, and $\{x^k\}$ (resp. $\{y^k\}$) converges to a primal feasible solution $\widetilde{x}$ (resp. a dual feasible solution $\widetilde{y}$) verifying local optimality conditions and

$$\widetilde{x} \in \partial g^*(\widetilde{y}), \qquad \widetilde{y} \in \partial h(\widetilde{x}). \tag{2.6}$$

These two sequences $\{x^k\}$ and $\{y^k\}$ are determined in the way that x^{k+1} (resp. y^k) is a solution to the convex program (P_k) (resp. (D_k)) defined by

$$\inf\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle : x \in \mathbb{R}^n\} \quad (P_k)$$

$$\inf\{h^*(y) - g^*(y^{k-1}) - \langle y - y^{k-1}, x^k\rangle : y \in \mathbb{R}^n\} \quad (D_k).$$

The *first interpretation* of DCA is simple: at each iteration one replaces in the primal DC program (P_{dc}) the second component h by its affine minorization $h_k(x) := h(x^k) + \langle x - x^k, y^k\rangle$ at a neighborhood of x^k to give birth to the convex program (P_k) whose the solution set is nothing but $\partial g^*(y^k)$. Likewise, the second DC component g^* of the dual DC program (D_{dc}) is replaced by its affine minorization $(g^*)_k(y) := g^*(y^k) + \langle y - y^k, x^{k+1}\rangle$ at a neighborhood of y^k to obtain the convex program (D_k) whose $\partial h(x^{k+1})$ is the solution set. DCA performs so a double linearization with the help of the subgradients of h and g^* and the DCA then yields the next scheme:

$$y^k \in \partial h(x^k); \quad x^{k+1} \in \partial g^*(y^k). \tag{2.7}$$

First of all, it is worth noting that our works involve the convex DC components g and h but not the DC function f itself. Moreover, a DC function f *has infinitely many DC decompositions which have crucial impacts on the qualities* (speed of convergence, robustness, efficiency, globality of computed solutions,...) of DCA. For a given DC program, the choice of *optimal* DC decompositions is still open. Of course, this depends strongly on the very specific structure of the problem being considered. In order to tackle the large scale setting, one tries in practice to choose g and h such that sequences $\{x^k\}$ and $\{y^k\}$ can be easily calculated, i.e. either they are in explicit form or their computations are inexpensive.

We mention now the main convergence properties of DCA [19–24, 32, 33]. In this paragraph, denote by C (resp. D) a convex set containing the sequence $\{x^k\}$ (resp. $\{y^k\}$) and $\rho(g, C)$ (or $\rho(g)$ if $C = \mathbb{R}^n$) the modulus of strong convexity of g on C given by:

$$\rho(g, C) = \sup\{\rho \geq 0 : g - (\rho/2)\|\cdot\|^2 \text{ be convex on } C\}.$$

DCAs convergence properties [19–24, 32, 33]. DCA is a descent method without linesearch which enjoys the following properties:

(i) The sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing and
- $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ iff $y^k \in \partial g(x^k) \cap \partial h(x^k), y^k \in \partial g(x^{k+1}) \cap \partial h(x^{k+1})$ and $[\rho(g, C) + \rho(h, C)]\|x^{k+1} - x^k\| = 0$. Moreover if g or h are strictly convex on C then $x^k = x^{k+1}$.
  In such a case DCA terminates at the kth iteration (finite convergence of DCA);
- $h^*(y^{k+1}) - g^*(y^{k+1}) = h^*(y^k) - g^*(y^k)$ iff $x^{k+1} \in \partial g^*(y^k) \cap \partial h^*(y^k), x^{k+1} \in \partial g^*(y^{k+1}) \cap \partial h^*(y^{k+1})$ and $[\rho(g^*, D) + \rho(h^*, D)]\|y^{k+1} - y^k\| = 0$. Moreover if g^* or h^* are strictly convex on D, then $y^{k+1} = y^k$.
  In such a case DCA terminates at the kth iteration (finite convergence of DCA).

(ii) If $\rho(g, C) + \rho(h, C) > 0$ (resp. $\rho(g^*, D) + \rho(h^*, D) > 0$)) then the series $\{\|x^{k+1} - x^k\|^2$ (resp. $\{\|y^{k+1} - y^k\|^2\}$ converges.

(iii) If the optimal value α of problem (P_{dc}) is finite and the infinite sequences $\{x^k\}$ and $\{y^k\}$ are bounded then every limit point $\widetilde{x}$ (resp. $\widetilde{y}$) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is a critical point of $g - h$ (resp. $h^* - g^*$).

(iv) DCA has a linear convergence for general DC programs.

(v) DCA has a finite convergence for polyhedral DC programs.

Before closing this outline of DCA, it is crucial to keep in mind the *second interpretation* of DCA:

Let x^* be an optimal solution of primal DC program (P_{dc}) and $y^* \in \partial h(x^*)$. In virtue of (2.5) y^* is an optimal solution of the dual DC program (D_{dc}). Let h_* be the affine minorization of h defined by

$$h_*(x) := h(x^*) + \langle x - x^*, y^* \rangle$$

and consider the next convex program

$$\alpha_* := \inf\{g(x) - h_*(x) : \mathbb{R}^n\} = \inf\{f(x) + h(x) - h_*(x) : x \in \mathbb{R}^n\}. \tag{2.8}$$

Since the function $f_*(x) = f(x) + h(x) - h_*(x)$ is a convex majorization of f, $\alpha_* \geq \alpha$. But $f_*(x^*) = f(x^*) = \alpha$. Hence $\alpha_* = \alpha$. On the other hand, the optimal solution set of (2.8) is $\partial g^*(y^*)$ that is contained in the optimal solution set $\mathcal{P}$ of (P_{dc}), following (2.5). Taking into account of (2.5) and the decrease of the sequence $\{g(x^k) - h(x^k)\}$, one can understand better the role played by the linearized programs (P_k) and (2.8) and explain partially the reason why DCA converges to an optimal solution of (P_{dc}) from a good initial point.

For a complete study of DC programming and DCA the reader is referred to [19–24, 32, 33] and references therein. The solution of a nonconvex program by DCA must be composed of two stages: the search for an *appropriate* DC decomposition and that for a *good* initial point. We shall apply *all these DC enhancement features* to solve problem (1) in its equivalent DC program given in the next section.

3 Solving the clustering problem (1.1) by DCA

To simplify related computations in DCA for solving problem (1.1) it is crucial, as will be seen, in the next to recast this problem in the matrix vector space $\mathbb{R}^{k\times n}$ of $(k \times n)$ real matrices. The variables are then $X \in \mathbb{R}^{k\times n}$ whose ith row X_i is equal to x^i for

$i = 1, \ldots, k$. The Euclidean structure of $\mathbb{R}^{k\times n}$ is defined with the help of the usual scalar product

$$\mathbb{R}^{k\times n} \ni X \longleftrightarrow (X_1, X_2, \ldots, X_k) \in (\mathbb{R}^n)^k, \qquad X_i \in \mathbb{R}^n \quad (i = 1, \ldots, k),$$
$$\langle X, Y\rangle := Tr(X^T Y) = \sum_{i=1}^{k} \langle X_i, Y_i\rangle$$

and its Euclidean norm $\|X\|^2 := \sum_{i=1}^{k} \langle X_i, X_i\rangle = \sum_{i=1}^{k} \|X_i\|^2$ (Tr denotes the trace of a square matrix). We will reformulate problem (1.1) as a DC program in the matrix space $\mathbb{R}^{k\times n}$ and then describe DCA for solving it.

3.1 Formulation of (1.1) as a DC program

According to the property

$$\min_{\ell=1,\ldots,k} \left\|x^\ell - a^i\right\|^2 = \sum_{\ell=1}^{k} \left\|x^\ell - a^i\right\|^2 - \max_{r=1,\ldots,k} \sum_{\ell=1,\ell\neq r} \left\|x^\ell - a^i\right\|^2,$$

and the convexity of the functions

$$\sum_{\ell=1}^{k} \left\|x^\ell - a^i\right\|^2, \quad \max_{r=1,\ldots,k} \sum_{\ell=1,\ell\neq r} \left\|x^\ell - a^i\right\|^2,$$

we can say that clustering problem (1.1) is a DC program.

We can advantageously express (1.1) in the matrix space $\mathbb{R}^{k\times n}$ as follows:

$$(1.1) \Leftrightarrow \min \quad \{F(X) := G(X) - H(X) :\ X \in \mathbb{R}^{k\times n}\}, \tag{3.1}$$

where the DC components G and H are given by

$$G(X) = \sum_{i=1}^{m}\sum_{\ell=1}^{k} G_{i\ell}(X), G_{i\ell}(X) = \frac{1}{2}\left\|X_\ell - a^i\right\|^2 \quad \text{for } i = 1, \ldots, m, \ell = 1, \ldots, k \tag{3.2}$$

and

$$H(X) = \sum_{i=1}^{m} H_i(X),\ H_i(X) = \max_{j=1,\ldots,k} \sum_{\ell=1,\ell\neq j}^{k} \frac{1}{2}\left\|X_\ell - a^i\right\|^2 \quad \text{for } i = 1, \ldots, m. \tag{3.3}$$

It is interesting to note that the function G is a strictly convex quadratic form. More precisely we have, after simple calculations:

$$G(X) = \frac{m}{2}\|X\|^2 - \langle B, X\rangle + \frac{k}{2}\|A\|^2, \tag{3.4}$$

where $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{k\times n}$ are given by

$$\begin{aligned} A_i &:= a^i \quad \text{for} \quad i = 1, \ldots, m, \\ B_\ell &:= a = \sum_{i=1}^{m} a^i \quad \text{for } \ell = 1, \ldots, k. \end{aligned} \tag{3.5}$$

Remark 3.1 In the matrix space $\mathbb{R}^{k\times n}$, the DC program (3.1) then is minimizing the difference of the simplest convex quadratic function (3.4) and the nonsmooth convex one (3.3). This nice feature is very convenient for applying DCA, which consists in solving a sequence of approximate convex quadratic programs whose solutions are explicit.

According to Sect. 2, determining the DCA scheme applied to (3.1) amounts to computing the two sequences $\{X^{(p)}\}$ and $\{Y^{(p)}\}$ in $\mathbb{R}^{k\times n}$ such that

$$Y^{(p)} \in \partial H(X^{(p)}), X^{(p+1)} \in \partial G^*(Y^{(p)}).$$

We shall present below the computation of $\partial H(X)$ and $\partial G^*(Y)$.

3.2 Calculation of $\partial H(X)$

We can write, for $i = 1,\ldots,m$:

$$H_i(X) = \max_{j=1,\ldots,k} H_{ij}(X) \text{ where } H_{ij}(X) := \sum_{\ell=1,\ell\neq j}^{k} \frac{1}{2}\left\|X_\ell - a^i\right\|^2 \quad \text{for } j = 1,\ldots,k.$$

Let $K_i(X) := \{j = 1,\ldots,k : H_{ij}(X) = H_i(X)\}$. Then we have [15]:

$$\partial H_i(X) = \text{co}\{\cup_{j\in K_i(X)} \partial H_{ij}(X)\}, \tag{3.6}$$

where co stands for the convex hull. Let now $L_{\ell i}$ be the convex function on $\mathbb{R}^{k\times n}$ defined by

$$L_{\ell i}(X) := \frac{1}{2}\left\|X_\ell - a^i\right\|^2.$$

So $H_{ij} = \sum_{\ell=1,\ell\neq j}^{k} L_{\ell i}$, and $L_{\ell i}$ is differentiable with (for $r = 1,\ldots,k$)

$$[\nabla L_{\ell i}(X)]_r = X_\ell - a^i \quad \text{if } r = \ell, 0 \text{ otherwise.}$$

It follows that for $r = 1,\ldots,k$

$$[\nabla H_{ij}(X)]_r = \sum_{\ell=1,\ell\neq j}^{k} [\nabla L_{\ell i}(X)]_r = 0 \quad \text{if } r = j,\ X_r - a^i\,. \tag{3.7}$$

The calculation of $\partial H(X)$ is then immediate from the relations (3.6), (3.7) and

$$\partial H(X) = \sum_{i=1}^{m} \partial H_i(X). \tag{3.8}$$

But from the computational point of view, the formula (3.7) makes the summation (3.8) cumbersome. We use instead another formula as follows:

We first express the convex function H_{ij} by

$$\begin{aligned} H_{ij}(X) &= \sum_{\ell=1}^{k} \frac{1}{2}\left\|X_\ell - a^i\right\|^2 - \frac{1}{2}\left\|X_j - a^i\right\|^2 \\ &= \frac{1}{2}\left\|X - A^{[i]}\right\|^2 - \frac{1}{2}\left\|X_j - a^i\right\|^2, \end{aligned} \tag{3.9}$$

where $A^{[i]} \in \mathbb{R}^{k\times n}$ is the matrix whose rows are all equal to a^i.

That gives

$$\nabla H_{ij}(X) = X - A^{[i]} - e_j^{[k]}(X_j - a^i) \tag{3.10}$$

with $\{e_j^{[k]} : j = 1, \ldots, k\}$ being the canonical basis of $\mathbb{R}^k$.

Hence, according to (3.6) and (3.8) we get the following simpler matrix formula for computing ∂H

$$Y \in \partial H(X) \Leftrightarrow Y = \sum_{i=1}^{m} Y^{[i]} \text{ with } Y^{[i]} \in \partial H_i(X) \quad \text{for } i = 1, \ldots, k, \tag{3.11}$$

where $Y^{[i]}$ is a convex combination of $\{\nabla H_{ij}(X) : j \in K_i(X)\}$, i.e.,

$$Y^{[i]} = \sum_{j \in K_i(X)} \lambda_j^{[i]} \nabla H_{ij}(X) \text{ with } \lambda_j^{[i]} \geq 0 \text{ for } j \in K_i(X) \quad \text{and} \quad \sum_{j \in K_i(X)} \lambda_j^{[i]} = 1. \tag{3.12}$$

In other words with the help of (3.10)

$$Y^{[i]} = X - A^{[i]} - \sum_{j \in K_i(X)} \lambda_j^{[i]} e_j^{[k]}(X_j - a^i). \tag{3.13}$$

Finally $Y \in \partial H(X)$ if and only if

$$Y = mX - B - \sum_{i=1}^{m} \sum_{j \in K_i(X)} \lambda_j^{[i]} e_j^{[k]}(X_j - a^i), \tag{3.14}$$

where $B = \sum_{i=1}^{m} A^{[i]}$ is already defined in (3.5).

In particular we can take for $i = 1, \ldots, m$

$$Y^{[i]} = X - A^{[i]} - e_{j(i)}^{[k]}(X_{j(i)} - a^i) \quad \text{for some } j(i) \in K_i(X) \tag{3.15}$$

and the corresponding $Y \in \partial H(X)$ defined by

$$Y = mX - B - \sum_{i=1}^{m} e_{j(i)}^{[k]}(X_{j(i)} - a^i). \tag{3.16}$$

3.3 Calculation of $\partial G^*(Y)$

Since the function G is strictly convex quadratic, its conjugate G^* is differentiable and we have from (3.4)

$$X = \nabla G^*(Y) \Longleftrightarrow Y = \nabla G(X) = mX - B$$

or again

$$X = \frac{1}{m}(B + Y). \tag{3.17}$$

Remark 3.2 According to (2.7) and the explicit calculations (3.14) and (3.17) of the subdifferentials ∂H and ∂G^* in the DC program (3.1), the sequences $\{X^{(p)}\}$ and $\{Y^{(p)}\}$ generated by DCA are explicitly computed.

We are now in a position to describe the DCA for solving problem(1.1) via the DC decomposition (3.1).

3.4 Description of DCA to solve the clustering problem (1.1)

Initialization: Let $\epsilon > 0$ be given, $X^{(0)}$ be an initial point in $\mathbb{R}^{k\times n}$, set $p := 0$;
Repeat
Calculate $Y^{(p)} \in \partial H(X^{(p)})$ by using (3.14)

$$Y^{(p)} = mX^{(p)} - B - \sum_{i=1}^{m} \sum_{j\in K_i(X^{(p)})} \lambda_j^{[i]} e_j^{[k]} (X_j^{(p)} - a^i)$$

$$\text{with } \lambda_j^{[i]} \geq 0 \quad \text{for } j \in K_i(X^{(p)}) \quad \text{and} \quad \sum_{j\in K_i(X^{(p)})} \lambda_j^{[i]} = 1, \tag{3.18}$$

and calculate $X^{(p+1)}$ according to (3.17)

$$X^{(p+1)} := X^{(p)} - \frac{1}{m} \sum_{i=1}^{m} \sum_{j\in K_i(X^{(p)})} \lambda_j^{[i]} e_j^{[k]} (X_j^{(p)} - a^i). \tag{3.19}$$

Set $p + 1 \leftarrow p$
until $\|X^{(p+1)} - X^{(p)}\| \leq \epsilon(\|X^{(p)}\| + 1)$ or $\left|F(X^{(p+1)}) - F(X^{(p)})\right| \leq \epsilon\left(\left|F(X^{(p)})\right| + 1\right)$.

Remark 3.3 The DC decomposition (3.1) gives birth to a very simple DCA. It requires only elementary operations on matrices (the sum and the scalar multiplication of matrices) and can so handle large-scale clustering problems.

4 A combined K-means-DCA procedure for initializing DCA

We are now interested in the second question while using DCA: *find a good starting point for the algorithm*?

Like the K-means algorithm, we can randomly choose $X_i^{(0)}$ among the points a^i. Nevertheless it is known that the performance of the K-means algorithm much depends on the initial point. For exploiting simultaneously the efficiency of DCA and the K-means algorithm we combine alternatively the two procedures. More precisely, starting with a point $X^{(0)}$ with $X_i^{(0)}$ randomly chosen among the points a^i we perform one iteration of DCA, namely set $Y^{(0)} \in \partial H(X^{(0)})$ and $Z^{(1)} \in \partial G^*(Y^{(0)})$, and then improve $Z^{(1)}$ by one iteration of K-means to obtain $X^{(1)}$. This procedure can be then repeated some times to provide a good initial point for the main DCA as will be shown in numerical simulations.

K-means [27] is one of the simplest unsupervised learning algorithms for the clustering problem. Starting with k points randomly chosen as centroids among the given set of points, each iteration of the algorithm is composed of two steps:

- Assign each point a^i to the group π_ℓ that has the closest centroid.
- When all points have been assigned, recalculate the positions of the k centroids.

Repeat these two steps until the centroids no longer move.

The combined K-means—DCA procedure, denoted KMDCA, to finding a good initial point for the main DCA can be then described as follows:

Procedure KMDCA. let q be a positive integer.

Let $X^{(0)} \in \mathbb{R}^{k\times n}$ such that $X_i^{(0)}$ is randomly chosen among the points of $\mathcal{A} = \{a^1, \ldots, a^m\}$

For t = 0, 1, . . . , q do

t1. Compute $Y^{(t)}$ by (3.18) and $X^{(t+1)}$ by (3.19)

t2. Assign each point $a^i \in \mathcal{A}$ into the group that has the closest centroid $X^{(t+1)}_{S(a^i)}$: let $S(a^i) := \arg\min_{\ell\in\{1,\ldots,k\}} \left\|X^{(t+1)}_\ell - a^i\right\|^2$ and then assign a^i to $\pi_{S(a^i)}$.

t3. For each $\ell \in \{1, \ldots, k\}$ **recompute** Z_ℓ as the centroids of the group $\pi_\ell := \{a^i : S(a^i) = \ell\}$, i.e.,

$$Z_\ell := \arg\min\left\{\sum_{a^i\in\pi_\ell} \|x - a^i\|^2 : x \in \mathbb{R}^n\right\}.$$

$$\text{Update } X^{(t+1)}_\ell := Z_\ell \text{ for } \ell = 1, \ldots k.$$

enddo

Ouput: set $X^{(0)} := X^{(q)}$.

From the numerical experiments we see that it suffices to take $q = 2$ for obtaining a very good starting point for DCA. We also note that the alternative KMDCA procedure is better than the combination of the complete K-means (until the convergence) and DCA.

5 Numerical experiments

Our algorithms are coded in C++, and run on a Pentium 2.930 GHz of 1024 DDRAM with the following choice of subgradients for computing the sequences $\{X^{(p)}\}, \{Y^{(p)}\}$ generated by DCA:

(i) In the description of DCA for solving (1.1) we use (3.16) for (3.18) and the corresponding $X^{(p+1)}$ given by

$$X^{(p+1)} := X^{(p)} - \frac{1}{m}\sum_{i=1}^{m} e^{[k]}_{j(i)}(X^{(p)}_{j(i)} - a^i). \tag{5.1}$$

where $j(i) \in K_i(X^{(p)})$ for $i = 1, \ldots, m$.

(ii) The same formulas are used in (t1) of the procedure KMDCA.

We have tested our code on two sets of data. The first is composed of ten problems among them the first seven ones are given in [4, 41, 42, 43]), and the problems 8, 9, 10 are randomly generated. The second set contains four problems with real data.

- "PAPILLON" is a well known dataset called "jeux de papillon". Several works of clustering have discussed this dataset (see Revue Modulad - Le MOnde Des Utilisateurs de L'Analyse de Données, numéro **11**, 7–44, 1993).
- "IRIS" is the IRIS dataset which is perhaps the best known dataset found in pattern recognition literature. The dataset consists of three classes, 50 instances each and 4 numeric attributes where each class refers to a type of iris plant namely Iris Setosa, Iris Versicolor, Iris Verginica. The first class is linearly separable from others while that latter are not linearly separable. The measurement consists of the sepal and petal lengths and widths in cms.
- "GENE" is a Gene Expression dataset containing 384 genes that we get from http://faculty.washington.edu/kayee/cluster/

- "VOTE" is the Congressional Votes dataset (Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL: Congressional Quarterly Inc. Washington, DC, 1985) consists of the votes for each of the U.S. House of Representative Congressmen, for 16 key votes, on different subjects (handicap, religion, immigration, army, education, ...). For each vote, three answers are possible: yes, nay, and unknown. The individuals are separated into two clusters: democrats (267) and republicans (168).
- "ADN" is the ADN dataset (ftp://genbank.bio.net) that consists of 3186 genes, described by 60 DNA sequence elements, called nucleotides or base pairs, with 4 possible categories (A, C, G or T). These genes are distributed into three different clusters: "intron → exon" or ie (sometimes called donors, 767 objects), "exon → intron" or ei (sometimes called acceptors, 765 objects), and neither, noted as *n* objects).

For the last two sets of data ("VOTE" and "ADN") we first transform the data by using the Chi-square metric. Consider a species abundance data table $Y = [y_{ij}]$ of size $(n \times p)$ with sites (rows) $i = \{1, \ldots, n\}$ and species (columns) $j = \{1, \ldots, p\}$; the row sums are noted y_{i+} and the column sums y_{+j}. The formula for the Chi-square metric between sites x_1 and x_2 across the p species is thus given by:

$$D_{\chi^2\text{metric}}(x_1, x_2) = \sqrt{\sum_{j=1}^{p} \frac{1}{y_{+j}} \left(\frac{y_{1j}}{y_{1+}} - \frac{y_{2j}}{y_{2+}} \right)^2}.$$

If the data $[y_{ij}]$ are transformed into $[y'_{ij}]$ as $y'_{ij} = \frac{y_{ij}}{y_{i+}\sqrt{y_{+j}}}$, then the Euclidean distance between row vectors of transformed data is identical to the Chi-square metric.

In Tables 1 and 2, we present the comparative numerical results provided by our algorithms and K-means which is available on the web site: http://www.bio.umontreal.ca/legendre/index.html. The results of the first set of data are reported in Table 1 while those of the real data are presented in Table 2. We considered two versions of the main DCA which differ from the initialization procedures: DCA with a randomly chosen starting point in the set $\mathcal{A}$ (denoted **DCA1**), and DCA with Procedure KMDCA (with $q = 2$) in the step initialization (denoted **DCA2**). Here "iter"and

Table 1 The performance of **DCA1** and **DCA2** and **K-means** for the first set of data

Data				K-means			DCA1			DCA2		
	m	*n*	*k*	Iter	CPU	Objval	Iter	CPU	Objval	Iter	CPU	Objval
1	88	7	4	12	0.02	21023.26	14	0.01	17686.21	6	0.00	17516.82
2	70	11	4	9	0.22	9107×10^{10}	9	0.01	2321×10^{9}	3	0.01	6684×10^{8}
3	284	11	6	7	0.18	1575×10^{10}	11	0.03	5239×10^{9}	4	0.00	2559×10^{9}
4	354	11	7	8	0.61	8603×10^{10}	9	0.01	7573×10^{9}	4	0.00	1666×10^{9}
5	853	4	8	19	1.03	1221×10^{5}	50	0.37	7067×10^{3}	8	0.01	2948×10^{3}
6	1123	8	12	26	2.03	1203.11	29	1.19	1041.85	8	0.33	989.47
7	200	30	5	13	0.81	613.11	16	0.06	400.92	9	0.05	392.04
8	3500	6	5	14	1.49	7533.11	23	1.55	5112.13	6	0.15	4982.10
9	8516	5	11	38	13.02	98110	45	5.12	65802.50	9	1.91	64782.96
10	11856	4	18	22	9.66	71210	37	6.11	49100	10	1.98	8032

Table 2 The performance of **DCA1** and **DCA2** and **K-means** for the real data

Real data				K-means				DCA1				DCA2			
	m	n	k	Iter	CPU	Objval	PMO	Iter	CPU	Objval	PMO	Iter	CPU	Objval	PMO
PAPILLON	23	4	4	3	0.01	629.00	3	3	0.00	564.10	0	3	0.00	564.1	0
IRIS	150	4	3	4	0.81	289.39	8	6	0.08	159.9	0.6	4	0.05	159.1	0.6
GENE	384	17	5	24	0.73	61303×10^4	25	32	1.12	446221×10^4	12.2	23	0.58	45734×10^4	11.7
VOTE	435	2	2	4	0.03	2307.46	18	7	0.09	1888.44	7.5	1	0.05	1880.7	7.1
ADN	3186	60	3	15	1.95	4686×10^2	21	21	2.77	4501×10^2	4.9	13	1.03	4484×10^2	4.8

"objval" denote, respectively, the number of iterations and the objective value while "PMO" means the percentage of misclassified objects (for the real data in Table 2). All CPU are computed in seconds.

From the numerical experiments we observe that

- Both **DCA1** and **DCA2** are better than **K-means**: the objective values given by **DCA1** and **DCA2** are much smaller than that computed by **K-means**.
- **DCA2** is the best among the three algorithms: it provides the best solution with the shortest time. **DCA2** is very fast and can then handle large-scale problems.
- DCA is very efficient for real data: for "PAPILLON" dataset, all objects are well classified - DCA gives exactly the right clustering; for IRIS dataset only one element is misclassified; likewise, for all other datasets the percentage of misclassified objects is small.

6 Conclusion

We have proposed, for solving a clustering problem with the squared Euclidean distance, a new and efficient approach based on DC programming and DCA. The considered clustering problem has been recast as a DC program in its elegant matrix formulation and with a natural choice of DC decomposition, in order to make simpler and so much less expensive the computations in the resulting DCA. It fortunately turns out that the DC program (1.1) is minimizing the difference of the simplest convex quadratic function (3.4) and the nonsmooth convex one (3.3). This nice feature is very relevant for applying DCA, which consists in solving a sequence of approximate convex quadratic programs whose solutions are explicit. The DCA requires only sums and scalar multiplications of matrices. An interesting procedure that combined DCA and K-means is introduced for initializing DCA. Preliminary numerical simulations on real world database show the robustness, the efficiency and the superiority of DCA with respect to the K-means on the quality of computed solutions. Concerning the running time, their difference is negligible, moreover it is possible to better still improve the explicit computations in DCA in order to make them still further less time-consuming. The efficiency of DCA for this problem suggests to us investigating it in the solution of other models of clustering problems as well as multi-hierarchical clustering. Works in these directions are in progress.

References

1. Alon, N., Spencer, J.H.: The Probabilistic Method. Wiley, New York, NY (1991)
2. Arora, S., Kannan, R.: Learning mixtures of arbitrary Gaussians. In: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, July 6–8, pp. 247–257. Heraklion, Crete, Greece (2001)
3. Al-Sultan, K.: A Tabu search approach to the clustering problem. Pattern Recogn. **28**(9), 443–1453 (1995)
4. Bradley, P.S., Mangasarian, O.L., Street, W.N.: Clustering via concave minimization, Technical Report 96-03, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin May 1996. Advances In: Mozer, M.C., Jordan, M.I., Petsche, T. (eds.) Neural Information processing Systems 9, pp. 368–374. MIT Press, Cambridge, MA Available by ftp://ftp.cs.wisc.edu/math-prog/tech-trports/96-03.ps.Z.
5. Bradley, B.S., Mangasarian, O.L.: Feature selection via concave minimization and support vector machines. In: Shavlik, J. (eds.) Machine Learning Proceedings of the Fifteenth International Conferences(ICML'98), pp. 82–90. San Francisco, CA 1998, Morgan Kaufmann.

6. Charikar, M., Guha, S.: Improved combinatorial algorithms for facility location and k-median problems. In: Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, 17–18 October, pp. 378–388. New York, NY, USA (1999)
7. Charikar, M., Guha, S., Tardos, E., Shmoys, D.B.: A constant-factor approximation algorithm for the k-median problem. In: Proceedings of the 31st Annual ACM Symposium on Theory of Computing pp. 1–10 (1999)
8. De Leeuw J.: Applications of convex analysis to multidimensional scaling, Recent developments. In: Barra, J.R., et al. (eds.) Statistics, pp. 133–145. North-Holland Publishing company, Amsterdam (1997)
9. De Leeuw, J.: Convergence of the majorization method for multidimensional scaling. J. Classi. **5**, 163–180 (1988)
10. Dhilon, I.S., Korgan, J., Nicholas, C.: Feature Selection and Document Clustering. In: Berry, M.W. (ed.) A Comprehensive Survey of Text Mining, pp. 73–100. Springer-Verlag, Berlin (2003)
11. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. Wiley, New York (1972)
12. Feder, T., Greene, D.: Optimal algorithms for approximate clustering. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC) May 2–4, Chicago, Illinois, USA (1988)
13. Fisher, D.: Knowledge acquisition via incremental conceptual clusterin. Mach. Learning, **2**, 139–172 (1988)
14. Fukunaga, K.: Statistical Pattern Recognition. Academic Press, NY (1990)
15. Hiriart Urruty, J.B., Lemarechal, C.: Convex Analysis and Minimization Algorithms. Springer Verlag, Berlin, Heidelberg (1993)
16. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data, Prentice-Hall Inc, Englewood Cliffs, NJ (1988)
17. Hartigan, J.A.: Clustering Algorithms. Wiley, New York (1975)
18. Le Thi Hoai An: Contribution à l'optimisation non convexe et l'optimisation globale: Théorie, Algoritmes et Applications. Habilitation à Diriger des Recherches, Université de Rouen, Juin (1997)
19. Le Thi Hoai An, Pham Dinh Tao: Solving a class of linearly constrained indefinite quadratic problems by DC algorithms. J. Global Optim. **11**(3), 253–285 (1997)
20. Le Thi Hoai An, Pham Dinh Tao: DC programming approach for large-scale molecular optimization via the general distance geometry problem. Nonconvex Optimization and Its Applications, Special Issue "Optimization in Computational Chemistry and Molecular Biology: Local and Global Approaches", pp. 301–339. Kluwer Academic Publishers, Dordrecht (2000) (This special issue contains refereed invited papers submitted at the conference on optimization in computational chemistry and molecular biology: local and global approaches held at Princeton University, 7–9 May 1999)
21. Le Thi Hoai An, Pham Dinh Tao: DC programming approach for solving the multidimensional scaling problem. Nonconvex optimizations and its applications: special issue "From local to global optimization", pp. 231–276. Kluwer Academic Publishers, Dordrecht (2001)
22. Le Thi Hoai An, Pham Dinh Tao: DC programming: theory, algorithms and applications. The state of the art. In Proceedings of The First International Workshop on Global Constrained Optimization and Constraint Satisfaction (Cocos' 02), 28 p. Valbonne-Sophia Antipolis, France, 2–4 October (2002)
23. Le Thi Hoai An, Pham Dinh Tao: Large Scale Molecular Optimization from distances matrices by a DC optimization approach. SIAM J. Optim. **14**(1), 77–116 (2003)
24. Le Thi Hoai An, Pham Dinh Tao: The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. Ann. Oper. Res. **133**, 23–46 (2005)
25. Liu, Y., Shen, X., Doss, H.: Multicategory ψ–learning and support vector machine (29 p.). Conference on Machine Learning, Statistics and Discovery, June 22–26, Department of Statistics, the Ohio State University (2003)
26. Mangasarian, O.L.: Mathematical programming in data mining. Data Mining and Knowl. discov. **1**, 183–201 (1997)
27. MacQueen, J.B.: Some Methods for classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability", Berkeley, University of California Press, **1**, 281–297 (1967)
28. Meyerson, A., O'Callaghan, L., Plotkin, S.: A k-Median algorithm with running time independent of data size. Machine Learn. **56**, 61–87 (2004)
29. Neumann, J., Schnörr, C., Steidl, G.: SVM-based feature selection by direct objective minimisation, Pattern Recognition. In: Proceeding of 26th DAGM Symposium, vol. 3175, pp. 212–219, LNCS (2004)

30. Pham Dinh Tao.: Convergence of subgradient method for computing the bound-norm of matrices, Linear Algebra Appl. **62**, 163–182 (1984)
31. Pham Dinh Tao.: Algorithmes de calcul d'une forme quadratique sur la boule unité de la norme du maximum. Numer. Math. **45**, 377–440 (1984)
32. Pham Dinh Tao, Le Thi Hoai An.: Convex analysis approach to d.c. programming: Theory, Algorithms and Applications. Acta Mathematica Vietnamica, (dedicated to Professor Hoang Tuy on the occasion of his 70th birthday), **22**(1), 289–355 (1997)
33. Pham Dinh Tao, Le Thi Hoai An.: DC optimization algorithms for solving the trust region subproblem. SIAM J. Optim. **8**, 476–505 (1998)
34. Rao, M.R.: Cluster analysis and mathematical programming. J. Amer. Stat. Associ., **66**, 622–626 (1971)
35. Rockafellar, R.T.: Convex Analysis. Princeton University, Princeton (1970)
36. Selim, S.Z., Ismail, M.A.: K-means-Type algorithms: a generalized convergence theorem and characterization of local optimilaty. (Book Series) IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6, 81–87 (1984)
37. Schüle, T., Schnörr, C., Weber, S., Hornegger, J.: Discrete tomography by convex-concave regularization and d.c. programming. Discr. Appl. Math. **151**, 229–243, (2005)
38. Weber, S., Schüle, T., Schnörr, C.: Prior learning and convex-concave regularization of binary tomography. Electr. Notes in Discr. Math. **20**, 313–327, (2005)
39. Weber, S., Schnörr, C., Schüle, T., Hornegger, J., Binary Tomography by Iterating Linear Programs, Klette, R., Kozera, R., Noakes, L., and Weickert, J., (eds.) Computational Imaging and Vision – Geometric Properties from Incomplete Data, Kluwer Academic Press, Dordrecht 2005.
40. Wong, T., Katz, R., McCanne, S.: A preference clustering protocol for large-Scale Multicast Applications, Proceedings of the First International COST264 Workshop on Networked Group Communication, November 17–20, LNCS, pp. 1–18. Pisa, Italy (1999)
41. Wolberg, W.H., Street, W.N., Mangasarian, O.L. Image analysis and machine learning applied to breast cancer diagnosis and prognosis. Anal. Quant. Cytol. Histol. **17**, 2, 77–87 (1995)
42. Wolberg, W.H., Street, W.N., Heisey, D.M., Mangasarian, O.L.: Computerized breast cancer diagnosis and prognosis from fine-needle aspirates. Arch. Surg. **130**, 511–516 (1995)
43. Wolberg, W.H., Street, W.N., Heisey, D.M., Mangasarian, O.L.: Computer-derived nuclear features distinguish malignant from benign breast cytology. Hum. Patholo., **26**, 792–796 (1995)
44. Yuille, A.L., Rangarajan, A.: The Concave-Convex Procedure (CCCP). Advances in Neural Information Processing Systems 14, pp. 1033–1040. MIT Press, Cambridge, MA, 2002