DC Programming and DCA for clustering using weighted dissimilarity measures

LE Hoai Minh, TA Minh Thuy, LE THI Hoai An
Laboratory of Theoretical and Applied Computer Science - LITA EA 3097
University of Lorraine
Ile de Saulcy, 57045 Metz, France
minh.le@univ-lorraine.fr
minh-thuy.ta5@etu.univ-lorraine.fr
hoai-an.le-thi@univ-lorraine.fr

PHAM DINH Tao
Laboratory of Mathematics, National Institute for Applied Sciences - Rouen
Avenue de l'Université, BP 8, 76801 Saint-Etienne-du-Rouvray cedex, France
pham@insa-rouen.fr

Abstract

The purpose of this paper is to develop new efficient approach based on DC (Difference of Convex functions) programming and DCA (DC Algorithm) for clustering using weighted dissimilarity measures which is formulated as a hard combinatorial optimization problem. DC reformulation technique and exact penalty in DC programming are developed to build an appropriate equivalent DC program which leads to an explicit DCA scheme. Numerical results on real word datasets show the efficiency, the scalability of DCA and its superiority with respect to standard algorithm.

1 Introduction

Clustering, which aims at dividing a data set into groups or cluster containing similar data, is a fundamental problem in unsupervised learning and has many applications in various domains. Usually, the distance function involves all attributes of the data set. It is applicable if most attributes are important to every cluster. However, the performance of clustering algorithms can be significantly degraded if many irrelevant attributes are used. In the literature, various approaches have been proposed to address this problem. The first strategy is feature selection that finds irrelevant features and removes them from the feature set before constructing a classifier. Feature weighting is an extension of the feature selection where the features are assigned continuous weights. Relevant features correspond to a high weight value, whereas a weight value close to zero represent irrelevant features.

Let $\mathcal{X} := \{x_1, x_2, ..., x_n\}$ be a data set of n entities in $\mathbb{R}^m$ and the known number of clusters k $(2 \leq k \leq n)$. Denote by Λ a $k \times m$ matrix defined as $\Lambda = (\lambda_{l,i})$ where $\lambda_{l,i}$ defines the relevance of i-th feature to the cluster C_l. Let $W = (w_{j,l}) \in \mathbb{R}^{n \times k}$ with $j = 1, \ldots, n$ and $l = 1, \ldots, k$ be the matrix defined by: $w_{j,l} := 1$ if $x_j \in C_l$; $w_{j,l} := 0$ otherwise. We are to partition the set $\mathcal{X}$ into k clusters in order to minimize the sum of squared distances from the entities to the centroid of their cluster. The dissimilarity measure is defined by m weighted attributes. Then a straightforward formulation of the clustering using weighted dissimilarity measures is (β is an exponent greater than 1):

$$\begin{cases} \min F(W,Z,\Lambda) := \sum\limits_{l=1}^{k}\sum\limits_{j=1}^{n}\sum\limits_{i=1}^{m} w_{jl}\lambda_{li}^{\beta}(z_{li}-x_{ji})^2 \\ s.t: \sum\limits_{l=1}^{k} w_{jl} = 1, j = 1..n, \\ \quad \sum\limits_{i=1}^{m} \lambda_{li} = 1, l = 1..k, \\ \quad w_{jl} \in \{0,1\}, j = 1..n, l = 1..k, \\ \quad \lambda_{li} \in [0,1], l = 1..k, i = 1..m. \end{cases} \tag{1}$$

Problem (1) is a hard combinatorial optimization problem for which no efficient global algorithm is available. The problem is difficult not only because it is combinatorial but also due to the nonconvexity of the objective function. In [2], the authors consider a K-means type algorithm, denoted **WF-K-means**, to solve the problem (1). At first, **WF-K-means** fixes Z, Λ and finds W to minimize $F(W,.,.)$. Then W, Λ are fixed for finding Z minimizing $F(.,Z,.)$. Finally, Λ is obtained by minimizing $F(.,.,\Lambda)$ with W and Z fixed. The process is repeated until no more improvement in the objective function can be made.

We investigate in this work, for solving the problem (1), an efficient nonconvex programming approach based on DC (Difference of Convex functions) programming and DCA (DC Algorithms) that were introduced by Pham Dinh Tao in a 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 increasingly popular (see e.g. [3, 4] and the list of references in [6]).

The remainder of the paper is organized as follows. The solution of the problem (1) by DCA is developed in Section 2. Finally, computational results are reported in the last section.

2 DCA for solving problem (1)

2.1 Outline of DC programming and DCA

DC programming and DCA constitute the backbone of smooth/nonsmooth nonconvex programming and global optimization. They address the problem of minimizing a function f which is the difference of two convex functions on the whole space $\mathbb{R}^d$ or on a convex set $C \subset \mathbb{R}^d$. Generally speaking, a DC program is an optimisation problem of the form :

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

where g, h are lower semi-continuous proper convex functions on $\mathbb{R}^d$. The idea of DCA is simple: each iteration l of DCA approximates the concave part $-h$ by its affine majorization (that corresponds to taking $y^l \in \partial h(x^l)$) and minimizes the resulting convex function (that is equivalent to determining a point $x^{l+1} \in \partial g^*(y^l)$ with g^* is the conjugate function of the convex function g).

The generic DCA scheme is shown below.
DCA scheme Initialization: Let $x^0 \in \mathbb{R}^d$ be a best guess, $r = 0$.

Repeat

- Calculate $y^r \in \partial h(x^r)$
- Calculate $x^{r+1} \in \arg\min\{g(x) - h(x^r) - \langle x - x^r, y^r\rangle : x \in \mathbb{R}^d\} \quad (P_l)$
- $r = r + 1$

Until convergence of $\{x^r\}$.

For a complete study of DC programming and DCA the reader is referred to [3, 4], and the references therein.

2.2 A continuous reformulation of problem (1)

Since $w_{j,l} \in \{0,1\}$ we can replace $w_{j,l}$ by $w_{j,l}^2$ and rewrite the objective function of (1) by $F(W,Z,\Lambda) := \sum\limits_{l=1}^{k}\sum\limits_{j=1}^{n}\sum\limits_{i=1}^{m} w_{jl}^2\lambda_{li}^{\beta}(z_{li}-x_{ji})^2$.

In the problem (1) the variables W and Λ are a priori bounded. One can also find a constraint for bound the variable Z. Indeed, let $\alpha_i := \min_{j=1,\dots,n} x_{j,i},\ \gamma_i := \max_{j=1,\dots,n} x_{j,i}$. Hence $z_l \in \mathcal{T}_l := \Pi_{i=1}^m [\alpha_i, \gamma_i]$ for all $l = 1, \dots, k$. Finally, $Z \in \mathcal{T} := \Pi_{l=1}^k \mathcal{T}_l$.

Let Δ_l (resp. $\mathcal{C}_j$) be the $(m-1)$-simplex in $\mathbb{R}^m$(resp. $(k-1)$-simplex in $\mathbb{R}^k$), for each $l \in \{1, \dots, k\}$ (resp. for each $j \in \{1, \dots, n\}$), defined by

$$\Delta_l := \left\{ \Lambda_l := (\lambda_{l,i})_l \in [0,1]^m : \sum_{i=1}^m \lambda_{l,i} = 1 \right\}, \quad \mathcal{C}_j := \left\{ W_j := (w_{j,l})_j \in [0,1]^k : \sum_{l=1}^k w_{j,l} = 1 \right\}$$

and $\mathcal{C} := \Pi_{j=1}^n \mathcal{C}_j, \mathcal{T} := \Pi_{l=1}^k \mathcal{T}_l, \Delta := \Pi_{l=1}^k \Delta_l$. The problem (1) can be rewritten as:

$$\min \left\{ F(W, Z, \Lambda) : W \in \mathcal{C} \cap \{0,1\}^{n \times k}, Z \in \mathcal{T}, \Lambda \in \Delta \right\}. \tag{2}$$

Consider the function p defined on $\mathbb{R}^{n\times k}$ by $p(W) := \sum_{j=1}^n \sum_{l=1}^k w_{j,l}(1 - w_{j,l})$. Clearly that p is finite concave on $\mathbb{R}^{n \times k}$, nonnegative on $\mathcal{C}$, and

$$\mathcal{C} \cap \{0,1\}^{n\times k} = \{W \in \mathcal{C} :\ p(W) = 0\} = \{W \in \mathcal{C} :\ p(W) \le 0\}.$$

$F(W, Z, \Lambda)$ is a DC function (cf. 2.3), the feasible set is bounded convex, using the exact penalty theorem [5], we can now write (1) in the form of the following nonconvex program in continuous variables ($t > t_0 \ge 0$ is called penalty parameter):

$$\min \left\{ F_1(W, Z, \Lambda) := F(W, Z, \Lambda) + tp(W) : W \in \mathcal{C}, Z \in \mathcal{T}, \Lambda \in \Delta \right\}, \tag{3}$$

In the next section, we will develop DC programming and DCA for solving (3).

2.3 A DC decomposition of (3)

Proposition 1 *There exists $\rho > 0$ such that the function $h(u, v, y) := \frac{\rho}{2}\left(u^2 + v^2 + y^2\right) - u^2 y^\beta (v - a)^2$ is convex on $(u, v, y) \in [0,1] \times [\alpha, \gamma] \times [0,1]$.*

Using the above proposition, for $u \leftarrow w_{jl}, v \leftarrow z_{li}, y \leftarrow \lambda_{li}$, the function

$$h_{lij}(w_{jl}, z_{li}, \lambda_{li}) = \frac{\rho}{2}\left(w_{jl}^2 + z_{li}^2 + \lambda_{li}^2\right) - w_{jl}^2 \lambda_{li}^\beta (z_{li} - x_{ji})^2 \tag{4}$$

is convex on $\{w_{jl} \in [0,1], z_{li} \in [\alpha_i, \gamma_i], \lambda_{li} \in [0,1]\}$.

As a consequence, the function $H(W, Z, \Lambda)$ defined by

$$H(W, Z, \Lambda) := \sum_{l=1}^k \sum_{j=1}^n \sum_{i=1}^m \left[\frac{\rho}{2}\left(w_{jl}^2 + z_{li}^2 + \lambda_{li}^2\right) - w_{jl}^2 \lambda_{li}^\beta (z_{li} - x_{ji})^2 \right] \tag{5}$$

is convex on $\{W \in \mathcal{C}, Z \in \mathcal{T}, \Lambda \in \Delta\}$. Finally, we can express our DC decomposition as follows:

$$F_1(W, Z, \Lambda) := G_1(W, Z, \Lambda) - H_1(W, Z, \Lambda) \tag{6}$$

with

$$G_1(W, Z, \Lambda) := \frac{\rho}{2} \sum_{l=1}^k \sum_{j=1}^n \sum_{i=1}^m \left(w_{jl}^2 + z_{li}^2 + \lambda_{li}^2\right); H_1(W, Z, \Lambda) := H(W, Z, \Lambda) - tp(W)$$

being clearly convex functions.

2.4 DCA applied to (3)

For designing a DCA applied to (3), we first need to compute $(\bar{W}^r, \bar{Z}^r, \bar{\Lambda}^r) \in \partial H_1(W^r, Z^r, \Lambda^r)$ and then have to solve the convex program

$$\min \left\{ \frac{\rho}{2} \sum_{l=1}^k \sum_{j=1}^n \sum_{i=1}^m \left(w_{jl}^2 + z_{li}^2 + \lambda_{li}^2\right) - \langle (W, Z, \Lambda), (\bar{W}^r, \bar{Z}^r, \bar{\Lambda}^r) \rangle : W \in \mathcal{C}, Z \in \mathcal{T}, \Lambda \in \Delta \right\}. \tag{7}$$

The function H_1 is differentiable and its gradient at the point (W^r, Z^r, Λ^r) is given by:

$$\begin{aligned}
\bar{W}^r &= \nabla_W H_1(W,Z,\Lambda) = \left(m\rho w_{jl} - \sum_{i=1}^{m} 2 w_{jl} \lambda_{li}^{\beta} (z_{li} - x_{ji})^2 + t(2w_{jl} - 1) \right)_{j=1..n}^{l=1..k}, \\
\bar{Z}^r &= \nabla_Z H_1(W,Z,\Lambda) = \left(n\rho z_{li} - \sum_{j=1}^{n} 2 w_{jl}^2 \lambda_{li}^{\beta} (z_{li} - x_{ji}) \right)_{l=1..k}^{i=1..m}, \\
\bar{\Lambda}^r &= \nabla_\Lambda H_1(W,Z,\Lambda) = \left(n\rho \lambda_{li} - \sum_{j=1}^{n} \beta w_{jl}^2 \lambda_{li}^{\beta-1} (z_{li} - x_{ji})^2 \right)_{l=1..k}^{i=1..m}.
\end{aligned} \tag{8}$$

The solution of the auxiliary problem (7) is explicitly computed as (Proj stands for the projection)

$$\begin{aligned}
&(W^{r+1})_j = \mathrm{Proj}_{\mathcal{C}_j}\left(\tfrac{1}{m\rho}(\bar{W}^r)_j\right) \; j = 1,...n; \quad (Z^{r+1})_{li} = \mathrm{Proj}_{[\alpha_i,\gamma_i]}\left(\tfrac{1}{n\rho}(\bar{Z}^r)_{li}\right) \; l = 1,..,k, i = 1,...m; \\
&(\Lambda^{r+1})_l = \mathrm{Proj}_{\Delta_l}\left(\tfrac{1}{n\rho}(\bar{\Lambda}^r)_l\right) \; l = 1,...k.
\end{aligned} \tag{9}$$

The algorithm can be described as follows.

WF-DCA: DCA applied to (3)

- **Initialization:** Choose W^0, Z^0 and Λ^0. Let $\epsilon > 0$ be sufficiently small, $r = 0$.
- **Repeat**
  - Compute $(\bar{W}^r, \bar{Z}^r, \bar{\Lambda}^r)$ via (8).
  - Compute $(W^{r+1}, Z^{r+1}, \Lambda^{r+1})$ via (9).
  - $r = r + 1$
- **Until** $\|(W^{r+1}, Z^{r+1}, \Lambda^{r+1}) - (W^r, Z^r, \Lambda^r)\| \leq \epsilon$ or $|F(W^{r+1}, Z^{r+1}, \Lambda^{r+1}) - F(W^r, Z^r, \Lambda^r)| \leq \epsilon$.

3 Numerical experiments

Numerical experiments were performed on 7 real-world datasets: *Breast Cancer Wiscosin, Ionosphere, Wine, Vote, Wave form, Pima and Magic* taken from UCI Machine Learning Repository. All algorithms clustering was implemented in the Visual C++ 2008, and performed on a PC Intel i5 CPU650, 3.2 GHz of 4GB RAM. The information about data sets is summarized in Table 1.

The following criteria were used to compare the performances of algorithms: the percentage of well classified points (PWCO), the CH value and the CPU time in seconds.

The PWCO is measured by the clustering accuracy defined as:

$$PWCO := \frac{1}{n}\sum_{j=1}^{k} a_j * 100\%$$

where a_j is the number of instances occurring in both cluster j and its corresponding generated cluster label. Good clustering is distinguished by large value of PWCO.

The CH value, introduced by Calinski and Harabasz ([7, 1]), is expressed as follows:

$$CH(k) := \frac{[traceB/(k-1)]}{[traceW/(n-k)]}$$

where

$$traceB := \sum_{i=1}^{k} |C_i| \, \|\overline{C_i} - \overline{x}\|^2; \;\; traceW := \sum_{i=1}^{k} \sum_{j \in C_i} \|x_j - \overline{C_i}\|^2$$

with $|C_i|$ is the number of objects assigned to the cluster C_i ($i = 1, \ldots, k$); $\overline{C_i} = \frac{1}{|C_i|} \sum_{j \in C_i} x_j$ and $\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$.

Table 1: Comparative results between **WF-DCA** and **WF-K-means**

Data				WF-K-means			WF-DCA		
	n	m	k	PWCO	CH	Time	PWCO	CH	Time
Breast Cancer	683	9	2	95,01%	979	**0,04**	**96,43%**	**1007**	0,08
Ionosphere	351	34	2	64,76%	21	**0,03**	**72,65%**	**157**	0,30
Wine	178	13	3	92,98%	80	**0,03**	**95,06%**	**81**	0,05
Vote	435	16	2	84,94%	149	**0,04**	**87,84%**	**157**	0,07
Wave	5000	40	3	49,49%	885	14,90	**60,03%**	**1705**	**4,47**
Pima	768	8	2	64,06%	11	0,09	**71,09%**	**502**	**0,02**
Magic	19020	3	2	55,31%	1266	**4,22**	**65,29%**	**9348**	13,68
Average				72,34%	498	**2,78**	**78,14%**	**1845**	2,90

We report in Table 1, the average results (of ten executions) of **WF-DCA** and **WF-K-means**([2]) algorithms.

From the numerical results, we observe that: in all experiments, **WF-DCA** gives better solutions (not only by the value PWCO, but also by the value CH) without a big increase in the CPU time.

Conclusion: we have rigorously studied the DC programming and DCA for clustering using weighted feature. Based on the reformulation technique and exact penalty in DC programming, the hard combinatorial optimization model has been recast as a DC program. It fortunately turns out that the corresponding DCA consists in computing, at each iteration, the projection of points onto a simplex and/or a rectangle, that all are given in the explicit form. Computational experiments show the efficiency and the superiority of DCA with respect to the standard algorithm **WF-K-means**([2]).

References

[1] Calinski T. and Harabasz J. (1974). A dendrite method for cluster analysis. Communications in Statistics Simulation and Computation, 3(1), 1–27.

[2] Chan E.Y., Ching W.K., Michael K.N. and Huang Z.J. (2004). An optimizationalgorithm for clustering using weighted dissimilarity measures Pattern Recognition 37(5),943-952.

[3] Le Thi, H.A. (1997). Contribution à l'optimisation non convexe et l'optimisation globale: Théorie, Algoritmes et Applications. Habilitation à Diriger des Recherches, Uni. Rouen.

[4] Le Thi, H.A. & Pham Dinh, T. (2005). The DC (Difference of Convex functions) Programming and DCA revisited with DC models of real world nonconvex optimization problems. Annals of Operations Research, 133, 23–46.

[5] Le Thi, H.A., Pham Dinh, T. & Huynh, V.N (2011). Exact penalty techniques in DC programming. Journal of Global Optimization, 1–27, doi:10.1007/s10898-011-9765-3.

[6] Le Thi, H.A. DC Programming and DCA. http://lita.sciences.univ-metz.fr/∼lethi.

[7] Vendramin L., Ricardo J. G. B. Campello, and Eduardo R. Hruschka (2009). On the comparison of relative clustering validity criteria. Proceedings of the Ninth SIAM International Conference on Data Mining, April 2009, Nevada, 733–744.