HAL
open science

Learning with sparsity and uncertainty by Difference of Convex functions optimization

Xuan Thanh Vo

To cite this version:

Xuan Thanh Vo. Learning with sparsity and uncertainty by Difference of Convex functions optimization. Other [cs.OH]. Université de Lorraine, 2015. English. NNT : 2015LORR0193 . tel-01754450

HAL Id: tel-01754450

https://hal.univ-lorraine.fr/tel-01754450v1

Submitted on 30 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE LORRAINE

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

THÈSE

en vue de l'obtention du titre de

DOCTEUR DE L'UNIVERSITÉ DE LORRAINE

(arrêté ministériel du 7 Août 2006)

Spécialité Informatique

présentée par

Vo Xuan Thanh

Titre de la thèse :

Apprentissage avec la parcimonie et sur des données incertaines par la programmation DC et DCA

—

Learning with sparsity and uncertainty by Difference of Convex functions optimization

soutenue le 15 octobre 2015

Composition du Jury :

Président	HENROT Antoine	*Professeur, Université de Lorraine*
Rapporteurs	CARRIZOSA Emilio	*Professeur, University of Seville*
	VERT Jean-Philippe	*Directeur de recherche, MINES ParisTech*
Examinateurs	CHEVALEYRE Yann	*Professeur, Université Paris 13*
	LERAY Philippe	*Professeur, Polytech Nantes*
	PHAM DINH Tao	*Professeur émérite, INSA de Rouen*
Directeur de thèse	LE THI Hoai An	*Professeur, Université de Lorraine*

Thèse préparée au sein de laboratoire
d'Informatique Théorique et Appliquée (LITA)
Université de Lorraine, Metz, France

Remerciements

Cette thèse a été réalisée au sein du Laboratoire d'Informatique Théorique et Appliquée (LITA) de l'Université de Lorraine, sous la direction du Madame le Professeur LE THI Hoai An, Directrice du LITA, Université de Lorraine.

En premier lieu, je voudrais remercier Madame le Professeur LE THI Hoai An. En tant que Directeur de thèse, elle m'a dirigé mes travaux de thèse et m'a aidé à lancer une carrière scientifique. Tout au long de plus de trois années, elle m'a donné son soutien permanent avec l'extraordinaire patience. Je la remercie pour tous les conseils qui sont très utiles et importants de la science ainsi que de la vie.

J'adresse également mes sincères remerciements à Monsieur PHAM DINH Tao, professeur à l'INSA de Rouen pour ses conseils, et son suivi dans mes recherches. Je voudrais lui exprimer toute ma reconnaissance pour les discussions très intéressantes qu'il a menées et pour m'avoir suggéré de nouvelles voies de recherche.

Je souhaite vivement remercier Monsieur CARRIZOSA Emilio, professeur à l'Université de Seville et Monsieur VERT Jean-Philippe, directeur de recherche à MINES ParisTech, de m'avoir fait l'honneur d'accepter la charge de rapporteur de ma thèse, ainsi que d'avoir participé à juger mon travail.

Je souhaite également exprimer ma gratitude à Monsieur CHEVALEYRE Yann, professeur à l'Université Paris 13, Monsieur LERAY Philippe, professeur à Polytech Nantes, et Monsieur HENROT Antoine, professeur à l'Université de Lorraine, de participer au jury.

Un grand merci à mes collègues du LITA ainsi que mes amis au Metz pour les aides et les moments agréables lors de mon séjour en France.

Enfin, les mots les plus simples étant les plus forts, j'adresse toute mon affection à ma famille.

Résumé

Ces dernières années ont vu une explosion d'intérêt d'apprentissage avec la parcimonie et/ou avec l'incertitude des données. Dans cette thèse, nous nous concentrons sur le développement des méthodes d'optimisation pour résoudre certaines classes de problèmes concernant ces deux sujets. Nos méthodes sont basées sur la programmation DC (Difference of Convex functions) et DCA (DC Algorithms) étant reconnues comme des outils puissants d'optimisation.

La thèse se compose de deux parties : La première partie concerne la parcimonie tandis que la deuxième partie traite l'incertitude des données. La première partie est composée de trois chapitres. Dans le premier chapitre, une étude approfondie pour la minimisation de la norme zéro a été réalisée tant sur le plan théorique qu'algorithmique. Nous considérons une approximation DC commune de la norme zéro qui inclut toutes les fonctions de pénalité standard. Nous développons quatre algorithmes basées sur la programmation DC et DCA pour résoudre le problème approché et nous prouvons que nos algorithmes couvrent tous les algorithmes standards existants dans le domaine. Dans le deuxième chapitre, nous étudions le problème de la factorisation en matrices non-négatives (NMF). Le problème consiste à approcher une matrice non négative par le produit de deux matrices non négatives de rang faible. Nous étudions la structure du problème considéré et fournissons des algorithmes appropriés basés sur la programmation DC et DCA. Nous étudions également le problème de NMF parcimonieuse. Poursuivant cette étude, dans le troisième chapitre, nous étudions le problème d'apprentissage de dictionnaire où la représentation parcimonieuse joue un rôle crucial. Dans ce problème, nous utilisons l'approche d'approximation précitée dans le cadre de la factorisation de matrice. L'application en traitement d'image est effectuée pour illustrer l'efficacité de notre méthode.

La deuxième partie se compose de deux chapitres. Nous exploitons la technique d'optimisation robuste pour traiter l'incertitude des données pour les deux problèmes importants dans l'apprentissage : la sélection de variables dans SVM (Support Vector Machines) et le clustering. Dans ce contexte, les données sont incertaines, mais varient dans un ensemble d'incertitude bornée. Différents modèles (rectangulaire / sphérique / ellipsoïdales) sont étudiés. Les algorithmes basés sur DCA sont développés pour résoudre ces problèmes. L'expérimentation sur différents types de données démontre l'efficacité des algorithmes proposés.

Abstract

Recent years have witnessed a surge of interest in sparsity and robust optimization for data uncertainty. In this thesis, we focus on developing optimization approaches for solving some classes of optimization problems in these two topics. Our methods are based on DC (Difference of Convex functions) programming and DCA (DC Algorithms) which are well-known as powerful tools in optimization.

This thesis is composed of two parts: the first part concerns with sparsity while the second part deals with uncertainty. The first part includes three chapters. In the first chapter, a unified DC approximation approach to optimization problem involving the zero-norm in objective is thoroughly studied on both theoretical and computational aspects. We consider a common DC approximation of zero-norm that includes all standard sparse inducing penalty functions, and develop general DCA schemes that cover all standard algorithms in the field. In the second chapter, the thesis turns to the nonnegative matrix factorization (NMF) problem that seeks to approximate a nonnegative matrix by the product of two low-rank nonnegative matrices. We investigate the structure of the considered problem and provide appropriate DCA based algorithms. To enhance the performance of NMF, the sparse NMF formulations are proposed. Continuing this topic, in the third chapter, we study the dictionary learning problem where sparse representation plays a crucial role. We use the aforementioned DC approximation approach to sparse optimization within the framework of matrix factorization. Application in image processing is conducted to illustrate the efficiency of our method.

The second part includes two chapters. We exploit robust optimization technique to deal with data uncertainty for two important problems in machine learning: feature selection in linear Support Vector Machines and clustering. In this context, individual data point is uncertain but varies in a bounded uncertainty set. Different models (box/spherical/ellipsoidal) related to uncertain data are studied. DCA based algorithms are developed to solve the robust problems. All the proposed algorithms have been verified on the synthetic and real-world datasets.

VO Xuan Thanh

Né le 12 Septembre, 1989 (Viet Nam)

Tél: 06 38 01 56 57

E-mail: xuan-thanh.vo@univ-lorraine.fr

Adresse personnelle: P1117, Res Univ Saulcy, Ile du Saulcy, 57010 Metz

Adresse professionnelle: Bureau E408, LITA – Université de Lorraine, Ile du Saulcy, 57045 Metz

Situation Actuelle

Depuis Septembre 2012	Doctorant au Laboratoire d'Informatique Théorique et Appliquée (LITA EA 3097) de l'Université de Lorraine. Encadré par Prof. Le Thi Hoai An. Sujet de thèse : **"Apprentissage avec la parcimonie et sur des données incertaines par la programmation DC et DCA"**

Experience Professionnelle

4/2012–6/2012	Stagiaire au laboratoire LITA, UFR MIM, Université de Lorraine, France. Responsable de stage: Prof. Le Thi Hoai An. Mémoire: **Robust optimization for classification under uncertain data**.
10/2011–3/2012	Enseignant, Université des Sciences - Ho Chi Minh Ville, Vietnam.

Diplôme et Formation

2012 au present	Doctorant en Informatique. LITA–Université de Lorraine, Metz.
2011–2012	Master 2 en Mathématiques, Université de Tours, France.
2007–2011	Diplôme universitaire en Mathématique et Informatiques, Université des Sciences - Ho Chi Minh Ville, Vietnam.

Publications

Refereed international journal papers

[1] Hoai An Le Thi, Tao Pham Dinh, Hoai Minh Le, Xuan Thanh Vo. DC approximation approaches for sparse optimization. European Journal of Operational Research 244 (1): 26–46 (2015).

[2] Hoai An Le Thi, Xuan Thanh Vo, Tao Pham Dinh. Feature Selection for SVMs under Uncertain Data: Robust optimization based on Difference of Convex functions Algorithms. Neural Networks 59: 36–50 (2014).

[3] Hoai An Le Thi, Xuan Thanh Vo, Tao Pham Dinh. Efficient Nonegative Matrix Factorization via DC programming and DCA. *Revised version to* Neural Computation.

[4] Xuan Thanh Vo, Hoai An Le Thi. Robust Optimization for Clustering Uncertain Data. *Submitted to* Pattern Recognition.

[5] Xuan Thanh Vo, Hoai An Le Thi. DC Programming and DCA for Dictionary Learning and Application in Image Denoising. *Submitted*.

Refereed papers in books / Refereed international conference papers

[1] Xuan Thanh Vo, Hoai An Le Thi, Tao Pham Dinh. Robust Optimization for Clustering. *Submitted to* ACIIDS 2016: 8th Asian Conference on Intelligent Information and Database Systems.

[2] Xuan Thanh Vo, Hoai An Le Thi, Tao Pham Dinh, Thi Bich Thuy Nguyen. DC Programming and DCA for Dictionary Learning. in M. Nunez et al. (Eds), ICCCI 2015: Proceedings of 7th International Conference on Computational Collective Intelligence Technologies and Applications, part I, LNAI 9329, pp. 295–304, Springer 2015.

[3] Thi Bich Thuy Nguyen, Hoai An Le Thi, Hoai Minh Le, Xuan Thanh Vo. DC approximation approach for L0–minimization in compressed sensing. in H.A. Le Thi et al. (Eds.), ICCSAMA 2015: Proceedings of 3rd International Conference on Computer Science, Applied Mathematics and Applications, AISC 358, pp. 37–48, Springer 2015.

[4] Hoai An Le Thi, Xuan Thanh Vo, Tao Pham Dinh. DC programming and DCA for nonnegative matrix factorization. in D. Hwang et al. (Eds), ICCCI 2014: Proceedings of 6th International Conference on Computational Collective Intelligence Technologies and Applications, LNAI 8733, pp. 573–582, Springer 2014.

[5] Hoai An Le Thi, Anh Vu Le, Xuan Thanh VO, Ahmed Zidna. A Filter Based Feature Selection Approach in MSVM Using DCA and Its Application in Network Intrusion Detection. in N.T. Nguyen et al. (Eds), ACIIDS 2014: Proceedings of the 6th Asian Conference on Intelligence Information and Database Systems, LNAI 8398, pp. 403–413, Springer 2014.

[6] Hoai An Le Thi, Xuan Thanh Vo, Tao Pham Dinh. Robust Feature Selection for SVMs under Uncertain Data, in P. Perner (Ed), ICDM 2013: Proceedings of the 13th Industrial Conference on Advances in Data Mining, LNAI 7987, pp. 151–165, Springer 2013.

Communications in national / International conferences

[1] Xuan Thanh Vo, Hoai An Le Thi. Robust Optimization for Clustering. Accepted for presentation in "The 27th European Conference on Operational Research, Glasgow, UK, July 12 - 15, 2015.

[2] Xuan Thanh Vo, Hoai An Le Thi, Tao Pham Dinh. DC programming and DCA for Dictionary Learning. The 20th Conference of the International Federation of Operational Research Societies (IFORS 2014) Barcelona, Spain, July 13 - 18, 2014.

[3] Xuan Thanh Vo, Hoai An Le Thi, Tao Pham Dinh. DC Programming and DCA for Nonnegative Matrix Factorization. Conférence sur l'Optimisation, la Planification et la Fusion Multi-Capteurs (PFMC 2013), Brest, France, October 23-24, 2013.

[4] Xuan Thanh Vo, Hoai An Le Thi, Tao Pham Dinh. DC Programming and DCA for Nonnegative Matrix Factorization. The 26th European Conference on Operational Research (EURO-INFORMS 2013), Rome, Italy, July 1 - 4, 2013.

Contents

Résumé **3**

Introduction générale **21**

1 Preliminary **27**

1.1 DC programming and DCA 27

1.1.1 Fundamental convex analysis 27

1.1.2 DC optimization 30

1.1.3 DC Algorithm (DCA) 33

1.1.4 Approximate DCA 37

1.2 Robust optimization 39

I Learning with sparsity **41**

2 DC approximation approaches for sparse optimization[1] **43**

2.1 Introduction 43

2.2 DC approximation approaches: consistency results 49

2.3 DC approximation functions 55

2.4 A deeper study on Capped-ℓ_1 approximation problems 57

2.4.1 Link between approximation and exact penalty approaches 57

2.4.2 A special case: link between the original problem (2.1) and Capped-ℓ_1 approximation problem . . . 60

2.4.3 Extension to other approximations . . . 62

2.5 DCA for solving the problem (2.8) . . . 63

2.5.1 The first DCA scheme for solving the problem (2.8) . . . 63

2.5.2 DCA2 - Relation with reweighted-ℓ_1 procedure . . . 65

2.5.3 DCA3 - Relation with reweighted-ℓ_2 procedure . . . 67

2.5.4 Discussion on the three DCA based algorithms 2.1, 2.2 and 2.3 . . . 70

2.5.5 DCA4: DCA applied on (2.8) with the piecewise linear (PiL) approximation . . . 71

2.5.6 Updating θ procedure . . . 73

2.6 Application to Feature selection in SVM . . . 73

2.6.1 Computational experiments . . . 77

2.7 Conclusion . . . 83

3 Algorithms for Nonegative Matrix Factorization[1] 85

3.1 Introduction . . . 85

3.2 The first approach: alternative DCA based algorithm for solving the NMF problem . . . 89

3.2.1 DCA for solving the NNLS problem . . . 89

3.2.2 Alternating DCA for computing NMF and convergence analysis . . . 95

3.3 The second approach: DCA applied on the whole NMF problem . . . 98

3.3.1 The first DCA based algorithm for the whole NMF problem . . . 98

3.3.2 The second DCA based algorithm for the whole NMF problem . . . 102

3.4 Extension of the NMF problem . . . 105

3.4.1 Constrained nonnegative matrix factorization . . . 105

3.4.2 Multilayer nonnegative matrix factorization . . . 109

3.4.3 Convex nonnegative matrix factorization 111
3.4.4 Symmetric nonnegative matrix factorization 112
3.5 Numerical experiments 113
3.5.1 Synthetic datasets 114
3.5.2 Real datasets 115
3.6 Sparse NMF 120
3.6.1 Related works. 121
3.6.2 Sparse NMF formulations 123
3.6.3 Alternative DCA based algorithms for solving the sparse NMF problems 124
3.6.4 Experiment 125
3.7 Conclusion 127

4 Dictionary Learning and Application in Image Denoising[1] 129

4.1 Introduction 129
4.2 Algorithm for Dictionary learning problem 133
4.2.1 General schema solution 133
4.2.2 Sparse coding phase: update W 134
4.2.3 Dictionary updating phase: update D 138
4.3 Application to image denoising 139
4.3.1 Image denoising protocol 139
4.3.2 Numerical experiment 141
4.4 Conclusion 142

II Learning with uncertainty 149

5 Feature Selection for linear SVMs under Uncertain Data[1] 151

5.1 Introduction 151
5.2 Feature Selection for SVMs under Uncertain Data 153
5.2.1 Feature Selection for Linear Two-class SVM Models 153
5.2.2 Data uncertainty model and robust counterpart 154
5.2.3 Ellipsoidal Uncertainty Model 154
5.2.4 Box uncertainty model 157
5.3 Solution methods based on DC programming and DCA 160
5.3.1 DCA for solving problem (5.9) 161
5.3.2 DCA for solving problem (5.10) 162
5.3.3 Robust Feature Selection using ℓ_1-regularizer 163
5.4 Numerical Experiments 164
5.4.1 Error Measures 164
5.4.2 Datasets 165
5.4.3 Experimental setups 166
5.4.4 Experiment on synthetic data 167
5.4.5 Experiments on real datasets 168
5.5 Conclusion 170

6 Robust optimization for clustering uncertain data[1] 179

6.1 Introduction 179
6.2 Robust formulation for clustering 183
6.2.1 Box uncertainty model 184
6.2.2 Spherical uncertainty model 185
6.2.3 Interpretation of the robust models 186
6.3 DCA for solving the robust clustering problems with box and spherical uncertainty models 188

6.3.1 DC formulation of the robust clustering problem 188
6.3.2 Calculation of $\partial H(V)$ 189
6.3.3 Solving the subproblem (6.16) 191
6.3.4 Case of the box uncertainty model 191
6.3.5 Case of the spherical uncertainty model 193
6.3.6 Description of DCA to solve the robust clustering problems with box and spherical uncertainty models 197
6.4 Numerical Experiments 198
6.4.1 Experiment on synthetic datasets 199
6.4.2 Experiment on real datasets 200
6.5 Conclusion 201

7 Conclusion **205**

Conclusion **205**

A Appendix **209**

A.1 Projection onto a nonnegative ball 209
A.2 Orthogonal matrix 210

List of Figures

2.1 Graphs of approximation functions. Except ℓ_p-norm$(0 < p < 1)$ and PiL, the others have the same derivative at 0. Here $\theta_{log} = 10$ for Log, $a = 4$ for SCAD, $p = -2$ for ℓ_p-norm$(p < 0)$. For ℓ_p-norm$(0 < p < 1)$, $\epsilon = 10^{-9}$ and $p = 0.2$. For PiL, $a = 5$ and $\theta_{PiL} = a\theta_{exp}$. 57

2.2 Graphs of functions: $r = 1 - e^{-2|x|}$, ν_1, ν_2 and ν_3 with $x^k = 0.5$. 71

3.1 Dense datasets. In these figures, the horizontal axis presents the elapsed time in second, and the vertical axis presents the relative error. 118

3.2 Sparse datasets. In these figures, the horizontal axis presents the elapsed time in second, and the vertical axis presents the relative error. 119

3.3 Features learned from the CBCL face image database. using our DCA based method and NMFSC method. 127

4.1 Lena 142

4.2 Barbara 143

4.3 Boat 144

4.4 House 145

4.5 Peppers 146

5.1 Crosses and stars represent patterns belonging to the two classes. The ellipsoid (resp. box) around the pattern denotes the uncertainty ellipsoid (resp. box). The dash line represents the nominal classifier, the solid line represents the robust classifier using ellipsoidal uncertainty and the dotted represents the robust classifier using box uncertainty. Wee see that the robust classifier using box uncertainty tends to be sparse than the robust classifier using ellipsoidal uncertainty. 159

5.2 Parkinsons dataset 170

5.3 Ionosphere dataset 171

5.4 WDBC dataset 172

5.5 Leukemia dataset 173

5.6 Lung Cancer dataset 174

List of Tables

2.1 ℓ_0-approximation functions r and the first DC decomposition φ. The second DC decomposition is $\psi = \varphi - r$. 56

2.2 Choice of η and expression of $\overline{z}_i^k \in \lambda\partial\psi(x_i^k)$ in Algorithm 2.1 and related works. 65

2.3 Expression of $\overline{z}_i^k$ in Algorithm 2.2 and relation with reweighted-ℓ_1 algorithms. 67

2.4 Expression of $\overline{z}_i^k$'s in Algorithm 2.3 and relation with reweighted-ℓ_2 algorithms. 69

2.5 Datasets 78

2.6 Comparison of different DCA schemes for Capped-ℓ_1 approximation 80

2.7 Comparison of different approximations 82

3.1 Numerical result on synthetic datasets with $m = 500$ and $n = 1000$. The suffix numbers in names of datasets indicate the sparsity of factors composing the datasets. The number in parenthesis indicates the number of trials that fail to achieve the target relative error within the limit time. 115

3.2 Numerical result on synthetic datasets with $m = 1000$ and $n = 2000$. The suffix numbers in names of datasets indicate the sparsity of factors composing the datasets. The number in parenthesis indicates the number of trials that fail to achieve the target relative error within the limit time. 115

3.3 Datasets 116

3.4 The comparisons of NMF algorithms. All algorithms start from the same initial point and stop when the stopping criteria (3.52) is satisfied with the tolerance $\tau = 10^{-6}$ or the running time exceeds 3600 seconds. Bold font indicates the best result in each row. The number in parenthesis is the relative difference computed by $100(F - F^*)/F^*$, where F is the corresponding "error" and F^* is the smallest value among F's in the upper row. 120

3.5 Numerical results on CBCL database. The better results are boldfaced . . . 126

4.1 PSNR comparison with KSVD and ℓ_1 penalty 147

5.1 Real datasets used in experiments. The numbers in bracket present class distribution +1/-1 166

5.2 Results on synthetic dataset of ℓ_1, ℓ_0(DCA1), ℓ_0(DCA2) approaches in term of the percentage of testing errors and number of selected features of nominal (No.) and of robust classifiers in ellipsoidal model (Ellip.) and box model (Box). The numbers in parentheses are number of correct model over 50 trials. Bold font indicates the best result in each approach. 167

5.3 Comparative results of ℓ_1, ℓ_0(DCA1), ℓ_0(DCA2) approaches in term of the percentage of expected errors (upper row) and percentage of selected features (lower row) of nominal (No.) and of robust classifiers in ellipsoidal model (Ellip.) and box model (Box). Bold font indicates the best result in each row. Uniform distribution of uncertainty is assumed. 175

5.4 Comparative results of ℓ_1, ℓ_0(DCA1), ℓ_0(DCA2) approaches in term of the percentage of expected errors of nominal (No.) and of robust classifiers in ellipsoidal model (Ellip.) and box model (Box). Bold font indicates the best result in each row. Gaussian distribution of uncertainty is assumed. 176

5.5 The running time in second. For Robust solution, the running time is the average on all considered values of parameters of noise. 176

6.1 Comparative results on the first synthetic dataset 199

6.2 Comparative results on the second synthetic datasets. The upper row presents results on the true dataset, the lower row presents results on the disturbed dataset. 200

6.3 Real datasets used in experiments 201

6.4 Comparative results on real datasets 202

Notation

Throughout the thesis, we use uppercase letters to denote matrices, and lowercase letters for vectors or scalars. Vectors are also regarded as matrices with one column. The table below summarizes some of the notation used in the thesis.

$\mathbb{R}$	set of real numbers
$\mathbb{R}_+$	set of nonnegative real numbers
$\mathbb{R}_{++}$	set of positive real numbers
$\mathbb{R}^n$	set of real column vectors of size n
$\mathbb{R}^{m\times n}$	set of real matrices of size m - by - n
$\mathbb{R}^n_+$	set of nonnegative real column vectors of size n
$\mathbb{R}^n_{++}$	set of positive real column vectors of size n
$\mathbb{R}^{m\times n}_+$	set of nonnegative real matrices of size m - by - n
$\mathbb{R}^{m\times n}_{++}$	set of positive real matrices of size m - by - n
$\|\cdot\|_p$	ℓ_p-norm $(0 < p < \infty)$, $\|x\|_p = (\sum_{i=1}^n \|x_i\|^p)^{1/p}$, $x \in \mathbb{R}^n$
$\|\cdot\|$	vector ℓ_2-norm/Euclidean norm, $\|x\| = (\sum_{i=1}^n \|x_i\|^2)^{1/2}$, $x \in \mathbb{R}^n$
	matrix ℓ_2-norm/spectral norm, $\|X\| = \max_{u\in\mathbb{R}^n, \|u\|=1} \|Xu\|$, $X \in \mathbb{R}^{m\times n}$
$\|\cdot\|_0$	ℓ_0-'norm', $\|x\|_0 = \|\{i : x_i \neq 0\}\|$, $\|X\|_0 = \|\{(i,j) : X_{ij} \neq 0\}\|$
$\|\cdot\|_F$	Frobenius norm, $\|X\|_F = (\sum_{i=1}^m \sum_{j=1}^n X_{ij}^2)^{1/2}$, $X \in \mathbb{R}^{m\times n}$
$\langle\ ,\ \rangle$	scalar product, $\langle X, Y\rangle = \sum_{i=1}^m \sum_{j=1}^n X_{ij}Y_{ij}$, $X, Y \in \mathbb{R}^{m\times n}$

$X_{i:}$	i^{th} row of X
$X_{:j}$	j^{th} column of X
X_{ij}	element located at the position (i, j) of X
X_{IJ}	submatrix of X with row (resp. column) indices in I (resp. J)
X^T	transpose of a matrix X, $(X^T)_{ij} = X_{ji}$
$\|X\|$	absolute of X, $\|X\|_{ij} = \|X_{ij}\|$ for all (i, j)
$\mathrm{sgn}(X)$	matrix of signs of X, $(\mathrm{sgn}(X))_{ij} = \mathrm{sgn}(X_{ij}) = -1$ if $X_{ij} < 0$, 1 if $X_{ij} > 0$, and 0 otherwise
$\max(X, \delta)$	matrix Z with $Z_{ij} = \max(X_{ij}, \delta)\ \forall(i, j)$, where $X \in \mathbb{R}^{m\times n}$, $\delta \in \mathbb{R}$
$X \geq \delta$	means $X_{ij} \geq \epsilon\ \forall(i, j)$, where $X \in \mathbb{R}^{m\times n}$, $\delta \in \mathbb{R}$
$\mathrm{diag}(X)$	vector of diagonal-elements of X, $(\mathrm{diag}(X))_i = X_{ii}$
$\mathrm{diag}(x)$	diagonal matrix whose the main diagonal is the vector x
$\mathrm{vec}(X)$	vector formed by stacking the columns of $X \in \mathbb{R}^{m\times n}$ into one vector of size mn

$X \circ Y$	Hadamard product between matrices X and Y, $(X \circ Y)_{ij} = X_{ij}Y_{ij}$
$\frac{[X]}{[Y]}$	Hadamard division between matrices X and Y, $\left(\frac{[X]}{[Y]}\right)_{ij} = \frac{X_{ij}}{Y_{ij}}$
$X \otimes Y$	Kronecker product between matrices X and Y
$X \preceq Y$	$Y - X$ is positive semi-definite matrix (all eigenvalues are nonnegative)

$\mathbf{1}_{m\times n}$	matrix of all ones
I_n	identity matrix of size n
$\mathrm{supp}(X)$	support (set of non-zero entries), $\mathrm{supp}(X) = \{(i, j) : X_{ij} \neq 0\}$, $X \in \mathbb{R}^{m\times n}$
$P_\Omega(X)$	projection of $X \in \mathbb{R}^{m\times n}$ onto $\Omega \subset \mathbb{R}^{m\times n}$
$[X]_+$	projection of $X \in \mathbb{R}^{m\times n}$ onto $\mathbb{R}_+^{m\times n}$, $[X]_+ = \max(X, 0)$

$\chi_C(\cdot)$	the indicator function of C, $\chi_C(x) = 0$ if $x \in C$ and $+\infty$ otherwise
$\nabla f(x)$	the gradient of f at x
$\nabla^2 f(x)$	the Hessian of f at x
$\partial f(x)$	the subdifferential of f at x

Introduction générale

Cadre général et nos motivations

Depuis quelques années, la société est confrontée au phénomène du Big Data, à savoir la disponibilité des données tellement massives qu'il faut des algorithmes puissants pour les analyser et explorer. La naissance du big data est lié aux progrès des systèmes de stockage, de fouille et d'analyse de l'information numérisée.

Dans cette thèse, nous nous intéressons à deux challenges en apprentissage et fouille de données dans le contexte du Big Data: apprentissage avec la parcimonie et/ou sur des données incertaines. Problèmes parcimonieux apparaissent dans plusieurs disciplines scientifiques. Par exemple, les techniques d'acquisition comprimée visent à trouver la solution la plus parcimonieuse d'un système linéaire sous-déterminé. En général, lorsqu'on désire expliquer un ensemble de réponses à partir de certaines observations, ou lorsqu'on souhaite compresser un grand volume de données, une solution simple, parcimonieuse, est souvent privilégiée face à d'autres alternatives plus "complexes". Pour la conception des modèles d'apprentissage, la modélisation parcimonieuse est basée sur la norme zéro (la norme zéro d'un vecteur est définie comme le nombre de ses termes non nulles). C'est la plus naturelle façon pour aborder la parcimonie, mais le problème d'optimisation correspondant est NP-difficile. Dans ces travaux, cette difficulté est surmontée par l'approximation de la norme zéro via une fonction DC (Difference of Convex functions), le problème résultant est ainsi un problème d'optimisation DC. Quant à l'incertitude des données, tous les problèmes d'optimisation dans les applications réelles sont liés à des paramètres ou des données incertaines. L'incertitude est considérée dans le sens suivant : on ne dispose pas des valeurs numériques exactes décrivant les données, on connaît en revanche un rectangle ou une boulle les contenant. L'optimisation robuste est la base méthodologique pour construire des modèles d'apprentissage sur ces données. Il s'agit des problèmes d'optimisation de la forme min-max (optimiser le pire des cas) qui peuvent être formulés comme des problèmes d'optimisation DC.

Sur le plan algorithmique, la thèse a proposé une approche unifiée, fondée sur la programmation DC et DCA (DC Algorithm), des outils puissant d'optimisation non convexe qui connaît

un grand succès, au cours de deux dernières décennies, dans la résolution de nombreux problèmes d'application dans diverses domaines de sciences appliquées, en particulier en machine learning et data mining. De nombreuses expérimentations numériques sur différents types de données (biologie, image, ...) réalisées dans cette thèse ont prouvé l'efficacité, la scalabilité, la rapidité des algorithmes proposés et leur supériorité par rapports aux méthodes standard.

La programmation DC et DCA considèrent le problème DC de la forme

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

où g et h sont des fonctions convexes définies sur $\mathbb{R}^n$ et à valeurs dans $\mathbb{R} \cup \{+\infty\}$, semi-continues inférieurement et propres. La fonction f est appelée fonction DC avec les composantes DC g et h, et $g - h$ est une décomposition DC de f. DCA est basé sur la dualité DC et des conditions d'optimalité locale. La construction de DCA implique les composantes DC g et h et non la fonction DC f elle-même. Or chaque fonction DC admet une infinité des décompositions DC qui influencent considérablement sur la qualité (la rapidité, l'efficacité, la globalité de la solution obtenue,...) de DCA. Ainsi, au point de vue algorithmique, la recherche d'une "bonne" décomposition DC et d'un "bon" point initial est très importante dans le développement de DCA pour la résolution d'un programme DC.

L'utilisation de la programmation DC et DCA dans cette thèse est justifiée par de multiple arguments (Pham Dinh and Le Thi (2014)):

- La programmation DC et DCA fournissent un cadre très riche pour les problèmes d'apprentissage et de fouille de données (Machine Learning and Data Mining - MLDM): MLDM constituent *une mine des programmes DC* dont la résolution appropriée devrait recourir à la programmation DC et DCA. En effet la liste indicative (non exhaustive) des références dans Le Thi (Website) témoigne de la vitalité, la puissance et la percée de cette approche dans la communauté de MLDM.
- DCA est une philosophie plutôt qu'un algorithme. Pour chaque problème, nous pouvons concevoir une famille d'algorithmes basés sur DCA. La flexibilité de DCA sur le choix des décomposition DC peut offrir des schémas DCA plus performants que des méthodes standard.
- L'analyse convexe fournit des outils puissants pour prouver la convergence de DCA dans un cadre général. Ainsi tous les algorithmes basés sur DCA bénéficient (au moins) des propriétés de convergence générales du schéma DCA générique qui ont été démontrées.
- DCA est une méthode efficace, rapide et scalable pour la programmation non convexe. A notre connaissance, DCA est l'un des rares algorithmes de la programmation non convexe, non différentiable qui peut résoudre des programmes DC de très grande dimension. La programmation DC et DCA ont été appliqués avec succès pour la modélisation DC et la résolution de nombreux et divers problèmes d'optimisation non convexes dans différents domaines des sciences appliquées, en particulier en MLDM (voir par exemple la liste des références dans Le Thi (Website)).
  Il est important de noter qu'avec les techniques de reformulation en programmation DC et

les décompositions DC appropriées, on peut retrouver la plupart des algorithmes existants en programmation convexe/non convexe comme cas particuliers de DCA.

En particulier, pour la communauté de Data mining-Machine learning, les méthodes très connus comme Expectation–Maximisation (EM) (Dempster et al., 1977), Succesive Linear Approximation (SLA) (Bradley and Mangasarian, 1998), ConCave–Convex Procedure (CCCP) (Yuille and Rangarajan, 2003), Iterative Shrinkage-Thresholding Algorithms (ISTA) (Chambolle et al., 1998) sont des versions spéciaux de DCA.

Nos contributions

L'objectif de la thèse est de développer des nouveaux modèles et méthodes pour cinq classes des problèmes difficiles et importants de l'apprentissage avec la parcimonie et/ou avec l'incertitude sur des données : apprentissage supervisée avec la parcimonie, factorisation des matrices non négatives (NMF) et NMF parcimonieuse, apprentissage dictionnaire parcimonieuse, classification supervisée des données incertaines, clustering des données incertaines.

Dans le premier temps, nous considérons le problème d'optimisation parcimonieuse qui a nombreuses applications en apprentissage comme la sélection de variables pour la classification supervisée, l'acquisition comprimée, la régression linéaire parcimonieuse, la sélection de portefeuilles, ... Nous réalisons une étude approfondie tant sur le plan théorique qu'algorithmique. Nous proposons une approche d'approximation non-convexe unifiée, avec des outils théoriques solides ainsi que des algorithmes efficaces basés sur la programmation DC et DCA, pour aborder la norme zéro et l'optimisation parcimonieuse. En considérant une approximation DC commune de la norme zéro qui inclut toutes les fonctions de pénalité standard, nous étudions la cohérence entre les minimums globaux (les minimums locaux) de problèmes approché et originaux. Nous montrons que, dans plusieurs cas, des minimums globaux (minimums locaux) du problème approché sont aussi ceux du problème original. En utilisant la technique de pénalité exacte dans la programmation DC, nous prouvons des résultats plus forts pour certaines approximations particulières, à savoir, le problème approché, avec les paramètres appropriés, est équivalent au problème original. L'efficacité de plusieurs fonctions de pénalité induisant parcimonie a été soigneusement analysé dans le détail. Nous développons quatre schémas de DCA pour résoudre le problème parcimonieux. Les algorithmes standards existant dans le domaine peuvent être considérés comme un algorithme de L1 perturbé / algorithme L1 repondéré/ algorithme L2 repondéré, et ils sont les versions spéciaux des schémas de DCA proposés. Comme application, nous mettons en pratique nos méthodes pour la sélection de variables pour SVM (Support Vector Machines) et effectuons des expériences numériques comparatives empiriques sur les algorithmes proposés avec diverses fonctions d'approximation.

Le deuxième problème traité dans cette thèse est la factorisation en matrices non-négatives (NMF) qui a nombreuse applications en traitement de l'image, traitement du texte, économie,

biologie, ... Le problème consiste à approcher une matrice non négative par le produit de deux matrices non négatives de rang faible. En général, ce problème est non convexe et il a été démontre que c'est un problème NP-hard. Nous proposons deux approaches pour résoudre le problème NMF qui utilise la distance euclidienne comme la fonction de coût. La première approche exploite la structure spéciale du problème: lorsque l'un des facteurs est fixé, le problème NMF se réduit à un problème des moindres carrés avec des contraintes non négatives qui peut être résolu par la programmation DC et DCA. Nous développons un algorithme basé sur la programmation DC et DCA pour résoudre le problème des moindres carrés non négatifs. Donc, en suivant la méthode alternative, nous obtenons un algorithme pour NMF qui à chaque itération alternativement fixe un facteur et minimise la fonction de coût par rapport à l'autre facteur utilisant l'algorithme précité. Nous démontrons que le schéma de cette approche couvre de nombreux algorithmes existants. La seconde approche applique directement la programmation DC et DCA au problème NMF. Nous proposons deux algorithmes qui calculent simultanément les deux facteurs à chaque itération. Nous prouvons que tous les algorithmes proposés décroissent la valeur du critère à chaque itération et convergent vers un point stationnaire.

Nous montrons que la première approche peut être facilement adaptée pour résoudre d'autres variantes de problème NMF inclut NMF contenant les contraints, multicouche-NMF et convexe-NMF. En outre, la seconde approche peut être appliqué à le problème NMF symétrique. Des expériences numériques montrent que l'algorithme proposé offre une bonne performance en terme de vitesse de calcul, en particulier sur de grands ensembles de données ou les ensembles de données parcimonieuse.

Dans beaucoup des applications, il est utile d'ajouter des contraintes de parcimonie à la NMF pour obtenir le problème NMF parcimonieux. En imposant la zéro-norme, nous cherchons une factorisation d'une matrice non négative tel que les matrices facteurs sont plus parcimonieux que les facteurs du NMF standard. Dans l'algorithme proposée, à chaque itération, nous faisons face aux problèmes des moindres carrés non-négative zéro-norme-régularisés. La méthode d'approximation DC considéré dans le premier problème est adapté pour résoudre ces problèmes.

Toujours dans le cadre de parcimonie, nous considérons le troisième problème - l'apprentissage de dictionnaires. L'objectif est d'apprendre un dictionnaire efficace pour adapter les données spécifiques, donnant lieu à une représentation parcimonieuse en utilisant seulement quelques-uns atomes du dictionnaire. Ce problème a la même formulation que le problème NMF parcimonieux sauf la contrainte de non-négativité. Comme pour les problèmes NMF et NMF parcimonieux, en suivant la méthode alternative, nous proposons un algorithme qui comporte deux phases: la première phase est le problème de codage parcimonieux et la deuxième phase est le problème de la mise à jour du dictionnaire. Dans la première phase, nous devons résoudre les problèmes des moindres carrés avec la norme zéro comme régularisation qui peut être traité d'une façon similaire au problème NMF parcimonieux. Une application de réduction de bruit d'image est testé pour vérifier l'efficacité de l'algorithme proposé.

Le quatrième problème est la sélection de variables dans SVM (Support Vector Machines) avec les données incertaines. C'est à dire, nous abordons le problème d'apprentissage supervisée avec la parcimonie et l'incertitude sur des données en même temps. En utilisant les principes de l'optimisation robuste, nous proposons les schémas robustes pour traiter les données avec le modèle ellipsoïdal et le modèle rectangle d'incertitude. Pour la sélection de variable (i.e. l'apprentissage avec la parcimonie) nous utilisons l'approximation DC étudiée dans le deuxième chapitre. Les résultats numériques montrent que les approches d'optimisation robustes proposées sont supérieures aux approches traditionnelles qui ne prennent pas compte la perturbation des données.

Le dernier problème est clustering des données incertaines. Toujours basant sur l'optimisation robuste, nous avons proposé les schémas robustes pour traiter les données avec le modèle sphérique et le modèle rectangle d'incertitude. Modifiant le modèle utilisant la distance euclidienne au carré, nous avons obtenu les formulations robustes qui sont non-convexes mais peut être reformulées comme une programme DC. Les algorithmes efficaces basés sur DCA sont proposés pour résoudre les problèmes résultants. Les expériences numériques montrent que nos algorithmes proposés offrent une bonne performance.

Organisation de la thèse

La thèse est composée de six chapitres.

Le premier chapitre décrit de manière succincte la programmation DC et DCA et de la technique d'optimisation robuste. Il présente les outils théoriques et algorithmiques servant des références aux autres chapitres. Cinq chapitres suivants sont divisés en deux parties: La première partie (chapitres 2, 3 et 4) concerne l'optimisation parcimonieuse tandis que la deuxième partie (chapitres 5 et 6) traite l'incertitude des données. Le seconde chapitre présente l'approche d'approximation DC pour la minimisation de la norme zéro. Le troisième chapitre est consacré au problème de la factorisation en matrices non-négatives (NMF) et le problème de NMF parcimonieuse. Dans le quatrième chapitre, nous étudions le problème d'apprentissage de dictionnaire. Les chapitres 5 et 6 sont destinés à traiter l'incertitude des données pour les deux problèmes: la sélection de variables dans SVM (Support Vector Machines) et le clustering. Le chapitre 7 fournit les conclusions et les perspectives de nos travaux.

Chapter 1

Preliminary

This chapter summarizes some basis concepts and results that will be the groundwork of the thesis.

1.1 DC programming and DCA

DC programming and DCA, which constitute the backbone of nonconvex programming and global optimization, were introduced by Pham Dinh Tao in their preliminary form in 1985 (Pham Dinh, 1986). Important developments and improvements on both theoretical and computational aspects have been completed since 1993 throughout the joint works of Le Thi Hoai An and Pham Dinh Tao. In this section, we present some basic properties of convex analysis and DC optimization and DC Algorithm that computational methods of this thesis are based on. The materials of this section are extracted from (Le Thi, 1997; Pham Dinh and Le Thi, 1997; Le Thi and Pham Dinh, 2005). In application, in section 1.1.4, we give a new result about approximate DCA.

Throughout this section, X denotes the Euclidean space $\mathbb{R}^n$ and $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$ is the set of extended real numbers.

1.1.1 Fundamental convex analysis

A subset C of X is said to be *convex* if $(1-\lambda)x + \lambda y \in C$ whenever $x, y \in C$ and $\lambda \in [0, 1]$.

Let f be a function whose values are in $\overline{\mathbb{R}}$ and whose domain is a subset S of X. The set

$$\{(x, t) : x \in S, t \in R, f(x) \leq t\}$$

is called the *epigraph* of f and is denoted by epi$f$.

We define f to be a *convex function* on S if epi$f$ is convex set in $X \times \mathbb{R}$. This is equivalent to that S is convex and

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y), \quad \forall x, y \in S, \forall \lambda \in [0,1]$$

The function f is *strictly convex* if the inequality above holds strictly whenever x and y are distinct in S and $0 < \lambda < 1$.

The *effective domain* of a convex function f on S, denoted by dom$f$, is the projection on X of the epigraph of f

$$\text{dom}f = \{x : \exists t \in \mathbb{R}, (x,t) \in \text{epi}f\} = \{x \mid f(x) < +\infty\}$$

and it is convex.

The convex function f is called *proper* if dom$f \neq \emptyset$ and $f(x) > -\infty$ for all $x \in S$.

The function f is said to be *lower semi-continuous* at a point x of S if

$$f(x) \leq \liminf_{y \to x} f(y)$$

Denote by $\Gamma_0(X)$ the set of all proper lower semi-continuous convex function on X.

Let $\rho \geq 0$ and C be a convex subset of X. One says that a function $\theta : C \mapsto \mathbb{R} \cup \{+\infty\}$ is ρ–convex if

$$\theta[\lambda x + (1-\lambda)y] \leq \lambda\theta(x) + (1-\lambda)\theta(y) - \frac{\lambda(1-\lambda)}{2}\rho\|x-y\|^2$$

for all $x, y \in C$ and $\lambda \in (0,1)$. It amounts to say that $\theta - (\rho/2)\|.\|^2$ is convex on C. The modulus of strong convexity of θ on C, denoted by $\rho(\theta, C)$ or $\rho(\theta)$ if $C = X$, is given by

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

One say that θ is *strongly convex* on C if $\rho(\theta, c) > 0$.

A vector y is said to be a *subgradient* of a convex function f at a point x^0 if

$$f(x) \geq f(x^0) + \langle x - x^0, y\rangle, \qquad \forall x \in X$$

The set of all subgradients of f at x^0 is called the *subdifferential* of f at x^0 and is denoted by $\partial f(x^0)$. If $\partial f(x)$ is not empty, f is said to be *subdifferentiable* at x.

For $\varepsilon > 0$, a vector y is said to be a ε*–subgradient* of a convex function f at a point x^0 if

$$f(x) \geq (f(x^0) - \varepsilon) + \langle x - x^0, y\rangle, \qquad \forall x \in X$$

The set of all ε–subgradients of f at x^0 is called the *ε–subdifferential* of f at x^0 and is denoted by $\partial_\varepsilon f(x^0)$.

We also have notations

$$\text{dom } \partial f = \{x \in X : \partial f(x) \neq \emptyset\} \text{ and } \text{range } \partial f(x) = \cup\{\partial f(x) : x \in \text{dom } \partial f\}$$

Proposition 1.1 *Let f be a proper convex function. Then*

1. *$\partial_\varepsilon f(x)$ is a closed convex set, for any $x \in X$ and $\varepsilon \geq 0$.*
2. $\text{ri}(\text{dom} f) \subset \text{dom } \partial f \subset \text{dom} f$
   where $\text{ri}(\text{dom} f)$ *stands for the* relative interior *of* $\text{dom} f$.
3. *If f has a unique subgradient at x, then f is differentiable at x, and $\partial f(x) = \{\nabla f(x)\}$.*
4. *$x_0 \in \text{argmin}\{f(x) : x \in X\}$ if and only if $0 \in \partial f(x_0)$.*

Conjugates of convex functions

The *conjugate* of a function $f : X \mapsto \overline{\mathbb{R}}$ is the function $f^* : X \mapsto \overline{\mathbb{R}}$ defined by

$$f^*(y) = \sup_{x \in X}\{\langle x, y\rangle - f(x)\}$$

Proposition 1.2 *Let $f \in \Gamma_0(X)$. Then we have*

1. *$f^* \in \Gamma_0(X)$ and $f^{**} = f$.*
2. *$f(x) + f^*(y) \geq \langle x, y\rangle$, for any $x, y \in X$.*
   Equality holds if and only if $y \in \partial f(x) \Leftrightarrow x \in \partial f^*(y)$.
3. *$y \in \partial_\varepsilon f(x) \Longleftrightarrow x \in \partial_\varepsilon f^*(y) \Longleftrightarrow f(x) + f^*(y) \leq \langle x, y\rangle + \varepsilon$, for all $\varepsilon > 0$.*

Polyhedral Functions

A *polyhedral* set is a closed convex set having form

$$C = \{x \in X : \langle x, b_i\rangle \leq \beta_i,\ \forall i = 1, \ldots, m\},$$

where $b_i \in X$ and $\beta_i \in \mathbb{R}$ for all $i = 1, \ldots, m$.

A function $f \in \Gamma_0(X)$ is said to be *polyhedral* if

$$f(x) = \max\{\langle a_i, x\rangle - \alpha_i : i = 1, \ldots, k\} + \chi_C(x), \quad \forall x \in X \tag{1.1}$$

where $a_i \in X, \alpha_i \in \mathbb{R}$ for $i = 1, \ldots, k$ and C is a nonempty polyhedral set. Notation χ_C stands for *indicator function* of C and is defined by $\chi_C(x) = 0$ if $x \in C$, and $+\infty$ otherwise. It is clear that $\text{dom } f = C$.

Proposition 1.3 *Let f be a polyhedral convex function, and $x \in \mathrm{dom} f$. Then we have*

1. *f is subdifferentiable at x, and $\partial f(x)$ is a polyhedral convex set. In particular, if f is defined by (1.1) with $C = X$ then*

$$\partial f(x) = \mathrm{co}\{a_i : i \in I(x)\}$$

where $I(x) = \{i \in \{1, \ldots, k\} : \langle a_i, x\rangle - \alpha_i = f(x)\}$.

2. *The conjugate f^* is a polyhedral convex function. Moreover, if $C = X$ then*

$$\mathrm{dom} f^* = \mathrm{co}\{a_i : i = 1, \ldots, k\}$$
$$f^*(y) = \inf\left\{\sum_{i=1}^{k} \lambda_i \alpha_i \Big| \sum_{i=1}^{k} \lambda_i a_i = y, \sum_{i=1}^{k} \lambda_i = 1, \lambda_i \geq 0, \forall i = 1, \ldots, k\right\}$$

In particular,

$$f^*(a_i) = \alpha_i, \qquad \forall i = 1, \ldots, k$$

Difference of convex (DC) functions

A function f is called DC function on X if it has the form

$$f(x) = g(x) - h(x), \quad x \in X$$

where g and h belong to $\Gamma_0(X)$. One says that $g - h$ is a *DC decomposition* of f and g, h are its *DC components*. If g and h are in addition finite on all of X then one says that $f = g - h$ is finite DC function on X. The set of DC functions (resp. finite DC functions) on X is denoted by $\mathcal{DC}(X)$ (resp. $\mathcal{DC}_f(X)$).

Remark 1.1 *Give a DC function f having a DC decomposition $f = g - h$. Then for every $\theta \in \Gamma_0(X)$ finite on the whole X, $f = (g + \theta) - (h + \theta)$ is another DC decomposition of f. Thus, a DC function f has finitely many DC decompositions.*

1.1.2 DC optimization

General DC program

In the sequel, we use the convention $+\infty - (+\infty) = +\infty$.

For $g, h \in \Gamma_0(X)$, a general *DC program* is that of the form

$$(P) \qquad \alpha = \inf\{f(x) = g(x) - h(x) : x \in X\}$$

and its dual counterpart

$$(D) \qquad \alpha^* = \inf\{h^*(y) - g^*(y) : y \in X\}$$

There is a perfect symmetry between primal and dual programs (P) and (D): the dual program to (D) is exactly (P), moreover, $\alpha = \alpha^*$.

Remark 1.2 *Let C be a nonempty closed convex set. Then, the constrained problem*

$$\inf\{f(x) = g(x) - h(x) : x \in C\}$$

can be transformed into an unconstrained DC program by using the indicator function χ_C, i.e.,

$$\inf\{f(x) = \phi(x) - h(x) : x \in X\}$$

where $\phi := g + \chi_C$ is in $\Gamma_0(X)$.

We will always keep the following assumption that is deduced from the finiteness of α

$$\operatorname{dom} g \subset \operatorname{dom} h \quad \text{and} \quad \operatorname{dom} h^* \subset \operatorname{dom} g^*. \tag{1.2}$$

Polyhedral DC program

In problem (P), if one of the DC components g and h is polyhedral function, we call (P) *polyhedral DC program*. This is an important class of DC optimization. It is often encountered in practice and has worthy properties.

Consider problem (P) where h is a polyhedral convex function given by

$$h(x) = \max\{\langle a_i, x\rangle - \alpha_i : i = 1, \ldots, k\}$$

By Proposition 1.3, the dual problem (D) has the form

$$\begin{aligned} \alpha^* &= \inf\{h^*(y) - g^*(y) : y \in X\} \\ &= \inf\{h^*(y) - g^*(y) : y \in \operatorname{co}\{a_i : i = 1, \ldots, k\}\} \\ &= \inf\{\alpha_i - g^*(a_i) : i = 1, \ldots, k\} \end{aligned}$$

Note that, if g is polyhedral convex and h is not, then by considering the dual problem (D) we have the similar formulation as above since g^* is polyhedral.

Optimality conditions for DC optimization

A point x^* is said to be a *local minimizer* of $g - h$ if $x^* \in \operatorname{dom} g \cap \operatorname{dom} h$ (so, $(g - h)(x^*)$ is finite) and there is a neighborhood U of x^* such that

$$g(x) - h(x) \geq g(x^*) - h(x^*), \qquad \forall x \in U. \tag{1.3}$$

A point x^* is said to be a *critical point* of $g - h$ if it verifies the generalized Kuhn–Tucker condition

$$\partial g(x^*) \cap \partial h(x^*) \neq \emptyset \tag{1.4}$$

Let $\mathcal{P}$ and $\mathcal{D}$ denote the solution sets of problems (P) and (D) respectively, and let

$$\mathcal{P}_\ell = \{x^* \in X : \partial h(x^*) \subset \partial g(x^*)\}, \quad \mathcal{D}_\ell = \{y^* \in X : \partial g^*(y^*) \subset \partial h^*(y^*)\}$$

Below, we present some fundamental results on DC programming (Pham Dinh and Le Thi, 1997).

Theorem 1.1 **i)** *Global optimality condition: $x \in \mathcal{P}$ if and only if $\partial_\varepsilon h(x) \subset \partial_\varepsilon g(x)$, $\forall \varepsilon > 0$.*

ii) *Transportation of global minimizers: $\cup\{\partial h(x) : x \in \mathcal{P}\} \subset \mathcal{D} \subset \operatorname{dom} h^*$.*
The first inclusion becomes equality if g^* is subdifferentiable in $\mathcal{D}$. In this case $\mathcal{D} \subset (\operatorname{dom} \partial g^* \cap \operatorname{dom} \partial h^*)$.

iii) *Necessary local optimality: if x^* is a local minimizer of $g - h$, then $x^* \in \mathcal{P}_\ell$.*

iv) *Sufficient local optimality: Let x^* is a critical point of $g-h$ and $y^* \in \partial g(x^*) \cap \partial h(x^*)$. Let U be a neighborhood of x^* such that $(U \cap \operatorname{dom} g) \subset \operatorname{dom} \partial h$. If for any $x \in U \cap \operatorname{dom} g$, there is $y \in \partial h(x)$ such that $h^*(y) - g^*(y) \geq h^*(y^*) - g^*(y^*)$, then x^* is a local minimizer of $g - h$. More precisely,*

$$g(x) - h(x) \geq g(x^*) - h(x^*), \qquad \forall x \in U \cap \operatorname{dom} g$$

iv) *Transportation of local minimizers: Let $x^* \in \operatorname{dom} \partial h$ be a local minimizer of $g - h$. Let $y^* \in \partial h(x^*)$ and a neighborhood U of x^* such that $g(x) - h(x) \geq g(x^*) - h(x^*)$, $\forall x \in U \cap \operatorname{dom} g$. If*

$$y^* \in \operatorname{int}(\operatorname{dom} g^*) \quad \text{and} \quad \partial g^*(y^*) \subset U$$

then y^* is a local minimizer of $h^* - g^*$.

Remark 1.3 *a) By the symmetry of the DC duality, these results have their corresponding dual part. For example, if y is a local minimizer of $h^* - g^*$, then $y \in \mathcal{D}_\ell$.*

b) The properties ii), iv) and their dual parts indicate that there is no gap between the problems (P) and (D). They show that globally/locally solving the primal problem (P) implies globally/locally solving the dual problem (D) and vice–versa. Thus, it is useful if one of them is easier to solve than the other.

c) The necessary local optimality condition $\partial h^*(x^*) \subset \partial g^*(x^*)$ is also sufficient for many important classes programs, for example (Le Thi and Pham Dinh, 2005), if h is polyhedral convex, or when f is locally convex at x^*, i.e. there exists a convex neighborhood U of x^* such that f is finite and convex on U. We know that a polyhedral convex function is almost everywhere differentiable, that is it is differentiable everywhere except on a set of measure zero. Thus, if h is a polyhedral convex function, then a critical point of $g - h$ is almost always a local solution to (P).

d) If f is actually convex on X, we call (P) a "false" DC program. *In addition, if* $\operatorname{ri}(\operatorname{dom} g) \cap \operatorname{ri}(\operatorname{dom} h) \neq \emptyset$ *and $x^0 \in \operatorname{dom} g$ such that g is continuous at x^0, then $0 \in \partial f(x^0) \Leftrightarrow \partial h(x^0) \subset \partial g(x^0)$ (Le Thi and Pham Dinh, 2005). Thus, in this case, the local optimality is also sufficient for the global optimality. Consequently, if in addition h is differentiable, a critical point is also a global solution.*

1.1.3 DC Algorithm (DCA)

Simplified form of DCA

The DCA consists in the construction of the two sequences $\{x^k\}$ and $\{y^k\}$ (candidates for being primal and dual solutions, respectively) which are easy to calculate and satisfy the following properties:

i) The sequences $(g-h)(x^k)$ and $(h^*-g^*)(y^k)$ are decreasing.

ii) Their corresponding limits x^∞ and y^∞ satisfy the local optimality condition $(x^\infty, y^\infty) \in \mathcal{P}_\ell \times \mathcal{D}_\ell$ or are critical points of $g-h$ and h^*-g^*, respectively.

From a given point $x^0 \in \operatorname{dom} g$, the *simplified* DCA (which will be called DCA for simplicity) generates these sequences by the scheme

$$y^k \in \partial h(x^k) = \arg\min\{h^*(y) - \langle y, x^k\rangle : y \in X\} \tag{1.5a}$$

$$x^{k+1} \in \partial g^*(y^k) = \arg\min\{g(x) - \langle x, y^k\rangle : x \in X\}. \tag{1.5b}$$

The interpretation of the above scheme is simple. At iteration k of DCA, we replace the second component h in the primal DC program by its affine minorant

$$h_k(x) = h(x^k) + \langle x - x^k, y^k\rangle, \tag{1.6}$$

where $y^k \in \partial h(x^k)$. Then the original DC program reduces to the *convex program*

$$(P_k) \qquad \alpha_k = \inf\{f_k(x) := g(x) - h_k(x) : x \in X\}$$

that is equivalent to (1.5a). It is easy to see that f_k is a majorant of f at x^k. Similarly, by replacing $g*$ with its affine minorant

$$g_k^*(y) = g^*(y^{k-1}) + \langle y - y^{k-1}, x^k\rangle, \tag{1.7}$$

where $x^k \in \partial g^*(y^{k-1})$, we lead to the convex problem

$$(D_k) \qquad \inf\{h^*(y) - g_k^*(y) : y \in X\}$$

whose solution set is $\partial h(x^k)$.

Remark 1.4 *a) Finding y^k, x^{k+1} by the scheme 1.5 is equivalent to solving the problems (D_k) and (P_k). Thus, DCA works by reducing a DC program to a sequence of convex program that can be solved efficiently.*

b) In practice, the calculation of the subgradient of the function h at a point x is usually easy if we know its explicit expression. But, the explicit expression of the conjugate of a given function g is unknown, so calculating x^{k+1} is done by solving the convex problem (D_k).

c) When h is polyhedral, the calculation of gradients $y^k = \partial h(x^k)$ is explicit by Proposition 1.3. With a fixed choice of subgradients of h, the set of y^k's will be finite. This leads to finite convergence of DCA.

d) DCA is constructed from DC convex components g and h and their conjugates but not from the DC function f itself, while a DC function has finitely many DC decompositions. Thus, it is useful to find a suitable DC decomposition since it may have crucial impacts on the efficiency of DCA.

Deeper insight into DCA

Denote by h^k the polyhedral convex minorants of h defined by

$$h^k(x) := \max\{h_i(x) : i = 0, 1, \ldots, k\}, \qquad \forall x \in \mathbb{R}^n, \tag{1.8}$$

where h_i's are the affine minorants defined by (1.6).

Consider the polyhedral DC program

$$\inf\{f^k(x) := g(x) - h^k(x) : x \in \mathbb{R}^n\}. \tag{1.9}$$

It is clear that for all $i = 0, 1, \ldots, k$ and $x \in \mathbb{R}^n$,

$$\begin{aligned} &h_i(x) \le h^k(x) \le h(x) \\ &h^k(x^i) = h_i(x^i) = h(x^i). \end{aligned}$$

This implies that $f(x) \le f^k(x) \le f_k(x)$ for any $x \in \mathbb{R}^n$, so (1.9) is a tighter relaxation of the original DC program (P) than (P_k). Interestingly enough, we will show that optimal solution of (P_k), x^{k+1}, is indeed an optimal solution of (1.9).

It is easy to see that $\{x^0, x^1, \ldots, x^k, x^{k+1}\}$ is the sequence generated by DCA applied to (1.9). Thus,

$$f^k(x^0) \ge f^k(x^1) \ge \cdots \ge f^k(x^k) \ge f^k(x^{k+1}).$$

Moreover, since $x^{i+1} \in \arg\min\{f_i(x) = g(x) - h_i(x) : x \in \mathbb{R}^n\}$ for all $i = 0, 1, \ldots, k$, and

$$f^k(x) = g(x) - \max_{i=0,1,\ldots,k} h_i(x) = \min_{i=0,1,\ldots,k} (g(x) - h_i(x)), \quad \forall x \in \mathbb{R}^n,$$

we have

$$\begin{aligned} \inf\{f^k(x) : x \in \mathbb{R}^n\} &= \min_{i=0,1,\ldots,k} \inf\{f_i(x) = g(x) - h_i(x) : x \in \mathbb{R}^n\} \\ &= \min_{i=0,1,\ldots,k} f_i(x^{i+1}) \\ &\ge \min_{i=0,1,\ldots,k} f^k(x^{i+1}) = f^k(x^{k+1}). \end{aligned}$$

Therefore, x^{k+1} is an optimal solution of (1.9).

Furthermore, $\{f^k\}$ is a decreasing sequence of majorants of f, i.e.

$$f^0(x) \geq f^1(x) \geq f^2(x) \geq \cdots \geq f(x), \quad \forall x \in \mathbb{R}^n$$

and

$$f^k(x^i) = f(x^i), \quad \forall k = 0, 1, 2, \ldots; i = 0, 1, \ldots, k.$$

This means that DCA successively generates improved solution and provides better convexifications for the original DC program (P).

Well definiteness of DCA

DCA is well defined if one can construct two sequences $\{x^k\}$ and $\{y^k\}$ as above from an arbitrary initial point x^0. The following Lemma is the necessary and sufficient condition for this property

Lemma 1.1 (Pham Dinh and Le Thi (1997)) *The sequences $\{x^k\}$ and $\{y^k\}$ in DCA are well defined if and only if*

$$\operatorname{dom} \partial g \subset \operatorname{dom} \partial h \quad \text{and} \quad \operatorname{dom} \partial h^* \subset \operatorname{dom} \partial g^*$$

Since for $\varphi \in \Gamma_0(X)$ we have $\operatorname{ri}(\operatorname{dom} \varphi) \subset \operatorname{dom} \partial\varphi \subset \operatorname{dom} \varphi$ (Proposition 1.1). Moreover, we also keep the assumptions $\operatorname{dom} g \subset \operatorname{dom} h$, $\operatorname{dom} h^* \subset \operatorname{dom} g^*$. So, we can say that DCA in general is well defined.

Complete form of DCA

In the simplified DCA, at iteration k, we know y^{k-1} and x^k with $x^k \in \partial g^*(y^{k-1})$. And we determine y^k and x^{k+1} by the scheme 1.5. Then, the solution space of y^k (resp. x^{k+1}) is $\partial h(x^k)$ (resp. $\partial g^*(y^k)$). The purpose of finding the best y^k (resp. x^{k+1}) leads to problems

$$\begin{array}{lcl} (S(x^*)) & & \inf\{h^*(y) - g^*(y) : y \in \partial h(^*)\} \\ & \Leftrightarrow & \inf\{\langle x^k, y\rangle - g^*(y) : y \in \partial h(^*)\} \end{array}$$

and

$$\begin{array}{lcl} (T(y^*)) & & \inf\{g(x) - h(x) : x \in \partial g^*(y^*)\} \\ & \Leftrightarrow & \inf\{\langle x, y^k\rangle - h(x) : x \in \partial g^*(y^*)\} \end{array}$$

where $x^*, y^* \in X$ are given.

Then, we find y^k and x^{k+1} as solutions of problems $S(x^k)$ and $T(y^k)$, respectively. Let $\mathcal{S}(x^*), \mathcal{T}(y^*)$ denote the solution sets of problems $(S(x^*)), (T(y^*))$, the complete DCA consists of constructing two sequences $\{x^k\}$ and $\{y^k\}$ defined by scheme

$$y^k \in \mathcal{S}(x^k); \qquad x^{k+1} \in \mathcal{T}(y^k).$$

with a starting point $x^0 \in \operatorname{dom} g$ given in advance.

Although the problems $(S(x^*)), (T(y^*))$ are simpler than $(D), (P)$ (we work on $\partial h(x^k)$ and $\partial g^*(y^k)$ with convex maximization problems), they remain nonconvex programs and thus are still hard to solve. In practice, we generally use the simplified DCA to solve DC programs.

Convergence properties of DCA

Complete convergence of DCA is given in the following results (Pham Dinh and Le Thi, 1997).

Let ρ_i and ρ_i^*, $(i = 1, 2)$ be real nonnegative numbes such that $0 \leq \rho_i < \rho(f_i)$ (resp. $0 \leq \rho_i^* < \rho(f_i^*)$) where $\rho_i = 0$ (resp. $\rho_i^* = 0$) if $\rho(f_i) = 0$ (resp. $\rho(f_i^*) = 0$) and ρ_i (resp. ρ_i^*) may take the value $\rho(f_i)$ (resp. $\rho(f_i^*)$) if it attained. We next set $f_1 = g$ and $f_2 = h$. Also let $dx^k := x^{k+1} - x^k$ and $dy^k := y^{k+1} - y^k$.

Theorem 1.2 *Suppose that the sequences $\{x^k\}$ and $\{y^k\}$ are generated by the simplified DCA. Then we have*

i)

$$\begin{aligned}
(g-h)(x^{k+1}) &\leq (h^*-g^*)(y^k) - \max\left\{\frac{\rho_2}{2}\|dx^k\|^2, \frac{\rho_2^*}{2}\|dy^k\|^2\right\} \\
&\leq (g-h)(x^k) - \max\left\{\frac{\rho_1+\rho_2}{2}\|dx^k\|^2, \frac{\rho_1^*}{2}\|dy^{k-1}\|^2\right. \\
&\quad \left. + \frac{\rho_2}{2}\|dx^k\|^2, \frac{\rho_1^*}{2}\|dy^{k-1}\|^2 + \frac{\rho_2^*}{2}\|dy^k\|^2\right\}.
\end{aligned}$$

ii)

$$\begin{aligned}
(h^*-g^*)(y^{k+1}) &\leq (g-h)(x^{k+1}) - \max\left\{\frac{\rho_1}{2}\|dx^{k+1}\|^2, \frac{\rho_1^*}{2}\|dy^k\|^2\right\} \\
&\leq (h^*-g^*)(y^k) - \max\left\{\frac{\rho_1^*+\rho_2^*}{2}\|dy^k\|^2, \frac{\rho_1^*}{2}\|dy^k\|^2\right. \\
&\quad \left. + \frac{\rho_2}{2}\|dx^k\|^2, \frac{\rho_1}{2}\|dx^{k+1}\|^2 + \frac{\rho_2}{2}\|dx^k\|^2\right\}.
\end{aligned}$$

Theorem 1.3 *Suppose that the sequences $\{x^k\}$ and $\{y^k\}$ are generated by the simplified DCA. Then we have*

i) *The sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing and*

- $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ *if and only if* $\{x^k, x^{k+1}\} \subset \partial g^*(y^k) \cap \partial h^*(y^k)$ *and* $[\rho(h) + \rho(g)]\|x^{k+1} - x^k\| = 0$.
- $h^*(y^{k+1}) - g^*(y^{k+1}) = h^*(y^k) - g^*(y^k)$ *if and only if* $\{y^k, y^{k+1}\} \subset \partial g(x^k) \cap \partial h(x^k)$ *and* $[\rho(h^*) + \rho(g^*)]\|y^{k+1} - y^k\| = 0$.

DCA terminates at the kth iteration if either of the above equalities holds.

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

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

iv) *DCA has a linear convergence for general DC program.*

v) *In polyhedral DC programs, the sequences $\{x^k\}$ and $\{y^k\}$ contain finitely many elements and DCA has a finite convergence.*

Remark 1.5 *If we construct three sequences $\{x^k\}, \{y^k\}$ and $\{z^k\}$ such that, for any $k = 0, 1, 2, \ldots,$*
– $y^k \in \partial h(x^k)$
– $z^k \in \partial g^*(y^k) = \arg\min\{g(x) - \langle x, y^k\rangle : x \in \mathbb{R}^n\}$
– x^{k+1} satisfies $f(x^{k+1}) \leq f(z^k)$.
Then the assertions of Theorems 1.2 and 1.3 are still valid except that z^k is in the place of x^{k+1} whenever this notation appears.

1.1.4 Approximate DCA

The general scheme of DCA requires at each iteration the computations of $y^k \in \partial h(x^k)$ and $x^{k+1} \in \partial g^*(y^k)$. However, these computations are not necessarily exact. Especially for the computation of x^{k+1} since we may have to face the subproblems of the form (1.5b) that are not easy to solve. Although the computation of y^k is usually explicit, we can not exclude the case the function h is complex so that we have to appeal to numerical methods for computing approximately ∂h.

In this section, we present an approximate version of DCA and prove its convergence. The term "approximate" here means that we only need to compute the approximate values of y^k and x^{k+1} upto a precision. Concretely, at each iteration we only need to compute an $y^k \in \partial_\varepsilon h(x^k)$ for a $\varepsilon > 0$. By virtue of Proposition 1.2, such a y^k satisfies

$$h(x^k) + h^*(y^k) \leq \langle x^k, y^k\rangle + \varepsilon \Leftrightarrow h^*(y^k) - \langle x^k, y^k\rangle \leq h^*(y) - \langle x^k, y\rangle + \varepsilon, \quad \forall y \in X.$$

This means that y^k is a ε-solution of problem (1.5a). Similarly, if $x^{k+1} \in \partial_\varepsilon g^*(y^k)$ then x^{k+1} is a ε-solution of problem (1.5b). Results relating to the approximate DCA are stated in two following theorems.

Theorem 1.4 *Let g and h be proper lower semi-continuous convex functions on $\mathbb{R}^n$ and an $\varepsilon > 0$. Suppose that $\{x^k\}$ and $\{y^k\}$ are sequences defined by (for all $k = 0, 1, 2, \ldots$)*

$$y^k \in \partial_\varepsilon h(x^k), \quad x^{k+1} \in \partial_\varepsilon g^*(y^k). \tag{1.10}$$

If (x^*, y^*) *is any limit point of* $\{(x^k, y^k)\}$*, then*

$$y^* \in \partial_\varepsilon h(x^*) \cap \partial_{3\varepsilon} g(x^*) \tag{1.11}$$

and

$$x^* \in \partial_\varepsilon h^*(y^*) \cap \partial_{3\varepsilon} g^*(y^*). \tag{1.12}$$

Proof : Since (x^*, y^*) is a limit point of $\{(x^k, y^k)\}$, there is a subset $\mathcal{K} \subset \mathbb{N}$ such that

$$\lim_{\substack{k\to\infty \\ k\in\mathcal{K}}} (x^k, y^k) = (x^*, y^*).$$

For any $k = 0, 1, 2, \ldots$, we have

$$y^k \in \partial_\varepsilon h(x^k) \Leftrightarrow h(x^k) + h^*(y^k) \le \langle x^k, y^k\rangle + \varepsilon, \tag{1.13}$$

Taking $k \to \infty, k \in \mathcal{K}$, (1.13) implies

$$h(x^*) + g(y^*) \le \langle x^*, y^*\rangle + \varepsilon \Leftrightarrow y^* \in \partial_\varepsilon h(x^*) \Leftrightarrow x^* \in \partial_\varepsilon h^*(y^*).$$

We need to prove that

$$y^* \in \partial_{3\varepsilon} g(x^*) \Leftrightarrow x^* \in \partial_{3\varepsilon} g^*(y^*) \Leftrightarrow g(x^*) + g^*(y^*) \le \langle x^*, y^*\rangle + 3\varepsilon.$$

By (1.13) and the facts that $g^*(y^k) + g(x^{k+1}) \ge \langle x^{k+1}, y^k\rangle$ and

$$x^{k+1} \in \partial_\varepsilon g^*(y^k) \Leftrightarrow y^k \in \partial_\varepsilon g(x^{k+1}) \Rightarrow g(x^k) \ge g(x^{k+1}) + \langle y^k, x^k - x^{k+1}\rangle - \varepsilon$$

we have, for all $k = 0, 1, 2, \ldots$,

$$h^*(y^k) - g^*(y^k) \le g(x^{k+1}) - h(x^k) - \langle x^{k+1} - x^k, y^k\rangle + \varepsilon \le g(x^k) - h(x^k) + 2\varepsilon.$$

Therefore,

$$\begin{aligned}
2\varepsilon &\ge \liminf_{k\to\infty}(g-h)(x^k) - \liminf_{k\to\infty}(h^*-g^*)(y^k) \ge \liminf_{k\to\infty}\big(g(x^k) + g^*(y^k) - h(x^k) - h^*(y^k)\big) \\
&\ge \liminf_{k\to\infty}\big(g(x^k) + g^*(y^k) - \langle x^k, y^k\rangle - \varepsilon\big) \ge \liminf_{\substack{k\to\infty \\ k\in\mathcal{K}}}\big(g(x^k) + g^*(y^k) - \langle x^k, y^k\rangle - \varepsilon\big) \\
&= \liminf_{\substack{k\to\infty \\ k\in\mathcal{K}}}\big(g(x^k) + g^*(y^k)\big) - \langle x^*, y^*\rangle - \varepsilon \ge g(x^*) + g^*(y^*) - \langle x^*, y^*\rangle - \varepsilon,
\end{aligned}$$

(the last inequality holds due to lower semi-continuity of g and g^*). This means that

$$g(x^*) + g^*(y^*) \le \langle x^*, y^*\rangle + 3\varepsilon \Leftrightarrow y^* \in \partial_{3\varepsilon} g(x^*) \Leftrightarrow x^* \in \partial_{3\varepsilon} g^*(y^*).$$

□

Theorem 1.5 *Let* g *and* h *be closed proper convex functions on* $\mathbb{R}^n$*,* $\{\varepsilon_k\} \subset \mathbb{R}_+$ *such that* $\varepsilon_k \to 0$*. Suppose that* $\{x^k\}$ *and* $\{y^k\}$ *are sequences well defined by (for all* $k = 0, 1, 2, \ldots$*)*

$$y^k \in \partial_{\varepsilon_k} h(x^k), \quad x^{k+1} \in \partial_{\varepsilon_k} g^*(y^k). \tag{1.14}$$

If (x^*, y^*) *is any limit point of* $\{(x^k, y^k)\}$*, then*

$$y^* \in \partial h(x^*) \cap \partial g(x^*) \Leftrightarrow x^* \in \partial h^*(y^*) \cap \partial g^*(y^*). \tag{1.15}$$

Proof : Note that if $\varepsilon < \varepsilon'$ and $f \in \Gamma_0(\mathbb{R}^n)$, we have the inference $y \in \partial_\varepsilon f(x) \Rightarrow y \in \partial_{\varepsilon'} f(x)$.

Since $\varepsilon_k \to 0$, for any $\varepsilon > 0$, there is a $N(\varepsilon) \in \mathbb{N}$ such that $\varepsilon_k < \varepsilon \ \forall k \geq N(\varepsilon)$. Then

$$y^k \in \partial_\varepsilon h(x^k) \text{ and } x^{k+1} \in \partial_\varepsilon g^*(y^k), \quad \forall k \geq N(\varepsilon).$$

By Theorem 1.4, we have

$$y^* \in \partial_\varepsilon h(x^*) \cap \partial_{3\varepsilon} g(x^*) \Leftrightarrow x^* \in \partial_\varepsilon h^*(y^*) \cap \partial_{3\varepsilon} g^*(y^*).$$

This is satisfied for arbitrary $\varepsilon > 0$, so we have (1.15). □

1.2 Robust optimization

Data uncertainty is common in real-world applications due to various reasons, including imprecise measurement, outdated sources and implementation errors. For example, radar data collected by a system of antennas are uncertain because of the noise in devices and affect of environmental conditions where signals pass through. In the case of missing values, using statistical methods such as imputation to estimate the values of the missing variables results in data with a certain degree of uncertainty. Another example, when a number of replicates of the same experiment are available, data points are often provided approximately, i.e. their features are only specified up to given intervals of confidence. Again, in image classification applications, some features may rely on image processing outputs that introduce errors. Hence classification problems based on the observed image features have noisy inputs. These kinds of uncertainty must be handled carefully, or else the results could be highly unreliable even with very small perturbations of the nominal data. Consequently, there exists a real need of a methodology capable to detect the cases when data uncertainty can heavily effect the quality of the nominal solution, and to generate a robust solution in such cases, one that is immunized against the effect of data uncertainty. Recently, the idea of using robust optimization to deal with such data uncertainty has attracted more interest from researchers. Robust optimization provides a novel and systematic approach for dealing with data uncertainty by solving a min-max optimization problem. By treating the uncertainty in the data as deterministic, the found solution tolerates changes in problem data up to a certain error, and guarantees a certain level of performance. An excellent review of the robust optimization can be found in (Ben-Tal et al., 2009). In what follows, we give a summary of this method.

A generic mathematical programming problem is of the form

$$\min_{x \in \mathbb{R}^n} \{ f_0(x, u) : f_i(x, u) \leq 0, i = 1, \ldots, m \}, \tag{P[u]}$$

where $x \in \mathbb{R}^n$ is a vector of decision variables, the function f_0 (the objective function) and $f_1, \ldots, f_m$ are *structural elements* of the problem, and u stands for the *data* specifying a particular problem instance. Since the data u can not be determined exactly, it is assumed to take arbitrary values in an *uncertainty set* $\mathcal{U}$ in the space of data. Then, we have to deal with an *uncertain optimization problem* defined as a collection of the usual ("certain") optimization problems

$$\{(P[u]) \mid u \in \mathcal{U}\}. \tag{1.16}$$

For the purpose of immunizing against the effect of data uncertainty, a meaningful candidate solution x of the uncertain problem (1.16) is required to be feasible for all realizations of the disturbances u within $\mathcal{U}$. That is, x is required to satisfy the semi-infinite system of constraints

$$f_i(x, u) \leq 0, i = 1, \ldots, m \quad \forall u \in \mathcal{U}.$$

Moreover, to quantify robustly the quality of the uncertain problem, we minimize the largest value $\hat{f}_0(x) = \sup_{u \in \mathcal{U}} f_0(x, u)$ – say *robust value* – of the "true" objective $f_0(x)$ over all realizations of the data from the uncertainty set. These lead to the *Robust Counterpart* of the uncertain problem (1.16)

$$\min_{x \in \mathbb{R}^n} \{\sup_{u \in \mathcal{U}} f_0(x, u) : \sup_{u \in \mathcal{U}} f_i(x, u) \leq 0, i = 1, \ldots, m\}. \tag{1.17}$$

The feasible/optimal solutions to the Robust Counterpart are called *robust feasible/optimal* solutions of the uncertain problem (1.16).

Part I

Learning with sparsity

Chapter 2

DC approximation approaches for sparse optimization[1]

Abstract: *Sparse optimization refers to an optimization problem involving the zero-norm in objective or constraints. In this chapter, we study nonconvex approximation approaches for sparse optimization with a unifying point of view in DC (Difference of Convex functions) programming framework. Considering a common DC approximation of the zero-norm including all standard sparsity-inducing penalty functions, we study the consistency between global minimums (resp. local minimums) of approximate and original problems. We show that, in several cases, some global minimizers (resp. local minimizers) of the approximate problem are also those of the original problem. Using exact penalty techniques in DC programming, we prove stronger results for some particular approximations, namely, the approximate problem, with suitable parameters, is equivalent to the original problem. The efficiency of several sparsity-inducing penalty functions have been fully analyzed. Four DCA (DC Algorithm) schemes are developed that cover all standard algorithms in nonconvex sparse approximation approaches as special versions. Three among the four DCA schemes can be viewed as an ℓ_1-perturbed algorithm / reweighted-ℓ_1 algorithm / reweighted-ℓ_2 algorithm. We offer a unifying nonconvex approximation approach, with solid theoretical tools as well as efficient algorithms based on DC programming and DCA, to tackle the zero-norm and sparse optimization.*

2.1 Introduction

The ℓ_0-norm is an important concept for modelling the sparsity of data and plays a crucial role in optimization problems where one has to select representative variables. Sparse optimization, which refers to an optimization problem involving the ℓ_0-norm in objective or

1. This chapter is published under the title:
[1]. Hoai An Le Thi, Tao Pham Dinh, Hoai Minh Le, Xuan Thanh Vo. DC approximation approaches for sparse optimization. *European Journal of Operational Research* 244 (1): 26–46 (2015).

constraints, has many applications in various domains (in particular in machine learning, image processing and finance), and draws increased attention from many researchers in recent years. The function ℓ_0, apparently very simple, is lower-semicontinuous on $\mathbb{R}^n$, but its discontinuity at the origin makes nonconvex programs involving $\|.\|_0$ challenging. Note that although one uses the term "norm" to design $\|.\|_0$, $\|.\|_0$ is not a norm in the mathematical sense. Indeed, for all $x \in \mathbb{R}^n$ and $\lambda \neq 0$, one has $\|\lambda x\|_0 = \|x\|_0$, which is not true for a norm.

Formally, a sparse optimization problem takes the form

$$\inf \{f(x,y) + \lambda \|x\|_0 : (x,y) \in K \subset \mathbb{R}^n \times \mathbb{R}^m \}, \tag{2.1}$$

(Weston et al., 2003; Zhang et al., 2006) where the function f corresponds to a given criterion and λ is a positive number, called the regularization parameter, that makes the trade-off between the criterion f and the sparsity of x. In some applications, one wants to control the sparsity of solutions, the ℓ_0-term is thus put in constraints, and the corresponding optimization problem is

$$\inf\{f(x,y) : (x,y) \in K, \|x\|_0 \leq k\}. \tag{2.2}$$

Let us mention some important applications of sparse optimization corresponding to these models.

Feature selection in classification learning: Feature selection is one of fundamental problems in machine learning. In many applications such as text classification, web mining, gene expression, micro-array analysis, combinatorial chemistry, image analysis, etc, data sets contain a large number of features, many of which are irrelevant or redundant. Feature selection is often applied to high-dimensional data prior to classification learning. The main goal is to select a subset of features of a given data set while preserving or improving the discriminative ability of the classifier. Given a training data $\{a_i, b_i\}_{i=1,\dots,q}$ where each $a_i \in \mathbb{R}^n$ is labeled by its class $b_i \in Y$, the discrete set of labels. The aim of classification learning is to construct a classifier function that discriminates the data points $A := \{a_i\}_{i=1,\dots,q}$ with respect to their classes$\{b_i\}_{i=1,\dots,q}$. The embedded feature selection in classification consists of determining the classifier which uses as few features as possible, that leads to a sparse optimization problem like (2.1).

Sparse Regression: Given a training data set $\{b_i, a_i\}_{i=1}^q$ of q independent and identically distributed samples composed of explanatory variables $a_i \in \mathbb{R}^n$ (inputs) and response variables $b_i \in \mathbb{R}$ (ouputs). Let $b := (b_i)_{i=1,\dots,q}$ denote the vector of outputs and $A := (a_{i,j})_{i=1,\dots,q}^{j=1,\dots,n}$ denote the matrix of inputs. The problem of the regression consists in looking for a relation which can possibly exist between A and b, in other words, relating b to a function of A and a model parameter x. Such a model parameter x can be obtained by solving the optimization

problem

$$\min\left\{f(x) := \sum_{i=1}^{q} L(b_i, a_i^T x) : x \in \mathbb{R}^n\right\}, \tag{2.3}$$

where $L : \mathbb{R}^n \to \mathbb{R}$ is called loss function. The *sparse regression* problem aims to find a sparse solution of the above regression model, it takes the form of (2.1):

$$\min_{x\in\mathbb{R}^n}\left\{\sum_{i=1}^{q} L(b_i, a_i^T x) + \rho\,\|x\|_0\right\}. \tag{2.4}$$

Sparse Fisher Linear Discriminant Analysis: Discriminant analysis captures the relationship between multiple independent variables and a categorical dependent variable in the usual multivariate way, by forming a composite of the independent variables. Given a set of q independent and identically distributed samples composed of explanatory variables $a_i \in \mathbb{R}^n$ and binary response variables $b_i \in \{-1, 1\}$. The idea of Fisher linear discriminant analysis is to determine a projection of variables onto a straight line that best separables the two classes. The line is so determined to maximize the ratio of the variances of between and within classes in this projection, i.e. maximize the function $f(\alpha) = \frac{\langle\alpha, S_B\alpha\rangle}{\langle\alpha, S_W\alpha\rangle}$, where S_B and S_W are, respectively, the between and within classes scatter matrix (they are symmetric positive semidefinite) given by

$$S_B := (q_+ - q_-)(q_+ - q_-)^T,\ S_W = S_+ + S_-,$$

$$S_+ = \sum_{i=1, b_i=+1}^{q} (x_i - q_+)(x_i - q_+)^T,\ S_- = \sum_{i=1, b_i=-1}^{q} (x_i - q_-)(x_i - q_-)^T.$$

Here, for $j \in \{\pm\}$, q_j is the mean vector of class j, l_j is the number of labeled samples in class j. If α is an optimal solution of the problem, then the classifier is given by $F(a) = \alpha^T a + c$, $c = 0.5\alpha^T(q_+ - q_-)$.
The sparse Fisher Discriminant model is defined by ($\rho > 0$)

$$\min\{\alpha^T S_W \alpha + \rho\,\|\alpha\|_0 : \alpha^T(q_+ - q_-) = b\}.$$

Compressed sensing: Compressed sensing refers to techniques for efficiently acquiring and reconstructing signals via the resolution of underdetermined linear systems. Compressed sensing concerns sparse signal representation, sparse signal recovery and sparse dictionary learning which can be formulated as sparse optimization problems of the form (2.1).

Portfolio selection problem with cardinality constraint: In portfolio selection problem, given a set of available securities or assets, we want to find the optimum way of investing a particular amount of money in these assets. Each of the different ways to diversify this money among the several assets is called a portfolio. In portfolio management one wants to

limit the number of assets to be investigated in the portfolio, that leads to a problem of the form (2.2).

Other applications: Other applications of sparse optimization include Sensor networks (Bajawa et al., 2006; Baron et al., 2006), Error correction (Candes and Tao, 2005; Candes and Randhall, 2006), Digital photography (Takhar et al., 2006)), etc.

Existing works. During the last two decades, research is very active in models and methods optimization involving the zero-norm. Works can be divided into three categories according to the way to treat the zero-norm: convex approximation, nonconvex approximation, and nonconvex exact reformulation.

In the machine learning community, one of the best known approaches, belonging to the group "convex approximation", is the ℓ_1 regularization approach proposed in (Tibshirani, 1996) in the context of linear regression, called LASSO (Least Absolute Shrinkage and Selection Operator), which consists in replacing the ℓ_0 term $\|x\|_0$ by $\|x\|_1$, the ℓ_1 -norm of the vector x. In (Gribonval and Nielsen, 2003), the authors have proved that, under suitable assumptions, a solution of the ℓ_0- regularizer problem over a polyhedral set can be obtained by solving the ℓ_1- regularizer problem. However, these assumptions are quite restrictive. Since its introduction, several works have been developed to study the ℓ_1-regularization technique, from the theoretical point of view to efficient computational methods (see (Hastie et al., 2009), Chapter 18 for more discussions on ℓ_1-regularized methods). The LASSO penalty has been shown to be, in certain cases, inconsistent for variable selection and biased (Zou, 2006). Hence, the Adaptive LASSO is introduced in (Zou, 2006) in which adaptive weights are used for penalizing different coefficients in the ℓ_1-penalty.

At the same time, nonconvex continuous approaches, belonging to the second group "nonconvex approximation" (the ℓ_0 term $\|x\|_0$ is approximated by a nonconvex continuous function) were extensively developed. A variety of sparsity-inducing penalty functions have been proposed to approximate the ℓ_0 term: exponential concave function (Bradley and Mangasarian, 1998), ℓ_p-norm with $0 < p < 1$ (Fu, 1998) and $p < 0$ (Rao and Kreutz-Delgado, 1999), Smoothly Clipped Absolute Deviation (SCAD) (Fan and Li, 2001), Logarithmic function (Weston et al., 2003), Capped-ℓ_1 (Peleg and Meir, 2008) (see (2.17), (2.18) and Table 2.1 in Section 2.2 for the definition of these functions). Using these approximations, several algorithms have been developed for resulting optimization problems, most of them are in the context of feature selection in classification, sparse regressions or more especially for sparse signal recovery: Successive Linear Approximation (SLA) algorithm (Bradley and Mangasarian, 1998), DCA (Difference of Convex functions Algorithm) based algorithms (Chen et al., 2010; Collober et al., 2006; Gasso et al., 2009; Guan and Gray, 2013; Le et al., 2013a; Le Thi et al., 2008, 2009b; Le Thi et al., 2013b; Le Thi and Nguyen, 2013; Newmann et al., 2005; Ong and Le Thi, 2012), Local Linear Approximation (LLA) (Zou and Li, 2008), Two-stage ℓ_1 (Zhang, 2009), Adaptive Lasso (Zou, 2006), reweighted-ℓ_1 algorithms (Candes et al., 2008)), reweighted- ℓ_2 algorithms such as Focal Underdetermined System Solver (FOCUSS)

(Gorodnitsky and Rao (1997); Rao and Kreutz-Delgado (1999); Rao et al. (2003)), Iteratively reweighted least squares (IRLS) and Local Quadratic Approximation (LQA) algorithm (Fan and Li, 2001; Zou and Li, 2008).

In the third category named nonconvex exact reformulation approaches, the ℓ_0-regularized problem is reformulated as a continuous nonconvex program. There are a few works in this category. In (Mangasarian, 1996), the author reformulated the problem (2.1) in the context of feature selection in SVM as a linear program with equilibrium constraints (LPEC). However, this reformulation is generally intractable for large-scale datasets. In (Thiao et al., 2008; Pham Dinh and Le Thi, 2014) an exact penalty technique in DC programming is used to reformulate (2.1) and (2.2) as DC programs. In (Thiao et al., 2010) this technique is used for Sparse Eigenvalue problem with ℓ_0-norm in constraint functions

$$\max\{x^T Ax : x^T x = 1, \|x\|_0 \leq k\}, \tag{2.5}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric and k an integer, and a DCA based algorithm was investigated for the resulting problem.

Beside the three above categories, heuristic methods are developed to tackle directly the original problem (2.1) by greedy based algorithms, e.g. matching pursuit, orthogonal matching pursuit, (Mallat and Zhang, 1993; Pati et al., 1993), etc.

Convex regularization approaches involve convex optimization problems which are so far "easy" to solve, but they do not attain the solution of the ℓ_0-regularizer problem. Nonconvex approximations are, in general, deeper than convex relaxations, and then can produce good sparsity, but the resulting optimization problems are still difficult since they are nonconvex and there are many local minima which are not global. Many issues have not yet been studied or proved in the existing approximation approaches. First, the consistency between the approximate problems and the original problem is a very important question but still is open. Only a weak result has been proved for two special cases in (Bradley et al., 1998) (resp. (Rinaldi et al., 2010)) when f is concave, bounded below on a polyhedral convex set K and the approximation term is an exponential concave function (resp. a logarithm function and/or ℓ_p-norm ($p < 1$)). It has been shown in these works that the intersection of the solution sets of the approximate problem and the original problem is nonempty. Moreover no result on the consistency between local minimum of approximate and original problems has been available, while most of the proposed algorithms furnish local minima. Second, several existing algorithms lack a rigorous mathematical proof of convergence. Hence the choice of a "good" approximation remains relevant. Two crucial questions should be studied for solving large scale problems, that are, *how to suitably approximate the zero-norm* and *which computational method to use* for solving the resulting optimization problem. The development of new models and algorithms for sparse optimization problems is always a challenge for researchers in optimization and machine learning.

We consider in this chapter the problem (2.1) where K is a convex set in $\mathbb{R}^n \times \mathbb{R}^m$ and f is a finite DC function on $\mathbb{R}^n \times \mathbb{R}^m$. We address all issues aforementioned for approximation

approaches and develop an unifying approach based on DC programming and DCA.

Firstly, considering a common DC approximate function, denoted by r_θ where θ is a parameter controling the tightness of approximation, we prove the consistency between the approximate problem and the original problem by showing the link between their global minimizers as well as their local minimizers. We demonstrate that any optimal solution of the approximate problem is in a $\epsilon-$neighbourhood of an optimal solution to the original problem (2.1). More strongly, if f is concave and the objective function of the approximate problem is bounded below on K, then some optimal solutions of the approximate problem are exactly solutions of the original problem. These new results are important and very useful for justifying the performance of approximation approaches.

Secondly, we provide an in-depth analysis of usual sparsity-inducing functions and compare them according to suitable parameter values. This study suggests the choice of good approximations of the zero-norm as well as that of good parameters for each approximation. A reasonable comparison via suitable parameters identifies Capped -ℓ_1 and SCAD as the best approximations.

Thirdly, we prove, via an exact reformulation approach by exact penalty techniques that, with suitable parameters ($\theta > \theta_0$ for some θ_0), nonconvex approximate problems resulting from Capped -ℓ_1 or SCAD functions are equivalent to the original problem. Moreover, when the set K is a box, we can show directly (without using exact penalty techniques) the equivalence between the original problem and the approximate Capped -ℓ_1 problem and give the value of θ_0 such that this equivalence holds for all $\theta > \theta_0$. These interesting and significant results justify our analysis on usual sparsity-inducing functions and the pertinence of these approximation approaches. It opens the door to study other approximation approaches which are consistent with the original problem.

Fourthly, we develop solution methods for all DC approximation approaches. Our algorithms are based on DC programming and DCA, because our main motivation is to exploit the efficiency of DCA to solve this hard problem. We propose three DCA schemes for three different formulations of a common model to all concave approximation functions. We show that these DCA schemes include all standard algorithms as special versions. The fourth DCA scheme is concerned with the resulting DC program given by the DC approximation (nonconcave piecewise linear) function in (Le Thi (2012)). Using DC programming framework, we unify all solution methods into DCA, and then convergence properties of our algorithms are guaranteed, thanks to general convergence results of the generic DCA scheme. It permits to exploit, in an elegant way, the nice effect of DC decompositions of the objective functions to design various versions of DCA. It is worth mentioning here the flexibility/versatility of DC programming and DCA: three among four proposed algorithms can be viewed as an ℓ_1-perturbed algorithm / a reweighted-ℓ_1 algorithm (intimately related to the ℓ_1-penalized LASSO approach) / a reweighted-ℓ_2 algorithm in case of convex objective functions.

Finally, as an application, we consider the problem of feature selection in SVM and perform

a careful empirical comparison of all approaches.

The rest of the chapter is organized as follows. The consistency between approximate problems and the original one, the link between their global minimizer as well as their local minimizer are studied in Section 2.2, while a comparative analysis on usual approximations is discussed in Section 2.3. A deeper study on Capped-ℓ_1 approximation and the relation between some approximate problems and exact penalty approaches is presented in Section 2.4. Solution methods based on DCA are developed in Section 2.5, while the application of the proposed algorithms for feature selection in SVM and numerical experiments are described in Section 2.6.

2.2 DC approximation approaches: consistency results

We focus on the sparse optimization problem with ℓ_0-norm in the objective function, called the ℓ_0-problem, that takes the form

$$\min \{F(x,y) = f(x,y) + \lambda\|x\|_0 : (x,y) \in K\}, \tag{2.6}$$

where λ is a positive parameter, K is a convex set in $\mathbb{R}^n \times \mathbb{R}^m$ and f is a finite DC function on $\mathbb{R}^n \times \mathbb{R}^m$. Suppose that f has a DC decomposition

$$f(x,y) = g(x,y) - h(x,y) \quad \forall (x,y) \in \mathbb{R}^n \times \mathbb{R}^m, \tag{2.7}$$

where g, h are finite convex functions on $\mathbb{R}^n \times \mathbb{R}^m$. Through the chapter, for a DC function $f := g - h$, $\partial f(x,y)$ stands for the set $\partial g(x,y) - \partial h(x,y)$. More precisely, the notation $(\overline{x}, \overline{y}) \in \partial f(x,y)$ means that $(\overline{x}, \overline{y}) = (x_g, y_g) - (x_h, y_h)$ for some $(x_g, y_g) \in \partial g(x,y)$, $(x_h, y_h) \in \partial h(x,y)$.

Define the step function $s : \mathbb{R} \to \mathbb{R}$ by $s(t) = 1$ for $t \neq 0$ and $s(t) = 0$ otherwise. Then $\|x\|_0 = \sum_{i=1}^n s(x_i)$. The idea of approximation methods is to replace the discontinuous step function by a continuous approximation r_θ, where $\theta > 0$ is a parameter controling the tightness of approximation. This leads to the approximate problem of the form

$$\min \left\{ F_{r_\theta}(x,y) = f(x,y) + \lambda \sum_{i=1}^n r_\theta(x_i) : (x,y) \in K \right\}. \tag{2.8}$$

Assumption 1 *$\{r_\theta\}_{\theta>0}$ is a family of functions $\mathbb{R} \to \mathbb{R}$ satisfying the following properties:*

i) $\lim_{\theta \to +\infty} r_\theta(t) = s(t)$, $\forall t \in \mathbb{R}$.
ii) For any $\theta > 0$, r_θ is even, i.e. $r_\theta(t) = r_\theta(|t|)$ $\forall t \in \mathbb{R}$) and r_θ is increasing on $[0, +\infty)$.
iii) For any $\theta > 0$, r_θ is a DC function which can be represented as

$$r_\theta(t) = \varphi_\theta(t) - \psi_\theta(t) \quad t \in \mathbb{R},$$

where $\varphi_\theta, \psi_\theta$ are finite convex functions on $\mathbb{R}$.

iv) $t\mu \geq 0\ \forall t \in \mathbb{R}, \mu \in \partial r_\theta(t)$.*where* $\partial r_\theta(t) = \{u - v : u \in \partial\varphi_\theta(t), v \in \partial\psi_\theta(t)\}$.
v) For any $a \leq b$ *and* $0 \notin [a, b]$: $\lim_{\theta\to+\infty} \sup\{|z| : z \in \partial r_\theta(t), t \in [a, b]\} = 0$.

First of all, we observe that by assumption ii) above, we get another equivalent form of (2.8)

$$\min_{(x,y,z)\in\Omega_1} \overline{F}_{r_\theta}(x, y, z) := f(x, y) + \lambda \sum_{i=1}^{n} r_\theta(z_i), \tag{2.9}$$

where

$$\Omega_1 = \{(x, y, z) : (x, y) \in K, |x_i| \leq z_i \quad \forall i = 1, \ldots, n\}.$$

Indeed, (2.8) and (2.9) are equivalent in the following sense.

Proposition 2.1 *A point* $(x^*, y^*) \in K$ *is a global (resp. local) solution of the problem* (2.8) *if and only if* $(x^*, y^*, |x^*|)$ *is a global (resp. local) solution of the problem* (2.9). *Moreover, if* (x^*, y^*, z^*) *is a global solution of* (2.9) *then* (x^*, y^*) *is a global solution of* (2.8).

Proof : Since r_θ is an increasing function on $[0, +\infty)$, we have

$$\overline{F}_{r_\theta}(x, y, z) \geq \overline{F}_{r_\theta}(x, y, |x|) = F_{r_\theta}(x, y) \quad \forall (x, y, z) \in \Omega_1.$$

Then the conclusion concerning global solutions is trivial. The result on local solutions also follows by remarking that if $(x, y, z) \in B((x^*, y^*, z^*), \delta)$ [2] then $(x, y) \in B((x^*, y^*), \delta)$, and if $(x, y) \in B((x^*, y^*), \frac{\delta}{2})$ then $(x, y, |x|) \in B((x^*, y^*, |x^*|), \delta)$. □

In standard nonconvex approximation approaches to ℓ_0-problem, all the proposed approximation functions r_θ are even and concave increasing on $[0, +\infty)$ (see Table 2.1 below) and the approximate problems were often considered in the form (2.9). Here we study the general case where r_θ is a DC function and consider both problems (2.8) and (2.9) in order to exploit the nice effect of DC decompositions of a DC program.

Now we show the link between the original problem (2.6) and the approximate problem (2.8). This result gives *a mathematical foundation* of approximation methods.

Theorem 2.1 *Let* $\mathcal{P}, \mathcal{P}_\theta$ *be the solution sets of the problem* (2.6) *and* (2.8) *respectively.*

i) Let $\{\theta_k\}$ *be a sequence of nonnegative numbers such that* $\theta_k \to +\infty$ *and* $\{(x^k, y^k)\}$ *be a sequence such that* $(x^k, y^k) \in \mathcal{P}_{\theta_k}$ *for any* k. *If* $(x^k, y^k) \to (x^*, y^*)$, *then* $(x^*, y^*) \in \mathcal{P}$.
ii) If K *is compact, then for any* $\epsilon > 0$ *there is* $\theta(\epsilon) > 0$ *such that*

$$\mathcal{P}_\theta \subset \mathcal{P} + B(0, \epsilon) \quad \forall \theta \geq \theta(\epsilon).$$

iii) If there is a finite set $\mathcal{S}$ *such that* $\mathcal{P}_\theta \cap \mathcal{S} \neq \emptyset\ \forall \theta > 0$, *then there exists* $\theta_0 \geq 0$ *such that*

$$\mathcal{P}_\theta \cap \mathcal{S} \subset \mathcal{P} \quad \forall \theta \geq \theta_0.$$

2. $B(u^*, \delta)$ stands for the set of vectors $u \in \mathbb{R}^d$ such that $\|u - u^*\| < \delta$

Proof : i) Let (x, y) be arbitrary in K. For any k, since $(x^k, y^k) \in \mathcal{P}_{\theta_k}$, we have

$$f(x, y) + \lambda \sum_{i=1}^{n} r_{\theta_k}(x_i) \geq f(x^k, y^k) + \lambda \sum_{i=1}^{n} r_{\theta_k}(x_i^k). \tag{2.10}$$

By Assumption 1 ii), if $x_i^* = 0$, we have

$$\liminf_{k \to +\infty} r_{\theta_k}(x_i^k) \geq \liminf_{k \to +\infty} r_{\theta_k}(0) = 0.$$

If $x_i^* \neq 0$, there exist $a_i \leq b_i$ and $k_i \in \mathbb{N}$ such that $0 \neq [a_i, b_i]$ and $x_i^k \in [a_i, b_i]$ for all $k \geq k_i$. Then we have

$$|r_{\theta_k}(x_i^k) - s(x_i^*)| \leq \max\{|r_{\theta_k}(a_i) - s(a_i)|, |r_{\theta_k}(b_i) - s(b_i)|\} \quad \forall k \geq k_i.$$

Since $\lim_{k \to +\infty} r_{\theta_k}(a_i) = s(a_i)$ and $\lim_{k \to +\infty} r_{\theta_k}(b_i) = s(b_i)$, we have $\lim_{k \to +\infty} r_{\theta_k}(x_i^k) = s(x_i^*)$. Note that f is continuous, taking $\liminf$ of both sides of (2.7), we get

$$f(x, y) + \lambda \sum_{i=1}^{n} s(x_i) \geq f(x^*, y^*) + \lambda \sum_{i=1}^{n} \liminf_{k \to \infty} r_{\theta_k}(x_i^k) \geq f(x^*, y^*) + \lambda \sum_{i=1}^{n} s(x_i^*).$$

Thus, $F(x, y) \geq F(x^*, y^*)$ for any $(x, y) \in K$, or $(x^*, y^*) \in \mathcal{P}$.

ii) We assume by contradiction that there exists $\epsilon > 0$ and a sequence $\{\theta_k\}$ such that $\theta_k \to +\infty$, and for any k there is $(x^k, y^k) \in \mathcal{P}_{\theta_k} \backslash (\mathcal{P} + B(0, \epsilon))$. Since $\{(x^k, y^k)\} \subset K$ and K is compact, there exists a subsequence $\{(x^{k_l}, y^{k_l})\}$ of $\{(x^k, y^k)\}$ converges to a point $(x^*, y^*) \in K$. By i), we have $(x^*, y^*) \in \mathcal{P}$. However, $\{(x^{k_l}, y^{k_l})\} \subset K \backslash (\mathcal{P} + B(0, \epsilon))$ that is a closed set, so $(x^*, y^*) \in K \backslash (\mathcal{P} + B(0, \epsilon))$. This contradicts the fact that $(x^*, y^*) \in \mathcal{P}$.

iii) Assume by contradiction that there is a sequence $\{\theta_k\}$ such that $\theta_k \to +\infty$, and for any k there is $(x^k, y^k) \in (\mathcal{P}_{\theta_k} \cap \mathcal{S}) \backslash \mathcal{P}$. Since $\mathcal{S}$ is finite, we can extract a subsequence such that $(x^{k_l}, y^{k_l}) = (\overline{x}, \overline{y})$ $\forall l$. Then we have $(\overline{x}, \overline{y}) \notin \mathcal{P}$. This contradicts the fact that $(\overline{x}, \overline{y}) \in \mathcal{P}$ following i). $\square$

Remark 2.1 *The assumption that r_θ is an even function is not needed for proving this theorem. More precisely, the theorem still holds when the assumption ii) is replaced by "for any $\theta > 0$, r_θ is decreasing on $(-\infty, 0]$ and is increasing on $[0, +\infty)$. For the zero-norm, since the step function is even, it is natural to consider its approximation r_θ as an even function.*

Theorem 2.1 shows that any optimal solution of the approximate problem (2.8) is in a $\epsilon-$neighboohord of an optimal solution to the original problem (2.6), and the tighter approximation of ℓ_0-norm is, the better approximate solutions are. Moreover, if there is a

finite set $\mathcal{S}$ such that $\mathcal{P}_\theta \cap \mathcal{S} \neq \emptyset \ \forall \theta > 0$, then any optimal solution of the approximate problem (2.8) contained in $\mathcal{S}$ solves also the problem (2.6). By considering the equivalent problem (2.9), we show in the following Corollary that such a set $\mathcal{S}$ exists in several contexts of applications (for instance, in feature selection in SVM).

Corollary 2.1 *Suppose that r is concave on $[0, +\infty)$, K is a polyhedral convex set having at least a vertex and f is concave, bounded below on K. Then Ω_1 defined in (2.9) is also a polyhedral convex set having at least a vertex. Let $\mathcal{V}$ be the vertex set of Ω_1 and*

$$\overline{\mathcal{P}}_\theta = \{(x, y) : \exists z \in \mathbb{R}^n \text{ s.t. } (x, y, z) \in \mathcal{V} \text{ is a global solution of } (2.9)\}.$$

Then $\overline{\mathcal{P}}_\theta \neq \emptyset \ \forall \theta > 0$ and there exists $\theta_0 > 0$ such that $\overline{\mathcal{P}}_\theta \subset \mathcal{P}$, $\forall \theta \geq \theta_0$.

Proof : By the assumptions, we have $\overline{F}_{r_\theta}$ is concave, bounded below on Ω_1, so $\overline{\mathcal{P}}_\theta \neq \emptyset \ \forall \theta > 0$. Let $\mathcal{S} = \{(x, y) : (x, y, z) \in \mathcal{V} \text{ for some } z \in \mathbb{R}^n\}$. By Proposition 2.1, we have $\overline{\mathcal{P}}_\theta \subset \mathcal{P}_\theta \cap \mathcal{S} \ \forall \theta > 0$. Since $\mathcal{V}$ is finite, so is $\mathcal{S}$. The property iii) of Theorem 2.1 implies the existence of $\theta_0 > 0$ such that

$$\overline{\mathcal{P}}_\theta \subset \mathcal{P}_\theta \cap \mathcal{S} \subset \mathcal{P} \ \forall \theta \geq \theta_0.$$

□

Note that the consistency between the solution of the approximate problem and the original problem have been carried out in (Bradley et al., 1998) (resp. (Rinaldi et al., 2010)) for the case where f is concave, bounded below on the polyhedral convex set K and r is the exponential approximation defined in Table 2.1 below (resp. r is the logarithm function and/or ℓ_p-norm ($p < 1$)). Here, besides general results carried out in Theorem 2.1, our Corollary 2.1 gives a much stronger result than those in (Bradley et al., 1998; Rinaldi et al., 2010) where they only ensure that $\overline{\mathcal{P}}_\theta \cap \mathcal{P} \neq \emptyset \ \forall \theta \geq \theta_0$.

Observing that the approximate problem is still nonconvex for which, in general, only local algorithms are available, we are motivated by the study of the consistency between local minimizers of the original and approximate problems. For this purpose, first, we need to describe characteristics of local solutions of these problems.

Proposition 2.2 *i) A point $(x^*, y^*) \in K$ is a local optimum of the problem (2.6) if and only if (x^*, y^*) is a local optimum of the problem*

$$\min\{f(x, y) : (x, y) \in K(x^*)\}, \tag{2.11}$$

where $K(x^*) = \{(x, y) \in K : x_i = 0 \ \forall i \notin supp(x^*)\}$.

ii) If $(x^*, y^*) \in K$ is a local optimum of the problem (2.6) then

$$\langle \overline{x}^*, x - x^* \rangle + \langle \overline{y}^*, y - y^* \rangle \geq 0 \quad \forall (x, y) \in K(x^*), \tag{2.12}$$

for some $(\overline{x}^*, \overline{y}^*) \in \partial f(x^*, y^*)$.

Proof : i) The forward implication is obvious, we only need to prove the backward one. Assume that (x^*, y^*) is a local solution of the problem (2.11). There exists a neighbourhood $\mathcal{V}$ of (x^*, y^*) such that

$$\text{supp}(x^*) \subset \text{supp}(x) \quad \text{and} \quad |f(x,y) - f(x^*,y^*)| < \lambda \quad \forall (x,y) \in \mathcal{V},$$

and

$$f(x^*,y^*) \leq f(x,y) \quad \forall (x,y) \in \mathcal{V} \cap K(x^*).$$

For any $(x,y) \in \mathcal{V} \cap K$, two cases occur:

- If $(x,y) \in K(x^*)$, then $\|x\|_0 = \|x^*\|_0$ and $f(x^*,y^*) \leq f(x,y)$.

- If $(x,y) \notin K(x^*)$, then $\|x^*\|_0 \leq \|x\|_0 - 1$ and $f(x^*,y^*) < f(x,y) + \lambda$.

In both cases, we have $f(x^*,y^*) + \lambda\|x^*\|_0 \leq f(x,y) + \lambda\|x\|_0$. Thus, (x^*,y^*) is a local solution of the problem (2.6).

ii) Since $f = g - h$ is a DC function, (2.11) is a DC program. Therefore, the necessary local condition of the problem (2.11) can be stated by

$$0 \in \partial(g + \chi_{K(x^*)})(x^*,y^*) - \partial h(x^*,y^*),$$

or equivalently, there exists $(\overline{x}^*, \overline{y}^*) \in \partial f(x^*,y^*)$ such that

$$-(\overline{x}^*, \overline{y}^*) \in \partial \chi_{K(x^*)}(x^*,y^*) \Leftrightarrow \langle \overline{x}^*, x - x^* \rangle + \langle \overline{y}^*, y - y^* \rangle \geq 0 \quad \forall (x,y) \in K(x^*),$$

□

As for the characteristics of local solutions of the problem (2.8), we follow the condition (1.4) for a DC program. Writing the problem (2.8) in form of a DC program

$$\min_{x,y} \{F_{r_\theta}(x,y) := G(x,y) - H(x,y)\}, \tag{2.13}$$

with

$$G(x,y) = \chi_K(x,y) + g(x,y) + \lambda \sum_{i=1}^{n} \varphi_\theta(x_i), \;\; H(x,y) = h(x,y) + \lambda \sum_{i=1}^{n} \psi_\theta(x_i). \tag{2.14}$$

Then for a point $(x^*,y^*) \in K$, the necessary local optimality condition (1.4) can be expressed as

$$0 \in \partial G(x^*,y^*) - \partial H(x^*,y^*),$$

which is equivalent to

$$\langle \overline{x}^*, x - x^* \rangle + \langle \overline{y}^*, y - y^* \rangle + \langle \overline{z}^*, x - x^* \rangle \geq 0 \quad \forall (x,y) \in K, \tag{2.15}$$

for some $(\overline{x}^*, \overline{y}^*) \in \partial f(x^*,y^*)$ and $\overline{z}_i^* \in \lambda \partial r_\theta(x_i^*) \;\; \forall i = 1, \ldots, n$.

Now we are able to state consistency results of local optimality.

Theorem 2.2 *Let $\mathcal{L}$ and $\mathcal{L}_\theta$ be the sets of $(x, y) \in K$ satisfying the conditions (2.12) and (2.15) respectively.*

i) Let $\{\theta_k\}$ be a sequence of nonnegative numbers such that $\theta_k \to +\infty$ and $\{(x^k, y^k)\}$ be a sequence such that $(x^k, y^k) \in \mathcal{L}_{\theta_k}, \forall k$. If $(x^k, y^k) \to (x^*, y^*)$, we have $(x^*, y^*) \in \mathcal{L}$.

ii) If K is compact then, for any $\epsilon > 0$, there is $\theta(\epsilon) > 0$ such that

$$\mathcal{L}_\theta \subset \mathcal{L} + B(0, \epsilon) \quad \forall \theta \geq \theta(\epsilon).$$

iii) If there is a finite set $\mathcal{S}$ such that $\mathcal{L}_\theta \cap \mathcal{L} \neq \emptyset, \forall \theta > 0$, then there exists $\theta_0 \geq 0$ such that

$$\mathcal{L}_\theta \cap \mathcal{S} \subset \mathcal{L} \quad \forall \theta \geq \theta_0.$$

Proof : i) By definition, there is a sequence $\{(\overline{x}^k, \overline{y}^k, \overline{z}^k)\}$ such that for all $k = 1, 2, \ldots$

$$(\overline{x}^k, \overline{y}^k) \in \partial f(x^k, y^k), \text{ and } \overline{z}_i^k \in \lambda \partial r_{\theta_k}(x_i^k) \quad i = 1, \ldots, n,$$

$$\langle \overline{x}^k, x - x^k \rangle + \langle \overline{y}^k, y - y^k \rangle + \langle \overline{z}^k, x - x^k \rangle \geq 0 \quad \forall (x, y) \in K. \tag{2.16}$$

For $k = 1, 2, \ldots,$ we have

$$(\overline{x}^k, \overline{y}^k) = (x_g^k, y_g^k) - (x_h^k, y_h^k),$$

where $(x_g^k, y_g^k) \in \partial g(x^k, y^k)$, and $(x_h^k, y_h^k) \in \partial h(x^k, y^k)$.

Since $\{(x^k, y^k)\}$ converges to (x^*, y^*), there is $k_0 \in \mathbb{N}$ and a compact set $\mathcal{S} \subset \mathbb{R}^n \times \mathbb{R}^m$ such that $(x^k, y^k) \in \mathcal{S}$, $\forall k \geq k_0$. It follows by Theorem 24.7 (Rockafellar (1970)) that $\partial g(\mathcal{S}) := \cup_{x \in \mathcal{S}} \partial g(x)$ and $\partial h(\mathcal{S}) := \cup_{x \in \mathcal{S}} \partial h(x)$ are compact sets. Thus, there is an infinite set $\mathcal{K} \subset \mathbb{N}$ such that the sequence $\{(x_g^k, y_g^k)\}_{k \in \mathcal{K}}$ converges to a point $(x_g^*, y_g^*) \in \partial g(\mathcal{S})$ and the sequence $\{(x_h^k, y_h^k)\}_{k \in \mathcal{K}}$ converges to a point $(x_h^*, y_h^*) \in \partial h(\mathcal{S})$. By Theorem 24.4 (Rockafellar (1970)), we have $(x_g^*, y_g^*) \in \partial g(x^*, y^*)$ and $(x_h^*, y_h^*) \in \partial h(x^*, y^*)$. Therefore, the sequence $\{(\overline{x}^k, \overline{y}^k)\}_{k \in \mathcal{K}}$ converges to $(\overline{x}^*, \overline{y}^*) = (x_g^*, y_g^*) - (x_h^*, y_h^*) \in \partial f(x^*, y^*)$.

By Assumption 1 iv), we have $\overline{z}_i^k x_i^k \geq 0 \ \forall i, k$. Moreover, for any $i \in supp(x^*)$, there exist $a_i \leq b_i$ and $k_i \in \mathbb{N}$ such that $0 \notin [a_i, b_i]$ and $x_i^k \in [a_i, b_i]$ for all $k \geq k_i$. By Assumption 1 v), we deduce that $\overline{z}_i^k \to 0$ as $k \to +\infty$.

For arbitrary $(x, y) \in K(x^*)$, (2.16) implies that

$$\begin{aligned} \langle \overline{x}^k, x - x^k \rangle + \langle \overline{y}^k, y - y^k \rangle &\geq \sum_{i \notin supp(x^*)} \overline{z}_i^k x_i^k - \sum_{i \in supp(x^*)} \overline{z}_i^k (x_i - x_i^k) \\ &\geq - \sum_{i \in supp(x^*)} \overline{z}_i^k (x_i - x_i^k) \quad \forall k. \end{aligned}$$

Taking $k \in \mathcal{K}, k \to +\infty$, we get

$$\langle \overline{x}^*, x - x^* \rangle + \langle \overline{y}^*, y - y^* \rangle \geq 0 \quad \forall (x, y) \in K(x^*).$$

Thus, $(x^*, y^*) \in \mathcal{L}$.

ii) and iii) are proved similarly as in Theorem 2.1. □

2.3 DC approximation functions

First, let us mention, in chronological order, the approximation functions proposed in the literature in different contexts, but we do not indicate the related works concerning algorithms using these approximations). The first was concave exponential approximation proposed in (Bradley and Mangasarian, 1998) in the context of feature selection in SVM, and ℓ_p-norm with $0 < p < 1$ for sparse regression (Fu (1998)). Later, the ℓ_p-norm with $p < 0$ was studied in (Rao and Kreutz-Delgado, 1999) for sparse signal recovery, and then the Smoothly Clipped Absolute Deviation (SCAD) (Fan and Li, 2001) in the context of regression, the logarithmic approximation (Weston et al., 2003) for feature selection in SVM, and the Capped-ℓ_1 (Peleg and Meir (2008)) applied on sparse regression.

A common property of these approximations is they are all even, concave increasing functions on $[0, +\infty)$. It is easy to verify that these function satisfy the conditions in Assumption 1 and so they are particular cases of our DC approximation r. More general DC approximation functions are also investigated, e.g., PiL (Le Thi (2012)) that is a (nonconcave) piecewise linear function defined in Table 2.1.

Note that, some of these approximation functions, namely logarithm (log), SCAD and ℓ_p-norm defined by

$$Log : \log(|t| + \epsilon),\ \epsilon > 0, \quad \ell_p : \mathrm{sgn}(p)(|t| + \epsilon)^p,\ 0 \neq p \leq 1, \epsilon > 0; \tag{2.17}$$

$$SCAD : \begin{cases} \gamma|t| & \text{if } 0 \leq |t| \leq \gamma, \\ \frac{-t^2+2a\gamma|t|-\gamma^2}{2(a-1)} & \text{if } \gamma < |t| < a\gamma, \quad a > 1, \gamma > 0 \\ \frac{(a+1)\gamma^2}{2} & \text{if } |t| \geq a\gamma, \end{cases} \tag{2.18}$$

do not directly approximate ℓ_0-norm. But they become approximations of ℓ_0-norm if we multiply them by an appropriate factor (which can be incorporated into the parameter λ), and add an appropriate term (such a procedure does not affect the original problem). The resulting approximation forms of these functions are given in Table 2.1. We see that r_{scad} is obtained by multiplying the SCAD function by $\frac{2}{(a+1)\gamma^2}$ and setting $\theta = \frac{1}{\gamma}$. Similarly, by taking $\theta = \frac{1}{\epsilon}$, we have

$$r_{log}(t) = \frac{\log(|t| + \epsilon)}{\log(1 + 1/\epsilon)} - \frac{\log \epsilon}{\log(1 + 1/\epsilon)}, \text{ and } r_{\ell_p^-}(t) = -\frac{(|t| + \epsilon)^p}{\epsilon^p} + 1.$$

Table 2.1: ℓ_0-approximation functions r and the first DC decomposition φ. The second DC decomposition is $\psi = \varphi - r$.

Approximation	Function r	Function φ
Exp (Bradley and Mangasarian (1998))	$r_{exp}(t) = 1 - e^{-\theta\|t\|}$	$\theta\|t\|$
$\ell_p(0 < p < 1)$(Fu (1998))	$r_{\ell_p^+}(t) = (\|t\| + \epsilon)^{1/\theta}$	$\frac{\epsilon^{1/\theta-1}}{\theta}\|t\|$
$\ell_p(p < 0)$(Rao and Kreutz-Delgado (1999))	$r_{\ell_p^-}(t) = 1 - (1 + \theta\|t\|)^p,\ p < 0$	$-p\theta\|t\|$
Log (Weston et al. (2003))	$r_{log}(t) = \frac{\log(1+\theta\|t\|)}{\log(1+\theta)}$	$\frac{\theta}{\log(1+\theta)}\|t\|$
SCAD (Fan and Li (2001))	$r_{scad}(t) = \begin{cases} \frac{2\theta}{a+1}\|t\| & 0 \le \|t\| \le \frac{1}{\theta} \\ \frac{-\theta^2t^2+2a\theta\|t\|-1}{a^2-1} & \frac{1}{\theta} < \|t\| < \frac{a}{\theta} \\ 1 & \|t\| \ge \frac{a}{\theta} \end{cases}$	$\frac{2\theta}{a+1}\|t\|$
Capped-ℓ_1 (Peleg and Meir (2008))	$r_{cap}(t) = \min\{1, \theta\|t\|\}$	$\theta\|t\|$
PiL (Le Thi, 2012)	$r_{PiL} = \min\left\{1, \max\left\{0, \frac{\theta\|t\|-1}{a-1}\right\}\right\}$	$\frac{\theta}{a-1}\max\left\{\frac{1}{\theta}, \|t\|\right\}$

For using ℓ_p-norm approximation with $0 < p < 1$, we take $\theta = \frac{1}{p}$. Note that $\lim_{\theta\to\infty} |t|^{1/\theta} = s(t)$. To avoid singularity at 0, we add a small $\epsilon > 0$. In this case, we require $\epsilon = \epsilon(\theta)$ satisfying $\lim_{\theta\to\infty} \epsilon(\theta)^{1/\theta} = 0$ to ensure that $\lim_{\theta\to\infty} r_{\ell_p^+}(t) = s(t)$.

All these functions satisfy Assumption 1 (for proving the condition iii) of Assumption 1 we indicate in Table 2.1 a DC decomposition of the approximation functions), so the consistency results stated in Theorems 2.1 and 2.2 are applicable.

Discussion. Except r_{PiL} that is differentiable at 0 with $r'_{PiL}(0) = 0$, the other approximations have the right derivative at 0 depending on the approximation parameter θ. Clearly the tightness of each approximation depends on related parameters. Hence, a suitable way to compare them is using the parameter θ such that their right derivatives at 0 are equal, namely

$$\theta_{cap} = \frac{2}{a+1}\theta_{scad} = \theta_{exp} = -p\theta_{\ell_p^-}.$$

In this case, by simple calculation we have

$$0 \le r_{\ell_p^-} \le r_{exp} \le r_{scad} \le r_{cap} \le s. \tag{2.19}$$

Comparing r_{cap} and r_{scad} with different values θ, we get

$$\begin{cases} 0 \le r_{scad} \le r_{cap} \le s, & \text{if } \frac{2\theta_{scad}}{a+1} \le \theta_{cap} \\ 0 \le r_{cap} \le r_{scad} \le s, & \text{if } \theta_{cap} \le \frac{\theta_{scad}}{a}. \end{cases} \tag{2.20}$$

Inequalities in (2.19) show that, with the parameter θ such that their right derivatives at 0 are equal, r_{scad} and r_{cap} are closer to the step function s than $r_{\ell_p^-}$ and r_{exp}.

As for r_{log} and $r_{\ell_p^+}$, we see that they tend to $+\infty$ when $t \to +\infty$, so they have poor approximation for t large. Whereas, the other approximations are minorants of s and larger t is, closer to s they are. For easier seeing, we depict these approximations in Figure 2.1.

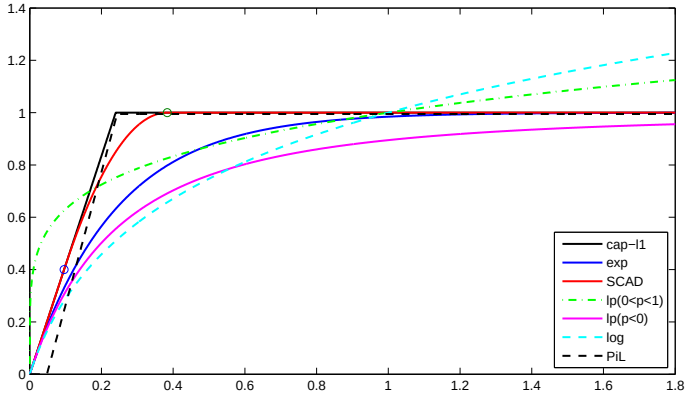


Figure 2.1: Graphs of approximation functions. Except ℓ_p-norm$(0 < p < 1)$ and PiL, the others have the same derivative at 0. Here $\theta_{log} = 10$ for Log, $a = 4$ for SCAD, $p = -2$ for ℓ_p-norm$(p < 0)$. For ℓ_p-norm$(0 < p < 1)$, $\epsilon = 10^{-9}$ and $p = 0.2$. For PiL, $a = 5$ and $\theta_{PiL} = a\theta_{exp}$.

Now, we give a deeper study on Capped-ℓ_1 approximation. Using exact penalty techniques related to ℓ_0-norm developed in (Thiao et al. (2008); Pham Dinh and Le Thi (2014); Le Thi et al. (2014a)) we prove a much stronger result for this approximation, that is the approximation problem (2.8) is equivalent to the original problem with appropriate parameters θ when K is a compact polyhedral convex set (this case quite often occurs in applications, in particular in machine learning contexts). Furthermore, when K is a box, we show (directly, without using the exact penalty techniques) that the Capped-ℓ_1 approximation problem is equivalent to the original problem and we compute an exact value θ_0 such that the equivalence holds for all $\theta > \theta_0$.

2.4 A deeper study on Capped-ℓ_1 approximation problems

2.4.1 Link between approximation and exact penalty approaches

Thanks to exact continuous reformulation via penalty techniques, we shall prove that, with some sparsity-inducing functions, the approximate problem is equivalent to the original problem. First of all, let us recall exact penalty techniques related to ℓ_0-norm (Thiao et al. (2008); Pham Dinh and Le Thi (2014)).

2.4.1.1 Continuous reformulation via exact penalty techniques

Denote by e the vector of ones in the appropriate vector space. We suppose that K is bounded in the variable x, i.e. $K \subset \Pi_{i=1}^{n}[a_i, b_i] \times \mathbb{R}^m$ where $a_i, b_i \in \mathbb{R}$ such that $a_i \leq 0 < b_i$ for $i = 1, ..., n$. Let $c_i := \max\{|x_i| : x_i \in [a_i, b_i]\} = \max\{|a_i|, |b_i|\}$ for $i = 1, ..., n$. Define the binary variable $u_i \in \{0, 1\}$ as

$$u_i = |x_i|_0 = \begin{cases} 1 \text{ if } x_i \neq 0 \\ 0 \text{ if } x_i = 0, \end{cases} \qquad \forall i = 1...n. \tag{2.21}$$

Then (2.1) can be reformulated as

$$\alpha := \inf\{f(x, y) + \lambda e^T u : (x, y) \in K, u \in \{0, 1\}^n, |x_i| \leq c_i u_i,\ i = 1, ..., n\}, \tag{2.22}$$

Let $p : [0, 1]^n \to \mathbb{R}$ be the penalty function defined by

$$p(u) := \sum_{i=1}^{n} \min\{u_i, 1 - u_i\}. \tag{2.23}$$

Then (2.1) can be rewritten as

$$\alpha = \inf\{f(x, y) + \lambda e^T u : (x, y) \in K, u \in [0, 1]^n, |x_i| \leq c_i u_i,\ i = 1, ..., n,\ p(u) \leq 0\}, \tag{2.24}$$

which leads to the corresponding penalized problems (τ being the positive penalty parameter)

$$\alpha(\tau) := \inf\{f(x, y) + \lambda e^T u + \tau p(u) : (x, y) \in K, u \in [0, 1]^n, |x_i| \leq c_i u_i,\ i = 1, ..., n\}. \tag{2.25}$$

It has been shown in (Thiao et al., 2008; Pham Dinh and Le Thi, 2014) that there is $\tau_0 \geq 0$ such that for every $\tau > \tau_0$ problems (2.1) and (2.25) are equivalent, in the sense that they have the same optimal value and $(x^*, y^*) \in K$ is a solution of (2.1) iff there is $u^* \in \{0, 1\}^n$ such that (x^*, y^*, u^*) is a solution of (2.25).

It is clear that if the function $f(x, y)$ is a DC function on K then (2.24) is a DC program.

Let us state now the link between the continuous problem (2.25) and the Capped-ℓ_1 approximation problem.

2.4.1.2 Link between (2.25) and Capped-ℓ_1 approximation problem

The Capped-ℓ_1 approximation is defined by:

$$\Psi_\theta(x) := \sum_{i=1}^{n} r_{cap}(x_i), \forall x = (x_i) \in \mathbb{R}^n, \text{ with } r_{cap}(t) := \min\{\theta |t|, 1\},\ t \in \mathbb{R}. \tag{2.26}$$

We will demonstrate that the resulting approximate problem of (2.1), namely

$$\beta(\theta) := \inf\left\{ f(x,y) + \lambda \sum_{i=1}^{n} r_{cap}\left(x_i\right) : (x,y) \in K \right\} \tag{2.27}$$

is equivalent to the penalized problem (2.25) with suitable values of parameters λ, τ and θ.

Let $M = \max\{c_i : i = 1, \ldots, n\}$, consider the problem (2.25) in the form

$$\alpha(\tau) := \inf\{f(x,y) + \lambda e^T u + \tau p(u) : (x,y) \in K, u \in [0,1]^n, |x_i| \leq M u_i,\ i = 1, \ldots, n\}. \tag{2.28}$$

Let $\varsigma : \mathbb{R} \to \mathbb{R}$ be the function defined by $\varsigma(t) = \min\{t, 1-t\}$. Then $p(u) = \sum_{i=1}^{n} \varsigma(u_i)$ and the problem (2.28) can be rewritten as

$$\alpha(\tau) := \inf\left\{ f(x,y) + \lambda \sum_{i=1}^{n} \left(u_i + \frac{\tau}{\lambda}\varsigma(u_i)\right) : (x,y) \in K, \frac{|x_i|}{M} \leq u_i \leq 1,\ i = 1, \ldots, n \right\}, \tag{2.29}$$

or again

$$\alpha(\tau) := \inf\left\{ f(x,y) + \lambda \sum_{i=1}^{n} \pi\left(u_i\right) : (x,y) \in K, \frac{|x_i|}{M} \leq u_i \leq 1,\ i = 1, \ldots, n \right\} \tag{2.30}$$

where $\pi : \mathbb{R} \to \mathbb{R}$ be the function defined by $\pi(t) := t + \frac{\tau}{\lambda}\varsigma(t)$.

Proposition 2.3 *Let $\theta := \frac{\tau+\lambda}{\lambda M}$. For all $\tau \geq \lambda$ problems (2.30) and (2.27) are equivalent in the following sense: (x^*, y^*) is an optimal solution of (2.27) iff (x^*, y^*, u^*) is an optimal solution of (2.30), where $u_i^* \in \left\{\frac{|x_i^*|}{M}, 1\right\}$ such that $\pi(u_i^*) = r_{cap}(x_i^*)$ for $i = 1, \ldots, n$. Moreover, $\alpha(\tau) = \beta(\theta)$.*

Proof : If (x^*, y^*, u^*) is an optimal solution of (2.30), then u_i^* is an optimal solution of the following problem, for every $i = 1, \ldots, n$

$$\min\left\{ \pi(u_i) : \frac{|x_i^*|}{M} \leq u_i \leq 1 \right\}. \tag{2.31}$$

Since ς is a concave function, so is π. Consequently

$$\min\left\{ \pi(u_i) : \frac{|x_i^*|}{M} \leq u_i \leq 1 \right\} = \min\left\{ \pi\left(\frac{|x_i^*|}{M}\right), \pi(1) \right\} = \min\left\{ \left(1 + \frac{\tau}{\lambda}\right)\frac{|x_i^*|}{M}, 1 \right\} = r_{cap}(x_i^*).$$

For an arbitrary $(x,y) \in K$, we will show that

$$f(x^*, y^*) + \lambda \sum_{i=1}^{n} r_{cap}(x_i^*) \leq f(x,y) + \lambda \sum_{i=1}^{n} r_{cap}(x_i). \tag{2.32}$$

By the assumption that (x^*, y^*, u^*) is an optimal solution of (2.30), we have

$$f(x^*, y^*) + \lambda \sum_{i=1}^{n} \pi(u_i^*) \leq f(x, y) + \lambda \sum_{i=1}^{n} \pi(u_i) \tag{2.33}$$

for any feasible solution (x, y, u) of (2.30). Let

$$u_i^x \in \arg\min \left\{ \pi(\xi) \ : \xi \in \left\{ \frac{|x_i|}{M}, 1 \right\} \right\} \subset \arg\min \left\{ \pi(\xi) : \frac{|x_i|}{M} \leq \xi \leq 1 \right\},$$

for all $i = 1, \ldots, n$. Then (x, y, u^x) is a feasible solution of (2.28) and

$$\pi(u_i^x) = \min \left\{ \pi(\xi) : \frac{|x_i|}{M} \leq \xi \leq 1 \right\} \ = r_{cap}(x_i), \quad \forall i = 1, \ldots, n.$$

Combining (2.33) in which u_i is replaced by u_i^x and the last equation we get (2.32), which implies that (x^*, y^*) is an optimal solution of (2.27).

Conversely, if (x^*, y^*) is a solution of (2.27), and let $u_i^* \in \left\{ \frac{|x_i^*|}{M}, 1 \right\}$ such that $\pi(u_i^*) = r_{cap}(x_i^*)$ for $i = 1, \ldots, n$. Then (x^*, y^*, u^*) is a feasible solution of (2.30) and for an arbitrary feasible solution (x, y, u) of (2.30), we have

$$\begin{aligned} f(x, y) + \lambda \sum_{i=1}^{n} \pi(u_i) &\geq f(x, y) + \lambda \sum_{i=1}^{n} r_{cap}(x_i) \\ \geq f(x^*, y^*) + \lambda \sum_{i=1}^{n} r_{cap}(x_i^*) &= f(x^*, y^*) + \lambda \sum_{i=1}^{n} \pi(u_i^*). \end{aligned}$$

Thus, (x^*, y^*, u^*) is an optimal solution of (2.30). The equality $\alpha(\tau) = \beta(\theta)$ is immediately deduced from the equality $\pi(u_i^*) = r_{cap}(x_i^*)$. □

We conclude from the above results that for $\theta = \frac{\tau+\lambda}{\lambda M}$ with $\tau > \max\{\lambda, \tau_0\}$, or equivalently $\theta > \theta_0 := \max\{\frac{2}{M}, \frac{\tau_0+\lambda}{\lambda M}\}$, the approximate problem (2.27) is equivalent to the original problem (2.1). The result justifies the goodness of the Capped-ℓ_1 approximation studied in Section 2.3 above.

2.4.2 A special case: link between the original problem (2.1) and Capped-ℓ_1 approximation problem

In particular, for a special structure of K, we get the following result.

Proposition 2.4 *Suppose that* $K = \prod_{i=1}^{n}[-\underline{l}_i, \overline{l}_i] \times Y$ $(0 \le \underline{l}_i, \overline{l}_i \le +\infty\ \forall i, Y \subset \mathbb{R}^m)$ *and* $\kappa > 0$ *is a constant satisfying*

$$|f(x,y) - f(x',y)| \le \kappa \|x - x'\|_2 \quad \forall (x,y),(x',y) \in K, \|x - x'\|_0 \le 1. \tag{2.34}$$

Then for $\theta > \frac{\kappa}{\lambda}$, *the problems* (2.1) *and* (2.27) *are equivalent.*

Proof : We observe that if $(x,y) \in K$ such that $0 < |x_{i_0}| < \frac{1}{\theta}$ for some i_0, let $(x',y) \in K$ determined by $x'_i = x_i\ \forall i \neq i_0$ and $x'_{i_0} = 0$, then

$$f(x,y) + \lambda\Phi(x) > f(x',y) + \lambda\Phi(x'),$$

where $\Phi(x) = \sum_{i=1}^{n} r_{cap}(x_i)$. Indeed, this inequality follows the facts that

$$|f(x,y) - f(x',y)| \le \kappa\|x - x'\|_2 = \kappa|x_{i_0}|$$

and

$$\Phi(x) - \Phi(x') = r_{cap}(x_{i_0}) = \theta|x_{i_0}| > \frac{\kappa}{\lambda}|x_{i_0}|.$$

For $x \in \mathbb{R}^n$, we define $t^x \in \mathbb{R}^n$ by $t^x_i = 0$ if $|x_i| < \frac{1}{\theta}$ and $t^x_i = x_i$ otherwise. By applying the above observation, for any $(x,y) \in K$, we have

$$f(x,y) + \lambda\Phi(x) \ge f(t^x,y) + \lambda\Phi(t^x).$$

The equality holds iff $|x_i| \ge \frac{1}{\theta}\ \forall i \in \mathrm{supp}(x)$.

Therefore, if (x^*,y^*) is a solution of (2.27), we have $|x^*_i| \ge \frac{1}{\theta}\ \forall i \in \mathrm{supp}(x^*)$. Then, for any $(x,y) \in K$,

$$f(x,y) + \lambda\|x\|_0 \ge f(x,y) + \lambda\Phi(x) \ge f(x^*,y^*) + \lambda\Phi(x^*) = f(x^*,y^*) + \lambda\|x^*\|_0.$$

This means that (x^*,y^*) is a solution of (2.1).

Conversely, assume that (x^*,y^*) is a solution of (2.1). Then for any $(x,y) \in K$, we have

$$\begin{aligned} f(x,y) + \lambda\Phi(x) &\ge f(t^x,y) + \lambda\Phi(t^x) = f(t^x,y) + \lambda\|t^x\|_0 \\ &\ge f(x^*,y^*) + \lambda\|x^*\|_0 \ge f(x^*,y^*) + \lambda\Phi(x^*). \end{aligned}$$

Thus, (x^*,y^*) is a solution of (2.27). □

For the problem of feature selection in SVM, we consider the loss function

$$f(x,b) = (1-\lambda)\left(\frac{1}{N_A}\|\max\{0, -Ax + eb + e\}\|_1 + \frac{1}{N_B}\|\max\{0, Bx - eb + e\}\|_1\right),$$

(cf. Sect. 2.6 for definition of notations).

It is easy to prove that for $u \in \mathbb{R}^n$, $\iota \in \mathbb{R}$ and $i \in \{1, \ldots, n\}$, we have

$$|\max\{0, \langle u, x\rangle + \iota\} - \max\{0, \langle u, x'\rangle + \iota\}| \leq |u|_i |x_i - x'_i|,$$

for all $x, x' \in \mathbb{R}^n$ such that $x_j = x'_j$ $\forall j \neq i$. Therefore, for $\kappa = (1 - \lambda) \max\limits_{i=1,\ldots,n} \left\{ \frac{1}{N_A} \sum\limits_{k=1}^{N_A} |A_{ki}| + \frac{1}{N_B} \sum\limits_{l=1}^{N_B} |B_{li}| \right\}$, we have

$$|f(x, b) - f(x', b)| \leq \kappa \|x - x'\|, \quad \forall b \in \mathbb{R}, \forall x, x' \in \mathbb{R}^n \text{ s.t. } \|x - x'\|_0 \leq 1.$$

By virtue of Proposition 2.4, in the case of feature selection in SVM, for $\theta > \theta^* := \frac{\kappa}{\lambda}$, the problems (2.1) and (2.27) are equivalent.

Remark 2.2 *Proposition 2.4 gives a very important result since it permits to tackle a class of box constrained sparse optimization problems including feature selection in SVM instances by solving their equivalent (continuous) DC program of the form (2.27). Global approaches such as Branch and Bound / interval analysis methods can be investigated for (2.27) in which efficient local approaches such as DCA is a suitable way for computing tight upper bounds. This is especially useful to interval analysis based methods such as IbexOpt (Trombettoni, 2011) where upper bounding procedures consist of randomly picking a point inside the extracted feasible box. On another hand, thanks to Proposition 2.4 we can improve the bounds of the variables by a simple procedure. Indeed, for an optimal solution $x^* \in \mathbb{R}^n$ of (2.27) we have: either $\|x_i^*\| > \frac{1}{\theta}$ or $x_i^* = 0$ for any $i = 1, \ldots, n$. Thus for $x_i \in [-\underline{l}_i, \overline{l}_i]$ ($0 \leq \underline{l}_i, \overline{l}_i \leq +\infty$; $i = 1, \ldots, n$) if $\underline{l}_i < \frac{1}{\theta}$ (resp. $\overline{l}_i < \frac{1}{\theta}$) we can reduce the interval $[-\underline{l}_i, \overline{l}_i]$ to $[0, \overline{l}_i]$ (resp. $[-\underline{l}_i, 0]$). And if $\underline{l}_i, \overline{l}_i < \frac{1}{\theta}$ we can reduce the interval $[-\underline{l}_i, \overline{l}_i]$ to $\{0\}$. It is worth noting that improving the bounds of the variables is the heart of interval-based solvers since it has strong impact on computing upper and lower bounds.*

2.4.3 Extension to other approximations

Proposition 2.5 *i) Suppose that σ is a function on $\mathbb{R}$ satisfying*

$$r_{cap}(t) \leq \sigma(t) \leq s(t) = \begin{cases} 0, & \text{if } t = 0, \\ 1, & \text{otherwise,} \end{cases}$$

for some $\theta_{cap} > \theta_0$. Then, the problems (2.1) and

$$\inf\{f(x, y) + \lambda \sum_{i=1}^{n} \sigma(x_i) : (x, y) \in K\} \tag{2.35}$$

are equivalent.
ii) In particular, if $\theta_{scad} > a\theta_0$ then for all $\tau \geq \lambda$ the approximate problem

$$\inf\{f(x, y) + \lambda \sum_{i=1}^{n} r_{scad}(x_i) : (x, y) \in K\}$$

is equivalent to (2.1).

Proof : As discussed before, since $\theta_{cap} > \theta_0$, the problems (2.1) and (2.27) are equivalent. Moreover, if (x^*, y^*) is a common solution then

$$f(x^*, y^*) + \lambda \sum_{i=1}^{n} r_{cap}(x_i^*) = f(x^*, y^*) + \lambda \|x^*\|_0.$$

Then i) is trivial by the fact that

$$f(x, y) + \lambda \sum_{i=1}^{n} r_{cap}(x_i) \leq f(x, y) + \lambda \sum_{i=1}^{n} \sigma(x_i) \leq f(x, y) + \lambda \|x\|_0, \quad \forall (x, y).$$

ii) is a direct consequence of i) and Propositions 2.3 and (2.20). □

2.5 DCA for solving the problem (2.8)

In this section, we will omit the parameter θ when this does not cause any ambiguity.

Usual sparsity-inducing functions are concave, increasing on $[0, +\infty)$. Therefore, first we present three variants of DCA for solving the problem (2.8) when r is concave on $[0, +\infty)$. We also suppose that r has the right derivative at 0, denoted by $r'(0)$, so $\partial(-r)(0) = \{-r'(0)\}$.

First, we consider the approximate problem (2.8).

2.5.1 The first DCA scheme for solving the problem (2.8)

We propose the following DC decomposition of r:

$$r(t) = \eta|t| - (\eta|t| - r(t)) \quad \forall t \in \mathbb{R}, \tag{2.36}$$

where η is a positive number such that $\psi(t) = \eta|t| - r(t)$ is convex. The next result gives a sufficient condition for the existence of such a η.

Proposition 2.6 *Suppose that r is a concave function on $[0, +\infty)$ and the (right) derivative at 0, $r'(0)$, is well-defined. Let $\eta \geq r'(0)$. Then $\psi(t) = \eta|t| - r(|t|)$ is a convex function on $\mathbb{R}$.*

Proof : Since r is concave on $[0,+\infty)$, the function $\eta|t|-r(t)$ is convex on $(0,+\infty)$ and on $(-\infty,0)$. Hence it suffices to prove that for any $t_1>0, t_2<0$ and $\alpha,\beta\in(0,1)$ such that $\alpha+\beta=1$, we have

$$\psi\left(\alpha t_1+\beta t_2\right)\leq\alpha\psi(t_1)+\beta\psi(t_2). \tag{2.37}$$

Without loss of generality, we assume that $\alpha|t_1|\geq\beta|t_2|$. Then (2.37) is equivalent to

$$\eta(\alpha|t_1|-\beta|t_2|)-2r\left(\alpha|t_1|-\beta|t_2|\right)\leq\eta(\alpha|t_1|+\beta|t_2|)-\alpha r(|t_1|)-\beta r(|t_2|)$$

which can be equivalently written as

$$\alpha r(|t_1|)+\beta r(|t_2|)-r\left(t_0\right)\leq 2\eta\beta|t_2|, \tag{2.38}$$

where $t_0=\alpha|t_1|-\beta|t_2|\geq 0$. Let $\mu\in\mathbb{R}$ such that $-\mu\in\partial(-r(t_0))$. Since r is concave on $[0,+\infty)$, we have

$$\alpha r(|t_1|)+\beta r(|t_2|)-r\left(t_0\right)\leq r\left(\alpha|t_1|+\beta|t_2|\right)-r\left(t_0\right)\leq 2\mu\beta|t_2|.$$

Hence (2.38) holds when $\mu\leq\eta$. By the concavity of r, we have

$$r\left(\frac{t_0}{2}\right)\leq r(0)+r'(0)\frac{t_0}{2},\quad\text{and}\quad r\left(\frac{t_0}{2}\right)\leq r(t_0)-\mu\frac{t_0}{2},$$

therefore

$$(z-r'(0))t_0\leq r(0)+r(t_0)-2r\left(\frac{t_0}{2}\right)\leq 0.$$

This and the condition $r'(0)\leq\eta$ imply that $\mu\leq r'(0)\leq\eta$. The proof is then complete. □

With $\eta\geq r'(0)$, a DC formulation of the problem (2.8) is given by

$$\min_{x,y}\{F_r(x,y):=G_1(x,y)-H_1(x,y)\}, \tag{2.39}$$

where

$$G_1(x,y)=\chi_K(x,y)+g(x,y)+\lambda\eta\|x\|_1,\ H_1(x,y)=h(x,y)+\lambda\sum_{i=1}^{n}\left(\eta|x_i|-r(x_i)\right),$$

and g,h are DC components of f.

By the definition $\psi(t)=\eta|t|-r(t)\ \forall t\in\mathbb{R}$, we have

$$\partial\psi(t)=\eta+\partial(-r)(t)\text{ if }t>0,\ -\eta-\partial(-r)(-t)\text{ if }t<0,\ [-\eta+r'(0),\eta-r'(0)]\text{ if }t=0. \tag{2.40}$$

Following the generic DCA scheme, DCA applied on (2.39) is given by Algorithm 2.1 below.

Table 2.2: Choice of η and expression of $\overline{z}_i^k \in \lambda\partial\psi(x_i^k)$ in Algorithm 2.1 and related works.

r	η	$\overline{z}_i^k \in \lambda\partial\psi(x_i^k)$	Related works	Context
r_{exp}	θ	$\mathrm{sgn}(x_i^k)\lambda\theta\left(1-e^{-\theta\lvert x_i^k\rvert}\right)$	(Le Thi et al., 2008) (Ong and Le Thi, 2012)	Feature selection in SVMs Learning sparse classifiers
$r_{\ell_p^+}$	$\frac{\epsilon^{1/\theta-1}}{\theta}$	$\mathrm{sgn}(x_i^k)\frac{\lambda}{\theta}\left[\epsilon^{1/\theta-1}-(\lvert x_i^k\rvert+\epsilon)^{1/\theta-1}\right]$		
$r_{\ell_p^-}$	$-p\theta$	$-\mathrm{sgn}(x_i^k)\lambda p\theta\left[1-(1+\theta\lvert x_i^k\rvert)^{p-1}\right]$		
r_{log}	$\frac{\theta}{\log(1+\theta)}$	$\mathrm{sgn}(x_i^k)\frac{\lambda\theta^2\lvert x_i^k\rvert}{\log(1+\theta)(1+\theta\lvert x_i^k\rvert)}$		
r_{scad}	$\frac{2\theta}{a+1}$	$\begin{cases} 0 & \lvert x_i^k\rvert \le \frac{1}{\theta} \\ \mathrm{sgn}(x_i^k)\frac{2\lambda\theta(\theta\lvert x_i^k\rvert-1)}{a^2-1} & \frac{1}{\theta} < \lvert x_i^k\rvert < \frac{a}{\theta} \\ \mathrm{sgn}(x_i^k)\frac{2\lambda\theta}{a+1} & \text{otherwise} \end{cases}$	(Le Thi et al., 2009b)	Feature selection in SVMs
r_{cap}	θ	$\begin{cases} 0 & \lvert x_i^k\rvert \le \frac{1}{\theta} \\ \mathrm{sgn}(x_i^k)\lambda\theta & \text{otherwise} \end{cases}$	(Ong and Le Thi, 2012)	Learning sparse classifiers

Algorithm 2.1 DCA for solving (2.8) (DCA1)

Initialize $(x^0, y^0) \in K$, $k \leftarrow 0$

repeat

1. Compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$ and $\overline{z}_i^k \in \lambda\partial\psi(x_i^k)$ $\forall i = 1, \ldots, n$ via (2.40).
2. Compute

$$(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)\in K} \left\{g(x,y) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle + \lambda\eta\|x\|_1 - \langle \overline{z}^k, x\rangle\right\}$$

3. $k \leftarrow k+1$.

until Stopping criterion

Instances of Algorithm 1 can be found in our previous works (Le Thi et al., 2008, 2009b; Ong and Le Thi, 2012) using exponential concave, SCAD or Capped$-\ell_1$ approximations (see Table 2.2). Note that for usual sparsity-inducing functions given in Table 2.2, this DC decomposition is nothing but that given in Table 2.1, i.e. $\varphi(t) = \eta|t|$.

Now we consider the approximate problem (2.9) and introduce a DCA scheme that includes all standard algorithms of reweighted-ℓ_1-type for sparse optimization problem (2.6).

2.5.2 DCA2 - Relation with reweighted-ℓ_1 procedure

The problem (2.9) can be written as a DC program as follows

$$\min_{x,y,z}\{\overline{F}_r(x,y,z) := G_2(x,y,z) - H_2(x,y,z)\}, \tag{2.41}$$

where

$$G_2(x,y,z) = \chi_{\Omega_1}(x,y,z) + g(x,y),\ \ H_2(x,y,z) = h(x,y) + \lambda\sum_{i=1}^{n}(-r)(z_i),$$

and g, h are DC components of f as stated in (2.7).

Assume that $(x^k, y^k, z^k) \in \Omega_1$ is the current solution at iteration k. DCA applied to DC program (2.41) updates $(x^{k+1}, y^{k+1}, z^{k+1}) \in \Omega_1$ via two steps:

- Step 1: compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$, and $\overline{z}_i^k \in \lambda\partial(-r)(z_i^k) \;\; \forall i = 1, \ldots, n$.
- Step 2: compute

$$\begin{aligned}(x^{k+1}, y^{k+1}, z^{k+1}) &\in \arg\min \left\{G_2(x, y, z) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle - \langle \overline{z}^k, z\rangle\right\} \\ &= \arg\min_{(x,y,z)\in\Omega_1} \left\{g(x, y) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle + \langle -\overline{z}^k, z\rangle\right\}.\end{aligned}$$

Since r is increasing, we have $-\overline{z}^k \geq 0$. Thus, updating $(x^{k+1}, y^{k+1}, z^{k+1})$ can be done as follows

$$\begin{cases}(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)\in K} \left\{g(x, y) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle + \langle -\overline{z}^k, |x|\rangle\right\} \\ z_i^{k+1} = |x_i^{k+1}| \;\; \forall i.\end{cases}$$

DCA for solving the problem (2.9) can be described as in Algorithm 2.2 below.

Algorithm 2.2 DCA for solving (2.9) (DCA2)

Initialize $(x^0, y^0, z^0) \in \Omega_1$, $k \leftarrow 0$

repeat

1. Compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$, $\overline{z}_i^k \in -\lambda\partial(-r)(z_i^k) \;\; \forall i = 1, \ldots, n$.
2. Compute

$$\begin{aligned}&(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)\in K} \left\{g(x, y) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle + \langle \overline{z}^k, |x|\rangle\right\} \\ &z_i^{k+1} = |x_i^{k+1}| \; \forall i = 1, \ldots, n.\end{aligned}$$

3. $k \leftarrow k + 1$.

until Stopping criterion

If the function f in (2.6) is convex, we can choose DC components of f as $g = f$ and $h = 0$. Then $(\overline{x}^k, \overline{y}^k) = 0 \; \forall k$. In this case, the step 2 in Algorithm 2.2 becomes

$$(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)\in K} \left\{f(x, y) + \sum_{i=1}^{n} \overline{z}_i^k |x_i|\right\}. \tag{2.42}$$

We see that the problem (2.42) has the form of a ℓ_1-regularization problem but with different weights on components of $|x_i|$. So Algorithm 2.2 iteratively solves the weighted–ℓ_1 problem

Table 2.3: Expression of $\overline{z}_i^k$ in Algorithm 2.2 and relation with reweighted-ℓ_1 algorithms.

Function r	expression of $\overline{z}_i^k$	Related works	Context
r_{exp}	$\lambda\theta e^{-\theta z_i^k}$	SLA (Bradley and Mangasarian (1998))	Feature selection in SVMs
$r_{\ell_p^+}$	$\frac{\lambda}{\theta(z_i^k+\epsilon)^{1-1/\theta}}$	Adaptive Lasso (Zou (2006))	Linear regression
$r_{\ell_p^-}$	$-\lambda p\theta(1+\theta z_i^k)^{p-1}$		
r_{scad}	$\begin{cases} \frac{2\lambda\theta}{a+1} & \text{if } z_i^k \le \frac{1}{\theta} \\ 0 & \text{if } z_i^k \ge \frac{a}{\theta} \\ \frac{\lambda\theta(a-\theta z_i^k)}{a^2-1} & \text{otherwise} \end{cases}$	LLA (Local Linear Approximation) (Zou and Li (2008))	
r_{cap}	$\begin{cases} \lambda\theta & \text{if } z_i^k \le 1/\theta \\ 0 & \text{otherwise} \end{cases}$	Two-stage ℓ_1 (Zhang (2009))	
r_{log}	$\frac{\lambda\theta}{\log(1+\theta)(1+\theta z_i^k)}$	Adaptive Lasso (Zou (2006)), Reweighted ℓ_1 (Candes et al. (2008)); AROM (Weston et al. (2003))	Sparse signal reconstruction; Feature selection in SVMs

(2.42) with an update of the weights $\overline{z}_i^k$ at each iteration k. The expression of weights $\overline{z}_i^k$ according to approximation functions are given in Table 2.3.

The update rule (2.42) covers standard algorithms of reweighted–ℓ_1–type for sparse optimization problem (2.6) (see Table 2.3). Some algorithms such as the two–stage ℓ_1 (Zhang (2009)) and the adaptive Lasso (Zou (2006)) only run in a few iterations (typically two iterations) and their reasonings bear a heuristic character. The reweighted–ℓ_1 algorithm proposed in (Candes et al., 2008) lacks of theoretical justification for the convergence.

Next, we introduce a slight perturbation of the formulation (2.8) and develop the third DCA scheme that includes existing algorithms of reweighted–ℓ_2–type for sparse optimization problem (2.6).

2.5.3 DCA3 - Relation with reweighted-ℓ_2 procedure

To avoid the singularity at 0 of the function $r(t^{1/2}), t \ge 0$, we add $\epsilon > 0$ and consider the perturbation problem of (2.8) which is defined by

$$\begin{cases} \min_{x,y} & \tilde{F}_r(x,y) := f(x,y) + \lambda \sum_{i=1}^n r((|x_i|^2+\epsilon)^{1/2}) \\ s.t. & (x,y) \in K, \end{cases} \quad \epsilon > 0. \tag{2.43}$$

The problem (2.43) is equivalent to

$$\min_{(x,y,z)\in\Omega_2} \hat{F}_r(x,y,z) := f(x,y) + \lambda \sum_{i=1}^n r((z_i+\epsilon)^{1/2}), \tag{2.44}$$

where $\Omega_2 = \{(x, y, z) : (x, y) \in K; \quad |x_i|^2 \leq z_i \ \forall i\}$. The last problem is a DC program of the form

$$\min_{x,y,z}\{\hat{F}_r(x, y, z) := G_3(x, y, z) - H_3(x, y, z)\}, \tag{2.45}$$

where

$$G_3(x, y, z) = \chi_{\Omega_2}(x, y, z) + g(x, y), \ H_3(x, y, z) = h(x, y) + \lambda \sum_{i=1}^{n}(-r)((z_i + \epsilon)^{1/2}),$$

and g, h are DC components of f as stated in (2.7). Note that, since the functions r and $(t+\epsilon)^{1/2}$ are concave, increasing on $[0, +\infty)$, $(-r)((t+\epsilon)^{1/2})$ is a convex function on $[0, +\infty)$.

Let $(x^k, y^k, z^k) \in \Omega_2$ be the current solution at iteration k. DCA applied to DC program (2.45) updates $(x^{k+1}, y^{k+1}, z^{k+1}) \in \Omega_2$ via two steps:

- Step 1: compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$, and $\overline{z}_i^k \in \frac{\lambda}{2(z_i^k+\epsilon)^{1/2}}\partial(-r)((z_i^k + \epsilon)^{1/2}) \quad \forall i = 1, \ldots, n$.
- Step 2: compute

$$\begin{aligned}(x^{k+1}, y^{k+1}, z^{k+1}) &\in \arg\min \left\{G_3(x, y, z) - \langle\overline{x}^k, x\rangle - \langle\overline{y}^k, y\rangle - \langle\overline{z}^k, z\rangle\right\} \\ &= \arg\min_{(x,y,z)\in\Omega_2} \left\{g(x, y) - \langle\overline{x}^k, x\rangle - \langle\overline{y}^k, y\rangle + \langle-\overline{z}^k, z\rangle\right\}\end{aligned}$$

Since r is increasing, we have $-\overline{z}^k \geq 0$. Thus, updating $(x^{k+1}, y^{k+1}, z^{k+1})$ can be done as follows

$$\begin{cases}(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)\in K} \left\{g(x, y) - \langle\overline{x}^k, x\rangle - \langle\overline{y}^k, y\rangle + \sum_{i=1}^{n}(-\overline{z}_i^k)x_i^2\rangle\right\} \\ z_i^{k+1} = |x_i^{k+1}|^2 \quad \forall i = 1, \ldots, n.\end{cases}$$

DCA for solving the problem (2.44) can be described as in Algorithm 2.3 below.

If the function f in (2.6) is convex, then, as before, we can choose DC components of f as $g = f$ and $h = 0$. Hence, in the step 1 of Algorithm 2.3, we have $(\overline{x}^k, \overline{y}^k) = 0 \ \forall k$. In this case, the step 2 in Algorithm 2.3 becomes

$$(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)\in K} \left\{f(x, y) + \sum_{i=1}^{n} \overline{z}_i^k x_i^2\right\}. \tag{2.46}$$

Thus, each iteration of Algorithm 2.3 solves a weighted-ℓ_2 optimization problem. The expression of weights $\overline{z}_i^k$ according to approximation functions are given in Table 2.4.

Note that if $\epsilon = 0$ then the update rule (2.46) encompasses standard algorithms of reweighted-ℓ_2 type for finding sparse solution (see Table 2.4). However, when $\epsilon = 0$ the (right) derivative at 0 of $r(t^{1/2})$ is not well-defined, that is why we take $\epsilon > 0$ in our algorithm. Note also that, in LQA and FOCUSS, if at an iteration k one has $x_i^k = 0$ then $x_i^l = 0$ for all $l \geq k$, by the way these algorithms may converge prematurely to bad solutions.

Algorithm 2.3 DCA for solving (2.44) (DCA3)

Initialize $(x^0, y^0, z^0) \in \Omega_2$, $k \leftarrow 0$

repeat

1. Compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$, $\overline{z}_i^k \in \frac{-\lambda}{2(z_i^k+\epsilon)^{1/2}} \partial(-r)(z_i^k + \epsilon)^{1/2})$ $\forall i = 1, \ldots, n$.
2. Compute

$$(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)\in K} \left\{ g(x,y) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle + \sum_{i=1}^n \overline{z}_i^k x_i^2 \right\},$$
$$z_i^{k+1} = |x_i^{k+1}|^2 \quad \forall i = 1, \ldots, n.$$

3. $k \leftarrow k + 1$.

until Stopping criterion

Table 2.4: Expression of $\overline{z}_i^k$'s in Algorithm 2.3 and relation with reweighted-ℓ_2 algorithms.

Function r	weight $\overline{z}_i^k (t_i^k = (z_i^k + \epsilon)^{1/2})$	Related works	Context
r_{exp}	$\frac{\lambda\theta}{2} \frac{e^{-\theta t_i^k}}{t_i^k}$		
$r_{\ell_p^+}$	$\frac{\lambda}{2\theta (t_i^k)^{2-\frac{1}{\theta}}}$	FOCUSS (Gorodnitsky and Rao, 1997),	Sparse signal
$r_{\ell_p^-}$	$\frac{-\lambda p \theta^p}{2t_i^k (\frac{1}{\theta} + t_i^k)^{1-p}}$	(Rao and Kreutz-Delgado, 1999; Rao et al., 2003);	reconstruction
r_{log}	$\frac{\lambda}{2\log(1+\theta)} \frac{1}{t_i^k(\frac{1}{\theta} + t_i^k)}$	IRLS (Chartrand and Yin, 2008)	
r_{cap}	$\begin{cases} \frac{\lambda\theta}{2t_i^k} & \text{if } \lvert t_i^k\rvert \le \frac{1}{\theta} \\ 0 & \text{otherwise} \end{cases}$		
r_{scad}	$\begin{cases} \frac{\lambda\theta}{(a+1)t_i^k} & \text{if } t_i^k \le \frac{1}{\theta} \\ 0 & \text{if } t_i^k \ge \frac{a}{\theta} \\ \frac{\lambda\theta(-\theta t_i^k + a)}{(a^2-1)t_i^k} & \text{otherwise} \end{cases}$	LQA (Fan and Li (2001); Zou and Li (2008))	Linear regression

2.5.4 Discussion on the three DCA based algorithms 2.1, 2.2 and 2.3

Algorithm 2.1 seems to be the most interesting in the sense that it addresses directly the problem (2.8) and does not need the additional variable z, then the subproblem has less constraints than that in Algorithms 2.2 and 2.3. Moreover, the DC decomposition (2.36) is more suitable since it results, in several cases, in a DC polyhedral program where both DC components are polyhedral convex (for instance, in feature selection in SVM with the approximations r_{scad}, r_{cap}) for which Algorithm 2.1 enjoys interesting convergence properties.

Algorithms 2.2 and 2.3 are based on two different formulations of the problem (2.8). In (2.9), we have linear constraints $|x|_i \leq z_i,\ i = 1, \ldots, n$ that lead to the subproblem of weighted–ℓ_1 type. Whereas, in (2.43), quadratic constraints $|x|_i^2 \leq z_i,\ i = 1, \ldots, n$ result to the subproblem of weighted–ℓ_2 type. With second order terms in subproblems, Algorithm 2.3 is, in general, more expensive than Algorithms 2.1 and 2.2. We also see that Algorithms 2.1 and 2.2 possess nicer convergence properties than Algorithm 2.3. Both Algorithms 2.1 and 2.2 have finite convergence when the corresponding DC programs are polyhedral DC. While (2.43) can't be a polyhedral DC program because the set Ω_2 and the functions $r((t+\epsilon)^{1/2})$ are not polyhedral convex.

To compare the sparsity of solutions given by the algorithms, we consider the subproblems in Algorithms 2.1, 2.2, and 2.3 which have the form

$$\min_{(x,y)\in K} \left\{ g(x,y) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle + \lambda \sum_{i=1}^{n} \nu(x_i, x_i^k) \right\}$$

where $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$,

$$\nu(x_i, x_i^k) = \begin{cases} \nu_1(x_i, x_i^k) = \eta|x_i| - \operatorname{sgn}(x_i^k)(\eta - \overline{z}_i^k)x_i + C_i^k & \text{for Algorithm 2.1} \\ \nu_2(x_i, x_i^k) = \overline{z}_i^k|x_i| + C_i^k & \text{for Algorithm 2.2} \\ \nu_3(x_i, x_i^k) = \frac{\overline{z}_i^k}{2|x_i^k|}|x_i|^2 + \frac{1}{2}\overline{z}_i^k|x_i^k| + C_i^k & \text{for Algorithm 2.3,} \end{cases}$$

with $\overline{z}_i^k \in -\partial(-r)(|x_i^k|)$, $C_i^k = r(x_i^k) - \overline{z}_i^k|x_i^k|$ and $\eta = r'(0)$.

All three functions ν_1, ν_2 and ν_3 attain minimum at 0 and encourage solutions to be zero. Denote by $\nu'_-(t)$ and $\nu'_+(t)$ the left and right derivative at t of ν respectively. We have

$$\begin{aligned} &\nu'_{1,-}(0, x_i^k) = -2\eta + \overline{z}_i^k, \quad \nu'_{2,-}(0, x_i^k) = -\overline{z}_i^k, \quad \nu'_{3,-}(0, x_i^k) = 0, \\ &\nu'_{1,+}(0, x_i^k) = \overline{z}_i^k, \quad \nu'_{2,+}(0, x_i^k) = \overline{z}_i^k, \quad \nu'_{3,+}(0, x_i^k) = 0. \end{aligned}$$

We also have $\eta \geq \overline{z}_i^k$ by the concavity of r on $[0, +\infty)$. Observe that if the range $[\nu'_-(0), \nu'_+(0)]$ is large, it encourages more sparsity. Intuitively, the values $\nu'_-(0)$ and $\nu'_+(0)$ reflect the slope of ν at 0, and if the slope is high, it forces solution to be zero. Here we have

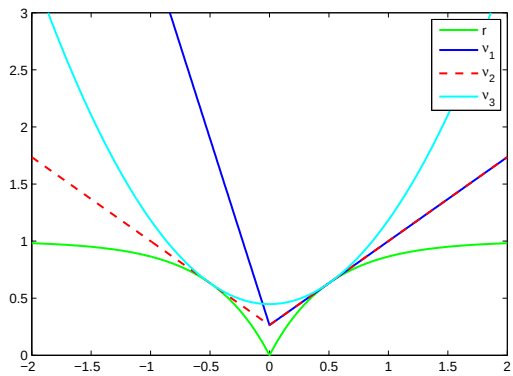


Figure 2.2: Graphs of functions: $r = 1 - e^{-2|x|}$, ν_1, ν_2 and ν_3 with $x^k = 0.5$.

$[\nu'_{3,-}(0, x_i^k), \nu'_{3,+}(0, x_i^k)] \subset [\nu'_{2,-}(0, x_i^k), \nu'_{2,+}(0, x_i^k)] \subset [\nu'_{1,-}(0, x_i^k), \nu'_{1,+}(0, x_i^k)]$. Thus, we expect that Algorithm 2.1 gives sparser solution than Algorithm 2.2, and Algorithm 2.2 gives sparser solution than Algorithm 2.3.

2.5.5 DCA4: DCA applied on (2.8) with the piecewise linear (PiL) approximation

We have proposed three DCA schemes for solving (2.8) or its equivalent form (2.9) when r is a concave function on $[0, +\infty)$. In this section we remove the assumption that r is concave on $[0, +\infty)$ and consider now the general case where r is a DC function satisfying Assumption 1. Hence the problem (2.8) can be expressed as a DC program (2.13) for which DCA is applicable. Each iteration of DCA applied on (2.13) consists of computing

- Compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$ and $\overline{z}_i^k \in \lambda \partial \psi(x_i^k)$ $\forall i = 1, \ldots, n$.

- Compute (x^{k+1}, y^{k+1}) as a solution of the following convex program

$$\min_{(x,y)\in K} \left\{ g(x, y) - \langle \overline{x}^k, x\rangle - \langle \overline{y}^k, y\rangle + \lambda \sum_{i=1}^{n} \varphi(x_i) - \langle \overline{z}^k, x\rangle \right\}. \tag{2.47}$$

The piecewise linear approximation function r_{PiL} is a DC function but not concave on $[0, +\infty)$. Hence we apply the above DCA scheme for solving the problem (2.8) with $r = r_{PiL}$

$$r_{PiL} = \min\left\{1, \max\left\{0, \frac{\theta|t| - 1}{a - 1}\right\}\right\} = \begin{cases} 0 & \text{if } |t| \le \frac{1}{\theta}, \\ \frac{\theta|t|-1}{a-1} & \text{if } \frac{1}{\theta} < |t| < \frac{a}{\theta}, \\ 1 & \text{otherwise,} \end{cases} \quad a > 1. \tag{2.48}$$

DC components of r_{PiL} are given by

$$\varphi_{PiL}(t) := \frac{\theta}{a-1}\max\left\{\frac{1}{\theta}, |t|\right\},\ \psi_{PiL}(t) := \frac{\theta}{a-1}\max\left\{\frac{a}{\theta}, |t|\right\} - 1 \quad \forall t \in \mathbb{R}, \tag{2.49}$$

that are polyhedral convex functions. Then the problem (2.8) can be expressed in form of a DC program as follows

$$\min_{x,y}\{F_{r_{PiL}}(x,y) := G_4(x,y) - H_4(x,y)\}, \tag{2.50}$$

where

$$G_4(x,y) = \chi_K(x,y) + g(x,y) + \lambda\sum_{i=1}^{n}\varphi_{PiL}(x_i),\ H_4(x,y) = h(x,y) + \lambda\sum_{i=1}^{n}\psi_{PiL}(x_i),$$

and g, h are DC components of f as stated in (2.7).

At each iteration k, DCA applied to (2.50) updates (x^{k+1}, y^{k+1}) from (x^k, y^k) via two steps:

- Compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$ and $\overline{z}_i^k \in \lambda\partial\psi_{PiL}(x_i^k)$ $\forall i = 1, \ldots, n$.

- Compute (x^{k+1}, y^{k+1}) as a solution of the following convex program

$$\min_{(x,y)\in K}\left\{g(x,y) - \langle\overline{x}^k, x\rangle - \langle\overline{y}^k, y\rangle + \frac{\lambda\theta}{a-1}\sum_{i=1}^{n}\max\left\{\frac{1}{\theta}, |x_i|\right\} - \langle\overline{z}^k, x\rangle\right\}. \tag{2.51}$$

Calculation of $\overline{z}_i^k$ $(i = 1, \ldots, n)$ is given by

$$\overline{z}_i^k = \begin{cases} \frac{\lambda\theta}{a-1} & \text{if } x_i^k > \frac{a}{\theta} \\ \frac{-\lambda\theta}{a-1} & \text{if } x_i^k < \frac{-a}{\theta} \\ 0 & \text{otherwise.} \end{cases} \tag{2.52}$$

Furthermore, (2.51) is equivalent to

$$\min_{(x,y,t)\in\Omega_3}\left\{g(x,y) - \langle\overline{x}^k, x\rangle - \langle\overline{y}^k, y\rangle + \frac{\lambda\theta}{a-1}\sum_{i=1}^{n}t_i - \langle\overline{z}^k, x\rangle\right\}, \tag{2.53}$$

where $\Omega_3 = \left\{(x,y,t) : (x,y) \in K, \frac{1}{\theta} \le t_i, x_i \le t_i, -x_i \le t_i\ \forall i = 1, \ldots, n\right\}$.

Algorithm 2.4 DCA applied to (2.50) (DCA4)

Initialize $(x^0, y^0) \in K$, $k \leftarrow 0$
repeat
1. Compute $(\overline{x}^k, \overline{y}^k) \in \partial h(x^k, y^k)$ and $\overline{z}_i^k \in \lambda\partial\psi_{PiL}(x_i^k)$ $\forall i = 1, \ldots, n$ via (2.52).
2. Solve the convex problem (2.53) to obtain (x^{k+1}, y^{k+1}).
3. $k \leftarrow k + 1$.

until Stopping criterion.

2.5.6 Updating θ procedure

According to consistency results, the larger θ is, the better approximate solution would be. However, from a computational point of view, with large values of θ, the approximate problems are difficult and the algorithms converge often to local minimums. We can overcome this bottleneck by using an update procedure for θ. Starting with a chosen value θ^0, at each iteration k, we compute (x^{k+1}, y^{k+1}) from (x^k, y^k) by applying the DCA based algorithms with $\theta = \theta^k$. The sequence $\{\theta^k\}_k$ is increasing by $\theta^{k+1} = \theta^k + \Delta\theta^k$. $\Delta\theta^k$ can be fixed or updated during the iterations (see Experiment 1 in the next section).

2.6 Application to Feature selection in SVM

In this section we focus on the context of Support Vector Machines learning with two-class linear models. Generally, the problem can be formulated as follows.

Given two finite point sets $\mathcal{A}$ (with label $+1$) and $\mathcal{B}$ (with label -1) in $\mathbb{R}^n$ represented by the matrices $A \in \mathbb{R}^{N_A \times n}$ and $B \in \mathbb{R}^{N_B \times n}$, respectively, we seek to discriminate these sets by a separating hyperplane ($x \in \mathbb{R}^n, b \in \mathbb{R}$)

$$P = \{w \in \mathbb{R}^n : w^T x = b\} \tag{2.54}$$

which uses as few features as possible. We adopt the notations introduced in (Bradley and Mangasarian, 1998) and consider the optimization problem proposed in (Bradley and Mangasarian, 1998) that takes the form ($e \in \mathbb{R}^n$ being the vector of ones):

$$\min_{x,b}(1-\lambda)\left(\frac{1}{N_A}\|\max\{0, -Ax + eb + e\}\|_1 + \frac{1}{N_B}\|\max\{0, Bx - eb + e\}\|_1\right) + \lambda\|x\|_0 \tag{2.55}$$

or equivalently

$$\begin{array}{ll} \min_{x,y,\xi,\zeta} & (1-\lambda)(\frac{1}{N_A}e^T\xi + \frac{1}{N_B}e^T\zeta) + \lambda\|x\|_0 \\ s.t. & -Ax + eb + e \leq \xi,\ Bx - eb + e \leq \zeta,\ \xi \geq 0,\ \zeta \geq 0. \end{array} \tag{2.56}$$

The nonnegative slack variables ξ_j ($j = 1, ..., N_A$) represent the errors of classification of $a_j \in \mathcal{A}$ while ζ_j ($j = 1, ..., N_B$) represent the errors of classification of $b_j \in \mathcal{B}$. More precisely, each positive value of ξ_j determines the distance between a point $a_j \in A$ (lying on the wrong side of the bounding hyperplane $w^T x = b + 1$ for $\mathcal{A}$) and the hyperplane itself. A similar explanation is for ζ_j, $\mathcal{B}$ and $w^T x = b - 1$. The first term of the objective function of (2.56) is the average error of classification, and the second term is the number of nonzero components of the vector x, each of which corresponds to a representative feature. Further, if an element of x is zero, the corresponding feature is removed from the dataset. Here λ is a control parameter of the trade-off between the training error and the number of selected features.

Observe that the problem (2.56) is a special case of (2.1) where the function f is given by

$$f(x,b,\xi,\zeta) := (1-\lambda)\left(\frac{1}{N_A}e^T\xi + \frac{1}{N_B}e^T\zeta\right) \tag{2.57}$$

and K is a polytope defined by

$$K := \left\{(x,b,\xi,\zeta) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}_+^{N_A} \times \mathbb{R}_+^{N_B} : -Ax + eb + e \le \xi,\ Bx - eb + e \le \zeta\right\}. \tag{2.58}$$

Then the approximate problem takes the form

$$\min\left\{F(x,b,\xi,\zeta) := f(x,b,\xi,\zeta) + \lambda\sum_{i=1}^n r(x_i) : (x,b,\xi,\zeta) \in K\right\}, \tag{2.59}$$

where r is one of the sparsity-inducing functions given in Table 2.1. This problem is also equivalent to

$$\min\left\{\overline{F}(x,b,\xi,\zeta,z) := f(x,b,\xi,\zeta) + \lambda\sum_{i=1}^n r(z_i) : (x,b,\xi,\zeta,z) \in \overline{K}\right\}, \tag{2.60}$$

where $\overline{K} = \{(x,b,\xi,\zeta,z) : (x,b,\xi,\zeta) \in K, -z_i \le x_i \le z_i\ \forall i = 1,\dots,n\}$.

Note that, since K is a polyhedral convex set, all the resulting approximate problems (2.59) with approximation functions given in Table 2.2 (except for $r = r_{PiL}$) are equivalent to the problem (2.56) in the sense of Corollary 2.1. More strongly, from Proposition 2.4, if $r = r_{cap}$ and $\theta > \theta^* := \frac{1-\lambda}{\lambda}\Delta$, where

$$\Delta := \max_{j=1,\dots,n}\left\{\frac{1}{N_A}\sum_{i=1}^{N_A}|A_{ij}| + \frac{1}{N_B}\sum_{i=1}^{N_B}|B_{ij}|\right\}, \tag{2.61}$$

then the problems (2.56) and (2.59) are equivalent.

Here the function f is simply linear, and DC components of f is taken as $g = f$ and $h = 0$. According to Algorithms 2.1, 2.2, 2.3 and 2.4, DCA for solving the problem (2.59) is described briefly as follows.

DCA1: For η given in Table 2.2, let $\psi(t) = \eta|t| - r(t)$. At each iteration k, DCA1 for solving (2.59) consists of

- Compute $\overline{z}_i^k \in \lambda\partial\psi(x_i^k)\ \forall i = 1,\dots,n$ as given in Table 2.2.

- Compute $(x^{k+1}, b^{k+1}, \xi^{k+1}, \zeta^{k+1})$ by solving the linear program

$$\min\left\{(1-\lambda)\left(\frac{1}{N_A}e^T\xi + \frac{1}{N_B}e^T\zeta\right) + \lambda\eta\sum_{i=1}^n z_i - \langle\overline{z}^k, x\rangle : (x,b,\xi,\zeta,z) \in \overline{K}\right\}. \tag{2.62}$$

Since f is linear and K is a polyhedral convex set, the first DC component G_1 in (2.39) is polyhedral convex. Therefore, (2.39) is always a polyhedral DC program. According to the convergence property of polyhedral DC programs, DCA1 applied to (2.59) generates a sequence $\{(x^k, b^k, \xi^k, \zeta^k)\}$ that converges to a critical point $(x^*, b^*, \xi^*, \zeta^*)$ after finitely many iterations. Furthermore, if $r = r_{cap}$ and $|x_i^*| \neq \frac{1}{\theta}$ $\forall i = 1, \ldots, n$, the second DC component H_1 in (2.39) is polyhedral convex and differentiable at $(x^*, b^*, \xi^*, \zeta^*)$. Using the DCA's convergence property, we deduce that $(x^*, b^*, \xi^*, \zeta^*)$ is a local solution of (2.59).

DCA2: At each iteration k, DCA2 for solving (2.59) consists of

- Compute $\overline{z}_i^k \in -\lambda\partial(-r)(|x_i^k|)$ $\forall i = 1, \ldots, n$ as given in Table 2.3.

- Compute $(x^{k+1}, b^{k+1}, \xi^{k+1}, \zeta^{k+1})$ by solving the linear program

$$\min \left\{ (1-\lambda)\left(\frac{1}{N_A}e^T\xi + \frac{1}{N_B}e^T\zeta\right) + \langle \overline{z}^k, z\rangle : (x, b, \xi, \zeta, z) \in \overline{K} \right\}.$$

Similar to the case of DCA1 mentioned above, (2.41) is also a polyhedral DC program. Thus, DCA2 applied to (2.60) generates a sequence $\{(x^k, b^k, \xi^k, \zeta^k, |x^k|)\}$ that converges to a critical point $(x^*, b^*, \xi^*, \zeta^*, |x^*|)$ after finitely many iterations. Furthermore, if $r = r_{cap}$ and $|x_i^*| \neq \frac{1}{\theta}$ $\forall i = 1, \ldots, n$, the second DC component H_2 in (2.41) is polyhedral convex and differentiable at $(x^*, b^*, \xi^*, \zeta^*, |x^*|)$. Then $(x^*, b^*, \xi^*, \zeta^*, |x^*|)$ is a local solution of (2.60).

DCA3: At each iteration k, DCA3 for solving (2.59) consists of

- Compute $\overline{z}_i^k \in \frac{-\lambda}{2(|x_i^k|^2+\epsilon)^{1/2}}\partial(-r)((|x_i^k|^2+\epsilon)^{1/2})$ $\forall i = 1, \ldots, n$ as given in Table 2.4.

- Compute $(x^{k+1}, b^{k+1}, \xi^{k+1}, \zeta^{k+1})$ by solving the quadratic convex program

$$\min \left\{ (1-\lambda)\left(\frac{1}{N_A}e^T\xi + \frac{1}{N_B}e^T\zeta\right) + \sum_{i=1}^{n} \overline{z}_i^k x_i^2 : (x, b, \xi, \zeta) \in K \right\}.$$

DCA4: Consider the case $r = r_{PiL}$. At each iteration k, DCA4 for solving (2.59) consists of

- Compute $\overline{z}_i^k \in \lambda\partial\psi_{PiL}(x_i^k)$ $\forall i = 1, \ldots, n$ via (2.52).

- Compute $(x^{k+1}, b^{k+1}, \xi^{k+1}, \zeta^{k+1})$ by solving the linear program

$$\begin{aligned} \min \quad & \left\{ (1-\lambda)\left(\frac{1}{N_A}e^T\xi + \frac{1}{N_B}e^T\zeta\right) + \frac{\lambda\theta}{a-1}\sum_{i=1}^{n} t_i - \langle \overline{z}^k, x\rangle \right\}, \\ s.t. \quad & (x, b, \xi, \zeta, t) \in \overline{K}, \frac{1}{\theta} \leq t_i \ \forall i = 1, \ldots, n. \end{aligned}$$

Since the second DC component H_4 in (2.50) is polyhedral convex, (2.50) is a polyhedral DC program. Thus, DCA4 applied to (2.59) generates a sequence $\{(x^k, b^k, \xi^k, \zeta^k)\}$ that

converges to a critical point $(x^*, b^*, \xi^*, \zeta^*)$ after finitely many of iterations. Moreover, if $|x_i^*| \neq \frac{1}{\theta}$ $\forall i = 1, \ldots, n$, then H_4 is differentiable at $(x^*, b^*, \xi^*, \zeta^*)$. This implies that $(x^*, b^*, \xi^*, \zeta^*)$ is a local solution of (2.59).

The stopping criterion of our algorithms is given by

$$\|x^{k+1} - x^k\| + |b^{k+1} - b^k| + \|\xi^{k+1} - \xi^k\| + \|\zeta^{k+1} - \zeta^k\| \leq \tau(1 + \|x^k\| + |b^k| + \|\xi^k\| + \|\zeta^k\|),$$

where τ is a small tolerance.

We have seen in Sect. 2.4 that the approximate problem using Capped-ℓ_1 and SCAD approximations are equivalent to the original problem if the parameter θ is beyond a certain threshold: $\theta \geq \theta_0$ (cf. Proposition 2.3 and Proposition 2.5). However, the computation of such a value θ_0 is in general not available, hence one must take large enough values for θ_0. But, as discussed in Sect. 2.5.6, a large value of θ makes the approximate problem hard to solve. For the feature selection in SVM, we can compute exactly a θ_0 as shown in (2.61), but it is quite large. Hence we use an updating θ procedure. On the other hand, in the DCA1 scheme, at each iteration, we have to compute $\bar{z}^k \in \partial\lambda\psi(x^k)$ and when ψ is not differentiable at x^k, the choice of $\bar{z}^k$ can influence on the efficiency of the algorithm. For Capped-ℓ_1 approximation, based on the properties of this function we propose a specific way to compute $\bar{z}^k$. Below, we describe the updating θ procedure for DCA1 with Capped-ℓ_1 approximation.

Initialization: $\Delta\theta > 0, \alpha^0 = +\infty$, $\theta^0 = 0$, $k = 0$. Let $(x^0, b^0, \xi^0, \zeta^0)$ be a solution of the linear problem (2.59).
Repeat

1. $I = \{i : 0 < |x_i^k| < \alpha^k\}$, $\alpha^{k+1} = \begin{cases} \max\{|x_i^k| : i \in I\} & \text{if } I \neq \emptyset, \\ \alpha^k & \text{otherwise.} \end{cases}$
2. Compute $\theta^{k+1} = \min\left\{\theta^*, \max\left\{\frac{1}{\alpha^{k+1}}, \theta^k + \Delta\theta\right\}\right\}$.
3. Compute $\overline{z}^k$: For $i = 1, \ldots, n$

   - If $|x_i^k| < \alpha^{k+1}$, $\overline{z}_i^k = 0$.
   - If $|x_i^k| > \alpha^{k+1}$, $\overline{z}_i^k = \text{sign}(x_i^k)\lambda\theta$.
   - If $|x_i^k| = \alpha^{k+1}$, compute F_i^- (resp. F_i^+) the left (resp. right) derivative of the function $u(x, b)$ w.r.t. the variable x_i at x_i^k, where

   $$u(x, b) = (1-\lambda)\left(\frac{1}{N_A}\|\max\{0, -Ax + eb + e\}\|_1 + \frac{1}{N_B}\|\max\{0, Bx - eb + e\}\|_1\right) + \lambda\sum_{j=1}^{n} r(x_j).$$

   Then $\overline{z}_i^k = \begin{cases} \text{sign}(x_i^k)\lambda\theta^{k+1} & \text{if } x_i^k(F_i^- + F_i^+) < 0 \\ 0 & \text{ortherwise.} \end{cases}$

4. Solve the linear problem (2.62) with $\eta = \theta^{k+1}$ to obtain $(x^{k+1}, b^{k+1}, \xi^{k+1}, \zeta^{k+1})$.
5. $k = k + 1$.

Until: Convergence of $\{x^k, b^k, \xi^k, \zeta^k\}$.

In the above procedure, the computation of $\overline{z}^k$ is slightly different from formula given in Table 2.2. When $|x_i^k| = \alpha^{k+1}$, $\partial r(x_i^k)$ is an interval. Taking into account information of derivative of u w.r.t. the variable x_i at x_i^k helps us judge which between two extreme values of $\partial r(x_i^k)$ may give better decrease of algorithm.

At each iteration, the value of θ increases at least $\Delta\theta > 0$ as long as it does not exceed θ^* – the value from which the problems (2.56) and (2.59) are equivalent. Moreover, we know that for each fixed θ, DCA1 has finite convergence. Hence, the above procedure also possesses finite convergence property.

If $F(x^{k+1}, b^{k+1}, \xi^{k+1}, \zeta^{k+1}) = F(x^k, b^k, \xi^k, \zeta^k)$ then $(x^k, b^k, \xi^k, \zeta^k)$ is a critical point of (2.59) with $r = r_{cap}$ and $\theta = \theta^{k+1}$. In addition, if $\alpha^{k+1} = \alpha^k$, which means that $|x_i^k| \geq \alpha^k \geq \frac{1}{\theta^k}$ for any $i \in \text{supp}(x^k)$, then $(x^k, b^k, \xi^k, \zeta^k)$ is a critical point of (2.59) for all $\theta \geq \theta^{k+1}$.

2.6.1 Computational experiments

The aim of our experiments is to give a fair comparison of DCA's versions to identify the best one. For this purpose we perform two experiments to identify

(i) the best DCA scheme among three proposed DCA1, DCA2 and DCA3;

(ii) the best among six existing / proposed approximations.

It is worth noting that comparative results between DCA based algorithms and several existing approaches have been presented in the previous works using DCA for feature selection in SVM. More precisely, the FSV algorithm in (Bradley and Mangasarian, 1998) (which is DCA2 with the exponential approximation) is the best with respect to SVM without regulation (Bennett et al., 1992) (named RLP), ℓ_2-SVM (Cortes and Vapnik, 1995), ℓ_1-SVM and ℓ_∞-SVM (Bradley and Mangasarian, 1998). In (Weston et al., 2003) the algorithm based on Franke and Wolfe method and the logarithm approximation (this is in fact DCA2 with the logarithm approximation) is better than RFE SVM (Guyon el al., 2002), CORR SVM (Weston et al., 2003), R2W2 SVM (Weston et al. , 2001), FSV (Bradley and Mangasarian, 1998). In (Newmann et al., 2005) the authors considered the two models using ℓ_2-ℓ_0-norm and ℓ_2-ℓ_1-norm with the exponential approximation and proposed a DCA scheme for the ℓ_2-ℓ_0-norm model. Their computational experiments indicated the superiority of DCA for ℓ_2-ℓ_0-norm model with respect to RLP, ℓ_2-SVM, FSV, and a SVM based filter method (Heiler et al., 2001). At last, according to comparative numerical results provided in (Le Thi et al., 2008), the DCA algorithm (DCA1 with the exponential approximation) outperforms RLP,

ℓ_2-SVM, ℓ_1-SVM and ℓ_∞-SVM and FSV. Thus, in the present work a comparison between DCA's versions and the "other approaches" above mentioned is not helpful. Meanwhile, in the second experiment dealing with the efficiency of approximations, we present also the numerical results of the ℓ_1-SVM algorithm (the ℓ_1-norm is used to approximate the ℓ_0-norm), the best known and widely used convex approximation approach for feature selection in SVM.

2.6.1.1 Datasets

Numerical experiments were performed on several real-word datasets taken from well-known UCI data repository and from challenging feature-selection problems of the NIPS 2003 datasets. In Table 2.5, the number of features, the number of points in training and test set of each dataset are given. The full description of each dataset can be found on the web site of UCI repository and NIPS 2003.

Table 2.5: Datasets

Data	#features	# points in training set	# points in test set
Ionosphere	34	234	117
WPBC (24 months)	32	104	51
WPBC (60 months)	32	380	189
Breast Cancer	24481	78	19
Leukemia	7129	38	34
Arcene	10000	100	100
Gisette	5000	6000	1000
Prostate	12600	102	21
Adv	1558	2458	821

2.6.1.2 Set up experiments

All algorithms were implemented in the Visual C++ 2008, and performed on a PC Intel i5 CPU650, 3.2 GHz of 4GB RAM. CPLEX 12.2 was used for solving linear/quadratic programs. We stop all algorithms with the tolerance $\epsilon = 10^{-5}$. The non-zero elements of x are determined according to whether $|x_i|$ exceeds a small threshold (10^{-5}).

For the comparison of algorithms, we are interested in the accuracy (PWCO - Percentage of Well Classified Objects) and the sparsity of obtained solution as well as the rapidity of the algorithms. $PWCO_1$ (resp. $PWCO_2$) denotes the PWCO on training set (resp. test set). The sparsity of solution is determined by the number (and percentage) of selected features (SF) while the rapidity of algorithms is measured by the CPU time in seconds.

2.6.1.3 Experiment 1

In this experiment, we study the effectiveness of the three proposed DCA schemes DCA1, DCA2 and DCA3 for a same approximation. From theoretical results in Section 5 we chose Capped-ℓ_1 approximation for this experiment. For each dataset, the same value of λ is used for all algorithms. We set $\lambda = 0.1$ for the first three datasets (*Ionosphere, WPBC(24), WPBC(60)*) while $\lambda = 0.001$ is used for five large datasets (*Adv, Arcene, Breast, Gisette, Leukemia*). To choose a suitable value of θ for each algorithm DCA1, DCA2 and DCA3, we perform them by 10 folds cross-validation procedure on the set $\{0.001, 0.005, 0.01, 0.1, 0.5, 1, 2, 3, 5, 10, 20, 50, 100, 500\}$ and then take the value corresponding to the best results. Once θ is chosen (its value is given in Table 2.6), we perform these algorithms 10 times from 10 random starting solutions and report, in the columns 3 - 5 of Table 2.6, the mean and standard deviation of the accuracy, the sparsity of obtained solutions and CPU time of the algorithm.

We are also interested on the efficiency of Updating θ procedure. For this purpose, we compare two versions of DCA1 - with and without Updating θ procedure (in case of Capped-ℓ_1 approximation). For a fair comparison, we first run DCA1 with Updating θ procedure and then perform DCA1 with the fixed value θ^* which is the last value of θ when the Updating θ procedure stops. Computational results are reported in the columns 6 (DCA1 with fixed θ) and 7 (DCA1 with Updating θ procedure) of Table 2.6.

To evaluate the globality of the DCA based algorithms we use CPLEX 12.2 for globally solving the exact formulation problem (2.22) via exact penalty techniques (Mixed 0-1 linear programming problem) and report the results in the last column of Table 2.6.

Bold values in the result tables correspond to best results for each data instance.

Comments on numerical results

- Comparison between DCA1, DCA2 and DCA3 (columns 3 - 5)
  - Concerning the correctness, DCA1 furnishes the best solution out of the three algorithms for all datasets (with an important gain of $6,9\%$ on dataset WPBC(24)). DCA2 and DCA3 are comparable in terms of correctness.
  - As for the sparsity of solution, all the three DCA schemes reduce considerably the number of selected features (up to 99% on large datasets such as *Arcene*, *Breast*, *Leukemia*, ...). Moreover, DCA1 gives better results than DCA2/DCA3 on 6 out of 7 datasets.
  - In terms of CPU Time, DCA1 and DCA2 are faster than DCA3. This is natural, since at each iteration, the first two algorithms only require solving one linear program while DCA3 has to solve one convex quadratic program. DCA1 is somehow a bit faster than DCA2 on 5 out 7 datasets.
  - Overall, we see that DCA1 is better than DCA2 and DCA3 on all the three evaluation criteria. Hence, it seems to be that the first DCA scheme is more appropriate than the other two for Capped-ℓ_1 approximation.

Table 2.6: Comparison of different DCA schemes for Capped-ℓ_1 approximation

		DCA1	DCA2	DCA3	DCA1 with θ^*	DCA1 with Updating θ	CPLEX
Ionosphere	θ	3	5	3	4,3	4,3	
	$PWCO_1$	86,2 ±1,5	85,2 ±1,7	84,8 ±1,8	84,0 ±1,2	**90,2**	**90,2**
	$PWCO_2$	80,3 ±1,6	75,3 ±1,3	74,3 ±1,3	80,3 ±1,4	**83,7**	**83,7**
	SF	3,5 (10,3%)	3,8 (11,2%)	3,8 (11,2%)	3,2 (9,4%)	**2 (5,9%)**	**2 (5,9%)**
	CPU	**0,2**	**0,2**	0,7	0,3	0,6	2,5
WPBC(24)	θ	1	0,1	0,1	661	661	
	$PWCO_1$	**84,3 ±1,4**	75,3 ±1,3	77,4 ±1,1	75,3 ±1,2	77,4	77,4
	$PWCO_2$	77,9 ±1,4	**80,2 ±1,6**	79,3 ±1,6	72,3 ±1,2	77,2	78,4
	SF	7,4 (23,1%)	8,5 (26,6%)	8,5 (26,6%)	8,4 (26,3%)	8 (25,0%)	**7 (21,9%)**
	CPU	**0,2**	0,3	0,8	**0,2**	1,1	6,4
WPBC(60)	θ	1	3	3	347	347	
	$PWCO_1$	96,2 ±1,3	95,2 ±1,3	95,2 ±1,3	**98,2 ±1,3**	96	**96**
	$PWCO_2$	92,5±1,4	92,5±1,4	90,8±1,8	**96,8±1,8**	95,3	**95,3**
	SF	4,7 (15,7%)	5,5 (18,3%)	5,7 (19,0%)	8,9 (29,7%)	**3 (10,0%)**	**3 (10,0%)**
	CPU	**0,4**	0,6	1,6	0,5	1	1,8
Breast	θ	5	10	2	435	435	
	$PWCO_1$	95,1±1,3	94,2±1,3	95,2±1,4	93,2±1,6	**96,8**	N/A
	$PWCO_2$	68,3±1,2	67,3±1,2	**70,3±1,6**	66,3±1,1	65,1	N/A
	SF	32,6 (0,1%)	47,5 (0,2%)	43,5 (0,2%)	52,3 (0,2%)	**28 (0,1%)**	N/A
	CPU	30	**25**	78	79	76	3600
Leukemia	θ	5	5	5	178	178	
	$PWCO_1$	**100**	**100**	**100**	**100**	**100**	N/A
	$PWCO_2$	**97,2±0,4**	97,1±0,4	96,8±0,3	94,8±0,7	**97,2**	N/A
	SF	8,2 (0,1%)	8,5 (0,1%)	8,5 (0,1%)	12,0 (0,2%)	**8 (0,1%)**	N/A
	CPU	**10**	**10**	75	14	17	3600
Arcene	θ	0,1	0,01	3	328	328	
	$PWCO_1$	**100**	**100**	**100**	**100**	**100**	N/A
	$PWCO_2$	80±1,6	**82±1,1**	81±1,9	61±1,1	70	N/A
	SF	78,5 (0,79%)	82,4 (0,82%)	82,4 (0,82%)	35 (0,35%)	**32 (0,32%)**	N/A
	CPU	**21**	26	273	30	118	3600
Gisette	θ	0,1	0,01	0,1	735	735	
	$PWCO_1$	**92,5±1,3**	88,5±1,3	88,5±1,3	90,5±1,2	91,2	N/A
	$PWCO_2$	**85,3±1,2**	83,4±1,2	83,1±1,6	84,1±1,1	83,2	N/A
	SF	339,4 (6,8%)	330,7 (6,6%)	332,2 (6,6%)	456 (9,1%)	**123 (2,5%)**	N/A
	CPU	87	**65**	253	71	387	3600
Adv	θ	0,1	0,01	0,1	321	321	
	$PWCO_1$	95,5±1,5	92,3±1,5	95,3±1,5	92,3±1,2	**97,2**	N/A
	$PWCO_2$	**94,2±1,1**	93,2±1,5	93,1±1,2	92,1±1,6	93,2	N/A
	SF	5,4 (0,35%)	6,2 (0,40%)	6,4 (0,41%)	6,5 (0,42%)	**5 (0,32%)**	N/A
	CPU	**2,1**	2,4	7,8	2,3	4,6	3600

- DCA1 with and without Updating θ procedure (columns 3, 6 and 7):
  - For all datasets, Updating θ procedure gives a better solution (on both accuracy and sparsity) than DCA1 with $\theta = \theta^*$.
  - Except for dataset *WPBC(24)*, Updating θ procedure is better than DCA1 with θ chosen by 10 folds cross-validation in terms of sparsity of solution. As for accuracy, the two algorithms are comparable.
  - The choice of the value of θ defining the approximation function is very important. Indeed, the results given in columns 3 and 6 are far different, due to the fact that, the value of θ chosen by 10 folds cross-validation is much more smaller than θ^*. These results confirm our analysis in Subsection 2.5.6 above: while the approximate function would be better with larger values of θ, the approximate problems become more difficult and it can happen that the obtained solutions are worse when θ is quite large. To overcome this

"contradiction" between theoretical and computational aspects, the proposed Updating θ procedure seems to be efficient.

- Comparison between DCA based algorithms and CPLEX for solving the original problem (2.22)
  - For *Ionosphere* and *WPBC(60)*, Updating θ procedure for Capped-ℓ_1 gives exactly the same accuracy and the same number of selected features as CPLEX. It means that Updating θ procedure reaches the global solution for those two datasets. For *WPBC(24)*, the two obtained solutions are slightly different (same accuracy on training set and 7 selected features for CPLEX instead of 8 for Updating θ procedure).
  - For large datasets, CPLEX can't furnish a solution with a CPU Time limited to 3600 seconds while DCA based algorithms give a good solution in a short time.

2.6.1.4 Experiment 2

In the second experiment, we study the effectiveness of different approximations of ℓ_0. We use DCA1 for all approximations except PiL for which DCA4 is applied (cf. Section 2.5.5).

In this experiment, for the trade-off parameter λ, we used the following set of candidate values $\{0.001, 0.002, 0.003, 0.004, 0.05, 0.1, 0.25, 0.4, 0.7, 0.9\}$. The value of parameter θ is chosen in the set $\{0.001, 0.005, 0.01, 0.1, 0.5, 1, 2, 3, 5, 10, 20, 50, 100, 500\}$. The second parameter a of SCAD approximation is taken from $\{1, 2, 3, 5, 10, 20, 30, 50, 100\}$. For each algorithm, we firstly perform a 10-folds cross-validation to determine the best set of parameter values. In the second step, we run each algorithm, with the chosen set of parameter values in step 1, 10 times from 10 starting random points and report the mean and standard deviation of each evaluation criterion. The comparative results are reported in Table 2.7.

We observe, among DCA's versions, that:

- In terms of sparsity of solution, the quality of all approximations are comparable. All the algorithms reduce considerably the number of selected features, especially for 5 large datasets (*Adv, Arcene, Breast, Gisette, Leukemia*). For *Breast* dataset, our algorithms select only about thirty features out of 24481 while preserving very good accuracy (up to $98,7\%$ correctness on train set).
- Capped-ℓ_1 is the best in terms of accuracy: it gives best accuracy on all train sets and 4 out of 7 test sets. The quality of other approximations are comparable.
- The CPU time of all the algorithms is quite small: less than 34 seconds (except for *Gisette*, CPU time of DCAs varies from 72 to 102 seconds).

Comparing DCA based algorithms with ℓ_1-SVM, not surprisingly, DCA is much better than ℓ_1-SVM in terms of both sparsity and accuracy, and ℓ_1-SVM is the fastest algorithm (it solves one linear program). The results show that the ℓ_1 approach is not efficient to deal with sparsity.

Table 2.7: Comparison of different approximations

		DCA1 Capped-l1	DCA1 SCAD	DCA1 Exp	DCA1 lp+	DCA1 lp-	DCA1 Log	DCA4 PiL	SVM-ℓ_1
Ionosphere	$PWCO_1$	**86,2 ±1,5**	80,1 ±1,4	82,1 ±1,4	81,5 ±1,3	83,1 ±1,4	81,2 ±1,4	83,2 ±1,4	77,3
	$PWCO_2$	80,3 ±1,6	73,5 ±1,6	**84,8 ±1,3**	75,1 ±1,1	70,3 ±1,2	73,1 ±2,1	83,5 ±1,6	75,3
	SF	3,5 (10,3%)	3,1 (9,1%)	**2,3 (6,8%)**	3,8 (11,2%)	3,1 (9,1%)	3,3 (9,7%)	2,6 (7,6%)	10 (29,4%)
	CPU	0,2	0,3	0,3	0,2	0,3	0,15	0,2	**0,01**
WPBC(24)	$PWCO_1$	**84,3 ±1,4**	77 ±1,3	**84,3 ±1,5**	81,3 ±1,2	81,9 ±1,2	71,3 ±1,4	84,2 ±1,4	73,3
	$PWCO_2$	77,9 ±1,4	**79,3 ±1,6**	74,3 ±1,9	78,4 ±1,2	**79,8 ±1,1**	68,4 ±1,6	78,5 ±1,4	72,1
	SF	7,4 (23,1%)	8,1 (25,3%)	**7,2 (22,5%)**	7,8 (24,4%)	7,5 (23,4%)	**7,2 (22,5%)**	7,6 (23,8%)	8 (25,0%)
	CPU	0,1	0,2	0,1	0,2	0,2	0,2	0,2	**0,01**
WPBC(60)	$PWCO_1$	**97,2 ±1,3**	93,5 ±1,7	95,1 ±1,6	93 ±1,2	94,5 ±1,1	89 ±1,5	95,2 ±1,3	84,3
	$PWCO_2$	**93,5±1,4**	89,1 ±1,9	92,3 ±1,9	85 ±1,2	90,6 ±1,2	80 ±1,6	88,5±1,1	85,5
	SF	5,4 (18,0%)	**5,2 (17,3%)**	**5,2 (17,3%)**	5,9 (19,7%)	5,7 (19,0%)	5,4 (18,0%)	5,4 (18,0%)	6 (20,0%)
	CPU	0,4	0,4	0,4	0,5	0,4	0,6	0,5	**0,01**
Breast	$PWCO_1$	**98,7±1,3**	91,9±1,4	96,3±1,4	93,2±1,4	91,9±1,4	91,2±1,4	92,4±1,2	94,3
	$PWCO_2$	68,3±1,2	69,1±1,6	**70%±1,4**	67,3±1,1	69,1±1,6	66,3±1,2	71,3±1,4	67,8
	SF	35,3 (0,1%)	37,0 (0,2%)	37,4 (0,2%)	40,3 (0,2%)	37,0 (0,1%)	45,3 (0,2%)	**26,5**	4325 (17,7%)
	CPU	30	31	25	32	31	31	31	**3,2**
Leukemia	$PWCO_1$	**100**	98,3±0,2	**100**	**100**	98,3±0,2	**100**	**100**	68,3
	$PWCO_2$	**97,2±0,4**	88,3±0,6	97,2±0,5	90,1±0,8	92,3±0,6	90,1±0,3	89,2±0,9	70,3
	SF	**8,2 (0,1%)**	**8,2 (0,1%)**	8,3 (0,1%)	27,9 (0,4%)	**8,2 (0,1%)**	27,3 (0,4%)	12,8 (0,2%)	2134 (29,9%)
	CPU	25	21	23	27	21	28	22	**3**
Arcene	$PWCO_1$	**100**	**100**	**100**	**100**	**100**	**100**	**100**	74,3
	$PWCO_2$	**80±1,6**	78,2±1,9	78,9±1,4	78,9±1,1	74,2±1,2	72,9±1,6	79±1,2	66,5
	SF	78,5 (0,79%)	72,5 (0,73%)	**69,4 (0,69%)**	71,1 (0,71%)	73,1 (0,73%)	72,3 (0,72%)	83,5 (0,84%)	2102 (21,02%)
	CPU	21	31	34	31	31	30	23	**3**
Gisette	$PWCO_1$	**92,5±1,3**	87,3±1,5	87,3±2,1	88,3±2,4	86,4±1,2	86,3±2,1	89,5±1,4	74,3
	$PWCO_2$	**85,3±1,2**	81,2±1,4	82,2±1,2	77,3±1,3	82,2±1,5	79,3±1,4	84,5±1,2	73,1
	SF	339,4 (6,8%)	340,1 (6,8%)	**330,1 (6,6%)**	341,5 (6,8%)	342,3 (6,8%)	354,5 (7,1%)	344,3 (6,9%)	1424 (28,5%)
	CPU	87	81	98	102	81	102	72	**12**
Adv	$PWCO_1$	**95,5±1,5**	94,2±1,3	**95,5±1,1**	93,2±1,1	92,2±1,5	95,2±1,6	94,1±1,8	84,3
	$PWCO_2$	94,2±1,1	94,4±1,9	**94,5±1,5**	80,2±1,5	88,1±1,2	92,2±1,5	90,2±1,1	77,8
	SF	5,4 (0,35%)	8,1 (0,52%)	**5,1 (0,33%)**	12,3 (0,79%)	6,4 (0,41%)	21,3 (1,37%)	7,4 (0,47%)	413 (26,5%)
	CPU	2,1	2,5	2,3	2,8	2,5	2,8	3,1	**12**

2.7 Conclusion

We have intensively studied DC programming and DCA for sparse optimization problem including the zero-norm in the objective function. DC approximation approaches have been investigated from both a theoretical and an algorithmic point of view. Considering a class of DC approximation functions of the zero-norm including all usual sparsity-inducing approximation functions, we have proved several novel and interesting results: the consistency between global (resp. local) minimizers of the approximate problem and the original problem, the equivalence between these two problems (in the sense that, for a sufficiently large related parameter, any optimal solution to the approximate problem solves the original problem) when the feasible set is a bounded polyhedral convex set and the approximation function is concave, the equivalence between Capped-ℓ_1 (and/or SCAD) approximate problems and the original problem with sufficiently large parameter θ (in the sense that they have the same set of optimal solutions), the way to compute such parameters θ in some special cases, and a comparative analysis between usual sparsity-inducing approximation functions. Considering the three DC formulations for a common model to all concave approximation functions we have developed three DCA schemes and showed the link between our algorithms with standard approaches. It turns out that all standard nonconvex approximation algorithms are special versions of our DCA based algorithms. A new DCA scheme has been also investigated for the DC approximation (piecewise linear) which is not concave as usual sparsity-inducing functions. Concerning the application to feature selection in SVM, among the four DCA schemes, three (resp. one) require solving one linear (resp. convex quadratic) program at each iteration and enjoy interesting convergence properties (except Algorithm 3): they converge after finitely many iterations to a local solution in almost all cases. Numerical experiments confirm the theoretical results: the Capped-ℓ_1 has been identified as the "winner" among sparsity-inducing approximation functions.

Our unified DC programming framework shed a new light on sparse nonconvex programming. It permits to establish the crucial relations among existing sparsity-inducing methods and therefore to exploit, in an elegant way, the nice effect of DC decompositions of objective functions. The first three algorithms can be viewed as an ℓ_1-perturbed algorithm / reweighted-ℓ_1 algorithm (intimately related to the ℓ_1-penalized LASSO approach / reweighted-ℓ_2 algorithm in case of convex objective functions. It specifies the flexibility/versatility of these theoretical and algorithmic tools. These results should enhance deeper developments of DC programming and DCA, in order to efficiently model and solve real-world nonconvex sparse optimization problems, especially in the large-scale setting.

Chapter 3

Algorithms for Nonegative Matrix Factorization[1]

Abstract: *In this chapter, we propose two approaches based on DC programming and DCA to the nonnegative matrix factorization (NMF) problem. The first approach follows the alternating framework where, at each iteration, we need to solve two nonnegativity-constrained least squares subproblems. We develop a DCA based scheme for solving the nonnegativity-constrained least square problem. Our NMF algorithm has global convergence property and can be adopted to many other extensions of NMF. The proposed method covers several existing algorithms for NMF and its extensions. The second approach applies directly DC programming and DCA to the NMF problem. Two algorithms, one computes all variables and one deploys a variable selection strategy, are proposed. The efficiency of the proposed algorithms are empirically demonstrated on both real-world and synthetic datasets. It turns out that our proposed algorithms compete favorably with the state-of-the-art alternating nonnegative least squares algorithms.*

3.1 Introduction

Nonnegative matrix factorization (NMF) is the problem of approximating a given nonnegative matrix by the product of two low-rank nonnegative matrices. Given a matrix $A \in \mathbb{R}^{m\times n}$ and a positive integer $r < \min\{n, m\}$, one desires to compute two nonnegative matrix factors

1. The material of this chapter is based on the following works:
[1]. Hoai An Le Thi, Xuan Thanh Vo, Tao Pham Dinh. DC programming and DCA for nonnegative matrix factorization. in *D. Hwang et al. (Eds), ICCCI 2014: Proceedings of 6th International Conference on Computational Collective Intelligence Technologies and Applications*, LNAI 8733, pp. 573–582, Springer 2014.
[2]. Hoai An Le Thi, Xuan Thanh Vo, Tao Pham Dinh. Efficient Nonegative Matrix Factorization via DC programming and DCA. *Revised version to* Neural Computation.

$U \in \mathbb{R}_+^{m\times r}$ and $V \in \mathbb{R}_+^{n\times r}$ such that

$$A \approx UV^T = \sum_{j=1}^{r} U_{:j}V_{:j}^T. \tag{3.1}$$

This problem was first introduced in 1994 by (Paatero and Tapper, 1994), and more recently received a considerable interest after the works of (Lee and Seung, 1999, 2001). The approximation (3.1) means that each data item $A_{:i}$ $(i = 1, \ldots, n)$ is approximately represented by a linear combination of the columns of U weighted by the components of $V_{i:}$. Therefore, U can be regarded as containing basis vectors that is optimized for the linear approximation of the data in A and V is the corresponding coefficient matrix. Since $r < \min(m, n)$, this results in a compact representation of the original large data that can be seen either as a feature extraction or a dimensionality reduction technique. The distinction between NMF and other factorization techniques such as singular value decomposition (SVD), principal component analysis (PCA) and vector quantization (VQ) is the nonnegativity constraints imposed on the matrix factors U and V. This feature makes NMF interpretable and capable of discovering structure that is latent in the data. Indeed, there are many types of data that have inherently nonnegative representation. For example, a corpus of documents can be characterized by matrix A where A_{ij} is the number of times the ith word appears in the document j^{th}. Or in image processing, digital images are represented by pixel intensities that are nonnegative. The nonnegativity of U enforces basis vectors (the columns of U) physically compatible with the original data. Together with the nonnegativity of V, each data item can be represented as an *additive* combination of basis vector, no subtractions can occur here.

The most useful property of NMF is that it often generates sparse basis vectors and sparse coefficient matrix. This implies two folds. Firstly, each basis vector will capture a certain feature or "part" from the data corresponding to positive elements. Secondly, each data item is represented by an additive combinations of only a few "active" basis vectors whose contributions are weighted by the corresponding coefficients. For example, in a facial database, basis vectors contain facial parts such as several versions of mouths, noses, ... and a whole face is a combination of this parts. For these reasons, the non-negativity constraints are compatible with the intuitive notion of combining parts (basis vectors) to form a whole (a data item), which is how NMF learns a parts-based representation. NMF has been successfully applied in many applications such as text mining (Shahnaz et al., 2006; Berry et al., 2006), image processing (Lee and Seung, 1999), bioinformatics (Kim and Park, 2007), spectral data analysis (Pauca et al., 2006; Berry et al., 2006), etc.

The tightness of approximation (3.1) can be assessed using various measures such as the Frobenius norm, the Kullback-Leibler divergence (Lee and Seung, 1999, 2001), the Bregman divergence (Dhillon and Sra, 2006), the Earth Mover's distance metric, and many more (see also (Cichocki et al., 2009) for a comprehensive survey). In this chapter, we focus on the most widely used measure which is the Frobenius norm $\|A - UV^T\|_F^2 = \sum_{ij}(A - UV^T)_{ij}^2$. The NMF problem is then formulated as the following optimization problem

$$\begin{aligned} \min \quad & F(U,V) := \frac{1}{2}\|A - UV^T\|_F^2 \\ s.t \quad & U \in \mathbb{R}_+^{m\times r},\ V \in \mathbb{R}_+^{n\times r}. \end{aligned} \tag{3.2}$$

Problem (3.2) is a non-convex optimization with respect to the pair (U, V), and it has been shown that solving NMF is NP-hard (Vavasis, 2009). However, problem (3.2) reduces to an efficiently solvable convex nonnegativity-constrained least squares problem (NNLS) when one of the factors U or V is fixed. Exploiting this fact, most of algorithms proposed for NMF share the following general framework

1. Initialize (U^0, V^0) with nonnegative elements.
2. For $k = 0, 1, 2, \ldots$ do
   (a) Fix U^k and find $V^{k+1} \geq 0$ by solving

$$\min_{V\geq 0} \frac{1}{2}\|A - U^k V^T\|_F^2. \tag{3.3a}$$

   (b) Fix V^{k+1} and find U^{k+1} by solving

$$\min_{U\geq 0} \frac{1}{2}\|A^T - V^{k+1} U^T\|_F^2. \tag{3.3b}$$

Alternatively, one may exchange the order of updating U and V. There are two different ways to compute U and V at each iteration. A natural way is to compute optimal solutions to the NNLS subproblems (3.3a) and (3.3b), which leads to a class of algorithms called alternating nonnegative least squares (ANLS), e.g. the projected gradient method (Lin, 2007b), the projected quasi-Newton method (Kim et al., 2007), the active set method (Kim and Park, 2008), and the block principal pivoting method (Kim and Park, 2011). However, these methods are quite expensive and complicated to implement. Recently, (Guan et al., 2012) proposed the so called NeNMF, which applies Nesterov's optimal gradient method to solve the NNLS subproblems. In (Gong and Zhang, 2012), the NNLS subproblems are solved by using a second-order algorithm called projected Newton method (PNM). Another method which was proposed in (Huang et al., 2014) uses a quadratic regularization projected Barzilai-Borwein (QRPBB) method to solve the NNLS subproblems.

Several algorithms only compute an approximate solution to the NNLS subproblem, sometimes very roughly, but with a cheaper computational cost. One of such a procedures is multiplicative updating (MU) algorithm proposed by Lee and Seung in their seminal papers (Lee and Seung, 1999, 2001). This algorithm has been one of the most commonly used for NMF, but there are some issues related to its performance and convergence (Gonzalez and

Zhang, 2005; Lin, 2007a,b; Kim et al., 2007). (Lin, 2007a) presented a modified version of MU to fix its convergence issue. However, the modification is more costly and it also has slow convergence rate. (Berry et al., 2006) gave a simple way to treat the NNLS problem. It first finds the solution to the unconstrained least squares problem then projects this solution onto the nonnegative orthant. This method is simple but it is unstable and lacks of convergence guarantee.

An algorithm that iteratively updates each column of V and U per iteration, called the Hierarchical Alternating Least Squares (HALS) algorithm or, under another name, Rank-one Residue Iteration (RRI), was introduced in (Cichocki et al., 2007; Cichocki and Phan, 2009; Ho, 2008; Gillis and Glineur, 2008). In fact, this method amounts to solving the NNLS subproblems (3.3a) and (3.3b) approximately by executing only one step the sequential coordinate-wise algorithm (Franc et al., 2005). By careful analyzing the computational cost needed at each iteration, (Gillis and Glineur, 2012) proposed an accelerated version of the HALS. The idea of this method is to repeat the sequential coordinate-wise algorithm several times with cheap additional cost. This is due to the fact that the computational cost of each sequential coordinate-wise update is much less than the computational cost of the preparation step.

Algorithms that simultaneously computes both factors U and V at each iteration have been also proposed. Works on this direction include (Chu et al., 2004; Lin, 2007b; Le Thi et al., 2014b).

In this chapter, we investigate DC programming and DCA and propose two approaches for solving the NMF problem. The first approach follows the general alternating framework and applies DCA for solving the NNLS subproblems. The alternating DCA based algorithm generates a solution that holds necessary conditions of a local optimum. This first approach provides a unified framework for several algorithms based on the alternating method. It can also be easily adopted to other extensions of NMF which have various applications. In the second approach, we apply directly DC programming and DCA to the NMF problem. This results in two algorithms that simultaneously computing both factors U and V at each iteration.

As aforementioned, NMF is a useful factorization technique for finding parts-based, linear representations of nonnegative data. However, it does not always result in parts-based representations since the nonnegativity constraints do not force enough sparsity. We propose to use in this chapter a sparsity-inducing function called capped-ℓ_1 (Peleg and Meir, 2008) for enhancing the sparsity of NMF factors and improve the factorization.

The rest of the chapter is organized as follows. In the next section, the alternative DCA based algorithm for computing NMF is described. Section 3.3 presents two algorithms based on DC programming and DCA for computing simultaneously two factors of the NMF at each iteration. Some extensions of NMF and their solution methods are considered in section 3.4. The numerical experiments are presented in section 3.5. Finally, section 3.6 considers the

sparse NMF problem.

3.2 The first approach: alternative DCA based algorithm for solving the NMF problem

As mentioned above, NMF algorithms following the alternating framework (3.3) will iteratively solve the subproblems (3.3a) and (3.3b). These subproblems take the form of a nonnegativity-constrained least squares (NNLS) problem with multiple right-hand side

$$\min\left\{\frac{1}{2}\|UV^T - A\|_F^2 : V \in \mathbb{R}_+^{n\times r}\right\},$$

which is separable with respect to rows of V. Hence, solving the last problem amounts to solving n NNLS problems with single right-hand side of the form

$$\min\left\{\frac{1}{2}\|Uv - a\|_2^2 : v \in \mathbb{R}_+^{r}\right\},$$

where v and a are transpose of rows of V and A accordingly.

In the next subsection, we will develop DCA based schemes for solving the NNLS problem with single right-hand side and then multiple right-hand side. Since the problem (3.3a) and (3.3b) are similar (by replacing A^T with A and changing the role of U and V), we will present the solution method of problem (3.3a).

3.2.1 DCA for solving the NNLS problem

3.2.1.1 DCA for solving the NNLS problem with a single right-hand side vector

Consider the NNLS problem with single right-hand side vector

$$\min\left\{f(v) = \frac{1}{2}\|Uv - a\|_2^2 : v \in \mathbb{R}_+^{r}\right\}, \tag{3.4}$$

where $U \in \mathbb{R}_+^{m\times r}$, $a \in \mathbb{R}^m$. By letting $Q = U^TU \in \mathbb{R}^{r\times r}$ and $q = U^Ta \in \mathbb{R}^r$, we have $f(v) = \frac{1}{2}v^TQv - \langle q, v\rangle + \frac{1}{2}\|a\|_2^2$.

The KKT conditions of problem (3.4) are given as follows

$$v \geq 0,\ \nabla f(v) \geq 0,\ \text{and } v \circ \nabla f(v) = 0, \tag{3.5}$$

where $\nabla f(v) = Qv - q = U^T(Uv - a)$. These conditions are equivalent to $I(v) = \emptyset$, where $I(v)$ is the set of the indices violating the KKT conditions (3.5) and it is given by

$$I(v) = \big\{1 \le i \le r : \min(v_i, \nabla_i f(v)) \neq 0\big\}. \tag{3.6}$$

Since $\nabla^2 f = Q$, the objective function f can be expressed as a DC function $f(v) = g(v) - h(v)$, where

$$g(v) = \frac{1}{2} v^T \mathrm{diag}(\rho) v, \quad h(v) = \frac{1}{2} v^T \mathrm{diag}(\rho) v - f(v),$$

and $\rho \in \mathbb{R}^r_{++}$ satisfies $Q \preceq \mathrm{diag}(\rho)$. Then problem (3.4) can be reformulated as a DC program

$$\min\{(g(v) + \chi_{\mathbb{R}^r_+}(v)) - h(x) : v \in \mathbb{R}^r\}. \tag{3.7}$$

Applying DCA to (3.7), at each iteration k, we have to determine

$$\overline{v}^k = \nabla h(v^k) = \mathrm{diag}(\rho) v^k - \nabla f(v^k),$$

and then compute the next iterate v^{k+1} by

$$\begin{aligned} v^{k+1} &= \arg\min \left\{ \frac{1}{2} v^T \mathrm{diag}(\rho) v - \langle \overline{v}^k, v \rangle : v \in \mathbb{R}^r_+ \right\} \\ &= \arg\min \left\{ \frac{1}{2} \left\| v - \frac{[\overline{v}^k]}{[\rho]} \right\|^2 : v \in \mathbb{R}^r_+ \right\} \\ &= \left[\frac{[\overline{v}^k]}{[\rho]} \right]_+ = \left[v^k - \frac{[\nabla f(v^k)]}{[\rho]} \right]_+ . \end{aligned}$$

For convenience, we drop the index k, then the above update procedure can be briefly described by

$$v \longleftarrow v^+ = \left[v - \frac{[\nabla f(v)]}{[\rho]} \right]_+ . \tag{3.8}$$

A common choice of ρ is $\rho = \lambda \mathbf{1}_{r\times 1}$ such that $\lambda \ge \|Q\|_2$ (Pham Dinh and Le Thi, 1998). In this case, the update rule (3.8) can be interpreted as a projected gradient scheme with the step size $\frac{1}{\lambda}$. Note that if the index $i \notin I(v)$, i.e. $\min(v_i, \nabla_i f(v)) = 0$, then the value of v_i does not change under the update rule (3.8) for any choice of $\rho > 0$. Thus, at each iteration, we do not need to consider variables v_i with $i \notin I(v)$. The following result will give us a way to choose ρ more flexibly.

Lemma 3.1 *Given an $r \times r$ symmetric matrix M and a vector $d \in \mathbb{R}^r_{++}$. Then*

$$M \preceq \mathrm{diag}\left(\frac{[|M|d]}{[d]} \right).$$

Proof : For any $v \in \mathbb{R}^r$, we have

$$\begin{aligned} v^T \operatorname{diag}\left(\frac{[|M|d]}{[d]}\right) v &= \sum_i \frac{[|M|d]_i}{d_i} v_i^2 = \sum_{i,j} |M_{ij}| v_i^2 \frac{d_j}{d_i} = \sum_{i,j} |M_{ij}| \frac{1}{2}\left(v_i^2 \frac{d_j}{d_i} + v_j^2 \frac{d_i}{d_j}\right) \\ &\geq \sum_{i,j} |M_{ij}||v_i v_j| \geq \sum_{i,j} M_{ij} v_i v_j = v^T M v. \end{aligned}$$

Thus, $M \preceq \operatorname{diag}\left(\frac{[|M|d]}{[d]}\right)$. □

From Lemma 3.1, we can choose $\rho = \frac{[\max(|Q|d,\delta)]}{[d]}$, where $d \in \mathbb{R}^r_{++}$ and $\delta > 0$ is a safety parameter. Then the update rule (3.8) becomes

$$v \longleftarrow v^+ = \left[v - d \circ \frac{[\nabla f(v)]}{[\max(|Q|d,\delta)]}\right]_+ . \tag{3.9}$$

Note that the update rule (3.9) is well-defined for any choice of $d \geq 0$. As mentioned above, we do not need to compute all elements of v but only a few (corresponding to nonzero elements of d). Moreover, we have the following result.

Lemma 3.2 *Suppose that $d \in \mathbb{R}^r_+$ satisfies the condition*

$$I(v) \cap supp(d) \neq \emptyset \quad \text{if} \quad I(v) \neq \emptyset. \tag{3.10}$$

Then if $v \in \mathbb{R}^r_+$ is not a critical point of problem (3.4), we have $f(v^+) < f(v)$, where v^+ is computed via the update rule (3.9).

Proof : The assumption that v is not a critical point of problem (3.4) implies that $I(v) \neq \emptyset$. Considering the DC program (3.7) restricting on variables $\{v_i : i \in I(v) \cap \operatorname{supp}(d)\}$, the assertion of Lemma 3.2 is a consequence of the general convergence properties of DCA.

□

Remark 3.1 *Consider the following general form of problem (3.5)*

$$\min\left\{f(v) = \frac{1}{2} v^T Q v - \langle q, v\rangle + c : v \in \mathbb{R}^r_+\right\}, \tag{3.11}$$

where $Q \in \mathbb{R}^{r\times r}$ is an symmetric matrix, $q \in \mathbb{R}^r$ and $c \in \mathbb{R}$. Then the update rule (3.9) and Lemma 3.2 are still valid for (3.11). We can also replace Q in the update rule (3.9) by a symmetric matrix $\overline{Q}$ such that $Q \preceq \overline{Q}$.

Since $U \geq 0$, by rewriting (3.9), an update rule for problem (3.4) has the form

$$v \longleftarrow v^+ = \left[v - d \circ \frac{[U^T U v - U^T a]}{[\max(U^T U d, \delta)]} \right]_+, \quad \delta > 0. \tag{3.12}$$

The construction of d would have important effect on the performance of the resulting algorithm. In the sequel, we propose a specific construction of d that will be used in this work.

A construction of d. For $v \in \mathbb{R}^r_+$, define

$$\bar{v} = \left[v - \frac{[\nabla f(v)]}{[\max(\mathrm{diag}(U^T U)], \delta)} \right]_+, \quad \Delta v = v - \bar{v} = \min\left(v, \frac{[\nabla f(v)]}{[\max(\mathrm{diag}(U^T U), \delta)]} \right).$$

Then $\bar{v}_t$ $(t = 1, \ldots, r)$ is the updated value of v_t by applying the update rule (3.12) with $d = e_t$ – the tth canonical basis vector in $\mathbb{R}^r$. And $(\Delta v)_t$ is the corresponding displacement. Define $\tilde{v}^t \in \mathbb{R}^r_+$ $(t = 1, \ldots, r)$ by $\tilde{v}^t_t = \bar{v}_t$ and $\tilde{v}^t_j = v_j$ for all $j \neq t$. For $t = 1, \ldots, r$, if we only update the tth component of v, the objective function f will decrease an amount of

$$(\Delta f)_t = f(\tilde{v}^t) - f(v) = \nabla_{v_t} f(v)(\Delta v)_t - \frac{1}{2}(U^T U)_{tt}(\Delta v)_t^2 \geq 0.$$

Let

$$t(v) = \arg\max\{(\Delta f)_t : t = 1, \ldots, r\}$$

and

$$J(v) := \left\{ 1 \leq t \leq r : f\left(\frac{\tilde{v}^{t(v)} + \tilde{v}^t}{2} \right) - f(v) \geq (\Delta f)_{t(v)} \right\}. \tag{3.13}$$

Let $\Delta f^0 = (\Delta f)_{t(v^0)}$, where v^0 is the starting point, and $\epsilon > 0$ is a small positive number. Then the vector $d \equiv d(U, v) \in \mathbb{R}^r_+$ in the update rule (3.9) is defined by

$$d_t = \begin{cases} (\Delta f)_t & \text{if } t \in J(v) \text{ and } (\Delta f)_{t(v)} > \epsilon \Delta f^0 \\ 0 & \text{otherwise} \end{cases}, \quad \forall t = 1, \ldots, r. \tag{3.14}$$

It is clear that if $\epsilon < 1$ and $v = v^0$, d defined by (3.14) satisfies the condition (3.10).

3.2.1.2 DCA for solving the NNLS problem with multiple right-hand side vectors

Now, we consider the NNLS problem with multiple right-hand side vectors

$$\min_{V \geq 0} F_U(V) := \frac{1}{2}\|A - UV^T\|_F^2, \tag{3.15}$$

where $U \in \mathbb{R}_+^{m\times r}$, $A \in \mathbb{R}^{m\times n}$ and $V \in \mathbb{R}^{n\times r}$.

Note that, by letting $a = \text{vec}(A)$, $v = \text{vec}(V^T)$ and $\mathbf{U} = I_n \otimes U$, we can express

$$F_U(V) = \frac{1}{2}\|\mathbf{U}v - a\|^2.$$

Thus, the NNLS problem with multiple right-hand side (3.15) can be cast as a NNLS problem with single right-hand side. However, we have to deal with a problem with $n \times r$ variables that is typically large. In fact, the objective function in (3.15) can be decoupled as follows

$$\frac{1}{2}\|A - UV^T\|_F^2 = \sum_{i=1}^{n} \frac{1}{2}\|a_i - Uv_i\|^2,$$

where $a_i = A_{:i}$ and $v_i = V_{i:}^T$, $\forall i = 1, \ldots, n$. Therefore, solving (3.15) amounts to solving n *separate problems* of the form (3.4), namely

$$\min\{f_i(v_i) := \frac{1}{2}\|Uv_i - a_i\|^2 : v_i \in \mathbb{R}_+^r\} \quad \forall i = 1, \ldots, n.$$

Let $d_i \in \mathbb{R}_+^r$ be computed by (3.14) with $f = f_i$ and $v = v_i$, then (3.12) gives us the update rule

$$v_i \longleftarrow v_i^+ = \left[v_i - d_i \circ \frac{[U^TUv_i - U^Ta_i]}{[\max(U^TUd_i, \delta)]}\right]_+, \quad \forall i = 1, \ldots, n.$$

Let $D \in \mathbb{R}_+^{n\times r}$ such that $D_{i:} = d_i^T$ $\forall i = 1, \ldots, n$. By taking transpose on both sides, the above update rule can be rewritten as

$$V_{i:} \longleftarrow V_{i:}^+ = \left[V_{i:} - D_{i:} \circ \frac{[V_{i:}U^TU - A_{i:}U^T]}{[\max(D_{i:}U^TU, \delta)]}\right]_+, \quad \forall i = 1, \ldots, n,$$

or under the matrix form

$$V \longleftarrow V^+ = \left[V - D \circ \frac{[\nabla F_U(V)]}{[\max(DU^TU, \delta)]}\right]_+, \tag{3.16}$$

where $\nabla F_U(V) = VU^TU - A^TU$.

The complete description of using DCA for solving (3.15) is summarized in algorithm DCA-NNLS. The following assertion is a consequence of Lemma 3.2.

Lemma 3.3 *Suppose that $D \in \mathbb{R}_+^{n\times r}$ verifies the condition*

$$supp(D) \cap I(V) \neq \emptyset \text{ if } I(V) \neq \emptyset, \tag{3.17}$$

where $I(V) = \{(i,j) : 1 \leq i \leq n, 1 \leq j \leq r$ and $\min(V_{ij}, (\nabla F_U(V))_{ij}) \neq 0\}$, and V^+ is computed via the update rule (3.16). Then if V is not a critical point of (3.15), i.e. $I(V) \neq \emptyset$, we have $F_U(V^+) < F_U(V)$, otherwise $V^+ = V$.

Consequently, let V^0, V^* be the input and output of algorithm DCA-NNLS. If V^0 is not a solution of problem (3.15) then $F_U(V^*) < F_U(V^0)$, otherwise $V^* = V^0$.

DCA-NNLS: DCA for solving the NNLS problem (3.15)
Input: $A \in \mathbb{R}^{m\times n}$, $U \in \mathbb{R}_+^{m\times r}$, $V^0 \in \mathbb{R}_+^{n\times r}$, $\delta > 0$ and $N_{iter} > 0$
Output: $V^* \in \mathbb{R}_+^r$ is an approximate solution to problem (3.15)

Compute $M = U^TU$, $N = A^TU$ and let $l = 1$
while $l \leq N_{iter}$ **do**
  Compute $\nabla F_U(V^{l-1}) = V^{l-1}M - N$
  Compute $D^l \in \mathbb{R}_+^{n\times r}$ row by row using (3.14)
  if $D^l \neq 0$ **then**
    Compute $V^l = \left[V^{l-1} - D^l \circ \frac{[\nabla F_U(V^{l-1})]}{[\max(D^lM,\delta)]}\right]_+$
  else
    break
  end if
  $l = l + 1$
end while
$V^* = V^l$

Link with the existing methods. Below, we show that with appropriate choices of D, we can recover update rules for solving the NNLS problem (3.15) that were used in some existing NMF/matrix fatorization algorithms.

1. If $A \geq 0$, we take $D = V$ and let the safety parameter $\delta = 0$, the update rule (3.16) becomes

$$V \longleftarrow V^+ = \left[V - V \circ \frac{[VU^TU - A^TU]}{[VU^TU]}\right]_+ = V \circ \frac{[A^TU]}{[VU^TU]}. \tag{3.18}$$

This rule is the multiplicative update rule of Lee and Seung (Lee and Seung, 2001). Note that this choice of D may not satisfy the condition (3.17). This situation occurs when $V_{ij} = 0$ for any $(i, j) \in I(V)$, and then the algorithm is stuck at a non-critical point. We can overcome this issue by taking D as follows

$$D_{ij} = \begin{cases} V_{ij} & \text{if } \nabla F_U(V)_{ij} \geq 0, \\ \max(V_{ij}, \delta) & \text{otherwise,} \end{cases} \tag{3.19}$$

with a $\delta > 0$. It is easy to see that $D_{ij} > 0$ for any $(i, j) \in I(V)$. The corresponding update rule is

$$V \longleftarrow V^+ = \left[V - D \circ \frac{[\nabla F_U(V)]}{[\max(DU^TU, \delta)]}\right]_+ = V - D \circ \frac{[\nabla F_U(V)]}{[\max(DU^TU, \delta)]}. \tag{3.20}$$

By replacing $\max(DU^TU, \delta)$ with $DU^TU + \delta\mathbf{1}_{n\times r}$, we get back the update rule proposed in (Lin, 2007a).

2. For $j = 1, \ldots, r$, by successively letting $D = D^j := \mathbf{1}_{n\times 1} e_j^T$, where $\{e_1, \ldots, e_r\}$ is the canonical basis of $\mathbb{R}^r$, and conducting the update rule (3.16), we get back the update rule used in the RRI/HALS algorithm (Ho, 2008; Cichocki and Phan, 2009). This means that, at each iteration of RRI/HALS, we do not consider entirely V at a time but column by column. Moreover, at each iteration of RRI/HALS, at least one of D^j $(j = 1, \ldots, r)$ will satisfy the condition (3.17), so we also have the assertion similar to Lemma 3.3 if V is not a critical point of (3.15) then $F_U(V^+) < F_U(V)$, where V^+ is the updated value of V after the iteration.

3. Note that in (3.15), we do not impose any assumption on the nonnegativity of A. Thus, the update rule (3.16) (or (3.8) for more generality) is applicable to a more general factorization problem called nonnegative factorization (NF) (Gillis and Glineur, 2008) where we desire to approximate an arbitrarily real matrix A by the product UV^T of nonnegative matrices U and V. By taking $D = V$ and $\delta = 0$, we get the multiplicative rule

$$V \longleftarrow V^+ = \left[V - V \circ \frac{[VU^TU - A^TU]}{[VU^TU]}\right]_+ = V \circ \frac{[A^TU]_+}{[VU^TU]}.$$

Moreover, if $A = B - C$ with $B, C \geq 0$, we can replace $\frac{[V]}{[VU^TU]}$ in the equation above with $\frac{[V]}{[VU^TU+C^TU]}$ [2]. Then the update rule becomes

$$V \longleftarrow V^+ = \left[V - V \circ \frac{[VU^TU - B^TU^T + C^TU]}{[VU^TU + C^TU]}\right]_+ = V \circ \frac{[B^TU]}{[VU^TU + C^TU]}.$$

This update rule was also introduced in (Gillis and Glineur, 2008).

3.2.2 Alternating DCA for computing NMF and convergence analysis

The alternating DCA based algorithm for solving the NMF problem (3.2) is described in algorithm ADCA.

ADCA: Alternative DCA based algorithm for computing NMF

Initialize (U^0, V^0) with nonnegative elements, $\delta > 0$, $N_{iter} > 0$, $k \leftarrow 0$

repeat

1. Compute $V^{k+1} = \text{DCA-NNLS}(A, U^k, V^k, \delta, N_{iter})$
2. Compute $U^{k+1} = \text{DCA-NNLS}(A^T, V^{k+1}, U^k, \delta, N_{iter})$
3. $k \leftarrow k + 1$

until Stopping criterion is satisfied.

2. This is equivalent to taking $\rho = \frac{[U^TUv+U^Tc]}{[v]}$ with $c \geq 0$ in (3.8). Since $U^Tc \geq 0$, we will have $U^TU \preceq \text{diag}(\rho)$

In the sequel, we will prove that any limit point of the sequence $\{(U^k, V^k)\}$ generated by algorithm ADCA is a critical point of problem (3.2), i.e. a point that satisfies the KKT conditions. Formally, the KKT optimality conditions for the problem (3.2) are given as follows

$$U \geq 0,\ \nabla_U F(U,V) \geq 0,\ U \circ \nabla_U F(U,V) = 0, \tag{3.21a}$$
$$V \geq 0,\ \nabla_V F(U,V) \geq 0,\ V \circ \nabla_V F(U,V) = 0, \tag{3.21b}$$

where

$$\nabla_U F(U,V) = UV^TV - AV,\ \nabla_V F(U,V) = VU^TU - A^TU.$$

This means that (U^*, V^*) is a KKT point of the problem (3.2) if and only if V^* and U^* are KKT points of the problems of the form (3.15) $\min_{V\geq 0} F(U^*, V)$ and $\min_{U\geq 0} F(U, V^*)$ respectively.

First, we derive a general result on convergence.

Theorem 3.1 *Consider the optimization problem*

$$\min_{x\in\Omega} f(x), \tag{P}$$

where f is a continuous real-valued function taken on the closed convex set $\Omega \in \mathbb{R}^n$. Suppose that $\{x^k\} \subset \Omega$ is a sequence satisfying the following conditions:

(a) The sequence $\{f(x^k)\}$ is monotonically decreasing,

(b) For any subsequence $\{x^{k_l}\}_l$, there exists a continuous function $\varphi : \Omega \to \Omega$ and a sub-subsequence $\{x^{k_{l_t}}\}_t$ such that

$$x^{k_{l_t}+1} = \varphi(x^{k_{l_t}}), \quad \forall t,$$

and $f(\varphi(x)) < f(x)$ if x is not a critical point of the problem (P).

Then, every limit point of $\{x^k\}$ is a critical point of the problem (P).

Proof : Assume that x^* is a limit point of $\{x^k\}$ and x^* is not a critical point. Then, there is a subsequence $\{x^{k_l}\}_l$ such that $\lim_{l\to\infty} x^{k_l} = x^*$. By the assumption, there is a function φ and a sub subsequence $\{x^{k_{l_t}}\}_t$ that satisfy the condition (b).

Using the fact that f is continuous and the condition (a), we have

$$f(x^*) = \lim_{l\to\infty} f(x^{k_l}) = \inf_k f(x^k).$$

Besides, we have $x^{k_{l_t}+1} = \varphi(x^{k_{l_t}}) \longrightarrow \varphi(x^*)$ as $t \to \infty$. Thus, $\varphi(x^*)$ is also a limit point of $\{x^k\}$, and we also have $f(\varphi(x^*)) = \inf_k f(x^k)$. But, the condition (b) and the assumption that x^* is not a critical point imply that $f(\varphi(x^*)) < f(x^*)$. We have a contradiction. The theorem has been proved. □

Proposition 3.1 *There is a finite set $\mathscr{V}$ of continuous functions from $\mathbb{R}_+^{m\times r}\times\mathbb{R}_+^{n\times r}\mapsto\mathbb{R}_+^{n\times r}$ such that for any $U\in\mathbb{R}_+^{m\times r}$, $V^0\in\mathbb{R}_+^{n\times r}$, if $V^*\in\mathbb{R}_+^{n\times r}$ is computed using algorithm DCA-NNLS, then $V^*=\Phi^V(U,V^0)$ for a $\Phi^V\in\mathscr{V}$.*

Proof : It is easy to see that the functions φ_{ij} $(i=1,\dots,m;j=1,\dots,r)$ defined by

$$\varphi_{ij}(U,V)=(\nabla F_U(V))_{ij}(\Delta V)_{ij}-\frac{1}{2}(U^TU)_{jj}(\Delta V)_{ij}^2,$$

where $(\Delta V)_{ij}=\min\left(V_{ij},\frac{(\nabla F_U(V))_{ij}}{[\max((U^TU)_{jj},\delta)]}\right)$, are continuous functions from $\mathbb{R}_+^{m\times r}\times\mathbb{R}_+^{n\times r}\mapsto\mathbb{R}_+$.

Let $\mathscr{D}$ be the set of functions from $\mathbb{R}_+^{m\times r}\times\mathbb{R}_+^{n\times r}\mapsto\mathbb{R}_+^{n\times r}$ given by

$$\mathscr{D}=\left\{\Phi^D=(\Phi_{ij}^D):\Phi_{ij}^D\in\{0,\varphi_{ij}\}\right\}.$$

Then $\mathscr{D}$ is a finite set of continuous functions. Moreover, it is clear that the function T defined by

$$T(U,V,D)=\left(V-D\circ\frac{[\nabla F_U(V)]}{[\max(DU^TU,\delta)]}\right)_+$$

is continuous w.r.t. (U,V,D). Therefore,

$$\mathscr{V}=\left\{\Phi^V=\Phi^{(l)}:1\le l\le N_{iter},\Phi(U,V)=T(U,V,\Phi^D(U,V))\text{ for some }\Phi^D\in\mathscr{D}\right\},$$

where $\Phi^{(l)}$ stands for the composite function $\Phi\circ\cdots\circ\Phi$ (l times), is a finite set of continuous functions from $\mathbb{R}_+^{m\times r}\times\mathbb{R}_+^{n\times r}\mapsto\mathbb{R}_+^{n\times r}$.

By the construction of DCA-NNLS, if $U\in\mathbb{R}_+^{m\times r}$, $V^0\in\mathbb{R}_+^{n\times r}$ and $V^*\in\mathbb{R}_+^{n\times r}$ are the inputs and output of DCA-NNLS, there is a $\Phi^V\in\mathscr{V}$ such that $V^*=\Phi^V(U,V^0)$. □

Theorem 3.2 *Every limit point (U^*,V^*) of the sequence $\{(U^k,V^k)\}_k$ generated by algorithm ADCA is a critical point of the NMF problem (3.2).*

Proof : According to Lemma 3.3, the sequence $\{(U^k,V^k)\}_k$ generated by algorithm ADCA verifies the condition (a) of Theorem 3.1.

By Proposition 3.1, there are finite families $\mathscr{U}$ and $\mathscr{V}$ of continuous functions from $\mathbb{R}_+^{n\times r}\times\mathbb{R}_+^{m\times r}\mapsto\mathbb{R}_+^{m\times r}$ and from $\mathbb{R}_+^{m\times r}\times\mathbb{R}_+^{n\times r}\mapsto\mathbb{R}_+^{n\times r}$ respectively such that, for $k=0,1,2,\dots,$

$$V^{k+1}=\Phi_V^k(U^k,V^k),\quad U^{k+1}=\Phi_U^k(V^{k+1},U^k),$$

for some $\Phi_V^k\in\mathscr{V}$, $\Phi_U^k\in\mathscr{U}$.

For any $k = 0, 1, 2, \ldots$, define a continuous function $\Upsilon^k : \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$ by

$$\Upsilon^k(U, V) = \Phi_U^k(\Phi_V^k(U, V), U).$$

Then we have

$$(U^{k+1}, V^{k+1}) = \Upsilon^k(U^k, V^k), \quad \forall k = 0, 1, 2, \ldots$$

Since the sets $\{\Phi_U^k : k = 0, 1, 2, \ldots\}$ and $\{\Phi_V^k : k = 0, 1, 2, \ldots\}$ are finite, and so is the set $\{\Upsilon^k : k = 0, 1, 2, \ldots\}$. This implies that the sequence $\{(U^k, V^k)\}_k$ generated by algorithm ADCA verifies the condition (b) of Theorem 3.1. By applying Theorem 3.1, we have conclusion. □

We have considered a very specific construction of D. However, the above convergence analysis is also applicable to other D, as long as it satisfies the condition (3.17) and the assumption: there is a finite sets of continuous function $\mathscr{D}$ from $\mathbb{R}_+^{m\times r} \times \mathbb{R}_+^{n\times r} \mapsto \mathbb{R}_+^{n\times r}$ such that each D^l in algorithm DCA-NNLS can be expressed as $D^l = \Phi^D(U, V^l)$ for a $\Phi^D \in \mathscr{D}$. The latter assumption simply requires that each element of D is computed from U and V via a finite number of formulations. It is easy to see that $D = V$, $D = \mathbf{1}_{n\times 1} e_j^T$ and D constructed by (3.19) satisfy this assumption. The two latter constructions of D also verify the condition (3.17), so convergence of the corresponding NMF algorithms is justified.

3.3 The second approach: DCA applied on the whole NMF problem

3.3.1 The first DCA based algorithm for the whole NMF problem

We first show that finding solution to problem (3.2) can be restricted to a bounded region.

Theorem 3.3 *The NMF problem (3.2) has a solution (U, V) such that*

$$\|U\|_F \le \mathbf{b}, \quad \|V\|_F \le \mathbf{b}. \tag{3.22}$$

where $\mathbf{b} = (\sqrt{r}\|A\|_2)^{1/2}$.

Proof : Suppose that U, V satisfy the KKT conditions (3.21). Then we have

$$0 = \sum_{ij} (U \circ \nabla_U F(U, V))_{ij} = \langle U, \nabla_U F(U, V)\rangle = \langle UV^T, UV^T - A\rangle.$$

Thus,

$$\|UV^T\|_F^2 = \langle A, UV^T\rangle \le \|A\|_2 \|UV^T\|_F \Longrightarrow \|UV^T\|_F^2 \le \|A\|_2^2.$$

Moreover,

$$\|UV^T\|_F^2 = \|\sum_{i=1}^r U_{:i}V_{:i}^T\|_F^2 = \sum_{i,j=1}^r \langle U_{:i}, U_{:j}\rangle\langle V_{:i}, V_{:j}\rangle \geq \sum_{i=1}^r \|U_{:i}\|^2\|V_{:i}\|^2.$$

By normalizing, we can always force $\|U_{:i}\| = \|V_{:i}\|$ for any $i = 1, \ldots, r$, and so $\|U\|_F = \|V\|_F$. Then we have

$$\|A\|_2^2 \geq \sum_{i=1}^r \|U_{:i}\|^4 \geq \frac{1}{r}\left(\sum_{i=1}^r \|U_{:i}\|^2\right)^2 = \frac{1}{r}\|U\|_F^4 = \frac{1}{r}\|V\|_F^4.$$

This implies the conclusion. □

By virtue of Theorem 3.3, instead of solving problem (3.2) on the whole space $U, V \geq 0$, we only solve on the subspace restricted by (3.22) and consider the problem

$$\min_{(U,V)\in\mathcal{S}_U\times\mathcal{S}_V} \quad F(U,V) := \frac{1}{2}\|A - UV^T\|_F^2, \tag{3.23}$$

where $\mathcal{S}_U = \left\{U \in \mathbb{R}_+^{m\times r} : \|U\|_F \leq \mathbf{b}\right\}$ and $\mathcal{S}_V = \left\{V \in \mathbb{R}_+^{n\times r} : \|V\|_F \leq \mathbf{b}\right\}$.

For any $(U,V) \in \mathbb{R}_+^{m\times r} \times \mathbb{R}_+^{n\times r}$ and for any $H \in \mathbb{R}^{m\times r}$ and $K \in \mathbb{R}^{n\times r}$ we have

$$F(U+H, V+K) = F(U,V) + DF(U,V)[H,K] + \frac{1}{2}D^2F(U,V)[H,K] + o(\|H\|_F^2 + \|K\|_F^2)$$

where

$$DF(U,V)[H,K] = \langle \nabla_U F(U,V), H\rangle + \langle \nabla_V F(U,V), K\rangle,$$

and

$$D^2F(U,V)[H,K] = \langle V^TV, H^TH\rangle + 2\langle UV^T - A, HK^T\rangle + 2\langle UK^T, HV^T\rangle + \langle U^TU, K^TK\rangle$$

are the first-order and the second-order derivatives of F respectively.

For any $(U,V) \in \mathcal{S}_U \times \mathcal{S}_V$, we have the estimation

$$\begin{aligned} D^2F(U,V)[H,K] &\leq \|V\|_F^2\|H\|_F^2 + (2\|U\|_F\|V\|_F + \|A\|_F)(\|H\|_F^2 + \|K\|_F^2) + \|U\|_F^2\|K\|_F^2 \\ &\leq (1 + 3\sqrt{r})\|A\|_2(\|H\|_F^2 + \|K\|_F^2). \end{aligned}$$

Therefore, for $\rho \geq (1 + 3\sqrt{r})\|A\|_2$, the function

$$h(U,V) = \frac{\rho}{2}(\|U\|_F^2 + \|V\|_F^2) - F(U,V)$$

is convex on the set $\mathcal{S} = \mathcal{S}_U \times \mathcal{S}_V$.
Let $g(U,V) = \frac{\rho}{2}(\|U\|_F^2 + \|V\|_F^2)$, we have a DC decomposition $g - h$ of F on $\mathcal{S}$ and corresponding DC program

$$\min\left\{g(U,V) - h(U,V) : (U,V) \in \mathcal{S}\right\}. \tag{3.24}$$

DCA applied to (3.24) consists of computing two sequences $\{(U^k, V^k)\}$ and $\{(\overline{U}^k, \overline{V}^k)\}$ with

$$\begin{aligned}
\overline{U}^k &= \nabla_U h(U^k, V^k) = \rho U^k - \nabla_U F(U^k, V^k), && (3.25)\\
\overline{V}^k &= \nabla_V h(U^k, V^k) = \rho V^k - \nabla_V F(U^k, V^k), && (3.26)\\
(U^{k+1}, V^{k+1}) &\in \arg\min \left\{ \frac{\rho}{2}(\|U\|_F^2 + \|V\|_F^2) - \langle \overline{U}^k, U\rangle - \langle \overline{V}^k, V\rangle : (U,V) \in \mathcal{S} \right\}.
\end{aligned}$$

Computing (U^{k+1}, V^{k+1}) can be split into two problems separately as follows

$$\begin{aligned}
U^{k+1} &\in \arg\min_{U\in\mathcal{S}_U} \left\{ \frac{\rho}{2}\|U\|_F^2 - \langle \overline{U}^k, U\rangle \right\} = \arg\min_{U\in\mathcal{S}_U} \left\{ \frac{1}{2}\|U - \frac{1}{\rho}\overline{U}^k\|_F^2 \right\},\\
V^{k+1} &\in \arg\min_{V\in\mathcal{S}_V} \left\{ \frac{\rho}{2}\|V\|_F^2 - \langle \overline{V}^k, V\rangle \right\} = \arg\min_{V\in\mathcal{S}_V} \left\{ \frac{1}{2}\|V - \frac{1}{\rho}\overline{V}^k\|_F^2 \right\}.
\end{aligned}$$

Hence U^{k+1} (resp. V^{k+1}) is the projection of the point $\frac{1}{\rho}\overline{U}^k$ (resp. $\frac{1}{\rho}\overline{V}^k$) onto $\mathcal{S}_U$ (resp. $\mathcal{S}_V$) and can be explicitly express as follows (see Appendix A.1 for projection on a nonnegative ball)

$$U^{k+1} = P_{\mathcal{S}_U}\left(\frac{1}{\rho}\overline{U}^k\right) = \begin{cases} \frac{1}{\rho}[\overline{U}^k]_+ & \text{, if } \|[\overline{U}^k]_+\|_F \le \rho\mathbf{b} \\ \frac{\mathbf{b}}{\|[\overline{U}^k]_+\|_F}[\overline{U}^k]_+ & \text{, if } \|[\overline{U}^k]_+\|_F > \rho\mathbf{b} \end{cases}, \tag{3.27}$$

$$V^{k+1} = P_{\mathcal{S}_V}\left(\frac{1}{\rho}\overline{V}^k\right) = \begin{cases} \frac{1}{\rho}[\overline{V}^k]_+ & \text{, if } \|[\overline{V}^k]_+\|_F \le \rho\mathbf{b} \\ \frac{\mathbf{b}}{\|[\overline{V}^k]_+\|_F}[\overline{V}^k]_+ & \text{, if } \|[\overline{V}^k]_+\|_F > \rho\mathbf{b}. \end{cases} \tag{3.28}$$

DCA for solving the problem (3.23) is summarized in algorithm DCA1. The following theorem shows that the solutions given by DCA1 are really the (critical) solutions of the NMF problem (3.2).

DCA1: the first DCA for solving the whole NMF problem

Initialize (U^0, V^0) satisfied (3.22), $k \leftarrow 0$

repeat

1. Compute $(\overline{U}^k, \overline{V}^k)$ using (3.25), (3.26)
2. Compute (U^{k+1}, V^{k+1}) using (3.27), (3.28)
3. $k \leftarrow k+1$

until Stopping criterion is satisfied.

Theorem 3.4 *Every limit point generated by Algorithm 1 is a stationary point of the problem (3.2). Moreover, the sequence $\{U^k, V^k)\}$ generated by Algorithm 1 has at least one limit point.*

Proof : Suppose that (U^*, V^*) is a limit point generated by algorithm DCA1. Then, by general convergence of DCA, (U^*, V^*) is a critical point of (3.24). Since both g and h are

continuously differentiable, (U^*, V^*) is an ordinary KKT point of (3.23). Thus, there exist $\alpha, \beta \geq 0$ and $\mu \in \mathbb{R}_+^{m\times r}$, $\nu \in \mathbb{R}_+^{n\times r}$ such that

$$\begin{cases} U^* \geq 0, & V^* \geq 0, \\ \nabla_U F(U^*, V^*) + 2\alpha U^* = \mu, & \nabla_V F(U^*, V^*) + 2\beta V^* = \nu, \\ \mu \circ U^* = 0, & \nu \circ V^* = 0, \\ \|U^*\|_F^2 \leq \mathbf{b}^2, & \|V^*\|_F^2 \leq \mathbf{b}^2, \\ \alpha \cdot (\|U^*\|_F^2 - \mathbf{b}^2) = 0, & \beta \cdot (\|V^*\|_F^2 - \mathbf{b}^2) = 0. \end{cases} \tag{3.29}$$

It is clear that if $\alpha = \beta = 0$ then U^* and V^* satisfy the KKT conditions (3.21). Now we assume $\alpha > 0$. Then $\|U^*\|_F^2 = \mathbf{b}^2$. From the first three properties in (3.29), we have

$$2\alpha\|U^*\|_F^2 = -\langle U^*, \nabla_U F(U^*, V^*)\rangle = -\langle V^*, \nabla_V F(U^*, V^*)\rangle = 2\beta\|V^*\|_F^2.$$

Combining this and the fifth property in (3.29), we deduce that $\beta = \alpha > 0$ and $\|V^*\|_F^2 = \|U^*\|_F^2 = \mathbf{b}^2$. For any $i = 1, \ldots, r$, from the first three properties in (3.29), we have

$$2\alpha\|U_{:i}^*\|^2 = -\langle U_{:i}^*, (\nabla_U F(U^*, V^*))_{:i}\rangle = -\langle V_{:i}^*, (\nabla_V F(U^*, V^*))_{:i}\rangle = 2\beta\|V_{:i}^*\|^2.$$

Therefore, $\|U_{:i}^*\|^2 = \|V_{:i}^*\|^2$ for any $i = 1, \ldots, r$. Then

$$\begin{aligned} \|U^*(V^*)^T\|_F^2 &= \|\sum_{i=1}^r U_{:i}^*(V_{:i}^*)^T\|_F^2 = \sum_{i,j=1}^r \langle U_{:i}^*, U_{:j}^*\rangle\langle V_{:i}^*, V_{:j}^*\rangle \\ &\geq \sum_{i=1}^r \|U_{:i}^*\|^2\|V_{:i}^*\|^2 = \sum_{i=1}^r \|U_{:i}^*\|^4 \geq \frac{1}{r}(\sum_{i=1}^r \|U_{:i}^*\|^2)^2. \end{aligned}$$

That is $\|U^*\|_F^2 \leq \sqrt{r}\|U^*(V^*)^T\|_F$.

Moreover, we have

$$\|U^*(V^*)^T\|_F^2 - \langle A, U^*(V^*)^T\rangle = \langle U^*, \nabla_U F(U^*, V^*)\rangle = -2\alpha\|U^*\|_F^2 < 0.$$

So

$$\|U^*(V^*)^T\|_F^2 < \langle A, U^*(V^*)^T\rangle \leq \|A\|_F\|U^*(V^*)^T\|_F.$$

$$\Rightarrow \|U^*\|_F^2 \leq \sqrt{r}\|U^*(V^*)^T\|_F < \sqrt{r}\|A\|_F = \mathbf{b}^2.$$

We have a contradiction. Thus, $\alpha = \beta = 0$ and (U^*, V^*) satisfy the KKT conditions (3.21). The last conclusion is trivial due to the fact that the sequence $\{U^k, V^k)\}$ is bounded. □

3.3.2 The second DCA based algorithm for the whole NMF problem

In the previous section, a DCA based algorithm for computing all elements of U and V at each iteration is proposed. In this section, we also compute U and V simultaneously at each iteration, but we will deploy the variable selection strategy to improve DCA1.

For any $(U,V) \in \mathbb{R}_+^{m\times r} \times \mathbb{R}_+^{n\times r}$ and $(H,K) \in \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$ we have

$$\begin{aligned} D^2F(U,V)[H,K] &\leq 2\langle V^TV, H^TH\rangle + 2\langle UV^T - A, HK^T\rangle + 2\langle U^TU, K^TK\rangle \\ &\leq 2\langle V^TV, H^TH\rangle + 2\langle U^TU, K^TK\rangle + \|UV^T - A\|_2(\|H\|_F^2 + \|K\|_F^2) \\ &= \sum_{i=1}^{m} H_{i:}(2V^TV + \|UV^T - A\|_2 I_r)H_{i:}^T + \\ &\quad + \sum_{j=1}^{n} K_{j:}(2U^TU + \|UV^T - A\|_2 I_r)K_{j:}^T. \end{aligned}$$

For any $D^U \in \mathbb{R}_{++}^{m\times r}$ and $D^V \in \mathbb{R}_{++}^{n\times r}$, by Lemma 3.1, we have

$$V^TV \preceq \operatorname{diag}\left(\frac{[V^TV(D_{i:}^U)^T]}{[(D_{i:}^U)^T]}\right), U^TU \preceq \operatorname{diag}\left(\frac{[U^TU(D_{j:}^V)^T]}{[(D_{j:}^V)^T]}\right), \forall 1 \leq i \leq m, 1 \leq j \leq n.$$

Given $U^{\min}, U^{\max} \in \mathbb{R}_+^{m\times r}$, and $V^{\min}, V^{\max} \in \mathbb{R}_+^{n\times r}$ such that $U^{\min} \leq U^{\max}$ and $V^{\min} \leq V^{\max}$. Consider the sets

$$\begin{aligned} S_U &= \{U \in \mathbb{R}_+^{m\times r} : U^{\min} \leq U \leq U^{\max}\}, \\ S_V &= \{V \in \mathbb{R}_+^{n\times r} : V^{\min} \leq V \leq V^{\max}\}, \end{aligned}$$

and let

$$\rho = \max\{\|UV^T - A\|_2 : (U,V) \in S_U \times S_V\}. \tag{3.30}$$

Then, for all $(U,V) \in S_U \times S_V$ and $(H,K) \in \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r}$, we have

$$\begin{aligned} D^2F(U,V)[H,K] &\leq \sum_{i=1}^{m} H_{i:}\left(2\operatorname{diag}\left(\frac{[V^TV(D_{i:}^U)^T]}{[(D_{i:}^U)^T]}\right) + \rho I_r\right)H_{i:}^T + \\ &\quad + \sum_{j=1}^{n} K_{j:}\left(2\operatorname{diag}\left(\frac{[U^TU(D_{j:}^V)^T]}{[(D_{j:}^V)^T]}\right) + \rho I_r\right)K_{j:}^T \\ &= \left\langle H, \frac{[2\,D^UV^TV + \rho D^U]}{[D^U]} \circ H\right\rangle + \left\langle K, \frac{[2\,D^VU^TU + \rho D^V]}{[D^V]} \circ K\right\rangle \\ &\leq \left\langle H, \frac{[2\,D^U(V^{\max})^TV^{\max} + \rho D^U]}{[D^U]} \circ H\right\rangle + \left\langle K, \frac{[2\,D^V(U^{\max})^TU^{\max} + \rho D^V]}{[D^V]} \circ K\right\rangle. \end{aligned}$$

Therefore, on $S_U \times S_V$, the objective function F of (3.2) has a DC decomposition given by $F(U,V) = G(U,V) - H(U,V)$, where

$$\begin{aligned} G(U,V) &= \frac{1}{2}\left\langle U, \frac{[2\,D^U(V^{\max})^T V^{\max} + \rho D^U]}{[D^U]} \circ U \right\rangle + \\ &\quad + \frac{1}{2}\left\langle V, \frac{[2\,D^V(U^{\max})^T U^{\max} + \rho D^V]}{[D^V]} \circ V \right\rangle, \\ H(U,V) &= G(U,V) - F(U,V) \end{aligned}$$

are convex functions on $S_U \times S_V$.

This leads to considering the DC program

$$\min\left\{G(U,V) - H(U,V) : (U,V) \in S_U \times S_V\right\}. \tag{3.31}$$

Applying DCA to (3.31), we compute the next iteration (U^+, V^+) from the current iteration (U,V) as follows.

- Compute $(\overline{U}, \overline{V}) = \nabla H(U,V)$. Specifically, we have

$$\begin{aligned} \overline{U} &= \frac{[2\,D^U(V^{\max})^T V^{\max} + \rho D^U]}{[D^U]} \circ U - \nabla_U F(U,V), \\ \overline{V} &= \frac{[2\,D^V(U^{\max})^T U^{\max} + \rho D^V]}{[D^V]} \circ V - \nabla_V F(U,V). \end{aligned}$$

- Then $(U^+, V^+) = \arg\min\{G(U,V) - \langle \overline{U}, U\rangle - \langle \overline{V}, V\rangle : (U,V) \in S_U \times S_V\}$. This is equivalent to

$$\begin{aligned} U^+ &= P_{[U^{\min}, U^{\max}]}\left(\frac{[D^U]}{[2\,D^U(V^{\max})^T V^{\max} + \rho D^U]} \circ \overline{U}\right), \\ V^+ &= P_{[V^{\min}, V^{\max}]}\left(\frac{[D^V]}{[2\,D^V(U^{\max})^T U^{\max} + \rho D^V]} \circ \overline{V}\right). \end{aligned}$$

These two steps can be compactly described by

$$U^+ = P_{[U^{\min}, U^{\max}]}\left(U - \frac{[D^U]}{[2\,D^U(V^{\max})^T V^{\max} + \rho D^U]} \circ \nabla_U F(U,V)\right), \tag{3.32}$$

$$V^+ = P_{[V^{\min}, V^{\max}]}\left(V - \frac{[D^V]}{[2\,D^V(U^{\max})^T U^{\max} + \rho D^V]} \circ \nabla_V F(U,V)\right). \tag{3.33}$$

Now, we discuss about determining the matrices D^U, D^V and bounds $U^{\min}, U^{\max}, V^{\min}$ and $V^{\max}$. Note that we have

$$0 \le \frac{D^U_{it}}{[2\,D^U(V^{\max})^T V^{\max} + \rho D^U]_{it}} \le \frac{1}{2\|V_{:t}\|^2}, \quad \forall 1 \le i \le m, 1 \le t \le r,$$

and similarly for V. This suggests that we can take

$$\begin{cases} U^{\min} = \min(U, \tilde{U}), & U^{\max} = \max(U, \tilde{U}) \\ V^{\min} = \min(V, \tilde{V}), & V^{\max} = \max(V, \tilde{V}), \end{cases} \tag{3.34}$$

where $\tilde{U} \in \mathbb{R}_+^{m\times r}$ and $\tilde{V} \in \mathbb{R}_+^{n\times r}$ are defined by

$$\tilde{U}_{:t} = \left[U_{:t} - \frac{(\nabla_U F(U,V))_{:t}}{2\|V_{:t}\|^2}\right]_+, \quad \tilde{V}_{:t} = \left[V_{:t} - \frac{(\nabla_V F(U,V))_{:t}}{2\|U_{:t}\|^2}\right]_+ \quad \forall 1 \le t \le r.$$

Then the projection in (3.32) is reduced to

$$U^+ = \left[U - \frac{[D^U]}{[2\, D^U (V^{\max})^T V^{\max} + \rho D^U]} \circ \nabla_U F(U,V)\right]_+. \tag{3.35}$$

And similarly,

$$V^+ = \left[V - \frac{[D^V]}{[2\, D^V (U^{\max})^T U^{\max} + \rho D^V]} \circ \nabla_V F(U,V)\right]_+. \tag{3.36}$$

For the matrices D^U and D^V, observing from (3.35) and (3.36), they do not need to be strictly positive. Instead, with the same reason in section 3.2.1.2, D^U and D^V satisfy

$$\text{supp}(D^U) \cap I \neq \emptyset \text{ if } I \neq \emptyset, \quad \text{and} \quad \text{supp}(D^V) \cap J \neq \emptyset \text{ if } J \neq \emptyset,$$

where $I = \{1 \le i \le m, 1 \le t \le r : \min(U_{it}, (\nabla_U F(U,V))_{it}) \neq 0\}$ and $J = \{1 \le j \le n, 1 \le t \le r : \min(V_{jt}, (\nabla_V F(U,V))_{jt}) \neq 0\}$.

For summary, we describe the algorithm for NMF problem in algorithm DCA2 below.

DCA2: the second DCA for solving the whole NMF problem

Initialize (U^0, V^0) with nonnegative elements, $k = 0$

repeat

1. Compute $U_k^{\min}, U_k^{\max}$ and $V_k^{\min}, V_k^{\max}$ by (3.34)
2. Compute $\rho^k = \max\{\|UV^T - A\|_2 : U_k^{\min} \le U \le U_k^{\max}, V_k^{\min} \le V \le V_k^{\max}\}$
3. Compute D_k^U, D_k^V row by row using (3.14)
4. Compute U^{k+1} and V^{k+1} by

$$\begin{aligned} U^{k+1} &= \left[U^k - \frac{[D_k^U]}{[2\, D_k^U (V_k^{\max})^T V_k^{\max} + \rho^k D_k^U]} \circ \nabla_U F(U^k, V^k)\right]_+ \\ V^{k+1} &= \left[V^k - \frac{[D_k^V]}{[2\, D_k^V (U_k^{\max})^T U_k^{\max} + \rho^k D_k^V]} \circ \nabla_V F(U^k, V^k)\right]_+. \end{aligned}$$

5. $k = k + 1$

until Stopping criterion is satisfied.

With the same arguments in Lemma 3.3, Proposition 3.1 and Theorem 3.2 and the fact that ρ^k in step 2 of algorithm DCA2 continuously depends on (U^k, V^k), we have the following convergence result of algorithm DCA2.

Theorem 3.5 *Let $\{(U^k, V^k)\}_k$ is the sequence generated by algorithm DCA2. Then*

i) $\{F(U^k, V^k)\}_k$ is monotonically decreasing.

ii) $F(U^{k+1}, V^{k+1}) < F(U^k, V^k)$ if (U^k, V^k) is not a critical point of the NMF problem (3.2).

iii) Every limit point (U^*, V^*) of the sequence $\{(U^k, V^k)\}_k$ is a critical point of the NMF problem (3.2).

In practice, computing ρ by (3.30) is difficult. We can replace it by an upper bound given by

$$\bar{\rho} = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \max\{(U_{i:}V_{j:}^T - A_{ij})^2 : U_{i:}^{\min} \leq U_{i:} \leq U_{i:}^{\max}, V_{j:}^{\min} \leq V_{j:} \leq V_{j:}^{\max}\} \right)^{1/2}. \quad (3.37)$$

Then the convergence result in Theorem 3.2 still holds.

3.4 Extension of the NMF problem

In this section, we present some extensions of NMF problem that can be solved by using an alternating framework similar to (3.3) and the method in Sect. 3.2.1 for solving subproblems. We also point out that some existing algorithms for solving these extensions are actually special cases of our method.

3.4.1 Constrained nonnegative matrix factorization

For various application purposes, additional constraints are incorporated to impose prior knowledge. To do this, regularization techniques are often used to extend the original problem (3.2) as follows

$$\min_{U,V \geq 0} F^r(U, V) := \frac{1}{2}\|A - UV^T\|_F^2 + \alpha J_1(U) + \beta J_2(V), \quad (3.38)$$

where the functions J_1 and J_2 are regularization terms enforcing certain dependent-application constraints on the solutions, and α, β are trade-off parameters.

3.4.1.1 Smooth regularization

Consider the smooth regularization, for example on the factor V, of the form

$$J_2(V) = \frac{1}{2}\langle L, VV^T\rangle,$$

where $L \in \mathbb{R}^{n\times n}$ is a regularization operator. If L is the identity matrix I_n, we have the well-known Tikhonov regularization (Berry et al., 2006; Pauca et al., 2006). Other choices than the identity for L include the graph Laplacian (Cai et al., 2008), the temporal smoothness operator (Chen and Cichocki, 2005). The update rule for V in this case amounts to solve the following problem

$$\min_{V\geq 0} f(V) = \frac{1}{2}\langle U^TU, V^TV\rangle - \langle A^TU, V\rangle + \frac{\beta}{2}\langle L, VV^T\rangle.$$

We can represent $f(V)$ by

$$\begin{aligned} f(V) &\equiv f(\mathrm{vec}(V)) \\ &= \frac{1}{2}\mathrm{vec}(V^T)^T(I_n\otimes U^TU)\mathrm{vec}(V^T) + \frac{\beta}{2}\mathrm{vec}(V)(I_r\otimes L)\mathrm{vec}(V) - \langle \mathrm{vec}(A^TU), \mathrm{vec}(V)\rangle \\ &= \frac{1}{2}\mathrm{vec}(V)^T\left(P^T(I_n\otimes U^TU)P + \beta(I_r\otimes L)\right)\mathrm{vec}(V) - \langle \mathrm{vec}(A^TU), \mathrm{vec}(V)\rangle, \end{aligned}$$

where $P \in \mathbb{R}^{nr\times nr}$ is a permutation matrix such that $P\mathrm{vec}(V) = \mathrm{vec}(V^T)$.

This problem is of the form (3.11). According to (3.9), for $D \in \mathbb{R}_+^{n\times r}$, an update rule for V will be

$$\mathrm{vec}(V) \longleftarrow \left[\mathrm{vec}(V) - \mathrm{vec}(D)\circ\frac{\left[Q\mathrm{vec}(V) - \mathrm{vec}(A^TU)\right]}{\left[\max(|Q|\,\mathrm{vec}(D), \delta)\right]}\right]_+, \tag{3.39}$$

where $Q = P^T(I_n\otimes U^TU)P + \beta(I_r\otimes L)$.

If L can be represented by $L = M - T$ where $M, T \geq 0$ are semi-positive symmetric matrices, we have

$$P^T(I_n\otimes U^TU)P + \beta(I_r\otimes L) \preceq P^T(I_n\otimes U^TU)P + \beta(I_r\otimes M).$$

As indicated in Remark 3.1, we can replace Q with $\overline{Q} = P^T(I_n\otimes U^TU)P + \beta(I_r\otimes M)$ in the denominator of (3.39). Thus, for $D \in \mathbb{R}_+^{n\times r}$, an update rule for V takes the form

$$\mathrm{vec}(V) \longleftarrow \left[\mathrm{vec}(V) - \mathrm{vec}(D)\circ\frac{\left[Q\mathrm{vec}(V) - \mathrm{vec}(A^TU)\right]}{\left[\max(|\overline{Q}|\,\mathrm{vec}(D), \delta)\right]}\right]_+. \tag{3.40}$$

Below, we will apply the update rules (3.39) and (3.40) for specific cases and show that they cover some existing methods.

a) Tikhonov regularization (Berry et al., 2006). If $L = I_n$, we have $J_2(V) = \frac{1}{2}\|V\|_F^2$ that is known as Tikhonov regularization. The update rule (3.39) becomes

$$\mathrm{vec}(V) \longleftarrow \left[\mathrm{vec}(V) - \mathrm{vec}(D) \circ \frac{\left[\mathrm{vec}(VU^TU) + \beta\mathrm{vec}(V) - \mathrm{vec}(A^TU)\right]}{\left[\max\left(\mathrm{vec}(VU^TU) + \beta\mathrm{vec}(V), \delta\right)\right]}\right]_+,$$

or under the matrix form,

$$V \longleftarrow \left[V - D \circ \frac{\left[VU^TU + \beta V - A^TU\right]}{\left[\max(VU^TU + \beta V, \delta)\right]}\right]_+.$$

By taking $D = V$ and $\delta = 0$, the preceding update rule becomes

$$V \longleftarrow V \circ \frac{[A^TU]}{[VU^TU + \alpha V]}.$$

This is the multiplicative update rule using in the algorithm CNMF (Berry et al., 2006).

b) Temporal smoothness constraint. (Chen and Cichocki, 2005) considered $L = \frac{1}{n}(I_n - T)^T(I_n - T)$, where $T \in \mathbb{R}_+^{n\times n}$ is a template operator. We can apply directly the update rules (3.39) and (3.40) by representing $L = \frac{1}{n}(I_n + T^TT) - \frac{1}{n}(T + T^T)$. In a special case, if $\beta > 0$ is sufficiently small such that $P^T(I_n \otimes U^TU)P + \beta(I_r \otimes L) \geq 0$, by taking $D = V$ and $\delta = 0$, (3.39) becomes

$$\begin{aligned}\mathrm{vec}(V) \quad &\longleftarrow \quad \mathrm{vec}(V) \circ \frac{[\mathrm{vec}(A^TU)]}{[(P^T(I_n \otimes U^TU)P + \beta(I_r \otimes L))\,\mathrm{vec}(V)]} \\ &= \quad \mathrm{vec}(V) \circ \frac{[\mathrm{vec}(A^TU)]}{[\mathrm{vec}(VU^TU) + \beta\mathrm{vec}(LV)]},\end{aligned}$$

or equivalently,

$$V \longleftarrow V \circ \frac{[A^TU]}{[VU^TU + \beta LV]}.$$

We obtain the update rule proposed in (Chen and Cichocki, 2005).

c) Graph Laplacian regularization. (Cai et al., 2008) considered $L = M - T$, where T is a symmetric edge weight matrix with $0 - 1$ elements and M is the diagonal matrix whose elements are column sums of T. Since M and T are nonnegative, we can apply the update rule (3.40). Specifically, if we take $D = V$ and $\delta = 0$, (3.40) becomes

$$\begin{aligned}\mathrm{vec}(V) \quad &\longleftarrow \quad \mathrm{vec}(V) \circ \frac{[\mathrm{vec}(A^TU) + \beta(I_r \otimes T)\mathrm{vec}(V)]}{[(P^T(I_n \otimes U^TU)P + \beta(I_r \otimes M))\,\mathrm{vec}(V)]} \\ &= \quad \mathrm{vec}(V) \circ \frac{[\mathrm{vec}(A^TU) + \beta\mathrm{vec}(TV)]}{[\mathrm{vec}(VU^TU) + \beta\mathrm{vec}(MV)]},\end{aligned}$$

or equivalently,

$$V \longleftarrow V \circ \frac{[A^TU + \beta TV]}{[VU^TU + \beta MV]}.$$

This is also the update rule proposed in (Cai et al., 2008).

3.4.1.2 Sparse regularization

Recall that one of important applications of NMF is to learn parts-base representations of nonnegative data. Although NMF itself usually produces sparse representations of data, it is useful to impose more sparsity constraint (Hoyer, 2002, 2004). We dedicate the subsection below to this topic.

3.4.1.3 Convex coding

In certain applications, one desires to find an approximation UV^T of an input matrix A such that $U, V \geq 0$ and $V\mathbf{1}_{r\times 1} = \mathbf{1}_{n\times 1}$. It means that each column of A is approximated by a convex combination of columns of U. Most of popular algorithms for NMF can not handle directly such constraints. To impose additivity to one on the rows of V, one adds to the objective function a penalty term of the form (Lee and Seung, 1997; Berry et al., 2006)

$$J_2(V) = \frac{1}{2}\|V\mathbf{1}_{r\times 1} - \mathbf{1}_{n\times 1}\|_F^2.$$

Then the subproblem for computing V has the form of (3.11). Of course, the full additivity is often not achieved. We are going to show in this section that we can directly deal with such constraints by using DC programming and DCA.

Consider the extension of NMF that involves additivity constraints on the coefficient as follows

$$\begin{aligned} \min \quad & F_1(U,V) = \frac{1}{2}\|A - UV^T\|_F^2, \\ s.t. \quad & U \in \mathbb{R}_+^{m\times r}, V \in \mathbb{R}_+^{n\times r}, \text{ and } V\mathbf{1}_{r\times 1} = \mathbf{1}_{n\times 1}. \end{aligned} \tag{3.41}$$

(3.42)

In the alternating framework, U is computed by solving a NNLS problem. While computing V amounts to solving a sequence of separate problems of the form

$$\min_{v\in\Delta^r} \quad f(v) = \frac{1}{2}\|Uv - a\|_2^2, \tag{3.43}$$

where Δ^r is the canonical simplex defined by

$$\Delta^r = \{v \in \mathbb{R}^r : v \geq 0, v^T\mathbf{1}_{r\times 1} = 1\}.$$

For $\rho \geq \|U^TU\|_2$, f has a DC decomposition given by $f(v) = g(v) - h(v)$, where

$$g(v) = \frac{\rho}{2}\|v\|_2^2, \quad h(v) = \frac{\rho}{2}\|v\|_2^2 - f(v).$$

DCA applied on this DC decomposition is iteratively processed as follows

- Compute $\overline{v} = \nabla h(v) = \rho v - \nabla f(v)$.
- Update

$$\begin{aligned} v \leftarrow v^+ &= \arg\min \left\{ \frac{\rho}{2}\|v\|_2^2 - \langle \overline{v}, v\rangle : v \in \Delta^r \right\} \\ &= \arg\min \left\{ \frac{1}{2}\left\| v - \frac{\bar{v}}{\rho}\right\|^2 : v \in \Delta^r \right\} \\ &= P_{\Delta^r}\left(\frac{\bar{v}}{\rho}\right). \end{aligned}$$

Note that, efficient algorithm for computing the projection of a point onto a simplex is available, see e.g. (Michelot, 1986).

3.4.2 Multilayer nonnegative matrix factorization

The multilayer nonnegative matrix factorization aims to represent a nonnegative data matrix A as a product of N nonnegative factor matrices $X_i \in \mathbb{R}^{m_i \times n_i}$ $(i = 1, \ldots, N)$ with compatible sizes (Cichocki and Zdunek, 2006). Measuring discrepancy by Euclidean distance, the multilayer nonnegative matrix factorization takes the form (Ho, 2008)

$$\min_{X_i \geq 0} F(X) = \frac{1}{2}\|A - X_1 X_2 \ldots X_N\|_F^2. \tag{3.44}$$

For $i = 1, 2, \ldots, N$, we set

$$U_i = X_1 X_2 \ldots X_i \quad \text{and} \quad V_i = X_N^T X_{N-1}^T \ldots X_i^T,$$

and $U_0 = I_m$ and $V_{N+1} = I_n$. Then, the objective function F in (3.44) can be expressed as

$$F(X) = \frac{1}{2}\|A - U_{i-1} X_i V_{i+1}^T\|_F^2, \quad \forall i = 1, \ldots, N.$$

According to alternating framework, we update X_i's as follows

for $i = 1, \ldots, N$ **do**
  Fix X_t's $(t \neq i)$, find X_i by solving $\min_{X_i \geq 0} F_i(X_i) := \frac{1}{2}\|A - U_{i-1} X_i V_{i+1}^T\|_F^2$.
end for

Similar to the usual NMF, the KKT conditions of the multiplayer NMF can be stated as follows

$$X_i \geq 0, \quad \nabla_{X_i} F(X) \geq 0, \quad X_i \circ \nabla_{X_i} F(X) = 0, \qquad \forall i = 1, \ldots, N,$$

where $\nabla_{X_i} F(X) = (U_{i-1}^T U_{i-1}) X_i (V_{i+1}^T V_{i+1}) - U_{i-1}^T A V_{i+1}$.

In the remaining of this section, we will discuss on the solution method for the subproblems $\min_{X_i \geq 0} F_i(X_i)$. These subproblems have the common form

$$\min_{X \geq 0} \mathcal{F}(X) = \frac{1}{2}\|A - CXB^T\|_F^2, \tag{3.45}$$

where $A \in \mathbb{R}^{m\times n}, C \in \mathbb{R}^{m\times k}, X \in \mathbb{R}^{k\times l}$ and $B \in \mathbb{R}^{n\times l}$.

Using the operator $\text{vec}(\cdot)$ and relation between the matrix product and the Kronecker product, we have (Van Loan, 2000)

$$\text{vec}(CXB^T) = (B \otimes C)\text{vec}(X).$$

Therefore,

$$\mathcal{F}(X) = \frac{1}{2}\|\text{vec}(A) - E\text{vec}(X)\|_F^2,$$

where $E = B \otimes C$. Therefore, solving the problem (3.45) amounts to solving a NNLS problem of the form (3.4). For computing gradient of $\mathcal{F}$, we have

$$\text{vec}(\nabla\mathcal{F}(X)) = (E^T E)\text{vec}(X) - E^T\text{vec}(A).$$

Since

$$E^T E = (B \otimes C)^T(B \otimes C) = (B^T B) \otimes (C^T C),$$
$$\Rightarrow (E^T E)\text{vec}(X) = (B^T B \otimes C^T C)\text{vec}(X) = \text{vec}(C^T CXB^T B),$$

and

$$E^T\text{vec}(A) = (B \otimes C)^T\text{vec}(A) = \text{vec}(C^T AB),$$

we get

$$\nabla\mathcal{F}(X) = (C^T C)X(B^T B) - C^T AB. \tag{3.46}$$

For any $D \in \mathbb{R}_{++}^{k\times l}$, we have

$$\frac{[|E^T E|\text{vec}(D)]}{[\text{vec}(D)]} = \frac{[\text{vec}((|C^T C|)D(|B^T B|))]}{[\text{vec}(D)]} \equiv \frac{[(|C^T C|)D(|B^T B|)]}{[D]}.$$

Then, by (3.9), an update rule for X will be

$$\begin{aligned} X \quad &\longleftarrow \quad \left[X - D \circ \frac{[\nabla\mathcal{F}(X)]}{[\max((|C^T C|)D(|B^T B|), \delta)]}\right]_+, \quad \delta > 0 \\ &= \quad \left[X - D \circ \frac{[(C^T C)X(B^T B) - C^T AB]}{[\max((|C^T C|)D(|B^T B|), \delta)]}\right]_+. \end{aligned} \tag{3.47}$$

Especially, if B and C are nonnegative, by letting $D = X$ and $\delta = 0$, we obtain a multiplicative update rule as follows

$$X \longleftarrow \left[X \circ \frac{[C^T AB]}{[(C^T C)X(B^T B)]}\right]_+. \tag{3.48}$$

3.4.3 Convex nonnegative matrix factorization

To enhance the interpretability of matrix factors, (Ding et al., 2010) proposed a variant of NMF, the so-called Convex-NMF. Differing from convex coding model where constraint is imposed on V, Convex-NMF imposes constraint on the first factor U so that its columns are convex combinations of columns of the data matrix A. This yields an interesting interpretation: each data point can be expressed as a weighted sum of given data points. The formulation of Convex-NMF proposed by (Ding et al., 2010) takes the form

$$\begin{aligned} \min \quad & F_2(W,V) = \frac{1}{2}\|A - AWV^T\|_F^2, \\ s.t. \quad & W \in \mathbb{R}_+^{n\times r}, V \in \mathbb{R}_+^{n\times r}, \text{ and } \mathbf{1}_{1\times n}W = \mathbf{1}_{1\times r}. \end{aligned} \tag{3.49}$$

Following the alternating method, we alternatively compute W and then V. Computing V is nothing but solving a NNLS problem (cf. Sect. 3.2.1.2). In the sequel, we address the computation of W that amounts to solving the convex problem

$$\min \left\{ \mathcal{F}(W) = \frac{1}{2}\|A - AWV^T\|_F^2 : W \in \mathbb{R}_+^{n\times r} \text{ and } \mathbf{1}_{1\times n}W = \mathbf{1}_{1\times r} \right\}. \tag{3.50}$$

Since the product WV^T is invariant under diagonal scaling (for a diagonal matrix $D \in \mathbb{R}^{r\times r}$ with positive diagonal elements, $WV^T = WD^{-1}(VD)^T$), computation of W can be carried out through two step as follows:

- Ignore the additivity constraint $\mathbf{1}_{1\times n}W = \mathbf{1}_{1\times r}$ and compute W using the method discussed at the end of Sect. 3.4.2.
- Normalization step: for $j = 1, \ldots, r$

$$s = \|W_{:j}\|_1, \; W_{:j} = \frac{1}{s}W_{:j}, \; V_{:j} = sV_{:j}.$$

We can also completely ignore the additive constraint and only apply the normalization step when the iteration process of computing W and V is finished. However, it may occur that some of columns of W are zero and the normalization step is invalid. We can directly solving problem (3.50) using DC programming and DCA as follows.

As in Sect. 3.4.2, we express $\mathcal{F}(W)$ in the least-square form

$$\mathcal{F}(W) = \frac{1}{2}\|\text{vec}(A) - E\text{vec}(W)\|_2^2,$$

where $E = V \otimes A$.

For $\rho \geq \|E\|_2$, we reformulate (3.50) as a DC program

$$\min \left\{ \mathcal{G}(W) - \mathcal{H}(W) : W \in \mathbb{R}_+^{n\times r} \text{ and } \mathbf{1}_{1\times n}W = \mathbf{1}_{1\times r} \right\},$$

where

$$\mathcal{G}(W) = \frac{\rho}{2}\|W\|_F^2, \quad \mathcal{H}(W) = \frac{\rho}{2}\|W\|_F^2 - \mathcal{F}(W).$$

Applying DCA to this DC program, we iteratively

- compute $Y = \nabla\mathcal{H}(W) = \rho W - \nabla\mathcal{F}(W)$ and then

- update

$$\begin{aligned} W &\longleftarrow \arg\min\left\{\frac{\rho}{2}\|W\|_F^2 - \langle Y, W\rangle : W \in \mathbb{R}_+^{n\times r} \text{ and } \mathbf{1}_{1\times n}W = \mathbf{1}_{1\times r}\right\} \\ &= \arg\min\left\{\frac{1}{2}\left\|W - \frac{Y}{\rho}\right\|_F^2 : W \in \mathbb{R}_+^{n\times r} \text{ and } \mathbf{1}_{1\times n}W = \mathbf{1}_{1\times r}\right\}, \end{aligned}$$

or equivalently,

$$W_{:i} \longleftarrow P_{\Delta_n}\left(\frac{Y_{:i}}{\rho}\right), \quad \forall i = 1, \ldots, r.$$

(Thurau et al, 2011) extended the idea of Convex-NMF to propose the so-call Convex-Hull NMF. This variant of NMF further imposes the constraint on the second factor V such that its rows are sum to one. That is, we add into problem (3.49) the constraint $V\mathbf{1}_{r\times 1} = \mathbf{1}_{n\times 1}$ as in the convex coding problem (3.41). Now each data point can be expressed as a convex combination of convex combinations of specific data points. For this problem, we can alternatively compute W using DCA as described above and compute V as discussed in Section 3.4.1.3. Note that in this case, V is not free, so the strategy of ignoring the constraint $\mathbf{1}_{1\times n}W = \mathbf{1}_{1\times r}$ then conducting normalization step is invalid.

It is worth mentioning that both (Ding et al., 2010) and (Thurau et al, 2011) did not directly deal with the constraint $\mathbf{1}_{1\times n}W = \mathbf{1}_{1\times r}$. (Ding et al., 2010) simply ignored this constraint and alternatively computing W and V using Majorize-Minimization method. Meanwhile, instead of computing W and the basis $U = AW$, (Thurau et al, 2011) computed directly U by (approximately) finding k appropriate vertices of the convex hull of A.

3.4.4 Symmetric nonnegative matrix factorization

Symmetric nonnegative matrix factorization takes the form

$$\min_{U\geq 0} F(U) = \frac{1}{4}\|A - UU^T\|_F^2, \tag{3.51}$$

where A is a $n \times n$ nonnegative matrix and $U \in \mathbb{R}^{n\times r}$.

The KKT conditions for this problem is as follows

$$U \geq 0, \quad \nabla F(U) \geq 0, \quad U \circ \nabla F(U) = 0,$$

where $\nabla F(U) = UU^TU - AU$.

Lemma 3.4 *If U is a solution of the prolem (3.51), then $\|U\|_F \leq \mathbf{b}_s$, where $\mathbf{b}_s = (\sqrt{r}\|A\|_2)^{1/2}$ as in theorem 3.3.*

Like the usual NMF, we only need to solve the problem (3.51) with the constraint

$$U \in \mathcal{S} = \{U \in \mathbb{R}^{n\times r} : U \geq 0, \|U\|_F \leq \mathbf{b}_s\}.$$

For any $U \in \mathcal{S}$ and for any $H \in \mathbb{R}^{n\times r}$,

$$\begin{aligned} D^2F(U)[H] &= \|UH^T\|_F^2 + \langle UH^T, HU^T\rangle + \langle UU^T - A, HH^T\rangle \\ &\leq (3\|U\|_F^2 + \|A\|_2)\|H\|_F^2 \leq (1+3\sqrt{r})\|A\|_2. \end{aligned}$$

Thus, for $\rho \geq (1+3\sqrt{r})\|A\|_2$, on the set $\mathcal{S}$, the objective function has a DC decomposition $F(U) = g(U) - h(U)$, where

$$g(U) = \frac{\rho}{2}\|U\|_F^2, \quad h(U) = \frac{\rho}{2}\|U\|_F^2 - F(U).$$

DCA applied to this problem can be summarized as follows

DCA-sym: DCA applied to Symmetric NMF

Initialize U^0 satisfied $\|U^0\|_F \leq \mathbf{b}_s$, $k \leftarrow 0$.

repeat

- Compute $\overline{U}^k = \nabla h(U^k) = \rho U^k - \nabla F(U^k)$.
- Compute $U^{k+1} = P_{\mathcal{S}}\left(\frac{1}{\rho}\overline{U}^k\right) = \begin{cases} \frac{1}{\rho}[\overline{U}^k]_+ & \text{, if } \|[\overline{U}^k]_+\|_F \leq \rho\mathbf{b}_s \\ \frac{\mathbf{b}_s}{\|[\overline{U}^k]_+\|_F}[\overline{U}^k]_+ & \text{, if } \|[\overline{U}^k]_+\|_F > \rho\mathbf{b}_s \end{cases}$.
- $k \leftarrow k+1$.

until Convergence of $\{U^k\}$.

3.5 Numerical experiments

In this section, we evaluate our proposed algorithms ADCA, DCA1, and DCA2. We compare these DCA based algorithms with the following state-of-the-art algorithms

- BPP[3]: Alternating nonnegative least squares with the block principal pivoting method (Kim and Park, 2011),

- HAac[4]: The accelerated hierarchical alternating least squares (Gillis and Glineur, 2012),

3. http://www.cc.gatech.edu/~hpark/
4. https://sites.google.com/site/nicolasgillis/home

- NeNMF[5]: An NMF solver based on Nesterov's optimal gradient method (Guan et al., 2012).

- PNM[6]: NMF via projected Newton method (Gong and Zhang, 2012).

- QRPBB: NMF using quadratic regularization projected Barzilai-Borwein method (Huang et al., 2014).

The source codes of these algorithms are available on the authors' homepages except QRPBB provided by the authors, and we used them for our comparison.

In all experiments, the settings $N_{iter} = r$, $\epsilon = 0.1$, $\delta = 10^{-12}$ are used for algorithm DCA-NNLS and D is row by row constructed using (3.14).

All these algorithms were executed with MATLAB R2008b, on a Intel(R) Core(TM) i5-2500S 2 × 2.7GHz processor and 4GB memory. The multi-threading option of MATLAB was disabled.

All initializations $(U^0, V^0) \in \mathbb{R}_+^{m\times r} \times \mathbb{R}_+^{n\times r}$ in experiments are computed as follows. First, each element of U^0 and V^0 is picked randomly in $[0, 1]$. Then we compute

$$\alpha = \arg\min_{\alpha\geq 0} \|A - \alpha U^0(V^0)^T\|_F^2 = \frac{\langle A, U^0(V^0)^T\rangle}{\|U^0(V^0)^T\|_F^2},$$

and set

$$U^0 = \sqrt{\alpha}U^0, \quad V^0 = \sqrt{\alpha}V^0.$$

3.5.1 Synthetic datasets

The synthetic datasets were generated as follows. For a triple (m, n, r) and each value $s \in \{20, 40, 80, 95\}$, we randomly generated two matrices $U \in \mathbb{R}_+^{m\times r}$ and $V \in \mathbb{R}_+^{n\times r}$ with sparsity $s\%$ (percentage of zeros elements). Then we formed the data matrix A by taking $A = UV^T$. In the experiments, we fixed $r = 30$ and considered two different values of (m, n). To ensure a fair comparison, we executed each algorithm from the same 10 different initializations and the average result of successful trials (the target relative error is achieved within a predefined limit time) is reported.

Tables 3.1 and 3.2 summarize the execution times for considered algorithms to achieve the relative error $\|A-UV^T\|_F/\|A\|_F \leq 10^{-4}$ with $(m, n) = (500, 1000)$ and $(m, n) = (1000, 2000)$ respectively. The limit time was set to be 600 seconds. It can be seen from Tables 3.1 and 3.2 that ADCA is the best algorithm. According to Table 3.1, it is faster than the BPP from 4 to 6 times, the HAac from 1.2 to 2.8 times, the NeNMF from 4.7 to 15 times, the PNM

5. https://sites.google.com/site/nmfsolvers/
6. http://www.public.asu.edu/~pgong5/

Table 3.1: Numerical result on synthetic datasets with $m = 500$ and $n = 1000$. The suffix numbers in names of datasets indicate the sparsity of factors composing the datasets. The number in parenthesis indicates the number of trials that fail to achieve the target relative error within the limit time.

Dataset	Time (in seconds)							
	ADCA	BPP	HAac	NeNMF	PNM	QRPBB	DCA1	DCA2
data20	**6.5**	29.1(1)	18.3	31.1	90.4(2)	13.0(1)	222.4	121.4
data40	**1.3**	5.3	2.6	6.7	19.8	4.3	31.3	13.7
data80	**0.21**	1.3	0.27	1.1	3.2	0.67	2.2	27.3
data95	**0.19**	0.99(1)	0.47	2.9(1)	4.8	1.1	2.3	33.9

Table 3.2: Numerical result on synthetic datasets with $m = 1000$ and $n = 2000$. The suffix numbers in names of datasets indicate the sparsity of factors composing the datasets. The number in parenthesis indicates the number of trials that fail to achieve the target relative error within the limit time.

Dataset	Time (in seconds)							
	ADCA	BPP	HAac	NeNMF	PNM	QRPBB	DCA1	DCA2
data20	**13.3**(2)	35.5	36.3	44.7	80.2(2)	26.8(2)	526.1(5)	325.2
data40	**4.1**	11.6	6.9	13.9	37.1	10.1	107.2	48.6
data80	**0.67**	2.6	0.82	2.4	7.5	1.8	6.7	88.9
data95	**0.67**	1.8	1.8	2.5	24.7	14.3	3.4	129.1

from 14 to 25 times, and the QRPBB from 2 to 5.7 times. Similarly, according to Table 3.2, ADCA is faster than the BPP (from 2.6 to 3.8 times), the HAac (from 1.2 to 2.7 times), the NeNMF (from 3.3 to 3.7 times), the PNM (from 6 to 36 times), and the QRPBB (from 2 to 21 times).

The algorithm DCA1 seems to have difficult convergence on dense datasets, but it performs well on sparse datasets. It is faster than the PNM on data80 and data95 from both experiments. On data95 ($m = 1000, n = 2000$), it is faster than the PNM (7 times) and the QRPBB (4 times). The algorithm DCA2 is only faster than the PNM on data40 from Table 3.1 and the DCA1 on datasets with 20% and 40% sparsity.

3.5.2 Real datasets

We have used six real-world datasets for our comparison, and their information is shown in Table 3.3. Among them, three facial image datasets are in dense format and three text datasets are in sparse format.

Table 3.3: Datasets

Dataset	Format	size
CBCL[4]	dense	361 × 2,429
ORL[5]	dense	10,304 × 400
PIE64[5]	dense	4,096 × 11,554
20NewsGroup[6]	sparse	26,214 × 11,314
TDT2[6]	sparse	19,448 × 9,394
Reuteur[6]	sparse	8,293 × 18,933

All algorithm are stopped when the following stopping condition is satisfied

$$\frac{|F(U^k, V^k) - F(U^{k+1}, V^{k+1})|}{\max(1, F(U^{k+1}, V^{k+1}))} < \tau, \tag{3.52}$$

where $\tau > 0$ is a small tolerance, or when the running time exceeds a limitation. In our experiment, we set $\tau = 10^{-6}$, and the time limit is 3600 seconds. Note that (Lin, 2007b) proposed to use the norm of the projected gradient to define the stopping condition. However, this quantity is not reliable (Ho, 2008; Kim and Park, 2011), since upto a scaling it can gets an arbitrary value without changing the objective value. Thus, we use the condition (3.52) instead.

The average results of 5 different initializations are reported in figures 3.1 and 3.2 and Table 3.4. For each dataset and each algorithm, we recorded the relative error $\|A - U^k(V^k)^T\|_F/\|A\|_F$ and the corresponding elapsed time after each iteration. From this information, we used linear interpolation to estimate the relative error at each time point in a fixed discrete grid and form a piece-wise linear function of relative error by elapsed time. For different initializations, we obtained different functions of this kind. The average function of these functions is calculated and plotted in the figures.

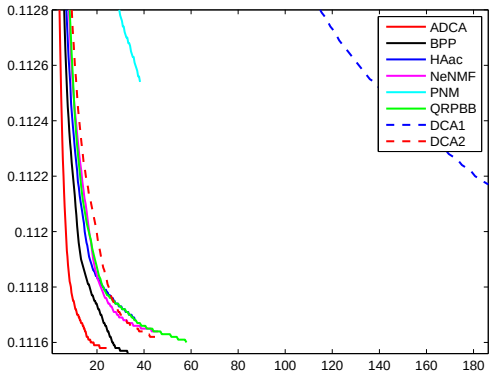
We observe from the figures 3.1 and 3.2 that our proposed algorithm ADCA always outperforms the BPP, the NeNMF, the PNM and the QRPBB (except for TDT2 with $r = 20$ and Reuteur with $r = 40$). As for the HAac, ADCA exhibits better performance except for ORL dataset where ADCA gives smaller errors but more running times. It can be noticed that our proposed algorithm is preferable to HAac on large datasets, especially on sparse datasets.

From Table 3.4, we see that our algorithm ADCA outperforms the BPP, NeNMF, PNM and QRPBB algorithms in terms of the relative error and the running time, except on CBCL with $r = 49$ (resp. TDT2 with $r = 20$) ADCA has a little higher error than the BPP (resp. the QRPBB) but has much less running time. The gain of DCA is more significant on sparse datasets. Comparing to HAac, ADCA has better performance on 8 out of 12 cases. Among

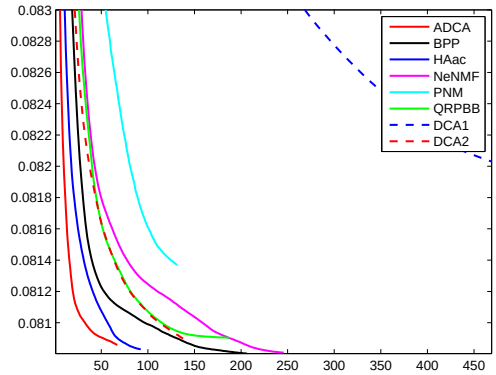
4. http://cbcl.mit.edu/software-datasets/FaceData2.html
5. http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html
6. http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html

the 4 remaining cases, for ORL dataset, ADCA has better error. And for 20NewsGroup dataset ($r = 40$), our algorithm is faster (2 times) but worse than HAac in term of the relative error. Overall, ADCA has the best relative error on 6 out of 12 cases. And on the remaining 6 cases, the relative difference between the result of ADCA and the best result is small (at most 0.06%). Moreover, ADCA has smallest running time except on ORL dataset.

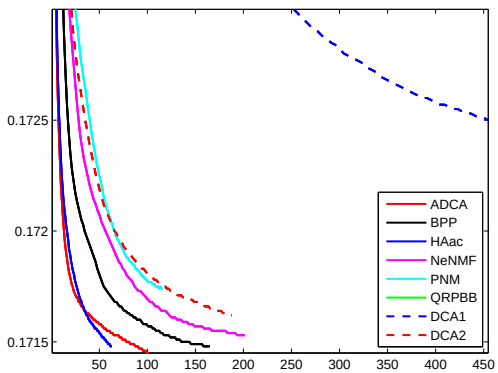
Unlike the result on synthetic datasets, the algorithm DCA2 has pretty good performance on real datasets. It appears as a medium methods among all competitors. The DCA1 has quite poor performance. It has relatively high error and takes much computational time.



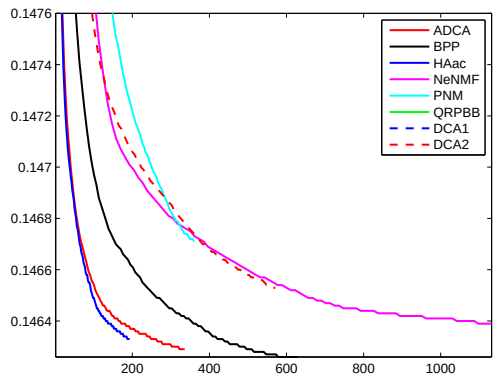
(a) CBCL, $r = 25$



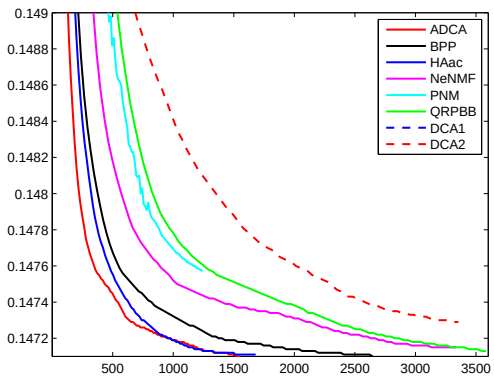
(b) CBCL, $r = 49$



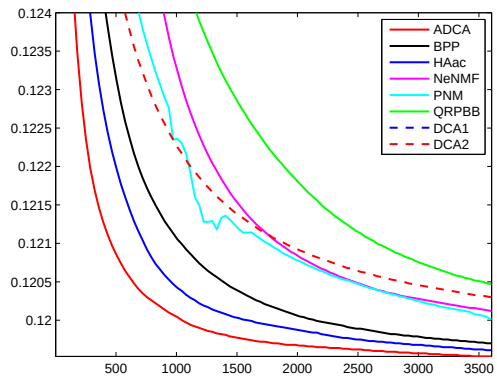
(c) ORL, $r = 25$



(d) ORL, $r = 49$

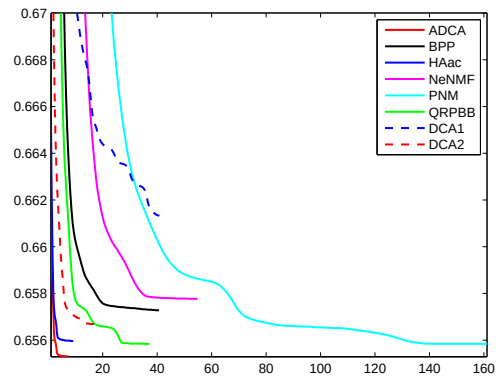


(e) PIE64, $r = 80$

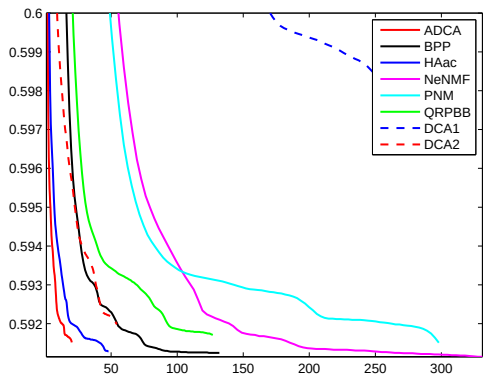


(f) PIE64, $r = 160$

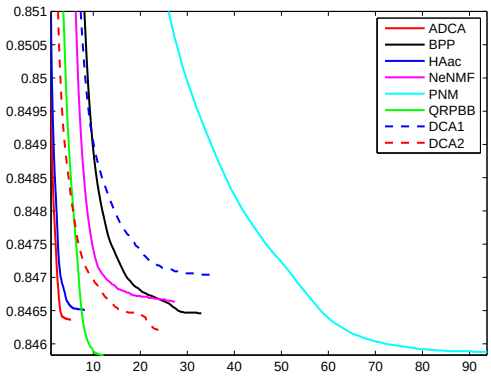
Figure 3.1: Dense datasets. In these figures, the horizontal axis presents the elapsed time in second, and the vertical axis presents the relative error.



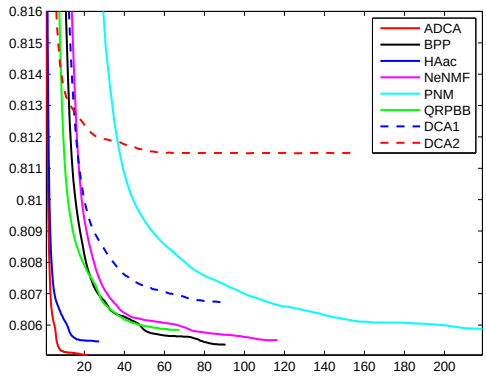
(a) 20News, $r = 20$



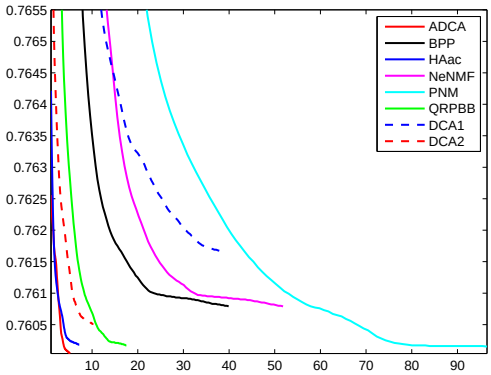
(b) 20News, $r = 40$



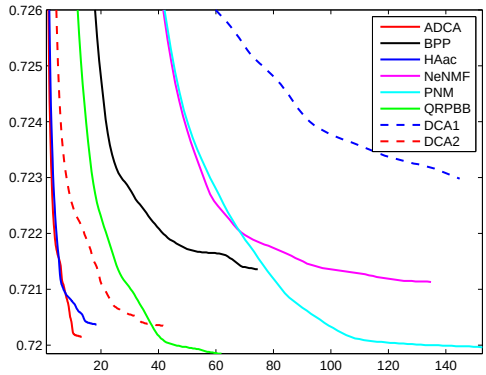
(c) TDT2, $r = 20$



(d) TDT2, $r = 40$



(e) Reuteur, $r = 20$



(f) Reuteur, $r = 40$

Figure 3.2: Sparse datasets. In these figures, the horizontal axis presents the elapsed time in second, and the vertical axis presents the relative error.

Table 3.4: The comparisons of NMF algorithms. All algorithms start from the same initial point and stop when the stopping criteria (3.52) is satisfied with the tolerance $\tau = 10^{-6}$ or the running time exceeds 3600 seconds. Bold font indicates the best result in each row. The number in parenthesis is the relative difference computed by $100(F - F^*)/F^*$, where F is the corresponding "error" and F^* is the smallest value among F's in the upper row.

Dataset	r	Criteria	ADCA	HAac	BPP	NeNMF	PNM	QRPBB	DCA1	DCA2
CBCL	25	error	0.11158	0.11168	**0.11156**	0.11164	0.11254	0.11160	0.11217	0.11161
			(0.018)	(0.107)	(0)	(0.072)	(0.88)	(0.036)	(0.54)	(0.045)
		time	**24.1**	36.1	33.1	46.0	38.2	58.0	186.2	46.263
	49	error	0.080842	0.080830	**0.080803**	0.080809	0.081369	0.080903	0.08203	0.080893
			(0.048)	(0.033)	(0)	(0.007)	(0.7)	(0.124)	(1.52)	(0.111)
		time	**85.9**	92.3	206.3	245.7	131.27	187.7	468.8	139.39
ORL	25	error	**0.17145**	0.17148	0.17148	0.17153	0.17174	0.17407	0.1725	0.17162
			(0)	(0.017)	(0.017)	(0.047)	(0.169)	(1.53)	(0.612)	(0.099)
		time	101.4.7	62.8	165.1	201.5	115.6	**5.75**	454.7	188.06
	49	error	0.14629	0.14633	**0.14626**	0.14639	0.14671	0.15009	0.1501	0.14653
			(0.02)	(0.048)	(0)	(0.089)	(0.307)	(2.619)	(2.62)	(0.185)
		time	335.7	192.7	629.1	1132.0	360.1	**14.9**	600.1	571.38
PIE64	80	error	**0.14710**	0.14711	**0.14710**	0.14715	0.14757	0.14713	0.15391	0.14729
			(0)	(0.007)	(0)	(0.034)	(0.32)	(0.02)	(4.63)	(0.13)
		time	1553.2	1680.4	2646.7	3332.8	**1240.4**	3582.9	3600	3354
	160	error	**0.11953**	0.11961	0.1197	0.12012	0.12001	0.12047	0.13385	0.1203
			(0)	(0.067)	(0.142)	(0.493)	(0.401)	(0.786)	(11.98)	(0.644)
		time	3600	3600	3600	3600	3600	3600	3600	3600
20News	20	error	**0.65529**	0.65596	0.65728	0.65777	0.65584	0.65584	0.66132	0.65923
			(0)	(0.102)	(0.304)	(0.379)	(0.084)	(0.084)	(0.92)	(0.601)
		time	**7.3**	9.1	40.6	54.8	161.1	37.2	40.7	27.987
	40	error	0.59152	0.59129	0.59125	**0.59115**	0.59151	0.59171	0.59813	0.59199
			(0.062)	(0.024)	(0.017)	(0)	(0.061)	(0.095)	(1.181)	(0.142)
		time	**20.2**	48.1	131.9	330.9	298.1	127.0	282.61	54.504
TDT2	20	error	0.84636	0.84651	0.84646	0.84663	0.84588	**0.84583**	0.84703	0.84621
			(0.063)	(0.08)	(0.074)	(0.094)	(0.006)	(0)	(0.142)	(0.045)
		time	**5.1**	8.1	32.9	27.3	93.7	12.1	35.8	23.9
	40	error	**0.80504**	0.80548	0.80537	0.80552	0.80587	0.80585	0.80673	0.81149
			(0)	(0.055)	(0.041)	(0.06)	(0.103)	(0.1)	(0.21)	(0.8)
		time	**21.4**	27.3	90.5	116.4	218.9	67.48	88.397	155.94
Reuteur	20	error	**0.76004**	0.76018	0.76079	0.76079	0.76015	0.76017	0.76167	0.76051
			(0)	(0.018)	(0.1)	(0.1)	(0.014)	(0.017)	(0.21)	(0.062)
		time	**5.1**	7.1	39.9	51.8	96.4	17.5	38.3	10.29
	40	error	0.72015	0.72037	0.72136	0.72113	0.71996	**0.71985**	0.72298	0.72034
			(0.042)	(0.072)	(0.21)	(0.178)	(0.015)	(0)	(0.435)	(0.068)
		time	**13.2**	18.4	74.5	134.8	152.7	61.9	144.7	42.07

3.6 Sparse NMF

As we have mentioned before, due to the nonnegativity constraints, NMF often generates sparse basis vectors and sparse coefficient matrix. However, the sparsity given by NMF is somewhat a side-effect rather than a goal. It has been reported that NMF generates holistic basis images instead of parts-based basis images for a facial image database (Li et al., 2001; Hoyer, 2004). Several approaches (Li et al., 2001; Hoyer, 2004) have been proposed to explicitly control the degree of sparsity of U and V. Depending on application, one can choose to impose sparsity constraint only on the basis factor U (Hoyer, 2004), only on the coefficient matrix V (Hoyer, 2002; Liu et al., 2003; Kim and Park, 2007), or on both U and

V (Li et al., 2001; Hoyer, 2004).

In this section, we introduce three NMF variants involving the constraints on the matrix factors U and V so that we can control the degree of sparsity in the matrix factors. We reconsider the problem (3.38)

$$\min_{U,V\geq 0} F(U,V) := \frac{1}{2}\|A - UV^T\|_F^2 + \alpha J_1(U) + \beta J_2(V). \tag{3.53}$$

In this section, we consider specifically the sparse NMF, i.e. at least sparsity constraints on U and/or V are imposed. Naturally, the sparsity is modeled using ℓ_0-norm, that is we let $J_1(U) = \|U\|_0$ and/or $J_2(V) = \|V\|_0$. However, as mentioned in chapter 2, the use of ℓ_0-norm leads to the problem hard to be solved. Motivated by the study in chapter 2, for the purpose of modeling sparsity, we use the capped-ℓ_1 penalty (Peleg and Meir, 2008) defined by

$$\varphi(x) = \min(1, \theta|x|), \quad \forall x \in \mathbb{R}, \tag{3.54}$$

where $\theta > 0$ is a parameter. We first present some related works and remarks on their resolution, then describe our proposed methods for the sparse NMF.

3.6.1 Related works.

The first attempt in this direction is to employ the ℓ_1-regularization on the factor V

$$J_2(V) = \|V\|_{1,1} = \sum_{i=1}^{n}\sum_{j=1}^{r} V_{ij}$$

(since $V \geq 0$). Then updating V amounts to solving the problem of the form (3.11) (via vectorizing matrix operations)

$$\min_{V\geq 0} \frac{1}{2}\langle U^TU, V^TV\rangle - \langle A^TU - \beta\mathbf{1}_{n\times r}, V\rangle.$$

If we take $\rho = \frac{[VU^TU+\beta\mathbf{1}_{n\times r}]}{[V]}$, the updating rule (3.8) takes the form

$$V \longleftarrow V^+ = \left[V - V \circ \frac{[VU^TU - A^TU + \beta\mathbf{1}_{n\times r}]}{[VU^TU + \beta\mathbf{1}_{n\times r}]}\right]_+ = V \circ \frac{[A^TU]}{[VU^TU + \beta\mathbf{1}_{n\times r}]}$$

that was also proposed in (Hoyer, 2002).

Another use of sparse regularization is the penalty function (Kim and Park, 2011)

$$J_2(V) = \frac{1}{2}\sum_{i=1}^{n}\|V_{i:}\|_1^2 = \langle \mathbf{1}_{r\times r}, V^TV\rangle.$$

Then updating V amounts to solving the problem of the form (3.11)

$$\min_{V\geq 0} \frac{1}{2}\langle U^TU + \beta \mathbf{1}_{r\times r}, V^TV\rangle - \langle A^TU, V\rangle.$$

By exploiting the relationship between the ℓ_1-norm and the ℓ_2-norm, (Hoyer, 2004) introduced a sparseness measure of a vector $x \in \mathbb{R}^n$ by

$$\text{sparseness(x)} = \frac{\sqrt{n} - \|x\|_1/\|x\|}{\sqrt{n}-1}.$$

Then the sparsity can be explicitly control by minimizing $E(A, UV^T) = \frac{1}{2}\|A - UV^T\|_F^2$ under the constraints

$$\text{sparseness}(U_{:t}) = s_u, \quad \text{sparseness}(V_{:t}) = s_v, \quad \forall t = 1, \ldots, r,$$

where s_u and s_v are the desired sparsity of U and V respectively.

This kind of sparsity constraint can be imposed as a penalty term of the form (Berry et al., 2006)

$$J_2(V) = \frac{1}{2}(\omega\|\text{vec}(V)\|_2 - \|\text{vec}(V)\|_1)^2,$$

where $\omega = \sqrt{nr} - \gamma(\sqrt{nr} - 1)$ with $0 \leq \gamma \leq 1$ representing desired sparsity.

Then updating V amounts to solve the nonconvex problem

$$\min_{V\geq 0} \frac{1}{2}\langle U^TU, V^TV\rangle - \langle A^TU, V\rangle + J_2(V).$$

Note that function J_2 can be expressed as a DC function $J_2(V) = J_g(V) - J_h(V)$ given by

$$J_g(V) = \omega^2\|\text{vec}(V)\|^2 + \|\text{vec}(V)\|_1^2, \quad J_h(V) = \frac{1}{2}(\omega\|\text{vec}(V)\|_2 + \|\text{vec}(V)\|_1)^2.$$

Thus, to solve this problem by applying DCA, we iteratively compute $\overline{V} \in \partial J_h(V)$ then solve the convex problem

$$\begin{aligned} & \min_{V\geq 0} \frac{1}{2}\langle U^TU, V^TV\rangle - \langle A^TU, V\rangle + \omega^2\|\text{vec}(V)\|^2 - \langle \overline{V}, V\rangle \\ \Leftrightarrow \quad & \min_{V\geq 0} \frac{1}{2}\langle U^TU + 2\omega^2 I, V^TV\rangle - \langle A^TU + \overline{V}, V\rangle \end{aligned}$$

that is of the form (3.11).

There are several works using the KL-divergence (instead of the Frobenius norm) as the discrepancy measure to find sparse NMF. (Li et al., 2001) argued that letting $J_1(U) = \|U^TU\|_1$ and $J_2(V) = -\|V\|_F^2$ leads an NMF such as U and V are as sparse as possible and columns of U are as orthogonal as possible. (Liu et al., 2003) used the same regularization as in (Hoyer, 2002) but for the NMF based on KL-divergence.

3.6.2 Sparse NMF formulations

Below, we give in details three formulations of sparse NMF corresponding to three cases: sparsity is imposed on both U and V (capNMF/B), and sparsity is imposed on either U (capNMF/L) or V (capNMF/R).

capNMF/B: To imposed sparsity constraints on both U and V, we let $J_1(U) = \sum_{i=1}^m \sum_{t=1}^r \varphi(U_{it})$ and $J_2(V) = \sum_{j=1}^n \sum_{t=1}^r \varphi(V_{jt})$. Then the corresponding sparse NMF formulation takes the form

$$\min_{U,V\geq 0} F_B(U,V) := \frac{1}{2}\|A - UV^T\|_F^2 + \alpha \sum_{i=1}^m \sum_{t=1}^r \varphi(U_{it}) + \beta \sum_{j=1}^n \sum_{t=1}^r \varphi(V_{jt}). \tag{3.55}$$

capNMF/L: To imposed sparsity constraints only on the first factor U, we can let $J_1(U) = \sum_{i=1}^m \sum_{t=1}^r \varphi(U_{it})$ while restrict V to the set

$$S_V = \{V \in \mathbb{R}_+^{n\times r} : \|V_{:t}\|_2 \leq 1 \ \forall t = 1,\ldots,r\}.$$

That is, we let $J_2(V) = \chi_{S_V}(V)$. Then the sparse NMF in this case in given by

$$\min \left\{ F_L(U,V) := \frac{1}{2}\|A - UV^T\|_F^2 + \alpha \sum_{i=1}^m \sum_{t=1}^r \varphi(U_{it}) : U \geq 0, V \in S_V \right\}. \tag{3.56}$$

The reason of restricting V to the set S_V is as follows. Since φ is an increasing function on $\mathbb{R}$, for any $0 < \lambda < 1$, we have $F_L(\lambda U, \frac{1}{\lambda}V) \leq F_L(U,V)$. Therefore, without the constraints $\|V_{:t}\|_2 \leq 1 \ \forall t = 1,\ldots,r$, we can replace (U,V) by $(\lambda U, \frac{1}{\lambda}V)$ with arbitrarily small λ to have smaller objective value. Consequently, the optimization problem in this case is ill-posed.

capNMF/R: Similar to capNMF/L, the formulation for sparse NMF with sparsity constraints only on the second factor V has the form

$$\min \left\{ F_R(U,V) := \frac{1}{2}\|A - UV^T\|_F^2 + \beta \sum_{j=1}^n \sum_{t=1}^r \varphi(V_{jt}) : U \in S_U, V \geq 0 \right\}, \tag{3.57}$$

where $S_U = \{U \in \mathbb{R}_+^{m\times r} : \|U_{:t}\|_2 \leq 1 \ \forall t = 1,\ldots,r\}$.

In the next section, we will describe DCA based algorithms for solving the sparse NMF problems (3.55), (3.56) and (3.57) that have the common form (3.53).

3.6.3 Alternative DCA based algorithms for solving the sparse NMF problems

Similar to the NMF problem, we also use the alternating method for solving the sparse NMF problem. Since the computation of U and V are similar, we only describe the computation of V. Computing V amounts to solving a problem of the form

$$\min_{V\geq 0} f(V) := \frac{1}{2}\|UV^T - A\|_F^2 + \beta J_2(V), \tag{3.58}$$

where $U \in \mathbb{R}_+^{m\times r}, A \in \mathbb{R}_+^{m\times n}$, and J_2 is one of the two functions: $J_2(V) = \chi_{S_V}(V)$ or $J_2(V) = \sum_{j=1}^n \sum_{t=1}^r \varphi(V_{jt})$. We will respectively consider two cases of function J_2.

DCA for solving problem (3.58) with bound constraint

In this case, we have $J_2(V) = \chi_{S_V}(V)$ and (3.58) is a convex problem. Similar to DCA for solving the NNLS problem (section 3.2.1.2, chapter 3), by replacing $\mathbb{R}^{n\times r}$ with S_V, an update rule for computing V^{l+1} from V^l is given by

$$V^{l+1} \longleftarrow P_{S_V}\left(V^l - D \circ \frac{[V^l U^T U - A^T U]}{[DU^T U]}\right),$$

where $D \in \mathbb{R}_{++}^{n\times r}$. By simply taking $D = \mathbf{1}_{n\times r}$, the above update rule becomes

$$V^{l+1} \longleftarrow P_{S_V}\left(V^l - \frac{[V^l U^T U - A^T U]}{[\mathbf{1}_{n\times r} U^T U]}\right)$$

$$\Leftrightarrow V_{:t}^{l+1} \longleftarrow P_{V_{:t}\geq 0, \|V_{:t}\|\leq 1}\left(V_{:t}^l - \frac{(V^l U^T U - A^T U)_{:t}}{(\mathbf{1}_{1\times r} U^T U)_t}\right), \quad \forall t = 1, \ldots, r.$$

This can be explicitly computed by

$$\overline{V}^l = \left[V^l - \frac{[V^l U^T U - A^T U]}{[\mathbf{1}_{n\times r} U^T U]}\right]_+, \quad V_{:t}^{l+1} \longleftarrow \frac{\overline{V}_{:t}^l}{\max(1, \|\overline{V}_{:t}^l\|)}, \quad \forall t = 1, \ldots, r. \tag{3.59}$$

The update rule (3.59) is the summary of DCA for solving problem (3.58) with bound constraint $J_2(V) = \chi_{S_V}(V)$.

DCA for solving problem (3.58) with capped-ℓ_1 regularization

In this case, we have $J_2(V) = \sum_{j=1}^n \sum_{t=1}^r \varphi(V_{jt})$. Since φ is a DC function with a DC decomposition given by $\varphi(x) = \theta|x| - (\max(1, \theta|x|) - 1)$, J_2 can be expressed as a DC function

$$J_2(V) = \theta\|V\|_1 - \left(\sum_{j=1}^n \sum_{t=1}^r \max(1, \theta|V_{jt}|) - nr\right).$$

Consequently, (3.58) can be reformulated as a DC program as follows

$$\min_{V\geq 0} g(V) - h(V), \tag{3.60}$$

where $g(V) = \frac{1}{2}\|UV^T - A\|_F^2 + \beta\theta\|V\|_1$ and $h(V) = \beta\sum_{j=1}^{n}\sum_{t=1}^{r}\max(1, \theta|V_{jt}|) - \beta nr$.

Applied DCA to (3.60), at each iteration l, we compute

- $\overline{V}^l \in \partial h(V^l)$ given explicitly by

$$\overline{V}_{jt} = \begin{cases} \beta\theta & \text{if } \theta V_{jt} > 1 \\ -\beta\theta & \text{if } \theta V_{jt} < -1 \\ 0 & \text{otherwise} \end{cases} \quad \forall j = 1, \ldots, n; t = 1 \ldots, r.$$

- V^{l+1} as a solution to the problem

$$\begin{aligned} & \min_{V\geq 0} \frac{1}{2}\|UV^T - A\|_F^2 + \beta\theta\|V\|_1 - \langle \overline{V}^l, V\rangle \\ \Leftrightarrow \quad & \min_{V\geq 0} \frac{1}{2}\|UV^T - A\|_F^2 + \langle \beta\theta\mathbf{1}_{n\times r} - \overline{V}^l, V\rangle \end{aligned}$$

This problem can be solved by using the method developed in Section 3.2.1.1 (Remark 3.1).

One of the problem with this case is that (3.58) is a nonconvex problem due to the nonconvexity of φ. If we compute V^{k+1} by solving (3.58) with V^k as initialization, it is likely that we get stuck in a bad local minimizer. To overcome this bottleneck, we reinitialize our DCA based algorithm for solving (3.58) by solving the following convex problem

$$\min_{V\geq 0} \frac{1}{2}\|UV^T - A\|_F^2 + \beta\theta\|V\|_1.$$

And again, this problem can be solved by using the method developed in Section 3.2.1.1 (Remark 3.1).

3.6.4 Experiment

It has been proved that adding sparsity constraints to the standard NMF can be useful in learning part-based representations (Hoyer, 2004). In this section, we do experiments to show that our proposed methods are suitable for this task. We compare our methods with the NMF with sparseness constraints (NMFSC) proposed by (Hoyer, 2004). Our algorithms were implemented on Matlab, and for the algorithm by (Hoyer, 2004), we use his Matlab implementation. All these algorithms were executed with MATLAB R2008b, on a Intel(R) Core(TM) i5-3360M 2 × 2.8GHz processor and 4GB memory.

For our algorithms, we take $\alpha = s_u\|A\|_F^2/m^2$ and $\beta = s_v\|A\|_F^2/(nr)$, where $s_u, s_v \in \{0.1, \ldots, 0.9\}$ stand for desired sparsity. All the considered algorithms will be terminated if the following condition is satisfied

$$\|U^{k+1} - U^k\|_F < \epsilon(1 + \|U^{k+1}\|_F) \text{ and } \|V^{k+1} - V^k\|_F < \epsilon(1 + \|V^{k+1}\|_F),$$

where $\epsilon = 10^{-4}$, or the number of iterations exceeds 1000.

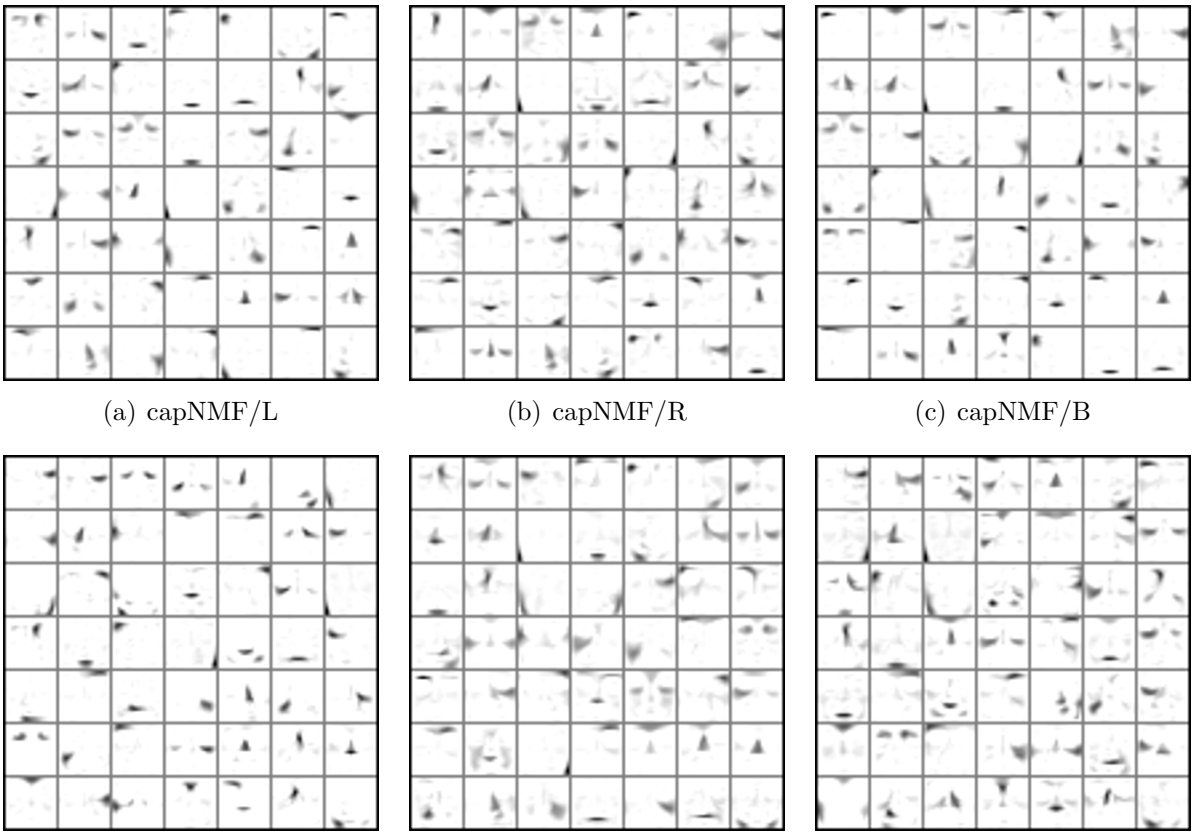
We test on the CBCL face image database[7] with the same settings as in (Hoyer, 2004). We consider three cases: the sparsity is imposed only on the first factor U (/L), only on the second factor V (/R), and on both factors U and V (/B). For each case and each algorithm, we compute the relative error $\|A - UV^T\|_F/\|A\|_F$, the sparsity on U: $\#U = \frac{\|U\|_0}{mr}$, and the sparsity on V: $\#V = \frac{\|V\|_0}{nr}$. Note that a number x is considered to be 0 if $|x| < 10^{-16}$. We run each algorithm with all sparsity level s_u, s_v and the result with the best relative error is reported. Numerical results are summarized in Table 3.5. Figure 3.3 represents the bases/features learned by our DCA based methods and the NMFSC methods.

Table 3.5: Numerical results on CBCL database. The better results are boldfaced

Method	/L			/R			/B		
	error	$\#U$	$\#V$	error	$\#U$	$\#V$	error	$\#U$	$\#V$
NMFSC	0.206	0.80	0.35	0.206	0.21	**0.51**	0.213	0.67	**0.51**
capNMF	**0.194**	**0.81**	**0.37**	**0.198**	**0.72**	0.47	**0.193**	**0.78**	0.43

We observe from Table 3.5 that our method always learn sparser basis and have smaller relative error then the NMFSC. For reference, the values: error, $\#U$, and $\#V$ of the standard NMF (using the multiplicative update algorithm of (Lee and Seung, 2001) implemented in (Hoyer, 2004)) are 0.203, 0.27, and 0.40 respectively. In all three cases, our method gives smaller relative error and higher sparsity on the basis U than the standard NMF, even in the case we only impose sparsity on the coefficients. This means that our method successfully learns part-bases representations. Figure 3.3 indicates that our DCA based method and the NMFSC method have quite similar performance.

7. http://cbcl.mit.edu/software-datasets/FaceData2.html



(a) capNMF/L (b) capNMF/R (c) capNMF/B

(d) NMFSC/L (e) NMFSC/R (f) NMFSC/B

Figure 3.3: Features learned from the CBCL face image database. using our DCA based method and NMFSC method.

3.7 Conclusion

In this chapter, two approaches for computing NMF based on DC programming and DCA were proposed. The first one iteratively and alternatively computes the factors of the factorization by solving NNLS subproblems. While the second one simultaneously computes the factors at each iteration. We have developed a scheme based on DC programming and DCA to solve the NNLS problem. Inheriting the convergence theory of DCA, our NMF algorithms holds global convergence property. The proposed alternating based method can easily be adopted to other extensions of NMF such as constrained NMFs, multilayer NMF, convex NMF and symmetric NMF. Our method for solving the NNLS problem cover many existing methods for finding approximate solution of NNLS problem. Thus, the proposed alternating based method also covers many existing methods for solving the NMF problem and its extensions. Numerical comparisons with recently developed NMF algorithms were performed using both real-world and synthetic datasets. The proposed algorithms compete

favorably with five state-of-the-art algorithms especially on large and sparse datasets.

As an extension of the standard NMF, we have proposed three novel sparse NMF formulations by using the capped-ℓ_1 to model sparsity. The DCA based algorithms for solving these sparse NMF problems have been presented. Experiments on a facial database showed that our methods can effectively learn parts-based representations.

Chapter 4

Dictionary Learning and Application in Image Denoising[1]

Abstract: *Sparse representations of signals based on learned dictionaries have drawn considerable interest in recent years. However, the design of dictionaries adapting well to a set of training signals is still a challenging problem. For this task, we propose a novel algorithm based on DC programming and DCA. The efficiency of proposed algorithm will be demonstrated in image denoising application.*

4.1 Introduction

Sparse signal representation.

Sparse representation plays a crucial role in signal processing techniques. Modeling a signal using a linear combination of only a few basis elements has shown to be very effective in many signal processing applications such as image compression, noise removal, texture synthesis, etc. The model of spare signal representation is concretely described as follows. Given a signal $x \in \mathbb{R}^m$ and a matrix $D \in \mathbb{R}^{m\times p}$ with with fewer rows than columns - so D is said to be "overcomplete". One desires to find a representation of x in the form $x = Dw$ such that $w \in \mathbb{R}^p$ is as sparse as possible. The matrix D is called *dictionary* or *codebook* while its columns are referred to as *atoms* or *bases*. The motivation for such an approach is that with an overcomplete dictionary composed of an appropriate set of basis vectors, one can efficiently represent a large class of signals compactly. Note that the same model also appears

1. The material of this chapter is based on the following works:
[1]. Xuan Thanh Vo, Hoai An Le Thi, Tao Pham Dinh, Thi Bich Thuy Nguyen. DC Programming and DCA for Dictionary Learning. in M. Nunez et al. (Eds): ICCCI 2015, Part I, LNAI 9329, pp. 295–304, Springer 2015.
[2]. Xuan Thanh Vo, Hoai An Le Thi. DC Programming and DCA for Dictionary Learning and Application in Image Denoising. *Submitted.*

in the compressed sensing, where one wish to recover an sparse object w from the observation $x = Dw$ and D is called *sensing matrix*.

In reality, the signals are often corrupted by noise. Thus, we should replace the above exact representation with an approximate one $x = Dw + \nu$, where ν is a vector of noises. With the assumption that the noises are zero-mean white and homogeneous Gaussian, finding the sparse representation of x via the dictionary D can be mathematically formulated as the optimization problem

$$\min_{w \in \mathbb{R}^p} L(x, D, w) := \frac{1}{2}\|Dw - x\|^2 + \lambda\phi(w), \tag{4.1}$$

where $\phi : \mathbb{R}^p \mapsto \mathbb{R}$ is a sparsity-inducing function and $\lambda > 0$ is a parameter that controls the trade-off between the reconstruction error and the sparsity.

Naturally, sparsity is modeled by ℓ_0-norm, i.e. $\phi = \|\cdot\|_0$. Unfortunately, this results in an NP-hard problem (Natarajan, 1995). We refer to chapter 2 for a rigorous review of methods for treating ℓ_0. In the context of signal processing, the most popular methods for this problem include the matching pursuit (Mallat and Zhang, 1993), orthogonal matching pursuit (Pati et al., 1993), basis pursuit denoising (Chen et al., 1998), FOCUSS (Gorodnitsky and Rao, 1997; Rao and Kreutz-Delgado, 1999).

Formulation of dictionary learning problem.

A great deal of works have been devoted to the construction of a good dictionary. Dictionaries used in the literature can be divided into two classes: ones are chosen as a predefined set of functions and ones are learned from a given set of signal examples. Popular dictionaries of the former class based on wavelets (Chen et al., 1998; Mallat, 1999) such as wavelets, curvelets, steerable wavelets, Gabor dictionaries, and more. Recently, designing dictionaries based on learning has received a lot of attention. The learning approach was proposed for the first time by (Olshausen and Field, 1996, 1997) and studied in a sequence of works (Engan et al., 1999; Kreutz-Delgado et al., 2003; Aharon et al., 2006; Lee et al., 2007; Mairal et al., 2010; Skretting and Engan, 2010). It has been proved that the use of learned dictionaries instead of off-the-shelf bases gives significantly better results for many image processing tasks (Elad and Aharon, 2006; Mairal et al., 2008; Protter and Elad, 2009; Yang et al., 2010; Mairal et al., 2014).

Let $L : \mathbb{R}^m \times \mathbb{R}^{m\times p} \mapsto \mathbb{R}$ be the loss function defined by

$$L(x, D) := \min_{w \in \mathbb{R}^p} L(x, D, w) = \min_{w \in \mathbb{R}^p} \frac{1}{2}\|Dw - x\|^2 + \lambda\phi(w).$$

The problem of dictionary learning aims at finding a dictionary $D = [d_1, \ldots, d_k] \in \mathbb{R}^{m\times p}$ that results in "good" sparse representations of signals, i.e. the *expected cost* $F(D) := \mathbb{E}_x[L(x, D)]$ is minimal, where $\mathbb{E}(X)$ denotes the expectation of a random variable X. To avoid any instability, it is useful to normalize columns of D. Thus, we will impose the constraint

$D \in \mathcal{C}$, where

$$\mathcal{C} = \{D = [d_1, \ldots, d_p] \in \mathbb{R}^{m\times p} : \|d_j\| \leq 1 \ \forall j = 1, \ldots, p\}.$$

In practice, given a set of training signals $X = [x_1, \ldots, x_n] \in \mathbb{R}^{m\times p}$, we replace the expected cost by the *empirical cost* $F_n(D) := \frac{1}{n}\sum_{i=1}^n L(x_i, D)$. Then learning the dictionary D amounts to solving the minimization problem

$$\min_{D\in\mathcal{C}} F_n(D) := \frac{1}{n}\sum_{i=1}^n L(x_i, D)$$

that is equivalent to

$$\min_{D\in\mathcal{C}, W\in\mathbb{R}^{p\times n}} F(D, W) := \sum_{i=1}^n \frac{1}{2}\|Dw_i - x_i\|^2 + \lambda\phi(w_i), \tag{4.2}$$

where $W = [w_1, \ldots, w_n]$ carries the representation coefficients of signals $x_1, \ldots, x_n$, λ and ϕ as in problem (4.1). It is clear that once the dictionary D is known and fixed, problem (4.2) reduces to n problems (4.1) of finding sparse representations of x_i's.

Denoting $\Phi(W) = \sum_{i=1}^n \phi(w_i)$, we can reformulate problem (4.2) in the form of matrix factorization as follows

$$\min_{D\in\mathcal{C}, W\in\mathbb{R}^{p\times n}} \frac{1}{2}\|DW - X\|_F^2 + \lambda\Phi(W). \tag{4.3}$$

This formulation has very close connection with other factorization techniques such as the NMF and the sparse NMF. All these problems aim to learn an "appropriate" basis that results in desired representation of data. Unlike the the NMF and the sparse NMF, dictionary learning does not impose nonnegativity constraints on matrix factors, instead it only concerns with the sparsity. In fact, the dictionary learning problem only differs from the sparse NMF in that it does not have the nonnegativity constraint. Therefore, as we will see, its solution method is similar to the solution method of the sparse NMF.

Existing methods for solving dictionary learning problem.

Most algorithms for dictionary learning iteratively alternate between two phases: sparse coding and dictionary updating. In the sparse coding phase, a sparse representation of signals is performed while the currently learned dictionary is fixed. In the dictionary updating phase, the learned dictionary is recomputed using the new sparse representation of signals. This kind of alternate minimization was first proposed by (Engan et al., 1999) under the name method of optimal directions (MOD), although (Olshausen and Field, 1996, 1997) actually used the same scheme in their works. We will describe here some representative methods for learning dictionary. More exhaustive reviews are referred to (Mairal et al., 2014).

(Olshausen and Field, 1996, 1997) employed several choices of ϕ including $\phi(w) = \|w\|_1/\sigma, \phi(w) = \sum_{j=1}^p \log(1 + w_j^2/\sigma^2)$ and $\phi(w) = -\sum_{j=1}^p e^{-w_j^2/\sigma^2}$, $w \in \mathbb{R}^p$ where $\sigma > 0$

is a scaling constant. The solution of the sparse coding phase is determined by sequentially setting the derivative w.r.t. W_{ij} be 0 and solving the resulting differential equation. Updating for the new dictionary D is accomplished by simple gradient descent. In these works, they did not explicitly impose any constraint on D, but the proposed algorithm includes a scaling mechanism to keep variables at an appropriate level.

With the MOD method (Engan et al., 1999), sparse coding phase can use either MP, OMP or FOCUSS. For updating the dictionary, it simply takes the analytic solution of the least squares minimization problem then normalizes the solution to have unit ℓ_2-norm.

(Kreutz-Delgado et al., 2003) used the ℓ_p-norm ($p \leq 1$) for modeling sparsity. They also used a simple gradient descent to compute D, while used the FOCUSS to compute sparse representations.

In (Lee et al., 2007), the authors used ℓ_1-norm to favor sparsity. An algorithm for solving the ℓ_1-regularized least squares problem has been proposed for learning the coefficients W, while learning the bases D is done by solving the Lagrange dual of a least squares problem with quadratic constraints.

One of the most popular method for dictionary learning is the K-SVD proposed by (Aharon et al., 2006). This method handles directly with ℓ_0-norm by using a greedy method such as OMP. To update the dictionary, it sequentially updates columns of D as follows. To update the column d_j, it find the subset Ω of signals that use d_j in their current sparse representation. Then the non-zero coefficients of the jth row of W are updated at the same time with d_j by performing a rank-one singular value decomposition of the residue $E_j = X_{:\Omega} - DW_{:\Omega} + d_j W_{\{j\}\Omega}$.

The aforementioned methods are referred to as "batch" learning since they access the whole training set at each iteration. Thus, these methods may not efficiently deal with very large training sets. To address this issue, (Mairal et al., 2010) proposed an *online* approach that processes the signals, one at a time, or in mini-batches. At each iteration t, for the sparse coding phase, this approach considers only one training signal x_t (can be extended to mini-batch) and computes its coefficient w_t using ℓ_1-norm as sparsity-penalty. For the dictionary updating phase, it compute D by using the block coordinate descent for solving the problem

$$\min_{D \in \mathcal{C}} \sum_{i=1}^{t} \frac{1}{2} \|Dw_i - x_i\|^2.$$

(Skretting and Engan, 2010) proposed a similar algorithm called "recursive least square dictionary learning". The differences are that the latter uses either MP, OPM or FOCUSS for sparse representation and updates the dictionary using a recursive least square method.

As we have seen, the dictionary learning problem arises from the sparse representation problem where we want to learn a dictionary adapting well data instead of using a predefined one. The challenge is to solve a huge number of sparse optimization problems corresponding to

training samples. Therefore, we need a method that is efficiently handle sparse optimization problem, especially in the large-scale setting. In this chapter, motivated by the study in chapter 2, we use the capped-ℓ_0 function (Peleg and Meir, 2008) to relax ℓ_0-norm for modeling the sparsity in the dictionary learning problem. We still consider the batch learning approach and follows the alternating framework with two phases: sparse coding and dictionary learning. Algorithms based on DC programming and DCA are developed to solve the subproblems in the two phases. The proposed dictionary learning method is applied to the problem of image denoising to evaluate its efficiency.

The rest of the chapter is organized as follows. Section 4.2 will present the algorithm based on DC programming and DCA for solving the dictionary learning problem. Some experiments of image denosing is reported in section 4.3 to evaluate the performance of our proposed method.

4.2 Algorithm for Dictionary learning problem

4.2.1 General schema solution

As mentioned in the previous section, we consider in the sequel problem (4.2) with ϕ is the capped-ℓ_1 function (Peleg and Meir, 2008) defined by

$$\phi(u) = \sum_{j=1}^{p} \min(1, \alpha|u_j|), \quad u = (u_1, \ldots, u_p) \in \mathbb{R}^p,$$

where $\alpha > 0$ is a parameter.

Following the alternating framework, the schema solution of problem (4.2) is as follows

Schema solution:
Initialized from an initial dictionary D^0, we iteratively alternate these two phases until convergence of D.

- Sparse coding phase: fix D, update W. This phase amounts to solving L problems

$$w_i \in \arg\min \left\{ \frac{1}{2}\|x_i - Dw\|^2 + \lambda\phi(w) : w \in \mathbb{R}^p \right\} \quad \forall i = 1, \ldots, n. \tag{4.4}$$

- Dictionary updating phase: fix W, update D by solving the problem

$$\min_{D \in \mathcal{C}} \frac{1}{2}\|DW - X\|_F^2$$

  or, equivalently,

$$\min_{D \in \mathcal{C}} \frac{1}{2}\langle A, D^T D\rangle - \langle B, D\rangle, \tag{4.5}$$

  where $A = WW^T, B = XW^T$.

As we have mentioned above, the schema solution of dictionary learning problem is similar to the NMF and sparse NMF problems. Since problems (4.4) are nonconvex, it is difficult to find global solutions of these problems. However, similar to the NMF problem, we do not solve these problems precisely. Instead, we compute approximate solutions that decrease the objective function $F(D, W)$ effectively. In the next sections, we will reformulate the subproblems (4.4) and (4.5) as DC programs and present DCA based algorithms for solving them.

4.2.2 Sparse coding phase: update W

For convenience, by omitting the subscript of x, we consider the common form of problems (4.4) as follows

$$\min_{w\in\mathbb{R}^p} \left\{ f_D(w) = \frac{1}{2}\|x - Dw\|^2 + \lambda\phi(w) \right\} \tag{4.6}$$

where $D \in \mathbb{R}^{m\times p}$ and $x \in \mathbb{R}^m$.

It is easy to see that $f_D(w) = f(w) + \lambda\phi(w)$ with f is convex and ϕ is the capped-ℓ_1 function. According to Algorithm 2.1 and Table 2.2 (cf. chapter 2), DCA for solving problem (4.6) iteratively (for $k = 0, 1, 2, \dots$)

- calculates $y^k \in \partial h_D(w^k)$ by

$$y_j^k = \begin{cases} \operatorname{sgn}(w_j^k)\lambda\alpha & \text{if } |w_j^k| > \frac{1}{\alpha} \\ 0 & \text{otherwise,} \end{cases}$$

- calculates $w^{k+1} \in \arg\min\{\frac{1}{2}\|x - Dw\|^2 + \lambda\alpha\|w\|_1 - \langle w, y^k\rangle : w \in \mathbb{R}^p\} \quad (P_k)$.

We will discuss below on how to solve problem (P_k).

DCA for solving problem (P_k).

Omitting the subscript of y, problem (P_k) takes the form

$$\min_{w\in\mathbb{R}^p} \overline{f}_D(w) := \frac{1}{2}\|x - Dw\|^2 + \lambda\alpha\|w\|_1 - \langle w, y\rangle. \tag{P_k}$$

Let $\rho \in \mathbb{R}^p_{++}$ such that $D^TD \preceq \operatorname{diag}(\rho)$, then $\overline{f}_D$ has a DC decomposition $\overline{f}_D = \overline{g}_D - \overline{h}_D$ given by

$$\overline{g}_D(w) = \sum_{j=1}^p \left(\frac{1}{2}\rho_j w_j^2 + \lambda\alpha|w_j| - y_j w_j \right), \quad \overline{h}_D(w) = \frac{1}{2}\sum_{j=1}^p \rho_j w_j^2 - \frac{1}{2}\|x - Dw\|^2.$$

DCA for solving problem (P_k) consists of (for $t = 0, 1, 2, \dots$)

– computing $z^t = \nabla \overline{h}_D(w^t) = \mathrm{diag}(\rho)w^t - D^T(Dw^t - x)$, and
– computing

$$w^{t+1} \in \arg\min \left\{ \sum_{j=1}^p \left(\tfrac{1}{2}\rho_j (w_j)^2 + \lambda\alpha |w_j| - y_j w_j - z_j^t w_j \right) : w \in \mathbb{R}^p \right\}$$
$$\Leftrightarrow \quad w_j^{t+1} = \arg\min_{w_j} \left\{ \tfrac{1}{2}\left(w_j - \tfrac{y_j + z_j^t}{\rho_j} \right) + \tfrac{\lambda\alpha}{\rho_j}|w_j| \right\} = \tfrac{S(y_j + z_j^t, \lambda\alpha)}{\rho_j} \quad \forall j = 1, \ldots, p,$$

where $S(u, \beta) = \mathrm{sgn}(u)(|u| - \beta)_+$ is the soft thresholding operator.

For simplicity, we rewrite the above updating rule in the vector form as follows

$$w^{t+1} = \frac{[S(y + z^t, \lambda\alpha)]}{[\rho]}, \tag{4.7}$$

where the operation S is component–wise, i.e. $(S(a, b))_j = S(a_j, b_j)\ \forall j = 1, \ldots, p$.

According to Lemma 3.1 (cf. chapter 3), we can chose $\rho = |D^T D|\mathbf{1}_{p\times 1}$. However, similar to the resolution of NNLS problem, we can do in a more effective way. Observe that if $w_j^t = 0$ and $|D_{:j}^T(Dw - x) - y_j| \leq \lambda\alpha$, we have $S(y_j + z_j^t, \lambda\alpha) = 0$ for any choice of ρ. Thus, the updating rule (4.7) makes no change on the element j^{th} of w^{t+1}. If we define

$$I(w, y) = \{j = 1, \ldots, p : w_j \neq 0 \text{ or } |D_{:j}^T(Dw - x) - y_j| > \lambda\alpha\}, \quad w, y \in \mathbb{R}^p,$$

then at the iteration t^{th}, we only need to consider variables $\{w_j : j \in I\}$ (here we write $I = I(w^t, y)$ for short). Repeat the above procedure with w_I (resp. $D_{:I}$ and y_I) replacing w (resp. D and y), we compute

$$z_I^t = \mathrm{diag}(\rho_I)w_I^t - D_{:I}^T(D_{:I}w_I^t - x_I)$$

and

$$w_I^{t+1} = \frac{[S(y_I + z_I^t, \lambda\alpha)]}{[\rho_I]}, \quad w_j^{t+1} = 0, \quad \forall j \notin I,$$

where $\rho_I = |D_{:I}^T D_{:I}|\mathbf{1}_{|I|\times 1}$ and $\rho_j = 0\ \forall j \notin I$.

These are equivalent to compute

$$\omega = \frac{[q]}{[|D^T D|q]},$$

where $q \in \mathbb{R}^p$ is defined by $q_j = 1$ if $j \in I$ and $q_j = 0$ otherwise.

Then we have

$$z^t = w^t - (D^T(Dw^t - x) - y) \circ \omega, \tag{4.8}$$
$$w^{t+1} = S(z^t, \lambda\alpha\omega). \tag{4.9}$$

Proposition 4.1 *Under the updating rules (4.8)–(4.9), the function $\overline{f}_D$ is decreasing. Moreover, if w^t is not a stationary point (also global solution since problem (P_k) is convex) of problem (P_k) then $\overline{f}_D(w^{t+1}) < \overline{f}_D(w^t)$.*

Proof : Since $\overline{f}_D$ is convex, w is a solution of problem (P_k) if and only if:

$$0 \in \partial \bar{f}_D(w) \Leftrightarrow \begin{cases} |D_{:j}^T(Dw - x) - y_j| \leq \lambda\alpha & \text{if } w_j = 0 \\ D_{:j}^T(Dw - x) - y_j = -\text{sign}(w_j) & \text{if } w_j \neq 0 \end{cases} \quad \forall j = 1, \ldots, p. \tag{4.10}$$

Thus, the variables $\{w_j : j \notin I(w, y)\}$ are already satisfying the conditions (4.10). By restricting problem (P_k) to variables $\{w_j : j \in I(w, y)\}$ the assertions of this proposition are consequences of general convergence properties of DCA. □

Note that, in the context of dictionary learning, w is expected to be very sparse. This implies that very few components of w need to be updated (corresponding to $\omega_j \neq 0$). To exploit this fact for solving problem (P_k), we will not recalculate ω after each iteration. Instead, we only compute ω from the beginning and keep using it later on. This means that we do not actually solve problem (P_k). We are now in a position to describe the DCA for solving problem (4.6).

DCA-SC: DCA for the sparse representation
Initialization: Initialize $w^0 \in \mathbb{R}^p$, $T > 0$ (maximum number of inner–iterations), $\epsilon > 0$ (stopping tolerance), $k \leftarrow 0$
Repeat
1. Compute $y^k \in \partial h(w^k)$ by $y_i^k = 0$ if $|w_i^k| \leq \frac{1}{\alpha}$ and $y_i^k = \text{sgn}(w_i^k)\lambda\alpha$ otherwise, for all $i = 1, \ldots, p$
2. Compute q^k by $q_i^k = 1$ if $i \in I(w^k, y^k)$ and $q_i^k = 0$ otherwise, for all $i = 1, \ldots, p$
3. Compute $\omega^k = \frac{[q^k]}{[|D^T D|q^k]}$, $w^{(k,0)} = w^k$, and set $t = 0$
   Repeat
   - $t = t + 1$
   - Compute $z^t = w^{(k,t-1)} - (D^T(Dw^{(k,t-1)} - x) - y^k) \circ \omega^k$
   - Compute $w^{(k,t)} = S(z^t, \lambda\alpha\omega^k)$
   Until $t = T$ or $\|w^{(k,t-1)} - w^{(k,t)}\| / \max(1, \|w^{k,t}\|) < \epsilon$
4. Set $w^{k+1} = w^{(k,t)}$
5. Set $k \leftarrow k + 1$
Until $\|w^k - w^{k-1}\| / \max(1, \|w^k\|) < \epsilon$.

Before going to the result concerning convergence of this algorithm, we describe the charac-

teristics of critical points of problem (4.6). We have

$$y \in \partial g_D(w) \Leftrightarrow \begin{cases} y_j - D_{:j}^T(Dw - x) = \text{sgn}(w_j)\lambda\alpha & \text{if } w_j \neq 0, \\ y_j - D_{:j}^T(Dw - x) \in [-\lambda\alpha, \lambda\alpha] & \text{if } w_j = 0, \end{cases} \quad \forall j = 1, \ldots, p$$

$$y \in \partial h_D(w) \Leftrightarrow \begin{cases} y_j = 0 & \text{if } |w_j| < \frac{1}{\alpha}, \\ y_j \in \text{sgn}(w_j)[0, \lambda\alpha] & \text{if } |w_j| = \frac{1}{\alpha}, \\ y_j = \text{sgn}(w_j)\lambda\alpha & \text{if } |w_j| > \frac{1}{\alpha}. \end{cases} \quad \forall j = 1, \ldots, p.$$

Therefore, w is a critical point of (4.6) (i.e. $\partial g_D(w) \cap \partial h_D(w) \neq \emptyset$) if and only if

$$\begin{cases} D_{:j}^T(Dw - x) \in [-\lambda\alpha, \lambda\alpha] & \text{if } w_j = 0, \\ D_{:j}^T(Dw - x) = -\text{sign}(w_j)\lambda\alpha & \text{if } |w_j| \in (0, \frac{1}{\alpha}), \\ D_{:j}^T(Dw - x) \in -\text{sign}(w_j)[0, \lambda\alpha] & \text{if } |w_j| = \frac{1}{\alpha}, \\ D_{:j}^T(Dw - x) = 0 & \text{if if } |w_j| > \frac{1}{\alpha}. \end{cases} \quad \forall j = 1, \ldots, p \tag{4.11}$$

Theorem 4.1 *Suppose that $\{w^k\}$ is the sequence generated by algorithm **DCA-SC**. Then $\{f_D(w^k)\}$ is a decreasing sequence and any limit point of the sequence $\{w^k\}$ is a critical point of problem (4.6).*

Proof : For any $k = 0, 1, 2, \ldots$, we have:

$$f_D(w) \leq \bar{f}_D(w) + C, \quad \forall w \in \mathbb{R}^p,$$

where $C = k\lambda - h_D(w^k) + \langle y^k, w^k \rangle$, and the equality holds if $w = w^k$. Thus, by Proposition 4.1, we have

$$f_D(w^{k+1}) \leq \bar{f}_D(w^{k+1}) + C \leq \bar{f}_D(w^{(k,1)}) + C \leq \bar{f}_D(w^{(k,0)}) + C = f_D(w^k).$$

The first assertion is proved. Moreover, if w^k is not a critical point of problem (4.6), then $y^k \notin \partial g_D(w^k)$. This also means that $w^k = w^{(k,0)}$ is not a critical point of problem (P_k) (not satisfying condition (4.10)) and that $I(w^k) \neq \emptyset$. By Proposition 4.1, $\bar{f}_D(w^{(k,1)}) < \bar{f}_D(w^{(k,0)})$, and consequently $f_D(w^{k+1}) < f_D(w^k)$. This implies that if $f_D(w^{k+1}) = f_D(w^k)$ then w^k is a critical point of problem (4.6) and algorithm **DCA-SC** terminates at the kth iteration.

Assume that w^* is an arbitrary limit point of the sequence $\{w^k\}_{k=0}^{\infty}$. Consider any subsequence $\{w^k\}_{k \in \mathcal{R}}$ with $\mathcal{R} \subseteq \{0, 1, 2, \ldots\}$ converging to w^*. Then we have

$$f_D(w^*) = \lim_{k \in \mathcal{R}, k \to +\infty} f_D(w^k) = \inf_{k=0,1,2,\ldots} f_D(w^k) \geq 0. \tag{4.12}$$

Note that $\{y^k\}$ (resp. $\{q^k\}$ and $\{\omega^k\}$) generated by **DCA-SC** has finite values. Thus, by passing to a subsequence, if necessary, we can assume that for any $k \in \mathcal{R}$, $y^k = y^*$, $q^k = q^*$

and $\omega^k = \omega^*$, for some $y^* \in \{0, \lambda\alpha, -\lambda\alpha\}^p$, $q^* \in \{0,1\}^k$ and $\omega^* = \frac{[q^*]}{[|D^T D|q^*]}$. Moreover, we also assume that for any $k \in \mathcal{R}$, computing w^{k+1} from w^k (loop **for** in this algorithm) takes the same number of inner iterations $t^* \in \{1, \ldots, T\}$.

Consider now the function $\psi : \mathbb{R}^p \to \mathbb{R}^p$ defined by:

$$\psi(w) = S(w - (D^T(Dw - x) - y^*) \circ \omega^*, \lambda\alpha\omega^*), \quad w \in \mathbb{R}^p.$$

We have ψ and $\Psi = \psi \circ \cdots \circ \psi$ (t^* times) are continuous functions, and:

$$w^{k+1} = \Psi(w^k), \quad \forall k \in \mathcal{R}.$$

This implies that $\{w^{k+1}\}_{k\in\mathcal{R}}$ converges to $\Psi(w^*)$ and $f_D(\Psi(w^*)) = f_D(w^*)$.
Moreover, since $\{w^k\}_{k\in\mathcal{R}}$ converges to w^*, there is an k_0 such that for any $k \in \mathcal{R}$ and $k \geq k_0$,

$$\begin{aligned}
&I(w^*, y^*) \subseteq I(w^k, y^*) = \{j : q_j^* = 1\},\\
&\left\{j : |w_j^*| < \frac{1}{\alpha}\right\} \subseteq \left\{j : |w_j^k| < \frac{1}{\alpha}\right\} \subseteq \left\{j : y_j^* = 0\right\},\\
&\left\{j : w_j^* > \frac{1}{\alpha}\right\} \subseteq \left\{j : w_j^k > \frac{1}{\alpha}\right\} = \left\{j : y_j^* = \lambda\alpha\right\},\\
&\left\{j : w_j^* < -\frac{1}{\alpha}\right\} \subseteq \left\{j : w_j^k < -\frac{1}{\alpha}\right\} = \left\{j : y_j^* = -\lambda\alpha\right\},\\
&y_j^* \in \{0, \lambda\alpha\} \text{ if } w_j^* = \frac{1}{\alpha}, \quad y_j^* \in \{0, -\lambda\alpha\} \text{ if } w_j^* = -\frac{1}{\alpha}.
\end{aligned}$$

Therefore, $y^* \in \partial h_D(w^*)$. By the same arguments as at the beginning of this proof, we have w^* is a critical point of problem (4.6). □

4.2.3 Dictionary updating phase: update D

For updating D we solve the optimization of the form

$$\min_{D\in\mathcal{C}} f_W(D) := \frac{1}{2}\langle A, D^T D\rangle - \langle B, D\rangle, \tag{4.13}$$

where $A = WW^T, B = XW^T$.

Let $\gamma = \mathbf{1}_{1\times p}|A|$, we can decompose f_W as $f_W = g_W - h_W$, where g_W and h_W are given by

$$g_W(D) = \frac{1}{2}\sum_{j=1}^{p}\gamma_j\|D_{:j}\|^2, \quad h_W(D) = \frac{1}{2}\sum_{j=1}^{p}\gamma_j\|D_{:j}\|^2 - \left(\frac{1}{2}\langle A, D^T D\rangle - \langle B, D\rangle\right).$$

It is clearly that g_W is convex and h_W is also a convex function by Lemma 3.1 (cf. chapter 3) and the fact that

$$h_W(D) = \sum_{i=1}^{m} \left[\frac{1}{2} \sum_{j=1}^{p} \gamma_j D_{ij}^2 - \left(\frac{1}{2} D_{i:} A D_{i:}^T - \langle B_{i:}, D_{i:} \rangle \right) \right].$$

DCA for solving problem (4.13) then consists of (for $t = 0, 1, 2, \dots$)

- Compute $\overline{D}^{(t)} = \nabla h_W(D^{(t)}) = \Gamma \circ D^{(t)} - (D^{(t)} A - B)$, where $\Gamma \in \mathbb{R}^{m \times p}$ is the matrix defined by $\Gamma_{i:} = \gamma$, $\forall i = 1, \dots, m$.
- Compute

$$\begin{aligned} D^{(t+1)} &= \arg\min \left\{ g_W(D) - \langle \overline{D}^{(t)}, D \rangle : D = [d_1, \dots, d_p] \in \mathcal{C} \right\} \\ &= \arg\min_{D \in \mathcal{C}} \sum_{j=1}^{p} \left(\frac{1}{2} \gamma_j \|d_j\|^2 - \langle \bar{d}_j^{(t)}, d_j \rangle \right) = \arg\min_{D \in \mathcal{C}} \sum_{j=1}^{p} \left\| d_j - \frac{1}{\gamma_j} \bar{d}_j^{(t)} \right\|^2 \\ \Leftrightarrow \quad d_j^{(t+1)} &= \operatorname*{Proj}_{\|d_j\| \le 1} \frac{\bar{d}_j^{(t)}}{\gamma_j} = \frac{\bar{d}_j^{(t)}}{\max\{\gamma_j, \|\bar{d}_j^{(t)}\|\}}, \quad \forall j = 1, \dots, p. \end{aligned} \tag{4.14}$$

We summarize this procedure in the following algorithm.

DCA-D: DCA for the dictionary updating stage
Initialization: Initial matrix $D^{(0)} \in \mathcal{C}$, $t \leftarrow 0$
Repeat
- Compute $\overline{D}^{(t)} = \Gamma \circ D^{(t)} - (D^{(t)} A - B)$.
- Compute $D^{(t+1)}$ by (4.14).
- Set $t \leftarrow t + 1$.

Until $\|D^{(t-1)} - D^{(t)}\| < \epsilon$.

Since the problem (4.13) is convex, general convergence of DCA implies that

Theorem 4.2 *Any limit point D^* of the sequence $\{D^{(t)}\}$ generated by the above algorithm is a global solution of problem (4.13).*

4.3 Application to image denoising

4.3.1 Image denoising protocol

In this section, we are interested in the image denoising problem. During the image acquisition, due to the effect of the acquisition device as well as of the environment, the measured

image will be affected by noise. Denoting by X the original image, the measured image Y can be modeled as

$$Y = X + \nu,$$

where ν stands for additive zero-mean white and homogeneous Gaussian noise with standard deviation σ. We desire to design an algorithm that can remove the noise from Y and get as close as possible to the original image X. We will present below the protocol for image denoising proposed in (Elad and Aharon, 2006).

Assume that size of X is $\sqrt{N} \times \sqrt{N}$. We consider image patches of size $\sqrt{n} \times \sqrt{n}$ ($n \ll N$) ordered lexicographically as column vectors $x \in \mathbb{R}^n$. The denoised image $\hat{X}$ (the estimate of the unknown original image X) can be found by solving the following denoising problem

$$\{\hat{X}, \hat{w}_{ij}, \hat{D}\} \in \underset{D\in\mathcal{C},W,x}{\arg\min}\, \beta\|X-Y\|_F^2 + \lambda \sum_{ij} \|w_{ij}\|_0 + \frac{1}{2}\sum_{ij} \|Dw_{ij} - R_{ij}X\|^2, \quad \beta, \lambda > 0. \quad (4.15)$$

In this formulation, D is dictionary of size $n \times p$. (i, j) is the index indicating the location of the patch $x_{ij} = R_{ij}X$ of size $\sqrt{n} \times \sqrt{n}$, while R_{ij} is an $n \times N$ matrix that extracts the (i, j) block from the image X. $w_{ij} \in \mathbb{R}^p$ is the sparse representation of x_{ij} over the dictionary D. By considering all image patches of size $\sqrt{n}\times\sqrt{n}$ in X with overlaps, we have $(\sqrt{N}-\sqrt{n}+1)^2$ patches in total. The first term in (4.4) imposes the proximity between the measured image Y and its denoised (and unknown) version X. The second and the third terms are the image prior that makes sure that in the constructed image X, every patch $x_{ij} = R_{ij}X$ has a sparse representation with a bounded error.

In the formulation (4.4), we have three unknowns: the denoised image X, the underlying dictionary D and the sparse representations w_{ij} per each location of X. Instead of addressing these three unknowns together, we conduct a three-step procedure as follows.

- Initialize $X = Y$.
- Step 1. Learn the dictionary $\widehat{D}$ from a training set of image patches $Z = [z_1, \ldots, z_M]$ of X by solving the dictionary learning problem

$$\{\widehat{D}, \widehat{W}\} \in \underset{D\in\mathcal{C},W\in\mathbb{R}^{n\times M}}{\arg\min}\, \frac{1}{2}\|DW - Z\|_F^2 + \lambda\|W\|_0. \quad (4.16)$$

- Step 2. Find sparse representations for all image patches x_{ij} over the dictionary $\widehat{D}$ learned in Step 1

$$\hat{w}_{ij} \in \underset{w\in\mathbb{R}^p}{\arg\min}\, \frac{1}{2}\|\widehat{D}w - x_{ij}\|^2 + \lambda\|w\|_0. \quad (4.17)$$

- Step 3. Compute the denoised image by

$$\hat{X} = \underset{X\in\mathbb{R}^{N\times N}}{\arg\min}\, \beta\|X - Y\|_F^2 + \frac{1}{2}\sum_{ij} \|\widehat{D}\hat{w}_{ij} - R_{ij}X\|^2.$$

$$\Leftrightarrow \quad \hat{X} = \left(2\beta I + \sum_{ij} R_{ij}^T R_{ij}\right)^{-1} \left(2\beta Y + \sum_{ij} R_{ij}^T \widehat{D}\hat{w}_{ij}\right). \quad (4.18)$$

The training set in step 1 can be the set of all patches of X or only a part of it. If we let Z is the set of all patches of X, the step 2 becomes redundant since we have already compute sparse representations of patches of X in step 1. Solving problems (4.16) and (4.17) have been presented in section 4.2, where we replace ℓ_0-norm by the capped-ℓ_1 function. The rather cumbersome expression (4.18) simply means that we compute the average of all the denoised patches $\widehat{D}\hat{w}_{ij}$, then we take a weighted average between the resulting image $\left(\sum_{ij} R_{ij}^T R_{ij}\right)^{-1}\left(\sum_{ij} R_{ij}^T \widehat{D}\hat{w}_{ij}\right)$ with the original noisy image Y to form the denoised one.

4.3.2 Numerical experiment

In this section, we carry out some experiments of image denosing to demonstrate the efficiency of our dictionary learning method. For the sake of comparison, we also implemented the closely related dictionary learning method based on ℓ_1-norm. Two these algorithms were implemented in the Matlab R2007a, and executed on a PC Intel i5 CPU650, 3.2 GHz of 4GB RAM. We also compare with the well-known standard algorithm K-SVD (Aharon et al., 2006). The experiments were conducted on five gray scale images: lena, barbara, boat, house, and peppers. The code of K-SVD and the testing images are taken from KSVD-Box (v3) package (`http://www.cs.technion.ac.il/~ronrubin/software.html`).

We generated noisy images by adding zero-mean white and homogeneous Gaussian noise with deviation $\sigma = 20$. For each testing image, the training set includes $L = 40000$ patches of size 8×8, which are regularly sampled from the original noisy image in an overlapping manner. Each patch is converted to a vector of size $n = 64$ and then normalized to have zero mean. The size of the dictionary was set to $k = 256$. The initialization of D in dictionary learning algorithm is chosen as an overcomplete DCT dictionary (Discrete Cosine Transforms).

For methods based on ℓ_0 and ℓ_1 norms, the value of the trade-off parameter λ was chosen in the set $\{500, 600, \ldots, 1500\}$. The parameter α of the capped-ℓ_1 function was set to 1. β in (4.18) was set to be 0.1.

We use the PSNR (Peak signal–to–noise ratio) to evaluate the denoising results obtained. The larger the PSNR is, the better the denoising is. Given a noise-free $m \times n$ monochrome image I and its noisy approximation J, the mean squared error (MSE) is defined as

$$\mathrm{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - J(i,j)]^2.$$

Then the PSNR is computed via the MSE by

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}_I^2}{\mathrm{MSE}}\right),$$

where MAX_I is the maximum possible pixel value of the image, which is 255 for an 8–bits image.

(a) Original image (b) Noise image

(c) denoised image by our method (d) denoised image by ℓ_1

Figure 4.1: Lena

Figures 4.1, 4.2, 4.3, 4.4, 4.5 show the denoising results for considered images. We observe that our dictionary learning method can provide better denoising result compared with the ℓ_1-norm method. Specifically, our results tend to be smoother and clearer.

Table 4.1 shows the peak signal-to-noise ratio (PSNR) of our method and the methods using ℓ_1 and K-SVD. It can be noticed that, our method achieves a significantly better performance of denoising in terms of PSNR than the ℓ_1-based method. And it is a little better than K-SVD.

4.4 Conclusion

In this chapter, we have studied the DC programming and DCA for dictionary learning problem and applied it to the denoising image problem. Inspired by the success of capped-ℓ_1

(a) Original image

(b) Noise image

(c) denoised image by our method

(d) denoised image by ℓ_1

Figure 4.2: Barbara

(a) Original image

(b) Noise image

(c) denoised image by our method

(d) denoised image by ℓ_1

Figure 4.3: Boat

(a) Original image

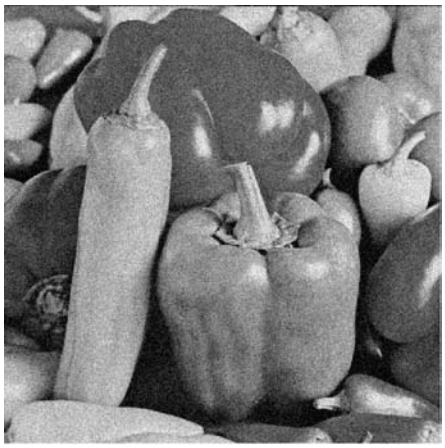
(b) Noise image

(c) denoised image by our method

(d) denoised image by ℓ_1

Figure 4.4: House

(a) Original image



(b) Noise image

(c) denoised image by our method

(d) denoised image by ℓ_1

Figure 4.5: Peppers

function in modeling the sparsity, we investigate it in the context of dictionary learning. Following the alternating framework, we alternate between two phases: sparse coding and dictionary updating. DCA algorithms have been developed to solve the subproblems in each phase. The efficiency of DCA for dictionary learning has been evaluated on the application of denoising gray images. The numerical results show that our method is promising. In future works, we will extend our method to online learning manner and apply to other applications of image processing.

Table 4.1: PSNR comparison with KSVD and ℓ_1 penalty

Method	barbara	boat	house	lena	peppers
our method	**31.00**	**30.48**	**33.42**	**32.54**	**32.39**
KSVD	30.86	30.37	33.18	32.42	32.22
ℓ_1	29.70	29.75	32.04	31.55	31.59

Part II

Learning with uncertainty

Chapter 5

Feature Selection for linear SVMs under Uncertain Data[1]

Abstract: *This chapter considers the problem of feature selection for linear SVMs on uncertain data that is inherently prevalent in almost all datasets. Using principles of Robust Optimization, we propose robust schemes to handle data with ellipsoidal model and box model of uncertainty. The difficulty in treating ℓ_0-norm in feature selection problem is overcome by using appropriate approximations and DC programming and DCA. The computational results show that the proposed robust optimization approaches are superior than a traditional approach in immunizing perturbation of the data.*

5.1 Introduction

Feature selection, which consists of choosing a subset of available features that capture the relevant properties of the data, is one of fundamental problems in machine learning. The goals are to remove the irrelevant and redundant features, reduce store space and execution time, and avoid the course of dimensionality to improve the prediction performance. Feature selection can help enhance accuracy in many machine learning problems, it can also improve the efficiency of training. Machine learning methods for feature selection can be divided into three classes (Rinaldi et al., 2009): wrapper (exploit a machine learning algorithm to evaluate the usefulness of features), filter (rank the features according to some discrimination measure and select features having higher ranks without using any learning algorithm), and

1. This chapter is published under the titles:
[1]. Hoai An Le Thi, Xuan Thanh Vo, Tao Pham Dinh. Robust Feature Selection for SVMs under Uncertain Data, in P. Perner (Ed), *ICDM 2013: Proceedings of the 13th Industrial Conference on Advances in Data Mining*, LNAI 7987, pp. 151–165, Springer 2013.
[2]. Hoai An Le Thi, Xuan Thanh Vo, Tao Pham Dinh. Feature Selection for SVMs under Uncertain Data: Robust optimization based on Difference of Convex functions Algorithms. *Neural Networks* 59: 36–50 (2014).

embedded methods (do not separate the learning from the feature selection part, integrate the selection of features in the model building).

In this chapter, we focus on an embedded approach for feature selection in the context of two-class linear Support Vector Machines (SVMs) with the presence of uncertain data. In the traditional feature selection in SVMs, the patterns (assumed as vectors $x \in \mathbb{R}^n$) belong to one of two classes (labeled by $+1$ or -1), and we seek to discriminate them by a hyperplane $\mathcal{H} = \{x \in \mathbb{R}^n : \langle w, x\rangle + b = 0\}$, $(w \in \mathbb{R}^n, b \in \mathbb{R})$ which uses as few features as possible. This aims to select a subset of relevant features while preserving the discriminative ability and improving the performance of classifier. For this traditional approach, the input data (x, δ)–patterns with corresponding labels–are given *exactly*. However, as mentioned above, this is unrealistic because uncertainty data is ubiquitous in many real world applications. The uncertainty can occur on the patterns x as well as on the labels $\delta \in \{+1, -1\}$. With noisy training data on x one can get a wrong SVM classifier because the classification problem is based on the observed data having noisy inputs. SVMs classifier are also sensitive to outliers of labels. For instance, the points in the training set far away from their own classes may give a SVM model with unbounded hinge loss (Wu and Liu (2007)). The methodology to tackle the uncertain data are different in each of case (noisy on x or on labels).

In the context of this chapter we will assume that the uncertainty is only in the patterns x and the labels $\delta \in \{+1, -1\}$ are known precisely whenever given. Motivated by robust optimization approach, the notion of uncertainty is made explicit by specifying the allowable values of a data point via an ellipsoid or a box. Borrowed from measurement error concept in statistics (Carroll et al., 2006), the additive uncertainty model assumes that $x^i_{true} = x^i + u^i$, where $\{x^i\}$ are given observed data and $\{u^i\}$ represent perturbation. In contrast to statistical models, robust optimization does not assume any probability distribution function on the perturbation and it protects against the uncertainty by minimizing the regularized training loss on all possible value of the perturbation u^i in some uncertainty set. This paradigm makes sense when the perturbations are not stochastic, or the distribution is not known or partially known. Moreover, the robust optimization has the ability to derive priori probability guarantees – e.g., probability of feasibility – that the solution to a robust optimization will satisfy up to a confidence level.

We take a look at existing works on classification under uncertain data. Bhattacharyya et al. (2004b) developed a Second Order Cone Programming (SOCP) SVM formulation to design a robust linear classifier when the uncertainty was described by multivariate normal distributions. This work has been generalized in (Shivaswamy et al., 2006) by proposing a SOCP formulation for designing robust binary classifier for arbitrary distributions having finite mean and covariance. The latter approach can be interpreted as the ellipsoidal bounded uncertainty. Bi and Zhang (2004) provided the Total Support Vector Classification (TSVC) formulation for bounded uncertainties. A similar work was developed in (Pant et al., 2011) for robust SVM classification of imbalanced and noisy data. In the above works, the authors use ℓ_2-norm for regularization. In (Bhattacharyya et al., 2004a), the authors developed robust sparse hyperplanes based on ellipsoidal data uncertainty model to uncertain molecular

profiling data using the sparsity-inducing regularizer ℓ_1. In another direction, Wu and Liu (2007) proposed a robust truncated hinge loss SVM to deal with data uncertainty on labels. A review of robust optimization in machine learning can be found in (Caramanis et al., 2011). Works on the feature selection problem on uncertain data are rarely encountered. Beside bounded uncertainty set approach, Bayesian setup was also addressed in works of Bi and Zhang (2004) and Caramanis et al. (2011) which give useful relations between robust optimization approach and statistical modeling approach.

In this chapter, we consider two models of uncertain data – ellipsoidal and box model which include the ℓ_0-norm as the regularizer term for feature selection purpose. We carefully explore and exploit robust optimization approaches based on DCA from both a theoretical and an algorithmic point of view. Two models (ellipsoid / box) related to uncertain data are studied. Dealing with the ℓ_0-norm, following the DC approximation approaches presented in Chapter 2, we consider its two approximations: the concave exponential approximation and the capped-ℓ_1. Related key questions are investigated: geometric / probabilistic interpretations of uncertainty models, the efficiency of algorithms in different uncertainty models, "good" approximations of ℓ_0-norm, the choices of DC decomposition to get interesting convergence properties of DCA. For example, the capped-ℓ_1 enjoys some advantages for the resulting DCA on the optimality conditions and the finiteness of convergence. Numerical experiments on several test problems with careful comparative studies between different models and methods are reported.

The rest of this chapter is organized as follows. The next section states the problem of feature selection and classification with uncertain data, and specifies the ellipsoidal model and the box model. In section 5.3, we show how to apply DC programming and DCA to solve our robust feature selection and classification problems. Section 5.4 presents the numerical experiments.

5.2 Feature Selection for SVMs under Uncertain Data

5.2.1 Feature Selection for Linear Two-class SVM Models

We first recall the model of feature selection for linear two-class SVM. Consider a two-class dataset consisting of N data points as well as labels, $\{(x^i, \delta_i)\}_{i=1}^N \subset \mathbb{R}^n \times \{-1, 1\}$. We suppose that $N = m + k$, and there are m data points belong to the class with label $+1$ while there are k data points belong to the class with label -1. The feature selection for SVM problem is formulated in (Bradley and Mangasarian, 1998) as follows:

$$\min_{w,b} \quad (1-\lambda)\left(\sum_{i=1}^{N} \sigma_i \left[1 - \delta_i(\langle w, x^i\rangle + b)\right]_+\right) + \lambda\|w\|_0, \tag{5.1}$$

where $\sigma_i = \frac{1}{m}$ if $\delta_i = 1$ and $\sigma_i = \frac{1}{k}$ if $\delta_i = -1$, and the parameter $\lambda \in [0, 1]$ is a measure of trade-off between misclassification and sparsity.

5.2.2 Data uncertainty model and robust counterpart

Assume that each input data x^i $(i = 1, \ldots, N)$ varies in a given *uncertainty set* $\mathcal{U}_i$. Then, *uncertain problem* corresponding to (5.1) is a collection of the form

$$\left\{ \min_{w,b} \quad (1-\lambda)\left(\sum_{i=1}^{N} \sigma_i \left[1 - \delta_i(\langle w, x^i\rangle + b)\right]_+\right) + \lambda\|w\|_0 \right\}_{\substack{x^i \in \mathcal{U}_i \\ i=1,\ldots,N}}. \tag{5.2}$$

Since uncertainty on data points x^i is separable, the *Robust Counterpart* of the uncertain problem (5.2) is given by

$$\min_{w,b} \left\{ (1-\lambda)\left(\sum_{i=1}^{N} \sigma_i \sup_{x^i \in \mathcal{U}_i} \left[1 - \delta_i(\langle w, x^i\rangle + b)\right]_+\right) + \lambda\|w\|_0 \right\},$$

or equivalently,

$$\begin{array}{ll} \min\limits_{w,b,\xi} & (1-\lambda)\sum_{i=1}^{N} \sigma_i \xi_i + \lambda\|w\|_0 \\ s.t & \xi_i \geq \sup\limits_{x^i \in \mathcal{U}_i} \left(1 - \delta_i(\langle w, x^i\rangle + b)\right), \xi_i \geq 0 \quad \forall i = 1, \ldots, N. \end{array} \tag{5.3}$$

5.2.3 Ellipsoidal Uncertainty Model

In this section, we consider a simple case when the input data uncertainty is described by ellipsoidal sets, called the ellipsoidal uncertainty model. This means that each input data x^i $(i = 1, \ldots, N)$ varies in an ellipsoid defined by

$$\mathcal{E}_i = \mathcal{E}(\overline{x}^i, P_i) = \{\overline{x}^i + P_i^{1/2} u : \|u\|_2 \leq 1\},$$

where $\overline{x}^i$ represents the centre, and the symmetric positive semidefinite matrix P_i represents the shape of the ellipsoid $\mathcal{E}_i$. The centre $\overline{x}^i$ is referred as *nominal value* of x^i. Substituting $x^i = \overline{x}^i + P_i^{1/2} u$, $(\|u\|_2 \leq 1)$, we have

$$\begin{aligned} \sup_{x^i \in \mathcal{E}_i} \left(1 - \delta_i(\langle w, x^i\rangle + b)\right) &= 1 - \delta_i(\langle w, \overline{x}^i\rangle + b) + \sup_{\|u\|_2 \leq 1} \langle P_i^{1/2} w, u\rangle \\ &= 1 - \delta_i(\langle w, \overline{x}^i\rangle + b) + \|P_i^{1/2} w\|_2. \end{aligned}$$

Then, the robust feature selection problem takes the form

$$\min \left\{ (1-\lambda)\sum_{i=1}^{N} \sigma_i \xi_i + \lambda\|w\|_0 : (w, b, \xi) \in K_e \right\}, \tag{5.4}$$

where

$$K_e = \Big\{(w, b, \xi) : \delta_i(\langle w, \overline{x}^i\rangle + b) \geq 1 - \xi_i + \|P_i^{1/2} w\|_2, \xi_i \geq 0, i = 1, \ldots, N\Big\}$$

is a closed convex set.

Below, we give some geometric interpretations for the ellipsoidal uncertainty model.

From the characterizations of the solution of the problem (5.4) given in Proposition 5.1 below (whose proof is trivial and will be omitted here), we see that, the robust constraints

$$\delta_i(\langle w, \overline{x}^i\rangle + b) - 1 \geq \|P_i^{1/2} w\|_2 - \xi_i \text{ and } \xi_i \geq 0$$

in the robust formulation mean that we try to find a hyperplane which separates not only the nominal value $\overline{x}^i$ but also entirely corresponding uncertainty set. And by solving the robust problem, we desire to reduce error in worst-case sense.

Proposition 5.1 *Suppose that* $(\hat{w}, \hat{b}, \hat{\xi})$ *is a solution to the problem* (5.4) *and* $\hat{w} \neq 0$. *Then we have, for any* $i = 1, \ldots, N$,

$$\hat{\xi}_i = \left[\sup_{x^i \in \mathcal{E}_i} \Big\{1 - \delta_i(\langle \hat{w}, x^i\rangle + \hat{b})\Big\}\right]_+ = \Big[1 - \delta_i(\langle \hat{w}, \hat{x}^i\rangle + \hat{b})\Big]_+, \tag{5.5}$$

where $\hat{x}^i$ *is determined by* $\hat{x}^i = \overline{x}^i - \delta_i \dfrac{P_i \hat{w}}{\|P_i^{1/2}\hat{w}\|_2} \in \mathcal{E}_i, \quad i = 1, \ldots, N.$

In fact, it is easy to see that $\hat{x}^i$ in Proposition 5.1 is on the boundary of the ellipsoid $\mathcal{E}_i$. If the separation constraints are violated, a part or entire uncertainty set is on wrong side and (5.5) shows that $\hat{x}^i$ is the *worst-case* – i.e. $\hat{x}^i$ is the most severely misclassified point. Therefore, the value of $\hat{\xi}^i$ will measure misclassification in worst-case. The worst-case error occurs if and only if $\hat{x}^i$ is misclassified, that is,

$$\delta_i(\langle w, \hat{x}^i\rangle + b) \leq 0 \quad \Leftrightarrow \quad \delta_i(\langle w, \overline{x}^i\rangle + b) \leq \|P_i^{1/2} w\|_2.$$

The worst-case error seem to be too pessimistic. A more optimistic measure is the concept of *expected error* (Shivaswamy et al., 2006). We assume that data is uniformly distributed in an uncertainty set. Then the expected error is computed as the ratio of the volume of the ellipsoid on the wrong side of the hyperplane to the entire volume of the ellipsoid. For $(\hat{w}, \hat{b}, \hat{\xi})$, as in Proposition 5.1, the part of the ellipsoid $\mathcal{E}_i$ on the wrong side of the bounding plane $\{x \in \mathbb{R}^n : \delta_i(\langle \hat{w}, x\rangle + \hat{b}) - 1 = 0\}$ is $S_i = \{x \in \mathbb{R}^n : \delta_i(\langle \hat{w}, x\rangle + \hat{b}) - 1 \leq 0\}$. The following Proposition 5.2 shows that $\mathbf{vol}\, S_i$ is monotonic to $\hat{\xi}_i / \|P^{1/2}\hat{w}\|_2 = \Big[1 - \delta_i(\langle \hat{w}, \hat{x}^i\rangle + \hat{b})\Big]_+ / \|P^{1/2}\hat{w}\|_2$. Hence $\hat{\xi}_i$ measures misclassification not only in worst-case but also in expected sense. Therefore, by solving the robust problem, we also desire to reduce the expected error.

Proposition 5.2 *Give ellipsoid* $\mathcal{E} = \{x_0 + P^{1/2}u : \|u\|_2 \le 1\}$, *where* $x_0 \in \mathbb{R}^n$ *and* $P \in \mathbb{R}^{n\times n}$ *is a positive definite matrix. For* $w \in \mathbb{R}^n \setminus \{0\}, b \in \mathbb{R}$, *let*

$$\hat{x} = x_0 - \frac{Pw}{\|P^{1/2}w\|_2}, \quad S(w,b) = \{x \in \mathcal{E} : \langle w, x\rangle + b \le 0\},$$

$$\zeta(w,b) = \frac{[-(\langle w,\hat{x}\rangle + b)]_+}{\|P^{1/2}w\|_2} = \frac{\left[-(\langle w, x_0\rangle + b) + \|P^{1/2}w\|_2\right]_+}{\|P^{1/2}w\|_2}.$$

Then, $\mathbf{vol}\, S(w,b)$ *is monotonic with respect to* $\zeta(w,b)$, *where* $\mathbf{vol}\, A$ *stands for volume of* A. *Precisely, we have*

$$\zeta(w,b) \le \zeta(w',b') \Longrightarrow \mathbf{vol}\, S(w,b) \le \mathbf{vol}\, S(w',b').$$

Proof : Consider the transformation $\phi : \mathbb{R}^n \to \mathbb{R}^n$ defined by $x \mapsto x_0 + P^{-1/2}(x - x_0)$. We have $\mathcal{B} := \phi(\mathcal{E}) = \{x_0 + u : \|u\|_2 \le 1\}$ is the unit ball at center x_0 in $\mathbb{R}^n$. Let

$$\overline{w} = P^{1/2}w, \quad \overline{b} = \langle x_0, w - P^{1/2}w\rangle + b, \quad \hat{y} = \phi(\hat{x}) = x_0 - \frac{\overline{w}}{\|\overline{w}\|_2}.$$

Then

$$\begin{aligned} \phi(S(w,b)) = \{y \in \mathcal{B} : \langle \overline{w}, y\rangle + \overline{b} \le 0\} &=: T(\overline{w},\overline{b}), \\ \zeta(w,b) = \frac{\left[-(\langle \overline{w},\hat{y}\rangle + \overline{b})\right]_+}{\|\overline{w}\|_2} &=: \gamma(\overline{w},\overline{b}), \end{aligned}$$

and $\mathbf{vol}\, \phi(S(w,b)) = (\det P)^{-1/2}\mathbf{vol}\ S(w,b)$. Therefore, it suffices to show that $\mathbf{vol}\, T(\overline{w},\overline{b})$ is monotonic with respect to $\gamma(\overline{w},\overline{b})$.

It is easy to verify that $\hat{y} = \operatorname{argmin}\{\langle \overline{w}, y\rangle + \overline{b} : y \in \mathcal{B}\}$. Thus, if $\gamma(\overline{w},\overline{b}) = 0$ or equivalently $\langle \overline{w},\hat{y}\rangle + \overline{b} \ge 0$, then $\langle \overline{w}, y\rangle + \overline{b} \ge 0\ \forall y \in \mathcal{B}$, and $\mathbf{vol}\, T(\overline{w},\overline{b}) = 0$.

Assume that $0 < \gamma(\overline{w},\overline{b}) \le \gamma(\overline{w}',\overline{b}')$ $(\overline{w},\overline{w}' \ne 0)$, or equivalently

$$0 < \frac{-(\langle \overline{w},\hat{y}\rangle + \overline{b})}{\|\overline{w}\|_2} \le \frac{-(\langle \overline{w}',\hat{y}'\rangle + \overline{b}')}{\|\overline{w}'\|_2} \Leftrightarrow \frac{\langle \overline{w}', x_0\rangle + \overline{b}'}{\|\overline{w}'\|_2} \le \frac{\langle \overline{w}, x_0\rangle + \overline{b}}{\|\overline{w}\|_2} < 1.$$

Since $\overline{w},\overline{w}' \ne 0$, by Lemma A.2 in Appendix A.2, there exists an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ and $k > 0$ such that $Q\overline{w}' = k\overline{w}$. Then, $\|\overline{w'}\|_2 = \|Q\overline{w'}\|_2 = k\|\overline{w}\|_2$. From the last inequality, we have

$$\frac{1}{k}(\langle \overline{w}', x_0\rangle + \overline{b}') - \langle \overline{w}, x_0\rangle \le \overline{b}. \tag{5.6}$$

Consider the transformation $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ defined by $y \mapsto x_0 + Q(y - x_0)$. We have

$$\begin{aligned} \varphi(T(\overline{w}',\overline{b}')) &= \left\{x_0 + Q(y - x_0) : y \in \mathcal{B}, \langle \overline{w}', y\rangle + \overline{b}' \le 0\right\} \\ &= \left\{y \in \mathcal{B} : \langle Q\overline{w}', y - x_0\rangle + \langle \overline{w}', x_0\rangle + \overline{b}' \le 0\right\} \\ &= \left\{y \in \mathcal{B} : \langle \overline{w}, y\rangle + \frac{1}{k}(\langle \overline{w}', x_0\rangle + \overline{b}') - \langle \overline{w}, x_0\rangle \le 0\right\}. \end{aligned}$$

By (5.6), we deduce that $T(\overline{w},\overline{b}) \subset \varphi(T(\overline{w}',\overline{b}'))$. Hence, $\mathbf{vol}\, T(\overline{w},\overline{b}) \leq \mathbf{vol}\, \varphi(T(\overline{w}',\overline{b}'))$. Moreover, Q is orthogonal, one has $|\det Q| = 1$. Thus,

$$\mathbf{vol}\, \varphi(T(\overline{w}',\overline{b}')) = |\det Q|\, \mathbf{vol}\, T(\overline{w}',\overline{b}') = \mathbf{vol}\, T(\overline{w}',\overline{b}').$$

Hence, $\mathbf{vol}\, T(\overline{w},\overline{b}) \leq \mathbf{vol}\, T(\overline{w}',\overline{b}')$. □

Probabilistic Interpretation Before closing this section, we give another interpretation that is meaningful.

For each $i = 1, \ldots, N$, we assume that x^i is a random vector with mean $\mathbf{E}(x^i) = \overline{x}^i$ and variance $\mathbf{Var}(x^i) = P_i$. Then, the quality $\zeta_i = 1 - \delta_i(\langle w, x^i\rangle + b)$ is also a random variable with mean $\mathbf{E}(\zeta_i) = 1 - \delta_i(\langle w, \overline{x}^i\rangle + b)$ and variance

$$\mathbf{Var}(\zeta_i) = \mathbf{Var}(\langle w, x^i\rangle) = w^T P_i w = \|P_i^{1/2} w\|_2^2.$$

For any $\rho > 0$, by Chebyshev's inequality, we have

$$\mathbf{Pr}\left(\zeta_i > \mathbf{E}(\zeta_i) + \rho\sqrt{\mathbf{Var}(\zeta_i)}\right) \leq \mathbf{Pr}\left(|\zeta_i - \mathbf{E}(\zeta_i)| > \rho\sqrt{\mathbf{Var}(\zeta_i)}\right) \leq \frac{1}{\rho^2}.$$

Especially, if x^i is the normal distribution and so is ζ_i, then

$$\mathbf{Pr}\left(\zeta_i > \mathbf{E}(\zeta_i) + \rho\sqrt{\mathbf{Var}(\zeta_i)}\right) = 1 - \Phi(\rho) \leq \exp(-\rho^2/2),$$

where Φ is the normal cumulative distribution function, that is

$$\Phi(u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} \exp\left(-\frac{s^2}{2}\right) ds.$$

Therefore, for large enough $\rho > 0$, the probability that $\zeta_i > \mathbf{E}(\zeta_i) + \rho\sqrt{\mathbf{Var}(\zeta_i)}$ is small. Let us choose a "safety parameter" $\rho > 0$ and ignore the "rare event" $\zeta_i > \mathbf{E}(\zeta_i) + \rho\sqrt{\mathbf{Var}(\zeta_i)}$, the robust value of $[1 - \delta_i(\langle w, x^i\rangle + b)]_+ = (\zeta_i)_+$ in the objective function of (5.1) can be taken at

$$\left[\mathbf{E}(\zeta_i) + \rho\sqrt{\mathbf{Var}(\zeta_i)}\right]_+ = \left[1 - \delta_i(\langle w, \overline{x}^i\rangle + b) + \rho\|P_i^{1/2} w\|_2\right]_+.$$

Then, a robust counterpart of (5.2) by this way is similar to (5.4) where x^i's are assumed to vary in $\mathcal{E}(\overline{x}^i, \rho P_i)$ respectively.

5.2.4 Box uncertainty model

In this section, we consider another model of uncertainty, that is the box model. This means that each input data $x^i, (i = 1, \ldots, N)$ varies in the box defined by

$$\mathcal{B}_i = \mathcal{B}(\overline{x}^i, d^i) = \{\overline{x}^i + \Delta x : |\Delta x| \leq d^i\},$$

where $d^i = (d_1^i, \ldots, d_n^i) \in \mathbb{R}_+^n$ and $\Delta x = (\Delta x_1, \ldots, \Delta x_n) \in \mathbb{R}^n$. The centre point $\overline{x}^i$ is also referred as *nominal value* of x^i, and d^i represents dimensions of uncertainty set $\mathcal{B}_i$.

Substituting $x^i = \overline{x}^i + \Delta x \, (|\Delta x| \leq d^i)$, we have

$$\begin{aligned} \sup_{x^i \in \mathcal{B}_i} \big(1 - \delta_i(\langle w, x^i \rangle + b)\big) &= 1 - \delta_i(\langle w, \overline{x}^i \rangle + b) + \sup_{|\Delta x| \leq d^i} \langle w, \Delta x \rangle \\ &= 1 - \delta_i(\langle w, \overline{x}^i \rangle + b) + \langle |w|, d^i \rangle. \end{aligned}$$

Then, the robust feature selection problem for box uncertainty model has the form

$$\min \left\{ (1 - \lambda) \sum_{i=1}^{N} \sigma_i \xi_i + \lambda \|w\|_0 : (w, b, \xi) \in K_b \right\}, \tag{5.7}$$

where

$$K_b = \big\{ (w, b, \xi) : \delta_i(\langle w, \overline{x}^i \rangle + b) \geq 1 - \xi_i + \langle |w|, d^i \rangle, \xi_i \geq 0, i = 1, \ldots, N \big\}$$

is a *polyhedral* convex set.

Similar to the ellipsoidal uncertainty model, we have a geometric interpretation for the case of box uncertainty model.

Proposition 5.3 *Suppose that* $(\hat{w}, \hat{b}, \hat{\xi})$ *is a solution to the problem* (5.7) *and* $\hat{w} \neq 0$. *Then we have, for any* $i = 1, \ldots, N$,

$$\hat{\xi}_i = \left[\sup_{x^i \in \mathcal{B}_i} \left\{ 1 - \delta_i(\langle \hat{w}, x^i \rangle + \hat{b}) \right\} \right]_+ = \left[1 - \delta_i(\langle \hat{w}, \hat{x}^i \rangle + \hat{b}) \right]_+,$$

where $\hat{x}^i = \overline{x}_i + \hat{\Delta x}$ *with* $\hat{\Delta x}$ *being determined by*

$$\hat{\Delta x}_j = \begin{cases} d_j^i & \text{if } \delta_i w_j < 0 \\ -d_j^i & \text{if } \delta_i w_j \geq 0 \end{cases}, j = \overline{1, n}.$$

Clearly, the point $\hat{x}^i$ in above Proposition is a vertex of the box $\mathcal{B}_i$, and is a point that is most severely misclassified. Similar to the ellipsoidal model, in case of box model, we also try to reduce misclassification in "worst-case".

From Proposition 5.3, we have a way to check whenever misclassification in "worst-case" occurs. The worst-case misclassification occurs if and only if the worst-case point $\hat{x}^i$ is misclassified.

We also have a probabilistic interpretation for box uncertainty model as follows. For each $i = 1, \ldots, N$, assume that x_j^i's $(j = 1, \ldots, n)$ are random variables with mean $\mathbf{E}(x_j^i) = \overline{x}_j^i$

and variance $\mathbf{Var}(x_j^i) = (d_j^i)^2$. Assuming that x_j^i's are independent, for any $\rho > 0$, we have

$$\mathbf{Pr}(|x^i - \overline{x}^i| \leq \rho d^i) = \prod_{j=1}^{n} \mathbf{Pr}(|x_j^i - \overline{x}_j^i| \leq \rho d_j^i) \geq \left(1 - \frac{1}{\rho^2}\right)^n.$$

Thus, for a confidence level $\kappa \in [0, 1]$, we can choose a $\rho > 0$ such that $x^i \in \mathcal{B}(\overline{x}^i, \rho d^i)$ with probability no less than κ. Then the robust value of $[1 - \delta_i(\langle w, x^i \rangle + b)]_+$ in the objective function of (5.1) can be taken at

$$\sup_{x^i \in \mathcal{B}(\overline{x}^i, \rho d^i)} \left[1 - \delta_i(\langle w, x^i \rangle + b)\right]_+ = \left[1 - \delta_i(\langle w, \overline{x}^i \rangle + b) + \rho \langle |w|, d^i \rangle\right]_+.$$

Note that, with the information of mean and variance of x_j^i $(j = 1, \ldots, n)$ as stated above, the variable ζ^i defined at the end of Sect. 5.2.3 has $\mathbf{E}(\zeta^i) = \overline{x}^i$ and $\mathbf{Var}(\zeta^i) = \sum_{j=1}^n (d_j^i w_j)^2$. Then with the same parameter ρ that determines the confidence interval of x_j^i's, the robust value of $[1 - \delta_i(\langle w, x^i \rangle + b)]_+$ given by ellipsoidal uncertainty model will be

$$\left[1 - \delta_i(\langle w, \overline{x}^i \rangle + b) + \rho \sqrt{\sum_{j=1}^{n} (d_j^i w_j)^2}\right]_+.$$

Since $\sqrt{\sum_{j=1}^n (d_j^i w_j)^2} \leq \langle |w|, d^i \rangle$, the robust objective value given by box uncertainty model is more conservative than the one given by ellipsoidal uncertainty model.

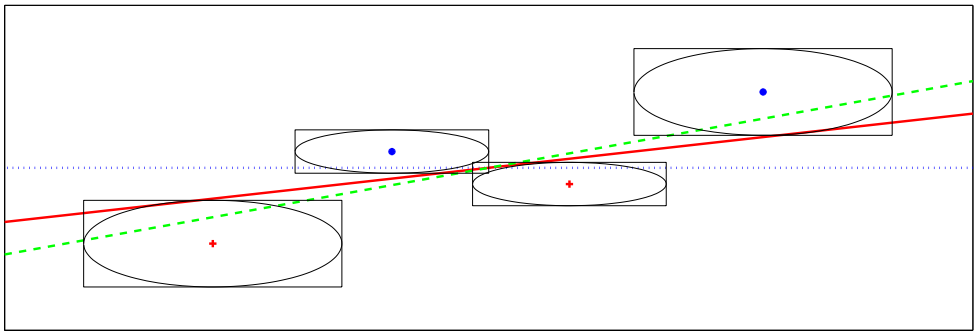


Figure 5.1: Crosses and stars represent patterns belonging to the two classes. The ellipsoid (resp. box) around the pattern denotes the uncertainty ellipsoid (resp. box). The dash line represents the nominal classifier, the solid line represents the robust classifier using ellipsoidal uncertainty and the dotted represents the robust classifier using box uncertainty. Wee see that the robust classifier using box uncertainty tends to be sparse than the robust classifier using ellipsoidal uncertainty.

We are now going to present solution methods based on DC programming and DCA for solving the robust feature selection problems (5.4) and (5.7).

5.3 Solution methods based on DC programming and DCA

In what follows we will use the common notation K to denote the set K_e (resp. K_b) in case of ellipsoidal (resp. box) uncertainty model.

Let

$$f(w, b, \xi) := (1 - \lambda) \sum_{i=1}^{N} \sigma_i \xi_i.$$

Then problems (5.4) and (5.7) have the common form

$$\min \{F(w, b, \xi) := f(w, b, \xi) + \lambda \|w\|_0 : (w, b, \xi) \in K\} \tag{5.8}$$

that is the form of problem (2.1). Following the DC approximation approach for sparse optimization (cf. chapter 2), we will approximate the ℓ_0-norm by a DC approximation function then apply DC programming and DCA for solving the resulting problem. Motivated by the success of the concave exponential function (Bradley and Mangasarian, 1998) and the capped-ℓ_1 function (Peleg and Meir, 2008) in linear SVMs (cf. section 2.6, chapter 2), we propose to use these approximations for the ℓ_0-norm in this chapter.

For $\theta > 0$, considering the concave exponential function

$$r_{exp}(t) = 1 - \exp(-\theta|t|), \quad t \in \mathbb{R},$$

we have $\|w\|_0$ is approximated by $\|w\|_0 \approx \sum_{i=1}^{n} r_{exp}(w_i)$. With this approximation, the robust feature selection problem (5.8) becomes

$$\min \left\{ F_1(w, b, \xi) := f(w, b, \xi) + \lambda \sum_{j=1}^{n} r_{exp}(w_j) : (w, b, \xi) \in K \right\}. \tag{5.9}$$

Similarly, by considering the capped-ℓ_1 function

$$r_{cap}(t) = \min(1, \theta|t|), \quad t \in \mathbb{R},$$

an approximation of $\|w\|_0$ is $\|w\|_0 \approx \sum_{i=1}^{n} r_{cap}(w_i)$. Then the robust feature selection problem (5.8) becomes

$$\min \left\{ F_2(w, b, \xi) := f(w, b, \xi) + \lambda \sum_{j=1}^{n} r_{cap}(w_j) : (w, b, \xi) \in K \right\}. \tag{5.10}$$

5.3.1 DCA for solving problem (5.9)

Note that f is a convex function. According to (2.39) (cf. chapter 2), we can reformulate (5.9) as the following DC program

$$\min\{G(X) - H_1(X) : X = (w, b, \xi) \in K\}, \tag{5.11}$$

where

$$G(w, b, \xi) := f(w, b, \xi) + \lambda\theta\|w\|_1, \tag{5.12}$$

and

$$H_1(w, b, \xi) := \lambda \sum_{j=1}^{n} (\theta|w_i| - r_{exp}(w_i)). \tag{5.13}$$

By virtue of Algorithm 2.1 and Table 2.2, DCA applied to (5.11) consists of computing two sequences $\{z^l\} \subset \mathbb{R}^n$ and $\{X^l = (w^l, b^l, \xi^l)\} \subset \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^N$ such that

$$z_j^l = \operatorname{sgn}(w_j^l)\lambda\theta(1 - \exp(-\theta|w_j^l|)), \quad \forall j = 1, \ldots, n, \tag{5.14}$$

and $X^l = (w^l, b^l, \xi^l)$ is a solution of the convex problem

$$\min\{G(X) - \langle z^l, w\rangle : X = (w, b, \xi) \in K\}.$$

that is equivalent to the following convex problem

$$\min\left\{(1-\lambda)\sum_{i=1}^{N}\sigma_i\xi_i + \lambda\alpha\sum_{j=1}^{n} t_j - \langle z^l, w\rangle : (w, b, \xi, t) \in \Omega\right\}, \tag{5.15}$$

where Ω is a closed convex set defined by

$$\Omega := \left\{(w, b, \xi, t) \in \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^N \times \mathbb{R}^n : (w, b, \xi) \in K, |w_j| \le t_j, j = 1, \ldots, n\right\}.$$

Note that, for ellipsoidal uncertainty model ($K = K_e$), the problem (5.15) is an instance of Second Order Cone Program (SOCP). In case of box uncertainty model, $K = K_b$ is a convex polyhedral set and then (5.15) is a Linear Program (LP). On another hand, g_α is convex polyhedral and so is G. Thus, in the box uncertainty model, (5.11) is a DC polyhedral program.

The DCA for solving (5.11) can be described as follows.

DCA1 (DCA applied to (5.11))

Initialization:

- Let ϵ be a tolerance sufficiently small, $l \leftarrow 0$.
- Choose a starting point $X^0 = (w^0, b^0, \xi^0) \in K$.
- Compute $F^0 = F_1(X^0)$.

Repeat

- Compute $z^l \in \mathbb{R}^n$ by $z_j^l = \text{sgn}(w_j^l)\lambda\theta(1 - \exp(-\theta|w_j^l|)), \quad \forall j = 1, \ldots, n$.
- Solve the convex problem (5.15) to obtain $X^{l+1} = (w^{l+1}, b^{l+1}, \xi^{l+1})$.
- $l \leftarrow l + 1$.

Until $\|X^l - X^{l+1}\|_1 < \epsilon(1 + \|X^l\|_1)$ or $|F_1(X^{l+1}) - F_1(X^l)| < \epsilon(1 + F_1(X^l))$.

Theorem 5.1 (Convergence properties of DCA1) *(i) DCA1 generates the sequence $\{X^l = (w^l, b^l, \xi^l)\}$ in K such that $\{F_1(X^l)\}$ is decreasing.*
(ii) If the sequence $\{X^l\}$ is bounded, then every limit point $X^* = (w^*, b^*, \xi^*)$ satisfies the necessary local optimality condition $\partial H_1(X^*) \subset \partial(G + \chi_K)(X^*)$.
(iii) In the case of box uncertainty model ($K = K_b$), the sequence $\{X^l\}$ converges to X^* after a finite number of iterations.

Proof : Observing that F_1 is bounded below by zero in K and H_1 is differentiable everywhere, (i) and (ii) are consequences of the convergence properties of general DC programs.

As mentioned above, in the case of box uncertainty model ($K = K_b$), (5.11) is a polyhedral DC program. Therefore, (iii) follows convergence properties of polyhedral DC programs. □

5.3.2 DCA for solving problem (5.10)

Similar to the previous section, (5.10) can be reformulated as the DC program

$$\min\{G(X) - H_2(X) : X = (w, b, \xi) \in K\}, \tag{5.16}$$

where $G(w, b, \xi)$ is given by (5.12), and $H_2(w, b, \xi) := \lambda \sum_{j=1}^{n} (\theta|w_i| - r_{cap}(w_i))$.

Note that, since $\theta|t| - r_{cap}(t) = \max(1, \theta|t|) - 1$ is convex polyhedral and so is H_2, the problem (5.16) is always a polyhedral DC program in both ellipsoidal and box uncertainty models.

According to Algorithm 2.1 and Table 2.2, DCA applied to DC program (5.16) is similar to DCA1. We simply replace the computation of z^l in (5.14) with

$$z_j^l = \begin{cases} 0 & \text{if } |w_j^l| \leq \frac{1}{\theta} \\ \text{sgn}(w_j^l)\lambda\theta & \text{otherwise} \end{cases}, \quad \forall j = 1, \ldots, n. \tag{5.17}$$

The DCA for solving (5.16) is described as follows.

DCA2 (DCA applied to (5.16))

Initialization:

- Let ϵ be a tolerance sufficiently small, $l \leftarrow 0$.
- Choose a starting point $X^0 = (w^0, b^0, \xi^0) \in K$.
- Compute $F^0 = F_2(X^0)$.

Repeat

- Compute $z^l \in \mathbb{R}^n$ by $z_j^l = \begin{cases} 0 & \text{if } |w_j^l| \leq \frac{1}{\theta} \\ \text{sgn}(w_j^l)\lambda\theta & \text{otherwise} \end{cases}, \quad \forall j = 1, \ldots, n.$
- Solve the convex problem (5.15) to obtain $X^{l+1} = (w^{l+1}, b^{l+1}, \xi^{l+1})$.
- $l \leftarrow l + 1$.

Until $F_2(X^{l+1}) = F_2(X^l)$.

Theorem 5.2 (Convergence properties of DCA2) *(i) DCA2 generates the sequence $\{X^l = (w^l, b^l, \xi^l)\}$ in K such that $\{F_2(X^l)\}$ is decreasing.*
(ii) The sequence $\{X^l\}$ converges to $X^* = (w^*, b^*, \xi^*)$ after a finite number of iterations.
(iii) The point X^* is a critical point of the function $(F_2 + \chi_K)$. Moreover, if

$$w_j^* \notin \left\{\frac{1}{\alpha}, -\frac{1}{\alpha}\right\}, \quad \forall j = 1, \ldots, n, \tag{5.18}$$

then X^* is a local minimizer of (5.16).

Proof : (i) and the first part of (iii) are direct consequences of the convergence properties of general DC programs while (ii) is a convergence property of a DC polyhedral program.

For the second part of (iii), observing that the second DC component of (5.16), say H_2, is a polyhedral function. If the condition (5.18) holds, then H_2 is differentiable at $X^* = (w^*, b^*, \xi^*)$. Using the DCA's convergence property, we deduce that X^* is a local minimizer of (5.16). □

5.3.3 Robust Feature Selection using ℓ_1-regularizer

We state here a formulation for the robust feature selection problem using ℓ_1-regularizer

$$\min \left\{ (1-\lambda) \sum_{i=1}^{N} \sigma_i \xi_i + \lambda \|w\|_1 : (w, b, \xi) \in K \right\}, \tag{5.19}$$

which is equivalent to

$$\min\left\{(1-\lambda)\sum_{i=1}^{N}\sigma_i\xi_i+\lambda\sum_{j=1}^{n}t_j : (w,b,\xi)\in K, -t_j\le w_j\le t_j \forall j=1,\dots,n\right\}. \tag{5.20}$$

(5.20) becomes a linear program when $K = K_b$ and a SOCP when $K = K_e$.

This formulation is slightly different from formulation studied in (Bhattacharyya et al., 2004a) when we put different weights on slack variables ξ_i. In the numerical experiments we will compare the solution of this formulation with the ones computed by our approaches. We also use the solution of this formulation as starting point for our algorithms.

5.4 Numerical Experiments

The numerical experiments aim to evaluate the performance of nominal solutions and robust solutions of the feature selection SVM problem under the impact of data uncertainty. We consider three approaches: ℓ_0(DCA1), ℓ_0(DCA2) (DCA applied to (5.9) and (5.10) respectively), and the standard approach based on the ℓ_1-regularizer model (5.19). Nominal classifiers and robust classifiers with different values of noise level ρ (see the definition below) were trained on training sets. The error rates (ordinary, worst-case and expected) were computed on the test set, in which the uncertainty of data is added by using the same shapes as for the training set.

The algorithms have been coded in VC++ and implemented on a Intel Core i5 CPU 2×2.74 GHz, RAM 4GB. All convex problems (5.20) and (P_k) (5.15) are solved by the commercial software CPLEX 11.2.

5.4.1 Error Measures

Here we briefly describe error measures which will be used to evaluate the performance of algorithms. Consider the separating hyperplane $\mathcal{H}(w,b) = \{x \,|\, \langle w,x\rangle + b = 0\}$, and data $\{(x^i,\delta_i)\}_{i=1}^N$ or $\{\mathcal{U}_i,\delta_i\}_{i=1}^N$, where $\{\mathcal{U}_i\}_{i=1}^N$ are uncertainty sets. For a data point x^i and the corresponding uncertainty set $\mathcal{U}_i$, we have:

- *Ordinary error* occurs when x^i is misclassified, i.e. $\mathrm{sgn}\{\langle w,x^i\rangle + b\}$ differs from δ_i.
- *Worst-case error* occurs when x^i has an ordinary error or $\mathcal{H}(w,b)$ intersects the corresponding uncertainty set $\mathcal{U}_i$. When the worst-case error occurs, we can always find a point within the uncertainty set that get misclassified. Otherwise, the entire uncertainty set is correctly classified. For each sample, the worst-case error is either zero or one. Computing the worst-case error for ellipsoidal and box models has been discussed in previous sections.
- *Expected error*, as mentioned in Sect. 5.2.3, is defined as follows. We determine the volume of the uncertainty set on the wrong side of the hyperplane. Then we use the ratio of this

volume to the volume of the entire uncertainty set as the expected error. So the expected error for each sample is between 0 and 1. In our experiments, this error was computed by generating a large number of uniformly distributed points in the uncertainty set and then taking the fraction of the number of points on the wrong side of the hyperplane to the total number of points generated. In our experiments, the number of generated points is set to be 1000.

Note that, these types of error are coincident if there is no noise or the noise level is zero.

5.4.2 Datasets

We evaluate the performance of various approaches on a synthetic dataset and a collection of real world datasets.

The synthetic data consists of data points with $n = 10$ features, where only the two first features are informative while the others are redundant. The data are generated as follows. First, we generate class label δ with $\mathbf{P}(\delta = 1) = 0.5$. Then given δ, if $\delta = 1$, $(x_1, x_2)^T$ is computed from a multivariate Gaussian distribution $N(\mu, \Sigma)$, where $\mu = (0, 2)^T$ and

$$\Sigma = 3u^T u + 0.4v^T v, \quad \text{with} \quad u = \frac{1}{\sqrt{2}}(1, -1)^T, \quad v = \frac{1}{\sqrt{2}}(1, 1)^T.$$

If $\delta = -1$, $(x_1, x_2)^T$ is computed from multivariate Gaussian distribution $N(-\mu, \Sigma)$. The remaining eight variables x_j $(j = 3, \ldots, 10)$ are drawn independently from $N(0, 5^2)$. For each $j \in \{1, \ldots, 10\}$, we generate a sample of 5 instances from Gaussian distribution $N(x_j, \omega_j^2)$, where $\omega_j = 0.2|x_j| + 0.5$ if $\delta = 1$ and $\omega_j = 0.2|x_j| + 0.1$ otherwise, then calculate its mean and standard deviation. These values are used as nominal value and perturbation of x_j.

The real world datasets consist of three real datasets from UCI repository [2] and two real microarray gene expression datasets. Three datasets from UCI involve the Parkinsons dataset, the Wisconsin Diagnostic Breast Cancer Database, and the Johns Hopkins University Ionosphere dataset. Two gene expression datasets are Leukemia (Golub et al., 1999) and Lung Cancer (Gordon et al., 2002). All the datasets are preprocessed by normalizing each dimension of the data to zero mean and unit variance. Detailed information of these datasets is summarized in Table 5.1.

Below, we present the way to create uncertainty sets for real datasets, some other options can be found in (Bhattacharyya et al., 2004a).

The centres and shapes of uncertainty sets The centers of ellipsoids or boxes are equated with observed data points, say $\overline{x}^i \equiv x^i$. We set $P_i = P^+$, $d^i = d^+$ if the label

2. http://archive.ics.uci.edu/ml

Table 5.1: Real datasets used in experiments. The numbers in bracket present class distribution +1/-1

Dataset	# Features	# Training points	# Test points
Parkinsons	22	130 (98/32)	65 (49/16)
WDBC	30	380 (142/238)	189 (70/119)
Ionosphere	34	234 (84/150)	117 (42/75)
Leukemia	7129	47 (31/17)	25 (16/8)
Lung Cancer	12533	121 (21/100)	60 (10/50)

$\delta_i = +1$, and $P_i = P^-$, $d^i = d^-$ if the label $\delta_i = -1$, where $P^\pm = \text{diag}(p_1^\pm, \ldots, p_n^\pm)$ and $d^\pm = (d_1^\pm, \ldots, d_n^\pm)$ are determined by the empirical standard deviations of features as follows

$$p_j^+ = (d_j^+)^2 = \frac{1}{m} \sum_{\substack{i=\overline{1,N} \\ \delta_i=1}} (x_j^i)^2 - \left(\frac{1}{m} \sum_{\substack{i=\overline{1,N} \\ \delta_i=1}} x_j^i \right)^2,$$

and

$$p_j^- = (d_j^-)^2 = \frac{1}{k} \sum_{\substack{i=\overline{1,N} \\ \delta_i=-1}} (x_j^i)^2 - \left(\frac{1}{k} \sum_{\substack{i=\overline{1,N} \\ \delta_i=-1}} x_j^i \right)^2$$

for any $j = 1, \ldots, n$.

The noise level parameter $\boldsymbol{\rho}$ To investigate the effect of different amounts of data uncertainty, we use a noise level parameter $\rho \geq 0$ to scale uncertainty sets. That is, we replace P and d by $\rho^2 P$ and ρd respectively. By this way, we can control the degree of the perturbation. When $\rho = 0$, there is no perturbation in data points.

5.4.3 Experimental setups

In experiments, we set $\theta = 5$ that gives a reasonable approximation of ℓ_0-norm as suggested in (Bradley and Mangasarian, 1998) and the stop tolerance $\epsilon = 10^{-6}$ for DCA. Concerning the parameter θ, from the theoretical point of view, the larger θ is, the better approximation of the ℓ_0-norm is. However, when we tried with larger θ (up to 100), the result is not improved.

The non-zero elements of w are determined according to whether its relative magnitude exceeds a small threshold $\tau > 0$, i.e. $|w_j| / \max_k(|w_k|) \geq \tau$. In these experiments, the threshold τ is set to be 10^{-4}.

Table 5.2: Results on synthetic dataset of ℓ_1, ℓ_0(DCA1), ℓ_0(DCA2) approaches in term of the percentage of testing errors and number of selected features of nominal (No.) and of robust classifiers in ellipsoidal model (Ellip.) and box model (Box). The numbers in parentheses are number of correct model over 50 trials. Bold font indicates the best result in each approach.

	Testing Error (%)			Avg. Number of selected features		
Approach	No.	Box	Ellip.	No.	Box	Ellip.
ℓ_1	4.07	**3.70**	3.93	7.12 (0)	**5.94 (0)**	8.44 (0)
ℓ_0(DCA1)	4.31	3.63	**3.52**	2.06 (47)	**2.0 (48)**	2.24 (40)
ℓ_0(DCA2)	4.37	**3.56**	**3.51**	2.08 (47)	**2.02 (49)**	2.04 (48)

The value of parameter λ is chosen through a grid search. It has been proved in (Thiao et al., 2008) that a good value of λ for the feature selection problem using zero-norm (5.1) should be smaller 0.5. Thus, we are suggested using the set of candidate values of λ given by

$$\Lambda = \{0.01, 0.02, 0.003, 0.004, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}.$$

For the experiment on synthetic data, the noise level ρ is also a tuning parameter and takes value over the range $\{0.1, 0.2, \ldots, 1.9, 2\}$.

5.4.4 Experiment on synthetic data

In this experiment, we generated training, tuning, and test datasets in the same manner as described in Sect. 5.4.2. The tuning datasets are used to choose the parameters λ and ρ, while the test datasets are used to measure the accuracy of various classifiers trained on the training datasets. We set the sample sizes of training, tuning, and testing datasets as 150, 1000 and 10000. We performed 50 trials for each experimental setting.

The experimental results on synthetic data are given in Table 5.2. In this table, the test error, the average number of selected features, as well as the number of correct models[3] over 50 trials are reported.

We observe from the Table 5.2 that, in terms of feature selection, ℓ_0(DCA1,DCA2) approaches give much better results than ℓ_1 approach. Among 50 trials, ℓ_1 approach does not give any correct model. Meanwhile, the ℓ_0(DCA1,DCA2) approaches retrieve quite correctly the informative features – the percentage of correct models varies from 94% to 98%, except for ℓ_0(DCA1) approach on ellipsoidal model where the percentage of correct models is 80%. We also see that the box model has the best performance in term of feature selection in all three approaches. In term of testing errors, the results are comparable. For nominal solutions the ℓ_1 approach is slightly better (0.7% and 0.69% versus the ℓ_0(DCA2) and ℓ_0(DCA1) respectively) while for robust solutions ℓ_0(DCA1) and ℓ_0(DCA2) are slightly better (the gains

3. A correct model here means a model only uses the two first features x_1 and x_2

vary from 0.07% to 0.42%). In each approach, not surprisingly, robust solutions have less errors than nominal solutions.

5.4.5 Experiments on real datasets

For the experiments on the real datasets, we used the cross-validation scheme to validate the performance of various classifiers. For a given noise, each dataset has been tested 10 times with different training and test sets. At each time, 2/3 samples were randomly chosen for training, and the remaning samples were employed for test. We test on different parameters defining noise.

The parameter λ was chosen via 5-fold cross-validation based on the ordinary error.

In Tables 5.3-5.4 we report the best average results (among all tests with different parameters defining noise) on 10 runs given by each algorithm. We are interested in the efficiency (the sparsity and the classification error) as well as the rapidity of the algorithms. We are also concerned with the stability of feature selection process under bootstrapping training samples. Hence we indicate in Table the standard deviation of numbers of selected future on 10 runs.

In the figures 5.2–5.6, we present the average results (ordinary errors, worst-case errors and percentage of selected features of nominal and robust classifiers) on 10 runs with different values of noise parameters.

The table 5.3 reports expected errors of different approaches. For each dataset, we considered the performance of all robust solutions corresponding to different ρ indicated in figures 5.2–5.6 and reported the best result. We also provided the average percentage of selected features and its standard deviation over ten cross-validation. This experiment assumed that data are uniformly distributed in the uncertainty set. This is rather a strong assumption, so we performed another experiment using the assumption that each data point obeys Gaussian distribution, i.e. $x_i \sim N(\overline{x}_i, \rho\Sigma_i)$, where $\Sigma_i = \mathrm{diag}(d_1^i, \ldots, d_j^i)$. We reported the result of this experiment in Table 5.4. The results relating to feature selection are similar to the uniform distribution case, hence we do not report here.

We observe from computational results that

Sparsity. For the ellipsoidal model, when the noise level is increasing, the percentage of selected features of robust classifier has tendency to increase, especially on sparse datasets (say the dataset in which the number of features is much larger than the number of samples) like Leukemia and Lung-Cancer. Meanwhile, robust classifiers of the box model maintain the sparsity when ρ is increasing (see figure 5.1 for an intuitive interpretation).In these experiments, the computation for the box model stops when the current solution becomes useless (the weight vector $w = 0$). We see that the box model is more sensitive to the noise

level than the ellipsoidal model. Its solution becomes useless more quickly than the solution of the ellipsoidal model when the noise is increasing. This is because the box model is more conservative than the ellipsoidal model.

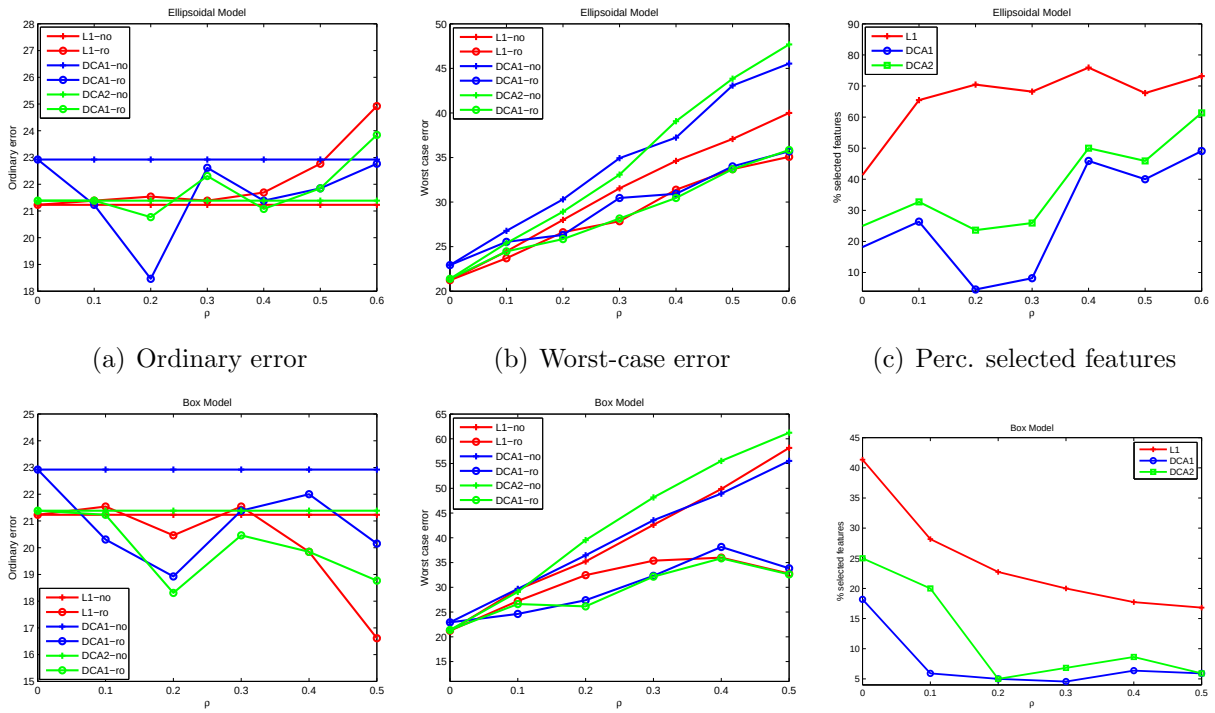
In all cases, the classifiers obtained by ℓ_0(DCA1) and ℓ_0(DCA2) approaches are sparser than those obtained by the regularizer ℓ_1 approach (in Table 5.3, the gains vary from 0.06% to 61%). For the ellipsoidal model, the classifiers obtained by ℓ_0(DCA1) are sparsest. In another hand, in almost cases, the standard deviation values are quite small (from 0 to 8.4%). So the feature selection process is stable.

Classification error. For the ordinary error, the robust classifier is better than the nominal classifier, especially for sparse datasets such as Leukemia, Lung Cancer (the gain is up to 6.5%). For the nominal solutions, except for Leukemia dataset, ℓ_1 and ℓ_0(DCA2) approaches are very comparable and better than ℓ_0(DCA1) approach. In all cases, differential accuracy between approaches is less than 2%. Meanwhile, ℓ_1 approach selects much more features than ℓ_0(DCA1,DCA2) approaches – the ratio of gain is 2 times on small datasets (Parkinsons, Ionosphere and WDBC) and more than 7 times on sparse datasets (Leukemia, Lung Cancer). Besides, these approaches are comparable for the robust solutions, except for Parkinsons and WDBC datasets, ℓ_0(DCA1,DCA2) approaches are better than ℓ_1 approach on ellipsoidal model.

For the worst-case error, the robust classifiers exhibit an advantage over the nominal classifiers. For nominal solutions, ℓ_1 approach is pretty better than ℓ_0(DCA1,DCA2) approaches, while they are very comparable for robust solutions.

For the expected error, with both assumptions on distribution of uncertainty, robust solutions perform better than nominal solutions, except ℓ_1 approach on WDBC dataset. We see that these results are consistent with the ordinary error. In fact, the centers of uncertainty sets (used for evaluating the ordinary error) can be regarded as the expected values of corresponding uncertainty sets. Comparing between ellipsoidal and box models, the box model gives better results than the ellipsoidal model on Parkinsons, Ionosphere and Lung Cancer datasets, while the ellipsoidal model is better than the box model on WDBC and Leukemia datasets. Comparing three approaches ℓ_1 and ℓ_0(DCA1,DCA2), for nominal solutions ℓ_1 approach has less expected error than ℓ_0(DCA1) approach. The difference of accuracy is from 1.7% (when ℓ_1 selects more features than ℓ_0 2.3 times) to 6.8% (when ℓ_1 selects more features than ℓ_0 9 times).

Training time. The training time is given in Table 5.5. We observe that the training time in the box model is less than that in the ellipsoidal model. This is reasonable because robust formulations of the ellipsoidal model requires solving SOCP programs, while nominal formulations and robust formulations of the box model need only solving linear programs. The training time of ℓ_1 approach is shorter than those of ℓ_0(DCA1, DCA2) approaches. This is not surprising because the former is a single convex problem, while the latter requires the resolution of some convex programs. We also see that ℓ_0(DCA2) takes less time than



(a) Ordinary error (b) Worst-case error (c) Perc. selected features

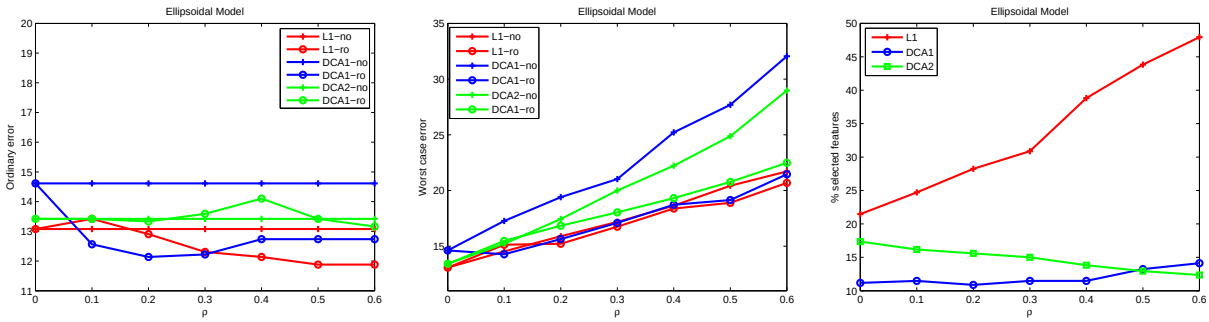
(d) Ordinary error (e) Worst-case error (f) Perc. selected features

Figure 5.2: Parkinsons dataset

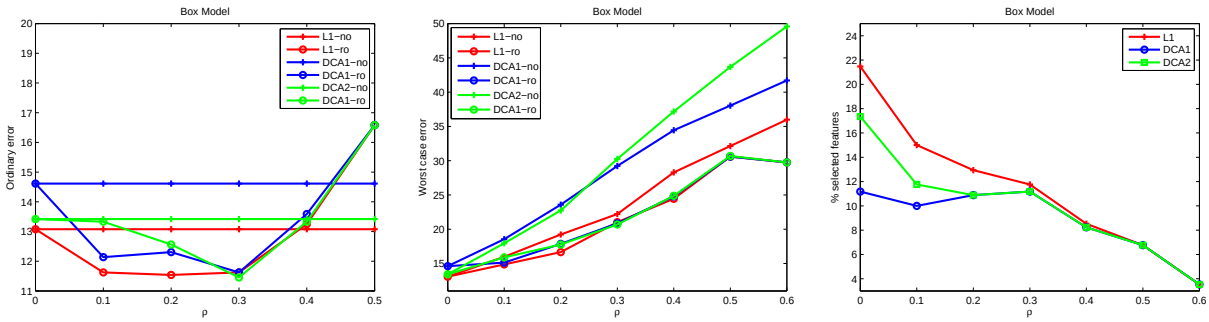
ℓ_0(DCA1). This can be explained by the fact that (5.16) is always a polyhedral DC program which has a finite convergence.

5.5 Conclusion

We have developed DC programming approaches for the feature selection SVM problem under data uncertainty. Robust optimization has been investigated from both a theoretical and an algorithmic point of view. We have proposed robust formulations that handle input uncertainty in ellipsoidal and box sets and provided geometric / probabilistic interpretations of these models. The zero-norm has been used to deal with feature selection, and efficient DCA based algorithms have been developed to solve the resulting optimization problems. Our approaches are motivated, on one hand, by the natural concept of sparsity of the zero-norm, and on another hand, by the efficiency of DCA for various large scale nonconvex problems. Computational experiments on synthetic and real datasets showed that the proposed robust formulations are more resilient than the nominal formulation: the robust solutions are able to immunize against the effect of uncertainty. They also proved that using the zero-norm is a good way to feature selection and DCA is an efficient approach to design nominal as well as robust algorithms for feature selection in SVMs. Moreover, ℓ_0(DCAs) approaches

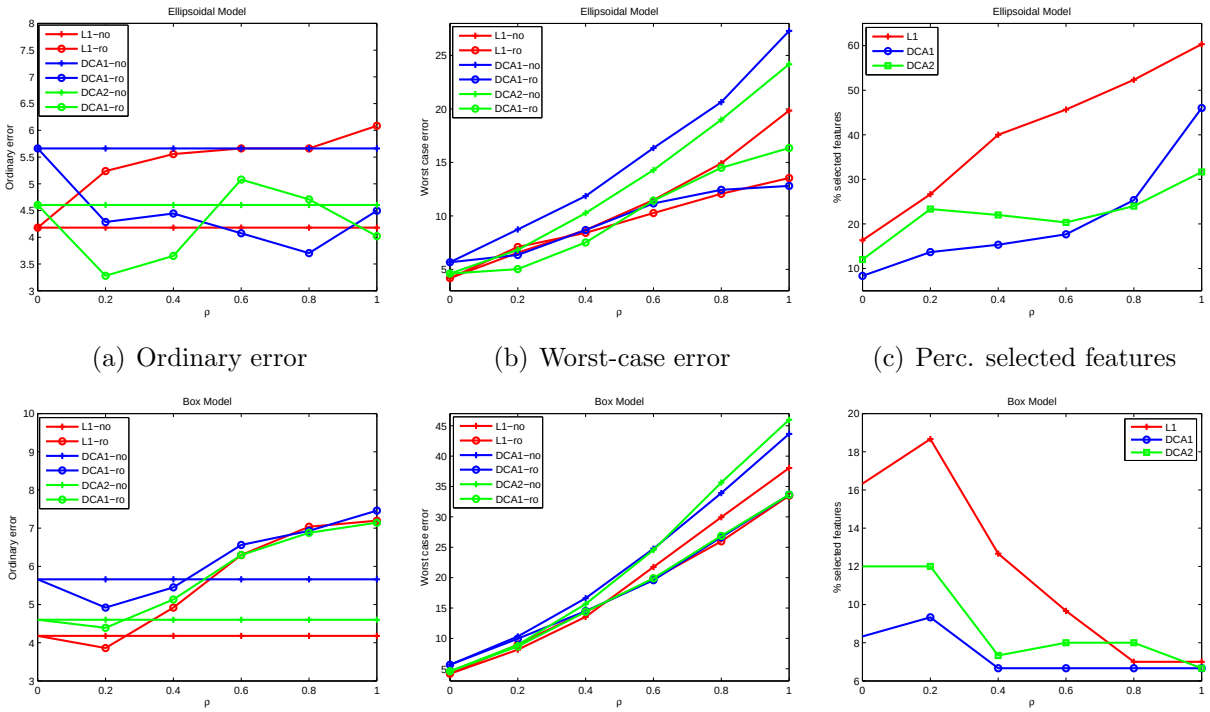


(a) Ordinary error (b) Worst-case error (c) Perc. selected features



(d) Ordinary error (e) Worst-case error (f) Perc. selected features

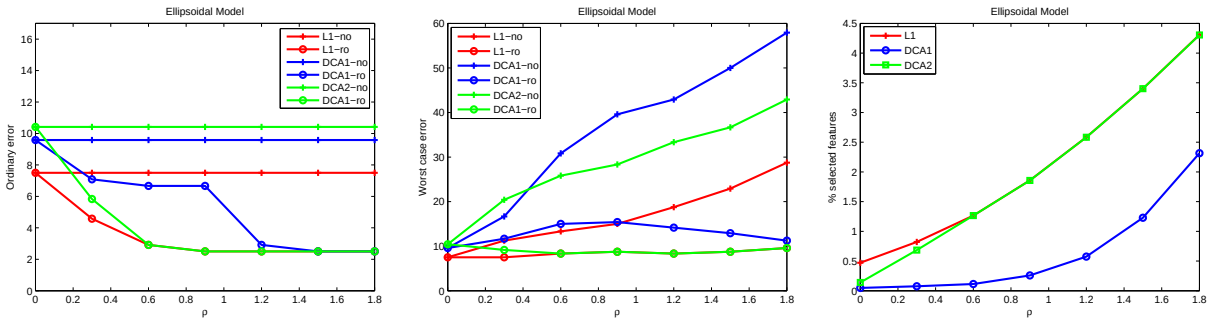
Figure 5.3: Ionosphere dataset



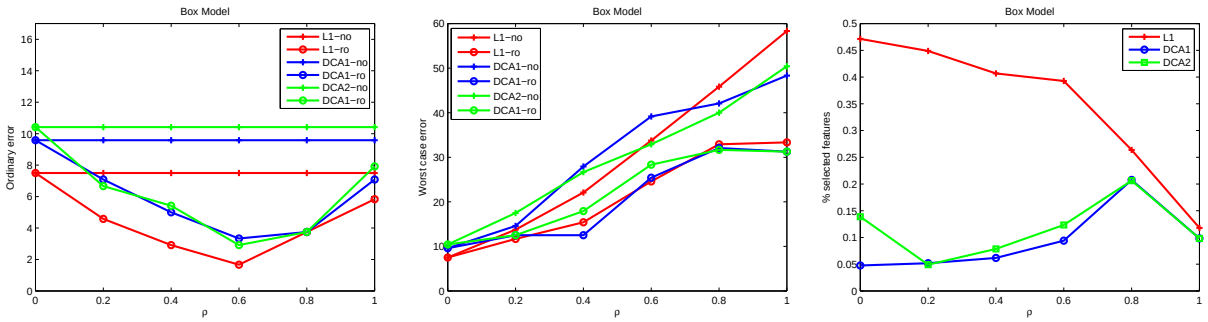
(a) Ordinary error (b) Worst-case error (c) Perc. selected features

(d) Ordinary error (e) Worst-case error (f) Perc. selected features

Figure 5.4: WDBC dataset

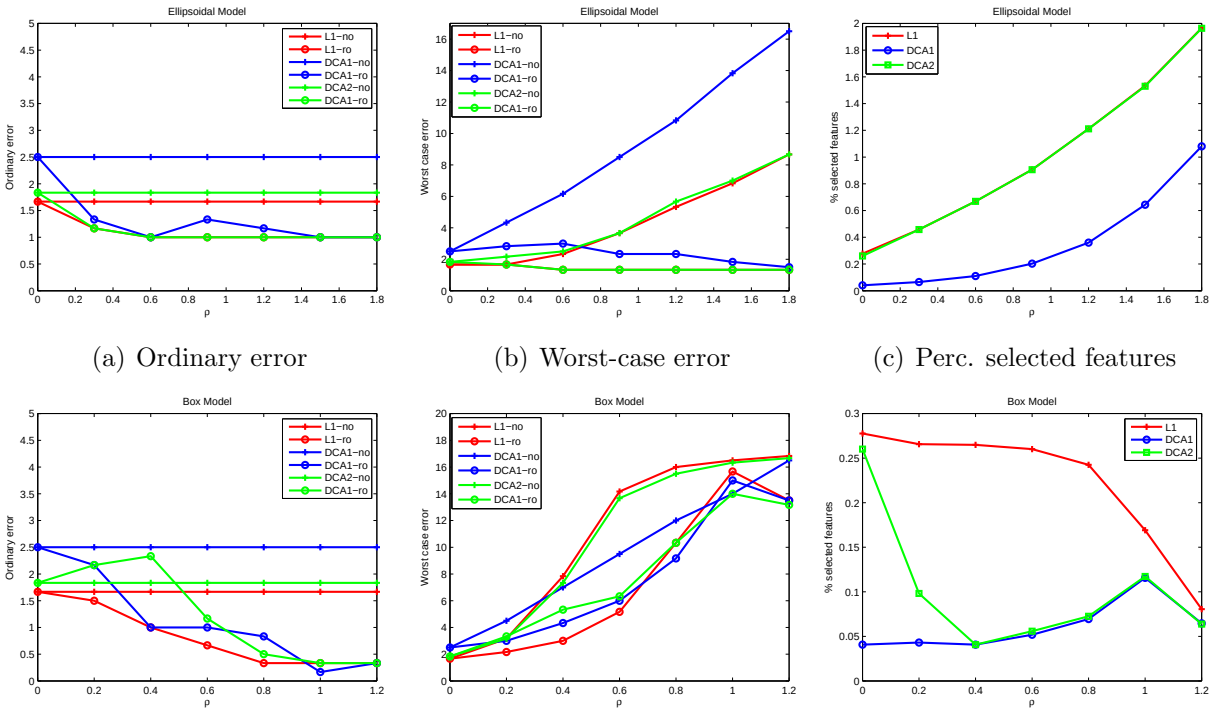


(a) Ordinary error (b) Worst-case error (c) Perc. selected features



(d) Ordinary error (e) Worst-case error (f) Perc. selected features

Figure 5.5: Leukemia dataset



(a) Ordinary error (b) Worst-case error (c) Perc. selected features

(d) Ordinary error (e) Worst-case error (f) Perc. selected features

Figure 5.6: Lung Cancer dataset

Table 5.3: Comparative results of ℓ_1, ℓ_0(DCA1), ℓ_0(DCA2) approaches in term of the percentage of expected errors (upper row) and percentage of selected features (lower row) of nominal (No.) and of robust classifiers in ellipsoidal model (Ellip.) and box model (Box). Bold font indicates the best result in each row. Uniform distribution of uncertainty is assumed.

Dataset		ℓ_1	ℓ_0(DCA1)	ℓ_0(DCA2)
Parkinsons	No.	**22.2**	23.9	23.4
		41.3 ± 5.5	**18.2 ± 3.5**	25.0 ± 6.8
	Box	**18.0**	21.1	20.6
		16.8 ± 3.5	**5.0 ± 1.3**	**5.0 ± 1.3**
	Ellip.	22.0	**20.7**	21.6
		65.5 ± 8.4	**4.5 ± 0**	35.0 ± 6.3
Ionosphere	No.	**13.1**	15.9	14.7
		21.5 ± 1.9	**11.2 ± 2.9**	17.3 ± 3.3
	Box	**12.2**	12.5	12.5
		12.9 ± 2.7	**11.2 ± 1.7**	**11.2 ± 1.7**
	Ellip.	**12.7**	**12.7**	13.8
		43.8 ± 3.3	13.2 ± 1.3	**12.9 ± 1.4**
WDBC	No.	**5.8**	8.5	7.3
		16.3 ± 3.1	**8.3 ± 1.7**	12.0 ± 2.2
	Box	**6.2**	7.5	6.7
		18.7 ± 3.7	**9.3 ± 1.3**	12.0 ± 1.6
	Ellip.	6.2	5.3	**5.2**
		40.0 ± 2.1	25.3 ± 1.6	**22.0 ± 2.2**
Leukemia	No.	**8.1**	14.9	13.2
		0.47 ± 0.03	**0.05 ± 0.02**	0.14 ± 0.17
	Box	**3.6**	4.0	4.4
		0.39 ± 0.03	**0.21 ± 0.04**	**0.21 ± 0.04**
	Ellip.	**3.0**	3.7	**3.0**
		3.4 ± 0.27	**1.23 ± 0.27**	3.4 ± 0.27
Lung Cancer	No.	**1.8**	3.4	**1.8**
		0.27 ± 0.02	**0.04 ± 0.01**	0.26 ± 0.05
	Box	0.5	0.5	0.5
		0.17 ± 0.02	**0.11 ± 0.02**	0.12 ± 0.02
	Ellip.	0.9	0.9	0.9
		1.96 ± 0.17	**0.64 ± 0.11**	1.96 ± 0.14

Table 5.4: Comparative results of ℓ_1, ℓ_0(DCA1), ℓ_0(DCA2) approaches in term of the percentage of expected errors of nominal (No.) and of robust classifiers in ellipsoidal model (Ellip.) and box model (Box). Bold font indicates the best result in each row. Gaussian distribution of uncertainty is assumed.

Dataset		ℓ_1	ℓ_0(DCA1)	ℓ_0(DCA2)
Parkinsons	No.	**21.6**	23.0	22.3
	Box	**17.3**	20.8	20.1
	Ellip.	21.7	**20.2**	21.2
Ionosphere	No.	**13.1**	15.6	14.4
	Box	**11.9**	12.3	12.3
	Ellip.	**12.5**	**12.5**	13.7
WDBC	No.	**5.3**	7.8	6.6
	Box	**5.6**	6.8	6.0
	Ellip.	6.0	**4.8**	**4.8**
Leukemia	No.	**8.0**	13.5	12.6
	Box	**3.4**	3.8	4.2
	Ellip.	**2.9**	3.5	**2.9**
Lung Cancer	No.	**1.7**	3.3	**1.7**
	Box	**0.4**	0.5	0.5
	Ellip.	**0.9**	**0.9**	**0.9**

Table 5.5: The running time in second. For Robust solution, the running time is the average on all considered values of parameters of noise.

Dataset	ℓ_1			ℓ_0(DCA1)			ℓ_0(DCA2)		
	No.	Box	Ellip.	No.	Box	Ellip.	No.	Box	Ellip.
Parkinsons	0.02	0.03	0.09	0.12	0.12	0.76	0.07	0.09	0.34
WDBC	0.06	0.10	0.48	0.42	0.38	4.22	0.30	0.30	1.88
Ionosphere	0.05	0.07	0.27	0.36	0.24	1.71	0.16	0.20	0.82
Leukemia	3.18	5.83	14.37	23.27	22.92	73.35	13.76	17.37	29.42
Lung Cancer	21.22	44.27	54.15	213.56	217.05	322.04	53.57	98.11	147.9

are more powerful when they are applied on robust formulation. Between the two models of uncertainty, intuitively the box model exhibits the advantage over the ellipsoidal model in terms of sparsity. This is confirmed by comparative numerical results.

Chapter 6

Robust optimization for clustering uncertain data[1]

Abstract: *In this chapter, we investigate the robust optimization for the minimum sum-of-squares clustering (MSSC) problem on uncertain data. Each data point is assumed to belong to an uncertainty set. Following the robust optimization paradigm, we propose robust clustering methods that can handle data with box model and spherical model of uncertainty. DCA-based algorithms are developed to solve the robust clustering problems. Preliminary numerical results on synthetic and real datasets show that the proposed robust optimization approaches exhibit good performance in dealing with uncertain data and work well in real situations.*

6.1 Introduction

Clustering is a powerful exploratory data mining technique and has many applications in various fields. It aims to divide a given dataset into groups (clusters) of similar objects according to a certain measure of similarity. There exist different approaches to clustering including density based clustering methods, hierarchical clustering methods and partitional clustering methods (see Hansen and Jaumard (1997); Jain et al. (1999); Berkhin (2006) and references therein). In this chapter, we focus on the partitional clustering problem that is described as follows. Given a dataset $\mathcal{X} = \{x_1, \ldots, x_m\}$ of m points in $\mathbb{R}^n$, a "distance" measure d defined on $\mathbb{R}^n \times \mathbb{R}^n$, and an integer $c\,(2 \leq c \leq m)$, the problem is to determine c centers (or centroids) v_i $(i = 1, \ldots, c)$ in $\mathbb{R}^n$ such that the sum of distances from data points in $\mathcal{X}$ to the center of their cluster (the cluster compactness criterion) is minimized. This

1. The material of this chapter is based on the following work:
[1]. Xuan Thanh Vo, Hoai An Le Thi. Robust Optimization for Clustering Uncertain Data. *Submitted to* Pattern Recognition.
[2]. Xuan Thanh Vo, Hoai An Le Thi, Tao Pham Dinh. Robust Optimization for Clustering. *Submitted to* ACIIDS 2016: 8th Asian Conference on Intelligent Information and Database Systems.

problem can be mathematically formulated as

$$\min_{U,V} \left\{ \frac{1}{2} \sum_{k=1}^{m} \sum_{i=1}^{c} u_{i,k} d(x_k, v_i) : U \in \mathcal{M},\, V = [v_1, \ldots, v_c] \in \mathbb{R}^{n \times c} \right\} \tag{6.1}$$

where $U = (u_{i,k}) \in \mathbb{R}^{c \times m}$ is the matrix of memberships, i.e. $u_{i,k} = 1$ if the object x_k is assigned to cluster i^{th} and 0 otherwise, and

$$\mathcal{M} = \{U = (u_{i,k}) \in \mathbb{R}^{c \times m} : \sum_{i=1}^{c} u_{i,k} = 1,\, u_{i,k} \in \{0, 1\},\ \forall i, k\}.$$

The constraint $U \in \mathcal{M}$ ensures that each point x_k is assigned to one and only one cluster, so the clusters are disjoint. It means that we are faced with *hard clustering*. This is in contrast to other clustering problems called *fuzzy clustering*, where the clusters are allowed to overlap and data points have degrees of membership in each cluster. Two most popular and best studied clustering measures in the literature are the squared ℓ_2-norm (squared Euclidean distance) and the ℓ_1-norm (Manhattan distance). If d is the squared Euclidean distance, problem (6.1) becomes the following well-known *minimum sum-of squares clustering* (MSSC) that was first formulated by (Vinod, 1969)

$$\min_{U,V} \left\{ \frac{1}{2} \sum_{k=1}^{m} \sum_{i=1}^{c} u_{i,k} \|x_k - v_i\|^2 : U \in \mathcal{M},\, V = [v_1, \ldots, v_c] \in \mathbb{R}^{n \times c} \right\}. \tag{6.2}$$

Problem (6.2) is a mixed-integer program with the nonconvex objective function. It has been proved that (6.2) is NP-hard with possibly many local minima (Aloise et al., 2009). There are different algorithms of mathematical programming have been developed to solve this problem (see Hansen and Jaumard (1997) for some reviews). Several global optimization methods have been explored to find the exact solution of the MSSC problem (Peng and Xiay, 2005; Sherali and Desai, 2005; Brusco, 2006). However, these methods are only confined to small datasets with only hundreds of records and the number of clusters is not large. Different heuristics can be used for solving large clustering problems and k-means (MacQueen, 1967) is the most popular one among such algorithms. At each iteration, k-means assigns each data point to the cluster whose center is closet to that data point according to the Euclidean distance, then recalculate the cluster centers as the barycenters (means) of the new clusters. Recently, (Le Thi and Pham Dinh, 2009; Le Thi et al., 2014e) proposed a scalable DCA based algorithm for solving problem (6.2). The mixed integer formulation of MSSC is carefully studied and is reformulated as a continuous optimization problem by using the exact penalty technique in DC programming. DCA is then investigated to the resulting problem.

Another equivalent formulation of the MSSC is based on bilevel programming

$$\min \left\{ \frac{1}{2} \sum_{k=1}^{m} \min_{i=1,\ldots,c} \|x_k - v_i\|^2 : V = [v_1, \ldots, v_c] \in \mathbb{R}^{n \times c} \right\}. \tag{6.3}$$

This is a nonconvex nonsmooth optimization problem containing only continuous variables. DC programming and DCA have been developed in (Le Thi et al., 2006) for solving this problem. Numerical experiments on real-world databases show the efficiency and the superiority of the DCA based algorithm with respect to the standard k-means algorithm. Other efficient approaches that have been proposed to address this bilevel problem include the hyperbolic smoothing clustering method (Xavier and Xavier, 2011), the nonsmooth optimization approach (Bagirov and Yearwood, 2006). (Le Thi et al., 2014e) considered a kernel version of the bilevel formulation (6.3), named GKMSSC, that is formulated as a DC program for which a simple and efficient DCA scheme is developed.

If d is the ℓ_1-norm distance metric, problem (6.1) is referred as the *k-median clustering* (Bradley et al., 1997). Similar to the MSSC, we also have two equivalent formulations of the k-median clustering: one is based on the mixed-integer programming and one is based on bilevel programming. The k-median algorithm (Bradley et al., 1997) has been introduced for solving this problem. The k-median algorithm is similar to the k-means algorithm in the description except that k-median algorithm uses the ℓ_1-norm for assignment step and cluster canters are recomputed by using the median in each single dimension. Both these algorithms are recognized as the most inexpensive and efficient method, especially for the large-scale setting.

In problem (6.1), the input data x_k's are assumed to be known exactly or certainly. However, this is unrealistic in real-world applications where data are commonly uncertain due to various reasons such as imprecise measurement, out dated sources, sampling discrepancy, etc. For example, due to speed, storage or confidentiality issues, raw data are not available in its original form, but one has access to compressed data such as in location-based devices. Another example is clustering of high–dimensional data, where one need to reduce dimensions of the data via a pre-processing step before applies a clustering algorithm. In these cases, data are no longer precise. Applying traditional method to uncertain data could seriously affect the quality of the clustering results. This motivates us to develop a new clustering method that is capable to handle real situations. In the following, we review some partitional approaches for clustering uncertain data.

Related works on clustering uncertain data

In (Chaudhuria and Bhowmik, 1998), the authors considered the situation that the true position of each datum x is unknown but anywhere in a ball $S(x)$ of radius r centered at the observed datum $P(x)$. This means that the noise is assumed to be zero mean, uniformly distributed and additive. The modified k-means algorithm has been proposed to deal with this situation. In this approach, the cluster centers are deterministic, but an object can be assigned to more than one cluster (in the extreme case) or assigned weights to different clusters (in the expected case), thus possibly resulting in ambiguously classified data.

Kumar and Patel (2007) considered a different setup. In their work, beside the observed measurement, each data point is associated with a covariance matrix representing the error.

The proposed *kError* algorithm is a generalization of k-mean algorithm where the Euclidean distance from a data point to its cluster center is replaced by the Mahalanobis distance defined via the covariance matrix of that data point. When all error matrices are equal and spherical, *kError* reduces to k-mean.

In another direction, each uncertain object is associated with a probability density function. Several methods has been proposed as an adaptation of the k-means algorithm for the context of uncertain data. In (Chau et al., 2006), each uncertain object is associated with a probability density function (pdf), then the proposed UK-means algorithm basically follows k-means algorithm except that it uses *expected distance* (ED) when determining which cluster an object should be assigned to. The cluster centers are deterministic and computed as the mean of the expected values of the objects within the corresponding clusters. The major computational cost of the UK-means algorithm is the evaluation of the EDs, which involves numerical integration using a large amount of sample points for the pdf's. Some improvements of UK-means in computing EDs were proposed in (Ngai et al., 2006) and (Lee, Kao and Cheng, 2007). Gullo and Tagarelli (2012) proposed another uncertain centroid based approach where the center of a cluster is an uncertain object (named U-centroid) defined as average of uncertain objects of that cluster. The proposed algorithm UCPC aims to minimize the total ED between uncertain objects and the corresponding U-centroids.

Hamdan and Govaert (2005) have addressed the problem of fitting mixture densities to uncertain data for clustering using a modified Expectation-Maximization (EM) algorithm. They supposed that data observed were the sampling results from a distribution mixture and aimed to find the maximum likelihood estimation of the mixture model parameters.

The major drawback of methods based on statistical information such as probability density function is the computational issue since the computation of the expected distances is very expensive despite of improvement techniques. Besides, the availability of probability density function for each data point is also a problem.

For the purpose of clustering uncertain data, we will investigate the Robust Optimization approach.

Our approach: robust optimization using DC programming and DCA

We assume that the input data are inaccurate by disturbances. It makes sense that noise caused by i.i.d. statistical distribution is usually of no concern but the completely arbitrary (potentially adversarial) disturbances are of real concern. In this setup, the disturbances are unknown and have no information on probability distribution function but we assume that these disturbances are bounded and vary in known subsets in the input space. Following the robust optimization approach, we minimize the worst possible objective value of (6.1) under such disturbances. We consider two models of uncertainty - box and spherical - for the MSSC clustering problem.

Our method for solving the robust problems is based on DC programming and DCA. This

choice is motivated by the fact that DCA is a powerful method for many (smooth or non-smooth) large-scale nonconvex programs in various domains of applied sciences, especially in Machine Learning (Le Thi et al., 2006, 2007a,c, 2008, 2009b; Le Thi et al., 2013a; Le Thi and Nguyen, 2013; Le Thi et al., 2013c, 2014a,e,f,g; Nguyen et al., 2015) for which they provide quite often a global solution and proved to be more robust and efficient than the standard methods. In particular, DC programming and DCA have been successfully applied to the clustering problem. As mentioned above, they have been extensively developed for the MSSC (Le Thi et al., 2006; Le Thi and Pham Dinh, 2009; Le Thi et al., 2014e). They are also investigated to solve other variants of the clustering problem such as the Fuzzy c-Mean (FCM) (Le Thi et al., 2007c), the hierarchical clustering (Le Thi et al., 2007a), the MSSC using weighted dissimilarity measures (Le and Ta, 2014), the block clustering (Le et al., 2013b). The DCA based algorithms have been shown to be superior to the related existing methods.

Robust optimization has been widely applied in Machine Learning, especially in classification, to deal with uncertainty (see Shivaswamy et al. (2006),Bhattacharyya et al. (2004a),Le Thi et al. (2014f) and the references therein). Caramanis et al. (2011) even proved that some regularization techniques in support vector machine and linear regression can be regarded as an application of robust optimization approach. However, to our knowledge, this is the first time the use of robust optimization in clustering is addressed. The purpose of this chapter is to develop robust clustering algorithms that work well in real applications. Our main contributions are two folds. Firstly, deriving from the MSSC problem and assuming point-wise independent disturbances in two models of uncertainty - box and spherical uncertainties - we propose two robust clustering formulations respectively. Secondly, we develop algorithms based on DC programming and DCA for solving the robust clustering problems.

The rest of this chapter is organized as follows. In the next section, we construct the robust formulations of clustering problem under the box and spherical uncertainty models. In section 6.3, we show how to apply DCA to solve our robust clustering problems. The numerical experiments are presented in section 6.4.

6.2 Robust formulation for clustering

We consider that the input data points x_k $(k = 1, \ldots, m)$ are subject to perturbations on each of their coordinates. These perturbations could be represented by additional vectors Δx_k $(k = 1, \ldots, m)$. The observed point x_k is called the nominal value, so the true value $\bar{x}_k$ of the data point k^{th} is expressed as $\bar{x}_k = x_k + \Delta x_k$. We consider here that the value of the vector Δx_k is *unknown*. But we assume that we are able to *bound* each Δx_k for all $k = 1 \ldots, m$, i.e. we know the *uncertainty set* $\mathcal{U}_k$ - the set of *admissible* disturbances of x_k - where Δx_k is allowed to belong to.

Considering problem (6.2) and following the robust optimization paradigm, we aim to find the optimal solution in the worst case sense. That is, we need to solve the following min–max problem

$$\min_{U,V}\left\{\frac{1}{2}\sum_{k=1}^{m}\max_{\Delta x_k\in\mathcal{U}_k}\left(\sum_{i=1}^{c}u_{i,k}\|(x_k+\Delta x_k)-v_i\|^2\right):U\in\mathcal{M},\,V\in\mathbb{R}^{n\times c}\right\}. \tag{6.4}$$

6.2.1 Box uncertainty model

We assume that the uncertainty sets are boxes in $\mathbb{R}^n$, i.e. $\mathcal{U}_k=\prod_{j=1}^{n}[-(w_k)_j,(w_k)_j]$ with $w_k\geq 0$ is known ($\forall k=1,\ldots,m$). Using the condition $\sum_{i=1}^{c}u_{i,k}=1$, we have

$$\begin{aligned}
& \arg\max_{\Delta x_k\in\mathcal{U}_k}\sum_{i=1}^{c}u_{i,k}\|(x_k+\Delta x_k)-v_i\|^2\\
= \quad & \arg\max_{|\Delta x_k|\leq w_k}\left\{\|\Delta x_k\|^2+2\langle\Delta x_k,\sum_{i=1}^{c}u_{i,k}(x_k-v_i)\rangle\right\}\\
= \quad & \operatorname{sgn}\left(\sum_{i=1}^{c}u_{i,k}(x_k-v_i)\right)\circ w_k.
\end{aligned}$$

Therefore,

$$\begin{aligned}
& \max_{\Delta x_k\in\mathcal{U}_k}\sum_{i=1}^{c}u_{i,k}\|(x_k+\Delta x_k)-v_i\|^2\\
= \quad & \sum_{i=1}^{c}u_{i,k}\|x_k-v_i\|^2+2\langle w_k,\left|\sum_{i=1}^{c}u_{i,k}(x_k-v_i)\right|\rangle+\|w_k\|^2\\
= \quad & \sum_{i=1}^{c}u_{i,k}\left(\|x_k-v_i\|^2+2\langle w_k,|x_k-v_i|\rangle+\|w_k\|^2\right).
\end{aligned}$$

The last equality is deduced from the conditions $\sum_{i=1}^{c}u_{i,k}=1$ and $u_{i,k}\in\{0,1\}$.

Then the robust problem (6.4) becomes

$$\min_{U,V}\left\{\psi^B(U,V):=\frac{1}{2}\sum_{k=1}^{m}\sum_{i=1}^{c}u_{i,k}d^B(x_k,v_i):U\in\mathcal{M},\,V\in\mathbb{R}^{n\times c}\right\}, \tag{6.5}$$

where $d^B(x_k,v_i)=\|x_k-v_i\|^2+2\langle w_k,|x_k-v_i|\rangle+\|w_k\|^2$ ($\forall k=1,\ldots,m;i=1,\ldots,c$).

This problem can be reformulated as a bilevel program as follows

$$\min\left\{F^B(V):=\frac{1}{2}\sum_{k=1}^{m}\min_{i=1,\ldots,c}d^B(x_k,v_i):v_i\in\mathbb{R}^n,\ i=1,\ldots,c\right\}. \tag{6.6}$$

The following result shows the equivalence between (6.5) and (6.6). Its proof is trivial so is omitted.

Proposition 6.1 *Problems (6.5) and (6.6) are equivalent in the following sense:*

(i) If $(U^*, V^*) \in \mathcal{M} \times \mathbb{R}^{n\times c}$ is a solution to the problem (6.5) then V^* is a solution to the problem (6.6).

(ii) Conversely, if $V^* \in \mathbb{R}^{n\times c}$ is a solution to the problem (6.6) then (U^*, V^*) is a solution to the problem (6.5), where $U^* \in \mathcal{M}$ satisfies, for any $k = 1, \ldots, m$,

$$u^*_{i_k,k} = 1 \text{ and } u^*_{i,k} = 0 \ \forall i \neq i_k, \text{ with a } i_k \in \arg\min_{j=1,\ldots,c} d^B(x_k, v^*_j).$$

6.2.2 Spherical uncertainty model

We assume that the uncertainty sets are balls in $\mathbb{R}^n$, i.e. there is a known scalar $\sigma_k \geq 0$ such that $\mathcal{U}_k = \{x \in \mathbb{R}^n : \|x\| \leq \sigma_k\}$ $(\forall k = 1, \ldots, m)$. Similarly to the box uncertainty model, we have

$$\begin{aligned}
& \arg\max_{\Delta x_k \in \mathcal{U}_k} \sum_{i=1}^{c} u_{i,k} \|(x_k + \Delta x_k) - v_i\|^2 \\
= \quad & \arg\max_{\|\Delta x_k\| \leq \sigma_k} \left\{ \|\Delta x_k\|^2 + 2\langle \Delta x_k, \sum_{i=1}^{c} u_{i,k}(x_k - v_i)\rangle \right\} \\
= \quad & \sigma_k \frac{\sum_{i=1}^{c} u_{i,k}(x_k - v_i)}{\|\sum_{i=1}^{c} u_{i,k}(x_k - v_i)\|},
\end{aligned}$$

and

$$\begin{aligned}
& \max_{\Delta x_k \in \mathcal{U}_k} \sum_{i=1}^{c} u_{i,k} \|(x_k + \Delta x_k) - v_i\|^2 \\
= \quad & \sum_{i=1}^{c} u_{i,k} \|x_k - v_i\|^2 + 2\sigma_k \|\sum_{i=1}^{c} u_{i,k}(x_k - v_i)\| + \sigma_k^2 \\
= \quad & \sum_{i=1}^{c} u_{i,k} \left(\|x_k - v_i\|^2 + 2\sigma_k \|x_k - v_i\| + \sigma_k^2 \right).
\end{aligned}$$

Then the robust problem (6.4) becomes

$$\min_{U,V} \left\{ \psi^E(U, V) := \frac{1}{2} \sum_{k=1}^{m} \sum_{i=1}^{c} u_{i,k} d^E(x_k, v_i) : U \in \mathcal{M},\ V \in \mathbb{R}^{n\times c} \right\}, \tag{6.7}$$

where $d^E(x_k, v_i) = \|x_k - v_i\|^2 + 2\sigma_k\|x_k - v_i\| + \sigma_k^2$ $(\forall k = 1, \ldots, m; i = 1, \ldots, c)$.

This problem can be reformulated as the following bilevel program

$$\min \left\{ F^E(V) := \frac{1}{2}\sum_{k=1}^{m} \min_{i=1,\ldots,c} d^E(x_k, v_i) : v_i \in \mathbb{R}^n,\ i = 1, \ldots, c \right\}. \tag{6.8}$$

The equivalence between (6.7) and (6.8) is formally stated below.

Proposition 6.2 *Problems (6.7) and (6.8) are equivalent in the following sense:*

(i) If $(U^*, V^*) \in \mathcal{M} \times \mathbb{R}^{n\times c}$ is a solution to the problem (6.7) then V^* is a solution to the problem (6.8).

(ii) Conversely, if $V^* \in \mathbb{R}^{n\times c}$ is a solution to the problem (6.8) then (U^*, V^*) is a solution to the problem (6.7), where $U^* \in \mathcal{M}$ satisfies, for any $k = 1, \ldots, m$,

$$u^*_{i_k,k} = 1 \text{ and } u^*_{i,k} = 0\ \forall i \neq i_k, \text{ with a } i_k \in \arg\min_{j=1,\ldots,c} d^E(x_k, v^*_j).$$

In Sect. 6.3, we will develop algorithms based on DC programming and DCA to solve the problems (6.6) and (6.8).

6.2.3 Interpretation of the robust models

We assume that the input data points x_k $(k = 1, \ldots, m)$ are subject to perturbations. With the same notations as above, the robust counterpart of (6.3) takes the form

$$\min_V \left\{ \frac{1}{2}\sum_{k=1}^{m} \max_{\Delta x_k \in \mathcal{U}_k} \left(\min_{i=1,\ldots,c} \|(x_k + \Delta x_k) - v_i\|^2 \right) : V \in \mathbb{R}^{n\times c} \right\}. \tag{6.9}$$

In this formulation, for each $k = 1, \ldots, m$, the quantity

$$\max_{\Delta x_k \in \mathcal{U}_k} \left(\min_{i=1,\ldots,c} \|(x_k + \Delta x_k) - v_i\|^2 \right) \tag{6.10}$$

is the worst-case distance of the kth data point to its cluster center over all disturbances $\Delta x_k \in \mathcal{U}_k$. However, solving (6.9) is difficult since it is complicated to evaluate the maximum in (6.10). We will relax (6.10) by

$$\max_{\Delta x_k \in \mathcal{U}_k} \min_{i=1,\ldots,c} \|(x_k + \Delta x_k) - v_i\|^2 \leq \min_{i=1,\ldots,c} \max_{\Delta x_k \in \mathcal{U}_k} \|(x_k + \Delta x_k) - v_i\|^2$$

with the assumption that evaluation of $\max_{\Delta x_k \in \mathcal{U}_k} \|(x_k + \Delta x_k) - v_i\|^2$ is tractable. Then a relaxation of (6.9) will be

$$\min_V \left\{ \frac{1}{2} \sum_{k=1}^{m} \min_{i=1,\ldots,c} \max_{\Delta x_k \in \mathcal{U}_k} \|(x_k + \Delta x_k) - v_i\|^2 : V \in \mathbb{R}^{n \times c} \right\}. \tag{6.11}$$

If the uncertainty set $\mathcal{U}_k$ $(k = 1, \ldots, m)$ are boxes (resp. balls), problem (6.11) becomes problem (6.6) (resp. (6.8)). Note that

$$\max_{\Delta x_k \in \mathcal{U}_k} \|(x_k + \Delta x_k) - v_i\|^2$$

is the worst–case squared Euclidean distance from v_i to the kth data point over all disturbances $\Delta x_k \in \mathcal{U}_k$. Thus, with formulation (6.11), the measure of similarity between a data point and a prototype is the worst–case squared Euclidean distance computed over all realization of the data point in its uncertainty set.

In the box uncertainty model, suppose that for any $k = 1, \ldots, m$, we have $w_k = (\lambda_k, \ldots, \lambda_k) \in \mathbb{R}^n$ with $\lambda_k > 0$. Considering a cluster C with the center v, its cluster-compactness can be expressed by

$$\sum_{x_k \in C} d^B(x_k, v) = \sum_{x_k \in C} \|x_k - v\|^2 + 2 \sum_{x_k \in C} \lambda_k \|x_k - v\|_1 + \sum_{x_k \in C} \|w_k\|^2. \tag{6.12}$$

Similarly, in the spherical uncertainty model, the cluster–compactness criterion of a cluster C with the center v has the form

$$\sum_{x_k \in C} d^E(x_k, v) = \sum_{x_k \in C} \|x_k - v\|^2 + 2 \sum_{x_k \in C} \sigma_k \|x_k - v\| + \sum_{x_k \in C} \sigma_k^2. \tag{6.13}$$

In both cases, the first term on the right hand side is the cluster–compactness criterion used in the MSSC problem, while the second term is the weighted sum of Manhattan/Euclidean distances that appears in the facility location problem including the Weber problem (Kuhn and Kuenne, 1962). We can interpret the second term as a regularization added to the MSSC to make it more robust. Indeed, if all λ_k are equal, the second term in (6.12) is also the cluster–compactness criterion used in the k-median clustering. Note that outliers have less influence on the k-median clustering than the MSSC (Bradley et al., 1997) since in the k-median, the cluster center determined by the entry-wise median of data points in C is less sensitive to outliers than the cluster center in MSSC that is determined by the means of data points. Thus, the presence of the second term might help the proposed robust clustering with the box uncertainty model be more robust to outliers than the original one. The robust clustering with the spherical uncertainty model has a similar property. Indeed, minimizing the cluster-compactness (6.13) implies that the differentiation of the right-hand side of (6.13) should be 0, that is,

$$(v_i - \bar{x}) + \frac{\sigma}{|C|} \sum_{x_k \in C} \frac{v_i - x_k}{\|x_k - v_i\|} = 0 \Leftrightarrow v_i = \Omega^{-1} \left(\bar{x} + \frac{\sigma}{|C|} \sum_{x_k \in C} \frac{x_k}{\|x_k - v_i\|} \right),$$

where $\Omega = 1 + \frac{\sigma}{|C|} \sum_{x_k \in C} \frac{1}{\|x_k - v_i\|}$. Thus, v_i can be expressed as a convex combination of $\bar{x}$ and x_k's. Moreover, data points x_k which are closer to v_i are associated with larger weights. Hence, the effect of outliers is relieved.

6.3 DCA for solving the robust clustering problems with box and spherical uncertainty models

In this section, we will develop DCA-based algorithms for solving problems (6.6) and (6.8). For convenience, we use the common notation d for either d^B or d^E. Then problems (6.6) and (6.8) have the common form (called the *robust clustering problem*)

$$\min \left\{ F(V) := \frac{1}{2} \sum_{k=1}^{m} \min_{i=1,\ldots,c} d(x_k, v_i) : V = [v_1, \ldots, v_c] \in \mathbb{R}^{n \times c} \right\}. \tag{6.14}$$

We will reformulate problem (6.14) as a DC program, then describe DCA for solving it and explain specifications for each particular case.

6.3.1 DC formulation of the robust clustering problem

Thank to the convexity of $d(x_k, v_i)$ w.r.t v_i ($\forall i = 1, \ldots, c; k = 1, \ldots, m$) and the fact that

$$\min_{i=1,\ldots,c} d(x_k, v_i) = \sum_{i=1,\ldots,c} d(x_k, v_i) - \max_{j=1,\ldots,c} \sum_{i=1, i \neq j}^{c} d(x_k, v_i),$$

we prove that the objective function F of (6.14) is a DC function. Specifically, problem (6.14) can be reformulated as a DC program as follows

$$\min \left\{ F(V) = G(V) - H(V) : V \in \mathbb{R}^{n \times c} \right\}, \tag{6.15}$$

where DC components G and H are given by

$$G(V) = \frac{1}{2} \sum_{k=1}^{m} \sum_{i=1}^{c} d(x_k, v_i), \quad H(V) = \sum_{k=1}^{m} H_k(V),$$

with $H_k(V) = \max_{j=1,\ldots,c} \frac{1}{2} \sum_{i=1, i \neq j}^{c} d(x_k, v_i)$ for any $k = 1, \ldots, m$.

DCA applied to (6.15) consists of constructing two sequences $\{V^{(t)}\}$ and $\{Y^{(t)}\}$ such that, for all $t = 0, 1, 2, \ldots$

- $Y^{(t)} = [y_1^{(t)}, \ldots, y_c^{(t)}] \in \partial H(V^{(t)})$ and

- $V^{(t+1)}$ solves the convex problem

$$\min\left\{\frac{1}{2}\sum_{k=1}^{m}\sum_{i=1}^{c} d(x_k, v_i) - \sum_{i=1}^{c}\langle y_i^{(t)}, v_i\rangle : v_i \in \mathbb{R}^n,\ i = 1, \ldots, c\right\}.$$

Or equivalently, for $i = 1, \ldots, c$, v_i^{t+1} solves the convex problem

$$\min\left\{\frac{1}{2}\sum_{k=1}^{m} d(x_k, v_i) - \langle y_i^{(t)}, v_i\rangle : v_i \in \mathbb{R}^n\right\}. \tag{6.16}$$

We will show below the computation of $Y \in \partial H(V)$ and the solution method of problem (6.16).

6.3.2 Calculation of $\partial H(V)$

We have $\partial H(V) = \sum_{k=1}^{m} \partial H_k(V)$. Therefore,

$$Y = [y_1, \ldots, y_c] \in \partial H(V) \Leftrightarrow Y = \sum_{k=1}^{m} Y^{[k]} \tag{6.17}$$

with $Y^{[k]} = [y_1^{[k]}, \ldots, y_c^{[k]}] \in \partial H_k(V) \quad$ for $k = 1, \ldots, m$.

We write, for $k = 1, \ldots, m$, $H_k(V) = \max\limits_{j=1,\ldots,c} H_{k,j}(V)$, where

$$H_{k,j}(V) = \frac{1}{2}\sum_{i=1, i\neq j}^{c} d(x_k, v_i).$$

Denote by $I_k(V) = \arg\max\limits_{j=1,\ldots,c} H_{k,j}(V) = \arg\min\limits_{j=1,\ldots,c} d(x_k, v_j)$. Then we have (Hiriart-Urruty and Lemarechal, 1993)

$$\partial H_k(V) = \mathrm{co}\{\cup_{j\in I_k(V)} \partial H_{k,j}(V)\},$$

where co stands for the convex hull. Thus, a subgradient $Y^{[k]} \in \partial H_k(V)$ is given by

$$Y^{[k]} = Y^{[k,j(k)]} \in \partial H_{k,j(k)}(V), \tag{6.18}$$

where $j(k) \in I_k(V)$. Besides,

$$\begin{aligned} & Y^{[k,j]} = [y_1^{[k,j]}, \ldots, y_c^{[k,j]}] \in \partial H_{k,j}(V) \\ \Leftrightarrow \quad & y_i^{[k,j]} \in \partial_{v_i} H_{k,j}(V), \quad \forall i = 1, \ldots, c \\ \Leftrightarrow \quad & y_j^{[k,j]} = 0; \quad y_i^{[k,j]} \in \frac{1}{2}\partial_{v_i} d(x_k, v_i),\ \forall i \neq j. \end{aligned}$$

Thus, by letting

$$\overline{Y}^{[k]} = [\bar{y}_1^{[k]}, \ldots, \bar{y}_c^{[k]}] \quad \text{such that} \quad \bar{y}_i^{[k]} \in \frac{1}{2}\partial_{v_i} d(x_k, v_i),$$

a subgradient $Y^{[k,j]} \in \partial H_{k,j}(V)$ is computed as

$$Y^{[k,j]} = \overline{Y}^{[k]} - \bar{y}_j^{[k]} e_j^T, \tag{6.19}$$

where $\{e_j : j = 1, \ldots, c\}$ is the canonical basis of $\mathbb{R}^c$.

From (6.17), (6.3.2) and (6.19), a subgradient $Y \in \partial H(V)$ is given by

$$Y = \sum_{k=1}^{m} \left([\bar{y}_1^{[k]}, \ldots, \bar{y}_c^{[k]}] - \bar{y}_{j(k)}^{[k]} e_{j(k)}^T \right), \tag{6.20}$$

where $j(k) \in I_k(V)$ and $\bar{y}_i^{[k]} \in \frac{1}{2}\partial_{v_i} d(x_k, v_i)$ for all $i = 1, \ldots, c; k = 1, \ldots, m$.

In the case of the box uncertainty ($d = d^B$), we have

$$\frac{1}{2}\partial_{v_i} d(x_k, v_i) = \left\{ v_i - x_k + w_k \circ \zeta : \zeta = (\zeta_1, \ldots, \zeta_n), \zeta_j \in \partial_{v_{ij}} |v_{ij} - x_{kj}| \ \forall j \right\}.$$

Note that $\operatorname{sgn}(v_{ij} - x_{kj}) \in \partial_{v_{ij}} |v_{ij} - x_{kj}|$, so a $\bar{y}_i^{[k]} \in \frac{1}{2}\partial_{v_i} d(x_k, v_i)$ is given by

$$\bar{y}_i^{[k]} = v_i - x_k + w_k \circ \operatorname{sgn}(v_i - x_k).$$

Then a subgradient $Y \in \partial H(V)$ can be calculated by

$$Y = m(V - \bar{X}) + \sum_{k=1}^{m} W^{[k]} \circ \operatorname{sgn}(V - X^{[k]}) - \sum_{k=1}^{m} [(v_{j(k)} - x_k) + w_k \circ \operatorname{sgn}(v_{j(k)} - x_k)] e_{j(k)}^T, \tag{6.21}$$

where $\bar{X}$ (resp. $X^{[k]}$, $W^{[k]}$) is the $n \times c$ matrix whose columns are all equal to $\bar{x} = \frac{1}{m}\sum_{k=1}^{m} x_k$ (resp. x_k, w_k).

In the case of the spherical uncertainty ($d = d^E$), we have

$$\frac{1}{2}\partial_{v_i} d(x_k, v_i) = v_i - x_k + \sigma_k \, \partial_{v_i} \|v_i - x_k\|.$$

Since

$$\partial_{v_i} \|v_i - x_k\| = \begin{cases} \frac{v_i - x_k}{\|v_i - x_k\|} & \text{if } v_i - x_k \neq 0 \\ \{\zeta : \|\zeta\| \leq 1\} & \text{otherwise,} \end{cases}$$

we take $\bar{y}_i^{[k]} \in \frac{1}{2}\partial_{v_i} d(x_k, v_i)$ as

$$\bar{y}_i^{[k]} = \left(1 + \frac{\sigma_k}{\|v_i - x_k\|}\right)(v_i - x_k)$$

with the convention $0 \times (+\infty) = 0$. Therefore, $Y \in \partial H(V)$ can be given by

$$Y = \sum_{k=1}^{m} (V - X^{[k]}) \mathrm{diag}\left(1 + \frac{\sigma_k}{\|v_1 - x_k\|}, \ldots, 1 + \frac{\sigma_k}{\|v_c - x_k\|}\right) - \\ - \sum_{k=1}^{m} \left(1 + \frac{\sigma_k}{\|v_j - x_k\|}\right)(v_j - x_k)e_j^T. \tag{6.22}$$

6.3.3 Solving the subproblem (6.16)

6.3.4 Case of the box uncertainty model

When $d = d^B$, (6.16) can be equivalently reformulated as

$$\min\left\{\frac{1}{2}\left\|v_i - \left(\frac{y_i^{(t)}}{m} + \bar{x}\right)\right\|^2 + \sum_{k=1}^{m}\langle\frac{w_k}{m}, |v_i - x_k|\rangle : v_i \in \mathbb{R}^n\right\},\ i = 1, \ldots, c.$$

Solving these problems amounts to solving $c \times n$ univariate convex problems

$$\min\left\{\frac{1}{2}\left[v_{ij} - \left(\frac{y_{ij}^{(t)}}{m} + \bar{x}_j\right)\right]^2 + \sum_{k=1}^{m}\langle\frac{w_{kj}}{m}, |v_{ij} - x_{kj}|\rangle : v_{ij} \in \mathbb{R}\right\},\ i = 1, \ldots, c; j = 1, \ldots, n.$$

These problems have the common form

$$\min\left\{f(x) := \frac{1}{2}(x - a)^2 + \sum_{i=1}^{m} b_i|x - a_i| : x \in \mathbb{R}\right\}, \tag{6.23}$$

where $a, a_i \in \mathbb{R}$, $b_i \in \mathbb{R}_{++}$ $(i = 1, \ldots, m)$. We will discuss below the resolution method for problem (6.23).

Denote by $f^-(x)$ (resp. $f^+(x)$) the left (resp. right) derivative of f at x. We have

$$f^-(x) = x - a + \sum_{i=1}^{m} b_i\delta_i, \quad f^+(x) = x - a + \sum_{i=1}^{m} b_i\sigma_i, \tag{6.24}$$

where $\delta_i = -1$ if $x \le a_i$ and 1 otherwise, $\sigma_i = -1$ if $x < a_i$ and 1 otherwise $(\forall i = 1, \ldots, m)$. Note that (6.23) is strongly convex, so the solution x^* is unique. We can find out the place where x^* is by using the following property.

Proposition 6.3 *Let* $\bar{a} \in \arg\min\{\sum_{i=1}^{m} b_i|x - a_i| : x \in \mathbb{R}\}$, $l_b = \min(a, \bar{a})$, *and* $u_b = \max(a, \bar{a})$. *We have the following assertions:*
i) $x^* \in [l_b, u_b]$.
ii) $f^-(a_i) > 0$ *if and only if* $x^* < a_i$.
iii) $f^+(a_i) < 0$ *if and only if* $x^* > a_i$

Proof : Given g is any finite convex function on $\mathbb{R}$. We have $\partial g(x) = [g^-(x), g^+(x)]$ for any $x \in \mathbb{R}$, and $\tilde{x} \in \arg\min_{x\in\mathbb{R}} g(x) \Leftrightarrow 0 \in \partial g(\tilde{x}) \Leftrightarrow g^-(\tilde{x}) \leq 0 \leq g^+(\tilde{x})$. Moreover, if $\tilde{x}$ is the unique optimum of g on $\mathbb{R}$, then for any $x \neq \tilde{x}$,

$$0 > g(\tilde{x}) - g(x) \geq y(\tilde{x} - x), \quad \forall y \in \partial g(x).$$

Therefore, $g^+(x) < 0$ iff $x < \tilde{x}$, and $g^-(x) > 0$ iff $x > \tilde{x}$. We have ii) and iii) are proved.

Let $f_1(x) = \frac{1}{2}(x-a)^2$ and $f_2(x) = \sum_{i=1}^m b_i|x - a_i|$. We have f_1 and f_2 are finite convex functions on $\mathbb{R}$, and a (resp. $\bar{a}$) is optimum of f_1 (resp. f_2) on $\mathbb{R}$. Without loss of generality we assume that $a \leq \bar{a}$. Then $f^+(a) = f_2^+(a) \leq 0$ and $f^-(\bar{a}) = f_1^-(\bar{a}) \geq 0$. This implies that $a \leq x^* \leq \bar{a}$. i) is proved. □

Once we find out the interval where x^* is and f is differentiable, x^* is easily determined by solving the equation $f'(x^*) = 0$. Without loss of generality, we assume that the sequence $\{a_i\}$ is in ascending order $a_1 \leq a_2 \leq \cdots \leq a_m$. For convenience, let $a_0 = -\infty$. Then the specific procedure for finding the solution x^* of problem (6.23) is given in Algorithm 6.1 below.

Algorithm 6.1 Solving problem (6.23)

Compute $\bar{a} \in \arg\min \{\sum_{i=1}^m b_i|x - a_i| : x \in \mathbb{R}\}$, $l_b = \min(a, \bar{a})$, and $u_b = \max(a, \bar{a})$
Let $i_f = \min\{i = 1, \ldots, m : a_i \geq l_b\}$, $i_l = \max\{i = 0, \ldots, m : a_i < u_b\}$
if $i_l = 0$ **then**
  Compute $f^-(a_{i_l})$
  if $f^-(a_{i_l}) > 0$ **then**
    $x^* = a + \sum_{i=1}^m b_i \Rightarrow$ STOP
  else
    $x^* = a_1 \Rightarrow$ STOP
  end if
end if
Compute $f^-(a_{i_f})$
if $f^-(a_{i_f}) \leq 0$ **then**
  $l_b = a_{i_f}$
else
  $x^* = a - \sum_{i=1}^{i_f - 1} b_i + \sum_{i=i_f}^m b_i \Rightarrow$ STOP
end if
while $i_f < i_l$ **do**
  Let $i_p = \lceil \frac{i_f + i_l}{2} \rceil$ [2]and compute $f^-(a_{i_p})$
  if $f^-(a_{i_p}) = 0$ **then**
    $x^* = a_{i_p} \Rightarrow$ STOP
  else
    if $f^-(a_{i_p}) > 0$ **then**

```
        u_b = a_{i_p}, i_l = max{i = 1, ..., i_p − 1 : a_i < u_b}
    else
        l_b = a_{i_p}, i_f = max{i = i_p, ..., m : a_i = l_b}
    end if
  end if
end while
Compute f^+(a_{i_f})
if f^+(a_{i_f}) ≥ 0 then
  x* = a_{i_f} ⇒ STOP
else
  x* = a − Σ_{i=1}^{i_f} b_i + Σ_{i=i_f+1}^{m} b_i ⇒ STOP
end if
```

6.3.5 Case of the spherical uncertainty model

When $d = d^E$, (6.16) is equivalent to

$$\min\left\{\frac{1}{2}\left\|v_i - \left(\frac{y_i^{(t)}}{m} + \frac{1}{m}\sum_{k=1}^{m} x_k\right)\right\|^2 + \sum_{k=1}^{m}\frac{\sigma_k}{m}\|v_i - x_k\| : v_i \in \mathbb{R}^n\right\},\ i = 1, \ldots, c.$$

Solving these problems amounts to solving c convex problems of the form

$$\min\left\{f(x) := \frac{1}{2}\|x - b_0\|^2 + \sum_{i=1}^{m} a_i\|x - b_i\| : x \in \mathbb{R}^n\right\}, \tag{6.25}$$

where $b_i \in \mathbb{R}^n$ ($\forall i = 0, 1, \ldots, m$), and a_i ($i = 1, \ldots, m$) are positive scalars. Clearly, this problem is a generation of (6.23). Note that without the squared Euclidean distance term, this problem becomes the well-known Weber problem (Kuhn and Kuenne, 1962) and the resolution method developed here can be straightforwardly extended to the Weber problem. It is easy to see that the function f is strongly convex and coercive, so this problem has a unique solution, denoted by x^*.

Without loss of generality, we assume that $b_1, \ldots, b_m$ are distinguished. Since $\partial_x\|x - b\| = \left\{\frac{x-b}{\|x-b\|}\right\}$ if $x \neq b$ and $\{\zeta \in \mathbb{R}^n : \|\zeta\| \leq 1\}$, the necessary and sufficient optimality condition of problem (6.25) is given by

$$0 \in \partial f(x) \Leftrightarrow \begin{cases} (x - b_0) + \sum_{i=1}^{m} \frac{a_i}{\|x-b_i\|}(x - b_i) = 0 & \text{if } x \neq b_i\ \forall i, \\ \left\|(x - b_0) + \sum_{j\neq i} \frac{a_j}{\|x-b_j\|}(x - b_j)\right\| \leq a_i & \text{if } x = b_i. \end{cases} \tag{6.26}$$

2. The notation $\lceil t \rceil$ denotes the the smallest integer not less than t

If $m = 1$, from the condition (6.26), it is easy to verify that x^* is given by

$$x^* = \begin{cases} b_1 & \text{if } \|b_1 - b_0\| \le a_1 \\ \left(1 - \frac{a_1}{\|b_1 - b_0\|}\right) b_0 + \frac{a_1}{\|b_1 - b_0\|} b_1 & \text{otherwise.} \end{cases} \tag{6.27}$$

In the sequel, we will develop an iterative algorithm for solving problem (6.25) in the general case $m > 1$. We do not apply directly DC programming and DCA to problem (6.25). But we use these methods to compute the next iterate x^+ from the current iterate x such that $f(x^+) \le f(x)$.

For an $i \in \{1, \dots, n\}$, (6.25) is equivalent to

$$\begin{aligned} \min_{x,t \in \mathbb{R}^n} \quad & F_i(x,t) := \frac{1}{2}\|x - b_0\|^2 + a_i\|x - b_i\| + \sum_{j \ne i} a_j r(t_j) \\ s.t. \quad & \|x - b_j\|^2 \le t_j \quad \forall j = 1, \dots, m, \ j \ne i, \end{aligned} \tag{6.28}$$

where $r(\cdot) = \sqrt{\cdot}$ is a concave function on $[0, \infty)$ (note that the ith element of t is not needed, but we denote in such a way for convenience).

Problem (6.28) can be reformulated as a DC program as follows

$$\min \{G_i(x,t) - H_i(x,t) : (x,t) \in C_i\} \tag{6.29}$$

where $C_i = \{(x,t) \in \mathbb{R}^n \times \mathbb{R}^n : \|x - b_j\|^2 \le t_j, \ \forall j = 1, \dots, m, \ j \ne i\}$ and

$$G_i(x,t) = \frac{1}{2}\|x - b_0\|^2 + a_i\|x - b_i\|, \quad H_i(x,t) = -\sum_{j \ne i} a_j r(t_j).$$

Supposed that we are in the iteration k ($k = 0, 1, 2, \dots$) with the current iterate x^k. Let $i_k \in \arg\min\{\|x^k - b_j\| : j = 1, \dots, m\}$ and consider the DC program (6.29) with $i = i_k$. By the assumption that $b_1, \dots, b_m$ are distinguished and the definition of i_k, we have $\|x^k - b_j\| \ge \frac{1}{2}\min\{\|b_p - b_q\| : p \ne q\} > 0 \ \forall j \ne i$. Let $t^k \in \mathbb{R}^n$ such that $t_j^k = \|x^k - b_j\|^2 \ \forall j \ne i$. Initializing from (x^k, t^k), one iteration of DCA applied to (6.29) consists of

- Compute $(\bar{x}^k, \bar{t}^k) \in \partial H_i(x^k, t^k)$ by $\bar{x}^k = 0$ and $\bar{t}_j^k = -\frac{a_j}{2\sqrt{t_j^k}} \ \forall j \ne i$.

- Compute the next iterate (x^{k+1}, t^{k+1}) as a solution to the convex problem

$$\min \left\{ \frac{1}{2}\|x - b_0\|^2 + a_i\|x - b_i\| + \sum_{j \ne i} \frac{a_j}{2\sqrt{t_j^k}} t_j : \|x - b_j\|^2 \le t_j, \forall j \ne i \right\} \tag{6.30}$$

This is equivalent to computing x^{k+1} as the solution to the convex problem

$$\min \left\{ \frac{1}{2}\|x - b_0\|^2 + a_i\|x - b_i\| + \sum_{j \ne i} \frac{a_j}{2\|x^k - b_j\|} \|x - b_j\|^2 : x \in \mathbb{R}^n \right\} \tag{6.31}$$

and setting $t_j^{k+1} = \|x^{k+1} - b_j\|^2 \ \forall j \neq i$. Let

$$y^k = b_0 + \sum_{j \neq i} \frac{a_j}{\|x^k - b_j\|} b_j, \quad \lambda^k = 1 + \sum_{j \neq i} \frac{a_j}{\|u - b_j\|}.$$

By virtue of (6.26), we have

$$\begin{aligned} x^{k+1} &= \arg\min \left\{ \frac{1}{2} \left\| x - \frac{y^k}{\lambda^k} \right\|^2 + \frac{a_i}{\lambda^k} \|x - b_i\| : x \in \mathbb{R}^n \right\} \\ &= \begin{cases} b_i & \text{if } \tau^k \leq a_i \\ \left(1 - \frac{a_i}{\tau^k}\right) y^k + \frac{a_i}{\tau^k} b_i & \text{otherwise,} \end{cases} \end{aligned}$$

where $\tau^k = \|\lambda^k b_i - y^k\|$.

We summarize this procedure in Algorithm 6.2 below. With this algorithm, we do not use a unique DC reformulation. Instead, at each iteration we use a different DC reformulation among m DC reformulations of the form (6.29) depending on the index i_k computed at step 1. Below we give a characteristic of a critical point of (6.29).

Proposition 6.4 *Suppose that (x^k, t^k) as in the above analysis. If (x^k, t^k) is a critical point of (6.29) then x^k is the solution to problem (6.25).*

Proof : If (x^k, t^k) is a critical point of (6.29) then (x^k, t^k) is a solution to problem (6.30). This means that x^k is a solution to problem (6.31). It is easy to verify that the optimality condition of problem (6.31) at x^k is the same as (6.26). Thus, $x^k = x^*$. □

Note that, without the assumption $i_k \in \arg\min\{\|x^k - b_j\| : j = 1, \ldots, m\}$ this property is still valid, as long as $t_j^k > 0 \ \forall j \neq i$. Now we are in a position to analyze the convergence of Algorithm 6.2.

Algorithm 6.2 Solving problem (6.25)

Initialize x^0, $k \leftarrow 0$

repeat

1. Compute $i_k \in \arg\min\{\|x^k - b_j\| : j = 1, \ldots, m\}$
2. Compute $y^k = b_0 + \sum_{j \neq i_k} \frac{a_j}{\|x^k - b_j\|} b_j$, $\lambda^k = 1 + \sum_{j \neq i_k} \frac{a_j}{\|x^k - b_j\|}$ and $\tau^k = \|\lambda^k b_{i_k} - y^k\|$
3. Compute $x^{k+1} = \begin{cases} b_{i_k} & \text{if } \tau^k \leq a_{i_k} \\ \left(1 - \frac{a_{i_k}}{\tau^k}\right) y^k + \frac{a_{i_k}}{\tau^k} b_{i_k} & \text{otherwise.} \end{cases}$
4. $k = k + 1$

until Convergence of $\{x^k\}$

Proposition 6.5 *Let $\{x^k\}$ be the sequence generated by Algorithm 6.2. Then we have*

i) The sequence $\{f(x^k)\}$ is decreasing.

ii) $\{x^k\}$ converges to the unique solution x^* of problem (6.25).

iii) If $x^* = b_i$ for some $i \in \{1, \ldots, m\}$ and $\left\|(b_i - b_0) + \sum_{j \neq i} \frac{a_j}{\|b_i - b_j\|}(x - b_j)\right\| < a_i$, then Algorithm 6.2 terminates after a finite number of iterations.

Proof : For any $k = 0, 1, \ldots,$ let $t^k, \bar{x}^k, \bar{t}^k \in \mathbb{R}^n$ be given by $\bar{x}^k = 0$, $t_j^k = \|x^k - b_j\|^2$ and $\bar{t}_j^k = -\frac{a_j}{2\sqrt{t_j^k}}$ $\forall j \neq i_k$. Then $\{(x^k, t^k)\}$ and $\{(\bar{x}^k, \bar{t}^k)\}$ are the sequence of primal and dual variables in the DCA frameworks respectively.

According to the general convergence properties of DCA, we have

$$f(x^{k+1}) = F_{i_k}(x^{k+1}, t^{k+1}) \leq F_{i_k}(x^k, t^k) = f(x^k).$$

Thus, i) is proved. Combing with the fact that f is coercive, we have $\{x^k\}$ is bounded and so is $\{t^k\}$. Moreover, $\|x^k - b_j\| \geq \frac{1}{2}\min\{\|b_p - b_q\| : p \neq q\} > 0$ $\forall j \neq i$, so $\{t^k\}$ is away from 0 and $\{(\bar{x}^k, \bar{t}^k)\}$ is bounded.

We will prove that x^* is the unique limit point of $\{x^k\}$. Suppose that $\bar{x}$ is a limit point of $\{x^k\}$, then there is a subsequence $\{x^{k_l}\}$ such that $x^{k_l} \to \bar{x}$ as $l \to \infty$. Since $\{i_k\}$ has finite elements, we can suppose (by extracting a subsequence if necessary) that there is an $i \in \{1, \ldots, m\}$ such that $i_{k_l} = i$ $\forall l$. Therefore, for any $l = 0, 1, 2, \ldots,$ (x^{k_l+1}, t^{k_l+1}) is computed from (x^{k_l}, t^{k_l}) via the same DC reformulation of the form (6.29). Let $\bar{t} = \lim t^{k_l}$, then $\bar{t}_j = \|\bar{x} - b_j\| > 0$ $\forall j \neq i$ since $\{t^k\}$ is away from 0. Then $(\bar{x}, \bar{t})$ is a critical point of (6.29). Let $(\bar{x}, \bar{t})$ play the role of (x^k, t^k) in Proposition 6.4, we deduce that $\bar{x}$ is the solution to problem (6.25), i.e. $\bar{x} = x^*$. Therefore, x^* is the unique limit point of $\{x^k\}$. Moreover, $\{x^k\}$ is bounded, so every its subsequence contains a subsubsequence converging to the same limit point x^*. This means that $\{x^k\}$ converges to x^*. We have ii) is proved.

If $x^* = b_i$, by virtue of iii), we have $i_k = i$ for k large enough. Let $y^* = b_0 + \sum_{j \neq i} \frac{a_j}{\|b_i - b_j\|} b_j$, $\lambda^* = 1 + \sum_{j \neq i} \frac{a_j}{\|b_i - b_j\|}$. Then

$$\tau^k \to \|\lambda^* b_i - y^*\| = \left\|(b_i - b_0) + \sum_{j \neq i} \frac{a_j}{\|b_i - b_j\|}(x - b_j)\right\| < a_i.$$

Thus, for k large enough, we have $\tau^k < a_i$ and $x^{k+1} = b_i$. □

We are in a position to describe the DCAs for solving problems (6.6) and (6.8) via the DC decomposition (6.15).

6.3.6 Description of DCA to solve the robust clustering problems with box and spherical uncertainty models

DCA-Box (DCA for solving the robust clustering problem (6.6))
Initialization: Let $\epsilon > 0$, $V^{(0)} \in \mathbb{R}^{n\times c}$, $t = 0$.
Repeat
1. Compute $Y^{(t)}$ by using (6.21)

$$Y^{(t)} = m(V^{(t)} - \bar{X}) + \sum_{k=1}^{m} W^{[k]} \circ \operatorname{sgn}(V^{(t)} - X^{[k]}) - \sum_{k=1}^{m} [(v_{j(k)}^{(t)} - x_k) + w_k \circ \operatorname{sgn}(v_{j(k)}^{(t)} - x_k)] e_{j(k)}^T$$

with $j(k) \in \arg\min_{j=1,\dots,c} d^B(x_k, v_j^{(t)})$, $k = 1, \dots, m$.
2. Compute $V^{(t+1)}$ by using Algorithm 6.1 to compute, $\forall i = 1, \dots, c$; $j = 1 \dots, n$,

$$v_{ij}^{(t+1)} = \arg\min \left\{ \frac{1}{2} \left(v_i - \left(\frac{y_{ij}^{(t)}}{m} + \bar{x}_j \right) \right)^2 + \sum_{k=1}^{m} \frac{w_{kj}}{m} |v_{ij} - x_{kj}| : v_{ij} \in \mathbb{R} \right\},$$

3. $t \leftarrow t + 1$.
Until $\|V^{(t-1)} - V^{(t)}\| < \epsilon(\|V^{(t)}\| + 1)$ or $|F^B(V^{(t-1)}) - F^B(V^{(t)})| < \epsilon(|F^B(V^{(t)})| + 1)$.

DCA-Sph (DCA for solving the robust clustering problem (6.8))
Initialization: Let $\epsilon > 0$, $V^{(0)} \in \mathbb{R}^{n\times c}$, $t = 0$.
Repeat
1. Compute $Y^{(t)}$ by using (6.22)

$$Y^{(t)} = \sum_{k=1}^{m} (V^{(t)} - X^{[k]}) \operatorname{diag}\left(1 + \frac{\sigma_k}{\|v_1^{(t)} - x_k\|}, \dots, 1 + \frac{\sigma_k}{\|v_c^{(t)} - x_k\|} \right) - \\ - \sum_{k=1}^{m} \left(1 + \frac{\sigma_k}{\|v_{j(k)}^{(t)} - x_k\|} \right) (v_{j(k)}^{(t)} - x_k) e_{j(k)}^T,$$

with $j(k) \in \arg\min_{j=1,\dots,c} \|v_j - x_k\|$, $k = 1, \dots, m$.
2. Compute $V^{(t+1)}$ by using Algorithm 6.2 to compute, $\forall i = 1, \dots, c$,

$$v_i^{(t+1)} = \arg\min \left\{ \frac{1}{2} \left\| v_i - \left(\frac{y_i^{(t)}}{m} + \frac{1}{m} \sum_{k=1}^{m} x_k \right) \right\|^2 + \sum_{k=1}^{m} \frac{\sigma_k}{m} \|v_i - x_k\| : v_i \in \mathbb{R}^n \right\},$$

3. $t \leftarrow t + 1$.
Until $\|V^{(t-1)} - V^{(t)}\| < \epsilon(\|V^{(t)}\| + 1)$ or $|F^E(V^{(t-1)}) - F^E(V^{(t)})| < \epsilon(|F^E(V^{(t)})| + 1)$.

6.4 Numerical Experiments

In this section, we carry out some experiments on both synthetic and real datasets to validate the proposed methods. We will compare the robust clusterings (6.6) (DCA-Box algorithm) and (6.8) (DCA-Sph algorithm) with three methods: the MSSC (DCA-MSSC algorithm (Le Thi et al., 2006)), the k-median clustering (k-median algorithm (Bradley et al., 1997)), and a method, called Alternative hard c-means clustering (AHCM) (Wu and Yang, 2002). The AHCM uses the exponential distance $d(x_k, v_i) = 1 - e^{-\beta\|x_k - v_i\|^2}$ ($\beta > 0$) instead of the squared Euclidean distance in (6.1). With the use of a new metric, AHCM is proved to be more robust than MSSC with noisy data and outliers (Wu and Yang, 2002).

The algorithms have been coded in VC++ and implemented on a Intel Core i5 CPU 2 × 2.74 GHz, RAM 4GB. The tolerance ϵ for the stopping criterion was set to be 10^{-6} for all algorithms.

For a set of uncertain data, each data point is represented as an uncertainty set that is a box or a ball. With this kind of data, our proposed methods are directly applicable. For certain methods such as k-median, DCA-MSSC or AHCM, each data point has to be a single point. To apply these methods to uncertain data, we can let the center of the uncertainty sets be the observed data. However, most of the benchmark datasets for clustering are in fact available as the observed data. We assume that they are uncertain and regard the observed data points as the centers of the uncertainty sets. The uncertainty information, i.e. the dimensions of boxes (resp. the radii of balls) in the box (resp. spherical) uncertainty model, is assumed to be computed by

$$(w_k)_j = w_j := \left(\frac{1}{m}\sum_{l=1}^{m}((x_l)_j - \bar{x}_j)^2\right)^{1/2}, \quad \forall k = 1, \ldots, m; j = 1, \ldots, n,$$

for the box uncertainty, and

$$\sigma_k = \left(\frac{1}{m}\sum_{l=1}^{m}\|x_l - \bar{x}\|^2\right)^{1/2}, \quad k = 1, \ldots, m,$$

for the spherical uncertainty, where $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_k$ is the mean of all data points.

Note that, all the considered clustering problems are nonconvex and the corresponding algorithms are local methods. Thus, for fair comparison, we run each algorithm several times from different initiamizations and select the solution corresponding to the smallest objective value.

The quality of clustering solutions was evaluated by two criteria: Adjusted Rand index Hubert and Arabie (1985) (Adj RI) and F-measure Gullo and Tagarelli (2012). The higher these criteria, the better clustering performance. We recall briefly here the definition of F-measure. Let us denote, for every instance x_k ($k = 1, \ldots, n$), its initial class by $I_{ref}(x_k)$

Table 6.1: Comparative results on the first synthetic dataset

Criterion	k-median	DCA-MSSC	DCA-Box	DCA-Sph	AHCM
Adj RI	0.659	0.640	**0.732**	0.713	0.713
F-measure	0.871	0.851	**0.902**	0.893	0.895

and its cluster obtained from the clustering algorithm by $I_{class}(x_k)$. For $i = 1, \ldots, c$, let $C_{ref}(i) = \{k : I_{ref}(x_k) = i\}$ and $C_{class}(i) = \{k : I_{class}(x_k) = i\}$. Then the F-measure is defined by (Gullo and Tagarelli (2012))

$$F(C_{ref}, C_{class}) = \frac{1}{n} \sum_{i=1}^{c} |C_{ref}| \max_{j=1,\ldots,c} F_{ij},$$

where $F_{ij} = 2P_{ij}R_{ij}/(P_{ij} + R_{ij})$ with

$$P_{ij} = \frac{|C_{ref}(i) \cap C_{class}(i)|}{|C_{class}|}, \quad R_{ij} = \frac{|C_{ref}(i) \cap C_{class}(i)|}{|C_{ref}|}, \quad \forall i, j = 1, \ldots, c.$$

6.4.1 Experiment on synthetic datasets

As mentioned above, the proposed methods could be capable of resisting outlier, beside the purpose of handling uncertain data. Thus, to evaluate both aspects, we test on two synthetic datasets with two different constructions: the first dataset contains outliers and the second dataset contains uncertain data.

The first synthetic dataset contains $m = 138$ points of three clusters are created as follows. We set three seed points $v_1 = (0, 0)$, $v_2 = (2, 2)$, $v_3 = (4, 0)$. For $k = 1, \ldots, 120$, we first generate the class label $i_k \in \{1, 2, 3\}$ with equal probability, then x_k is computed from a multivariate Gaussian distribution $N(v_{i_k}, \delta I)$ with $\delta = 1$. To make the dataset containing noisy data points, for $k = 121, \ldots, 138$, we generate x_k from $N((2, 1), 4I)$, then the class label is determined by $i_k = \arg\min\{\|x_k - v_i\| : i = 1, 2, 3\}$. We generate 10 datasets in this way. For each dataset, we run each algorithm 10 times from random initialization and select the solution corresponding to the smallest objective value. The average results of 10 datasets are presented in Table 6.1. It is clear that DCA-Box and DCA-Sph outperform the k-median and DCA-MSSC algorithms. DCA-Box has the best performance while DCA-Shp and AHCM is comparable.

The second synthetic dataset contains $m = 120$ points generated by the same manner as the first 120 points of the first synthetic dataset with $\delta = 0.7$ to make the clusters be more compact. This is referred to as the *true* dataset $\mathcal{X} = (x_1, \ldots, x_m)$. From this true dataset, a *disturbed* dataset $\overline{\mathcal{X}} = (\bar{x}_1, \ldots, \bar{x}_m)$ is generated as follows. For $k = 1, \ldots, m$, we take $\bar{\omega}_k = U(0, 1) \times \left[0.2|x_k| + (|N(0, \omega_1)|, |N(0, \omega_2)|)\right]$, where $U(0, 1)$ denotes a uniformly

Table 6.2: Comparative results on the second synthetic datasets. The upper row presents results on the true dataset, the lower row presents results on the disturbed dataset.

Criterion	k-median	DCA-MSSC	DCA-Box	DCA-Sph	AHCM
Adj RI	0.862	0.872	0.872	0.873	0.871
	0.744	0.752	**0.766**	0.752	0.753
F-measure	0.952	0.955	0.955	0.955	0.955
	0.906	0.909	**0.914**	0.908	0.909

distributed number, ω_i $(i = 1, 2)$ is the standard deviation of $\mathcal{X}$ on dimension i $(i = 1, 2)$. Then $\bar{x}_k = x_k + N(0, \bar{\omega}_k I)$ for any $k = 1, \ldots, m$. We also generate 10 datasets in this way. For each dataset, we run each algorithm 10 times from random initialization and select the solution corresponding to the smallest objective value. The average results of 10 datasets are presented in Table 6.2. We see that on the true datasets, four methods DCA-MSSC, DCA-Box, DCA-Sph and AHCM are comparable and better than k-median. Meanwhile, on the disturbed datasets, DCA-Box is the best.

6.4.2 Experiment on real datasets

We test on 13 real datasets taken from UCI[3] (Table 6.3). For datasets Lympho and ADN whose attribute characteristics are categorical, we first transform the data by using the Chi-square metric. Consider a species abundance data table $Z = (z_{ij})$ of size $(m \times n)$ with sites (rows) $i \in \{1, \ldots, m\}$ and species (columns) $j \in \{1, \ldots, n\}$, the row sums are denoted by z_{i+} and the column sums z_{+j}. The dataset $Z = (z_{ij})$ are transformed into $Z' = (z'_{ij})$ as $z'_{ij} = \frac{z_{ij}}{z_{i+}\sqrt{z_{+j}}}$ for all i, j. Then the Euclidean distance between rows k^{th} and l^{th} of the transformed data Z'

$$d(Z_{k:}, Z_{l:}) = \|Z_{k:} - Z_{l:}\| = \left(\sum_{j=1}^{n} \frac{1}{z_{+j}} \left(\frac{z_{kj}}{z_{k+}} - \frac{z_{lj}}{z_{l+}} \right)^2 \right)^{1/2}$$

is identical to the Chi-square metric between rows k^{th} and l^{th} of the original data Z.

For each dataset, we run each algorithm 50 times from random initialization and select the solution corresponding to the smallest objective value. The running time in second is average over 50 trials. The results are presented in Table 6.4.

We observe from computational results that.

Adjusted Rand index. DCA-Box has the best performance on 8 out of 13 datasets. It is better than k-median (resp. DCA-MSSC, AHCM) on 10 (resp. 7, 8) datasets. DCA-Sph

3. http://archive.ics.uci.edu/ml/

Table 6.3: Real datasets used in experiments

Dataset	no. of samples	no. of features	no. of classes
Iris	150	4	3
Lympho	148	18	4
Papillon	23	4	4
StatLog	4435	36	6
Vote	435	16	2
Waveform	5000	40	3
Wine	178	13	3
Comp	3891	10	3
Glass	214	10	6
Seeds	210	7	3
Ecoli	336	7	8
Yeast	1484	8	10
ADN	3186	60	3

has the best (resp. second best) performance on 6 (resp. 5) out of 13 datasets. It is better than k-median (resp. DCA-MSSC, AHCM) on 10 (resp. 8, 10) datasets. It also has the same performance with DCA-MSSC (resp. AHCM) on 3 (resp. 2) datasets.

F-measure. DCA-Box has the best (resp. second best) performance on 8 (resp. 2) out of 13 datasets. It is better than k-median (resp. DCA-MSSC, AHCM) on 11 (resp. 10, 10) datasets. DCA-Sph has the best (resp. second best) performance on 5 (resp. 6) out of 13 datasets. It is better than k-median (resp. DCA-MSSC, AHCM) on 10 (resp. 9, 10) datasets. It also has the same performance with DCA-MSSC (resp. AHCM) on 2 (resp. 2) datasets.

The running times of the DCA-Box and DCA-Sph are often longer than the DCA-MSSC. The reason is that at each iteration, both DCA-Box and DCA-Sph require to iteratively solve a convex subproblem. Meanwhile, the solution of the subproblem in DCA-MSSC can be computed explicitly.

6.5 Conclusion

In this work, we have applied the robust optimization approach to the clustering problem. By assuming uncertainty of the input data in which box uncertainty sets or spherical uncertainty sets are considered, we obtain robust formulations from the minimum sum-of-squares clustering model. The cluster-compactness criterion of these robust formulations can be expressed as a combination of the cluster-compactness criterion using the squared Euclidean distance and the one using Manhattan/Euclidean distance. Based on DC programming and DCA, we have developed efficient algorithms to solve the resulting optimization problems.

Table 6.4: Comparative results on real datasets

Dataset	Criterion	k-median	DCA-MSSC	DCA-Box	DCA-Sph	AHCM
Iris	Adj RI	0.717	0.730	**0.744**	0.729	0.730
	F-measure	0.884	0.891	**0.898**	0.892	0.891
	Time (s)	0.001	0.001	0.002	0.004	0.001
Lympho	Adj RI	0.159	0.207	0.188	**0.222**	0.137
	F-measure	0.537	0.603	0.617	**0.662**	0.544
	Time (s)	0.002	0.001	0.005	0.013	0.001
Papillon	Adj RI	0.315	0.517	0.315	**0.722**	0.414
	F-measure	0.616	0.797	0.616	**0.834**	0.709
	Time (s)	0.001	0.001	0.001	0.001	0.001
StatLog	Adj RI	**0.546**	0.534	0.519	0.534	0.533
	F-measure	0.724	0.708	**0.755**	0.718	0.716
	Time (s)	0.237	0.043	6.904	2.31	0.028
Vote	Adj RI	0.529	0.543	**0.584**	0.564	0.563
	F-measure	0.866	0.870	**0.884**	0.877	0.877
	Time (s)	0.008	0.001	0.388	0.004	0.001
Wine	Adj RI	0.371	0.371	**0.375**	**0.375**	0.371
	F-measure	0.719	0.714	**0.720**	**0.720**	0.714
	Time (s)	0.002	0.001	0.006	0.011	0.001
Wave	Adj RI	0.250	0.251	0.251	0.251	0.251
	F-measure	0.535	0.535	**0.539**	0.535	0.535
	Time (s)	1.107	0.019	1.334	1.05	0.010
Comp	Adj RI	0.902	**0.928**	0.911	0.921	0.915
	F-measure	0.966	**0.975**	0.969	0.972	0.970
	Time (s)	0.572	0.008	0.292	0.351	0.005
Glass	Adj RI	0.550	0.527	**0.568**	0.543	0.527
	F-measure	0.669	0.656	**0.676**	0.664	0.660
	Time (s)	0.003	0.001	0.991	0.031	0.001
Seeds	Adj RI	0.699	**0.716**	0.689	**0.716**	**0.716**
	F-measure	0.885	**0.895**	0.880	**0.895**	**0.895**
	Time (s)	0.002	0.001	0.122	0.007	0.001
Ecoli	Adj RI	0.386	0.401	**0.478**	0.423	0.417
	F-measure	0.614	0.635	0.685	**0.686**	0.675
	Time (s)	0.000	0.000	0.022	0.062	0.000
Yeast	Adj RI	0.114	0.132	**0.136**	**0.136**	0.134
	F-measure	0.384	0.431	**0.434**	0.427	0.424
	Time (s)	0.013	0.010	0.514	0.309	0.008
ADN	Adj RI	0.053	0.026	**0.179**	0.116	0.069
	F-measure	0.454	0.507	**0.598**	0.563	0.475
	Time (s)	0.618	0.003	0.507	0.307	0.005

Comparative numerical results also show the efficiency and the superiority of the proposed robust optimization approaches with respect to the DCA-MSSC, the k-median, and the AHCM algorithms.

The robust formulation developed so far can be regarded as a relaxation of the robust formulation (6.9) that would be more meaningful in dealing with uncertainty. In the future work, we will study the formulation (6.9) and develop numerical method for solving it. Moreover, more complex uncertainty models should be studied to incorporate more available information of error.

Chapter 7

Conclusion

We have addressed in this thesis several problems concerning with the sparsity and the uncertainty from various aspects, with theoretical, algorithmic and applied considerations. The main algorithmic methodologies that the thesis used are DC (Difference of Convex functions) programming and DCA (DC Algorithms) well-known as the powerful tools in optimization.

We have first proposed DC approximation approaches for sparse optimization problem including the zero-norm in the objective function. By considering a common DC approximation of the zero-norm that includes all standard sparsity-inducing penalty functions, we have studied the consistency between global minimums (local minimums) of approximate and original problems. We have also studied the connection between the approximation approach and the exact penalty technique. The efficiency of several sparsity-inducing penalty functions have been fully analyzed. Four DCA schemes have been developed that cover all standard algorithms in nonconvex sparse approximation approaches as special versions.

Our second contribution lies in the investigation of DC programming and DCA for solving the nonnegative matrix factorization (NMF) problem and its extensions. We have proposed two approaches for solving this problem: one is based on the alternating method and DCA and one is a directly application of DC programming and DCA. Resolution methods for numerous extensions of the NMF problem have showed that the approaches based on DC programming and DCA are flexible can be adapted easily to other situations. Using the capped-ℓ_1 function for enhancing sparsity, we have considered the sparse NMF problem. We have demonstrated that the new sparse formulations can effectively learn parts-based representations.

We then studied the dictionary learning problem with application in image denosing. The capped-ℓ_1 function is chosen for modeling the sparsity. Algorithm for dictionary learning alternates between two phases: dictionary updating and computing sparse representations of training data. Algorithms based on DC programming and DCA have been developed to

solve the subproblems in each phase.

Our fourth contribution concerns the problem of simultaneous classification and feature selection on uncertain data. We have proposed robust optimization approaches based on DCA. In particular, we have considered the ellipsoidal and box models for uncertain data in which the uncertainty of each data point is independent.

Finally, we have applied the robust optimization approach to the clustering problem. We have considered the box and spherical uncertainty models for the minimum-sum-of-square clustering (MSSC). The robust formulations can be regarded as the regularized formulations of the MSSC with the regularization term is a weighted sum of Manhattan/Euclidean distances. Efficient algorithms based on DCA have been developed for solving these robust clustering problems. Numerical results show that our new clustering methods are capable to handle uncertain data and work well in the real situations.

This thesis has explored some issues relating to modeling sparsity, sparse optimization, matrix factorization, and data uncertainty. Several issues for future research are derived from this research.

For sparse optimization, the theoretical results as well as DCA based algorithms presented in this thesis will be useful to develop global approaches such as Brand and Bound and/or interval analysis for sparse optimization problems. Since DCA has been shown to be efficient and scalable, it is worthwhile to suitably combine DCA with these global approaches in order to improve the quality of computed solutions. The remaining hard question is to find tight convex underestimations of functions involving the zero norm. Moreover, in our works, we have only considered the problem with the zero-norm in the objective function. However, there are a number of applications where we need to directly control sparsity under the constraints. Recently, some novel approaches for modeling sparsity such as group-sparsity, structured-sparsity (Bach et al., 2011) have been proposed. In future works, we will investigate the DC approximation approaches to these kinds of sparsity model.

For two factorization techniques - NMF and dictionary learning, the main purpose is to find a good basis/dictionary that can discover latent features of a corpus of data so that we can have appropriate representations of data in particular applications. The idea of online learning is promising to learn the bases/dictionary with massive training data. We will adopt this approach to our proposed batch methods and study their theoretical aspect.

We also plan to extend the proposed methods for NMF problem using Frobenius norm as discrepancy measure to other NMF formulations.

For feature selection under uncertain data, the materials in chapter 5 can be used / extended in other contexts dealing with sparsity and uncertainty. On another hand, we consider the ellipsoidal and box models for uncertain data in which the uncertainty of each data point is independent. This model is suitable for most practical data. However, other models of uncertainty should be interesting, such as the models with dependent uncertainty data on the

whole dataset. Furthermore, extensions of this problem to nonlinear case will be considered in future work. In another direction we will study robust optimization for classification under uncertain class labels.

So far, we have formulated the robust clustering problems from the mixed-integer formulation of the MSSC. These problems are relaxations of the robust formulations constructed from the bilevel formulation of the MSSC. With robust optimization approach, we should avoid redundant variables to keep the robust counterpart from being more conservative. Thus, the robust formulations constructed from the bilevel MSSC can be more meaningful than the considered robust formulations. Research on the resolution method for the robust counterpart of the bilevel MSSC is in progress. Moreover, more complex uncertainty models will be studied to incorporate more available information of the error.

Appendix A

Appendix

A.1 Projection onto a nonnegative ball

Lemma A.1 *Given $x^0 \in \mathbb{R}^n$ and $S = \{x \in \mathbb{R}^n : x \geq 0, \|x\| \leq b\}$, $b > 0$. Then the projection of the point x^0 onto the set S is as follows*

$$x^* = P_S(x^0) = \begin{cases} [x^0]_+, & \text{if } \|[x^0]_+\| \leq b, \\ \frac{b}{\|[x^0]_+\|}[x^0]_+, & \text{otherwise.} \end{cases} \tag{A.1}$$

Proof : The projection x^* is the unique solution of the following optimization problem

$$\begin{array}{ll} \min\limits_{x} & \frac{1}{2}\|x - x^0\|^2 \\ s.t & x \geq 0, \ \|x\|^2 \leq b^2. \end{array}$$

Since this is a convex problem, x^* will be solution iff there exists $(\lambda^*, \delta^*) \in \mathbb{R}^n_+ \times \mathbb{R}_+$ such that the following KKT conditions are satisfied

$$\begin{cases} x - x^0 - \lambda + 2\delta x = 0, \\ x \geq 0, \ \lambda \geq 0, \ \langle \lambda, x \rangle = 0, \\ \|x\|^2 \leq b^2, \ \delta \geq 0, \ \delta(\|x\|^2 - b^2) = 0. \end{cases}$$

We have two cases:

- If $\|[x^0]_+\| \leq b$: we take $x^* = [x^0]_+$, $\lambda^* = [-x^0]_+$ and $\delta^* = 0$.

- If $\|[x^0]_+\| > b$: we take $x^* = \dfrac{b}{\|[x^0]_+\|}[x^0]_+$, $\lambda^* = [-x^0]_+$ and $\delta^* = \dfrac{1-c}{2c}$, where $c = \dfrac{b}{\|[x^0]_+\|} < 1$. □

A.2 Orthogonal matrix

Définition A.2.1 *A matrix P is said to be* orthogonal *if it is nonsingular and $P^T = P^{-1}$.* □

Theorem A.1 (Howard (2004)) – *The transpose and the inverse of an orthogonal matrix are orthogonal.*
– *The product of two or more orthogonal matrices of order n is an orthogonal matrix of order n.*
– *The determinant of an orthogonal matrix is equal to $+1$ or -1.*
– *If c is an nth-order, nonzeros, real, normal (i.e. $\|c\|_2 = 1$) column vector, then there exists a real orthogonal matrix P having c as its first column.*

Lemma A.2 *Suppose that x and y are two n^{th}-order, nonzeros, real, normal column vectors. Then there exists a real orthogonal matrix P such that $y = Px$.*

Proof : By theorem A.1, there exist orthogonal matrix Q and R of order n such that x and y are the first columns of Q and R respectively. Let $P = RQ^T = RQ^{-1}$ be an orthogonal matrix. We have $R = PQ$. This implies that $y = Px$. □

Bibliography

Aharon, M., Elad, M. and Bruckstein, A. (2006). K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing 54(11),4311–4322.

D. Aloise, A. Deshpande, P. Hansen, P. Popat (2008). Np-Hardness of Euclidean Sum of-Squares Clustering. Mach Learn 75, 245–248.

Amaldi, E. and Kann, V. (1998). On the approximability of minimizing non zero variables or unsatisfied relations in linear systems. *Theoretical Computer Science, 209*, 237–260.

Bach F., Jenatton R., Mairal J. and Obozinski G. (2011). Optimization with Sparsity-Inducing Penalties. Foundations and Trends in Machine Learning 4(1), 1–106.

Bajwa, W., Haupt, J., Sayeed A., and Nowak, R. (2006). Compressive wireless sensing. *Proceedings of Fifth Int. Conf. on Information Processing in Sensor Networks*, pp.134–142.

Bagirov A.M. and Yearwood J. (2006). A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. European Journal of Operational Research 170, 578–596.

Baron, D., Wakin, M.B., Duarte, M.F., Sarvotham, S., and Baraniuk, R.G. (2009). Distributed compressed sensing. Technical Report ECE06-12, Electrical and Computer Engineering Department, Rice University.

Bennett, K.P., & Mangasarian, O. L. (1992). Robust linear programming discrimination of two linearly inseparable sets. Optimization Methods and Software, 1(1):23–34.

Ben-Tal A., El Ghaoui L., and Nemirovski A. (2009). Robust Optimization. Princeton University Press. Princeton, NJ.

Berkhin P. (2006). A Survey of Clustering Data Mining Techniques. Grouping Multidimensional Data, Springer Berlin Heidelberg, pp. 25-71.

Berry M., Browne M., Langville A., Pauca P., and Plemmons R. (2006). Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, *52*, 155–173.

Bhattacharyya C., Grate L.R., Jordan M.I., El Ghaoui L., and Mian I.S. (2004a). Robust sparse hyperplane classifier: application to uncertain molecular profiling data. *J Comput Biol, 11(6)*, 1073-1089.

Bhattacharyya C., Pannagadatta K.S., and Smola A.J. (2004). A second order cone programming formulation for classifying missing data. *in Advances in Neural Information Processing Systems 17 (NIPS17)*, MIT Press.

Bi J. and Zhang T. (2004). Support vector classification with input data uncertainty. *in Advances in Neural Information Processing Systems (NIPS17)*, MIT Press.

Bradley P.S., Mangasarian O.L., and Street W.N. (1997). Clustering via concave minimization in M.C. Mozer, M.I. Jordan, T. Petsche (Eds), NIPS 9, Cambridge, MA: MIT Press, pp. 368–374.

Bradley, P.S and Mangasarian, O.L. (1998). Feature Selection via concave minimization and support vector machines. *Proceeding of International Conference on Machina Learning ICML'98*.

Bradley, P.S., Mangasarian, O.L., and Rosen, J.B. (1998). Parsimonious Least Norm Approximation. *Comput. Optim. Appl., 11(1)*, 5–21.

Brusco M.J. (2006). A repetitive branch-and-bound procedure for minimum within-cluster sum of squares partitioning. Psychometrika 71, 347–363.

Cai D., He X., Wu X., and Han J. (2008). Nonnegative matrix factorization on manifold. *in Proc. IEEE Int. Conf. Data Mining*, pp. 63–72.

Candes, E.J. and Randall, P. (2006). Highly robust error correction by convex programming. *IEEE Trans. Inform. Theory, 54*, 2829–2840.

Candes, E.J and Tao, T. (2005). Decoding by linear programming. *IEEE Trans. Inf. Theory, 51(12)*, 4203–4215.

Candes, E.J., Wakin, M., and Boyd, S. (2008). Enhancing sparsity by reweighted-l_1 minimization. *J. Four. Anal. and Appli., 14*, 877–905.

Caramanis C., Mannor S., and Xu H. (2011). Optimization for Machine Learning. *in Robust optimization in machine learning*, MIT Press.

Carroll R., Stefanski L.A., Ruppert D., and Craincieanu C.M. (2006). Measurement error in nonlinear models. Chapman and Hall.

A. Chambolle, R. A. De Vore, Nam-Yong Lee, and B. J. Lucier. Nonlinear wavelet image processing: Variational problems, compression, and noise removal through wavelet shrinkage. IEEETrans. Image Process., 7(3):319–335, March 1998. ISSN 1057-7149.

Chan, A.B., Vasconcelos, N., and Lanckriet, R.G. (2007). Direct Convex Relaxations of Sparse SVM. *Proceeding ICML'07 Proceedings of the 24th international conference on Machine learning*, 145–153.

Chartrand, R. and Yin, W. (2008). Iteratively reweighted algorithms for compressive sensing. *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008*, pp. 3869–3872.

Chau M., Cheng R., Kao B., and Ng J. (2006). Uncertain data mining: An example in clustering location data. in: W.K. Ng, M. Kitsuregawa, J. Li (Eds.), PAKDD 2006, LNAI 3918, pp. 199–204.

Chaudhuria B.B. and Bhowmik P.R. (1998). An approach of clustering data with noisy or imprecise feature measurement, Pattern Recognition Letters 19(14), 1307–1317.

Chen Z. and Cichocki A. (2005). Nonnegative matrix factorization with temporal smoothness and/or spatial decorrelation constraints. [online] Available: http://www.bsp.brain.riken.jp/publications/2005/nips_nmf24-07.pdf

Chen, S., Donoho, D., and Saunders, M. (1998). Atomic decomposition by basis pursuit. SIAM Journal on Scientific Computing 20(1), 33–61.

Chen, X., Xu, F.M., and Ye, Y. (2010). Lower bound theory of nonzero entries in solutions of l2-lp minimization. *SIAM J. Sci. Comp., 32(5)*, 2832–2852.

Chu M., Diele F., Plemmons R. and Ragni S. (2004). Optimality, computation and interpretation of nonnegative matrix factorizations. *SIAM Journal on Matrix Analysis*, 4–8030.

Cichocki A. and Zdunek R. (2006). Multilayer nonnegative matrix factorization. *Electronics Letters*, *42(16)*, 947–948.

Cichocki A., Zdunek R., and Amari S.-I. (2007). Hierarchical als algorithms for nonnegative matrix and 3d tensor factorization. *in Lecture Notes in Computer Science 4666*, 169–176.

Cichocki A. and Phan A.-H. (2009). Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E92-A*, 708–721.

Cichocki A., Zdunek R., Phan A.-H., and Amari S.-I. (2009). Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation. John Wiley & Sons.

Cohen J.E. and Rothblum U.G. (1993). Nonnegative ranks, decompositions, and factorizations of nonnegative matrices. *Linear Algebra and its Applications*, *190*, 149–168.

Collobert, R., Sinz, F., Weston, J., and Bottou, L. (2006). Trading Convexity for Scalability. *Proceedings of the 23th International Conference on Machine Learning (ICML 2006)*, Pittsburgh, PA.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. Machine Learning 20(3):273–297.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B, 39:1–38, 1977.

Dhillon I. and Sra S. (2006). Generalized nonnegative matrix approximations with bregman divergences *In: Weiss Y. et al. (eds.), Advances in Neural Information Processing Systems 18, MIT Press, Cambridge, MA*, 283–290.

Ding C., Li T., and Jordan M.I. (2010). Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*, 45-55.

Elad, M. and Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. IEEE Trans. Image Process. 54(12), 3736–3745.

K. Engan, S. O. Aase, and J. H. Husoy. Method of optimal directions for frame design. In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 1999.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Amer. Stat. Ass., 96(456)*, 1348–1360.

Fawzi, A., Davies, M., and Frossard, P. (2014). Dictionary learning for fast classification based on soft-thresholding. submitted to *International Journal of Computer Vision*, http://arxiv.org/abs/1402.1973.

Franc V., Hlavac V., and Navara M. (2005). Sequential Coordinate-wise Algorithm for Non-negative Least Squares Problem. *In: Gagalowicz, A. and Philips, W. (eds.), CAIP 2005: Computer Analysis of Images and Patterns , LNCS 3691, Springer-Verlag, Berlin, Germany*, 407–414.

Fu, W.J. (1998). Penalized regression: the bridge versus the lasso. *J. Comp. Graph. Stat., 7*, 397–416.

Gasso, G., Rakotomamonjy, A. and Canu, S. (2009). Recovering sparse signals with a certain family of nonconvex penalties and dc programming. *IEEE Trans. Sign. Proc., 57*, 4686–4698.

Gillis N. and Glineur F. (2012). Accelerated Multiplicative Updates and Hierarchical ALS Algorithms for Nonnegative Matrix Factorization. *Neural Computation 24 (4)*, 1085–1105.

Gillis N. and Glineur F. (2008). Nonnegative Factorization and The Maximum Edge Biclique Problem. *CORE Discussion paper 2008/64.*

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. B., and Lander, E. S. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science, 286(5439),* 531-537.

Gong P. and Zhang C. (2012). Efficient Nonnegative Matrix Factorization via Projected Newton Method. *Pattern Recognition 45 (9)*, 3557–3565.

Gonzalez E.F. and Zhang Y. (2005). Accelerating the Lee-Seung algorithm for non-negative matrix factorization. *Tech Report*, Department of Computational and Applied Mathematics, Rice University.

Gordon, G. J., Jensen, R. V., Hsiao, L.-L., Gullans, S. R., Blumenstock, J. E., Ramaswamy, S., Richards, W. G., Sugarbaker, D. J., and Bueno, R. (2002). Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Res, 62(17)*, 4963–4957.

Gorodnitsky, I.F. and Rao, B.D. (1997). Sparse signal reconstructions from limited data using FOCUSS: A re-weighted minimum norm algorithm. *IEEE Trans. Signal Processing, 45*, 600–616.

Gribonval, R. and Nielsen, M. (2003). Sparse representation in union of bases. *IEEE Trans. on Information Theory, 49*, 3320–3325.

Guan, W. and Gray, A. (2013). Sparse high-dimensional fractional-norm support vector machine via DC programming. *Computational Statistics and Data Analysis, 67*, 136–148.

Guan N., Tao D., Lou Z., and Yuan B. (2012). NeNMF: An Optimal Gradient Method for Nonnegative Matrix Factorization. *IEEE Transactions on Signal Processing 60(6)*, 2882–2898.

Gullo F. and Tagarelli A. (2012). Uncertain Centroid based Partitional Clustering of Uncertain Data, Proceedings of the VLDB Endowment (ACM) 5(7), 610–621.

Guyon, I., Weston, J. Barnhill, S., and Vapnik, V.N. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning, 46(1–3)*, 389–422.

Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L.A. (2006). Feature extraction, foundations and applications. *Studies in Fuzziness and Soft Computing, 207*, Springer Berlin.

Hamdan H. and Govaert G. (2005). Mixture model clustering of uncertain data. in Proc. of IEEE ICFS Conference, pp. 879–884.

Hansen, P. and Jaumard, B. (1997). Cluster analysis and mathematical programming. Mathematical Programming, 79, 191–215.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). The elements of statistical learning. Springer Heidelberg, 2nd edition.

Heiler, M., Cremers, D., & Schnörr, C. (2001). Efficient feature subset selection for support vector machines. Technical Report TR-01-021, Comp. science series, Dept. of Mathematics and Computer Science, University of Mannheim.

Hiriart-Urruty J.B. and Lemarechal C. (1993). Convex Analysis and Minimization Algorithms. Springer-Verlag Berlin Heidelberg.

Ho N.-D. (2008). Nonnegative Matrix Factorization: Algorithms and Applications. *PhD Thesis*, University catholique de Louvain.

Howard W.E. (1966). Elementary matrix theory. Allyn and Bacon, Inc., Boston.

Hoyer P.O. (2002). Nonnegative sparse coding. *in Proc. IEEE Workshop on Neural Netw. Signal Process.*, pp. 557–565.

Hoyer P.O. (2004). Non-negative Matrix Factorization with Sparseness Constraints. *J. of Machine Learning Research*, *5*, 1457–1469.

Huang, J., Horowitz, J., and Ma, J. (2008). Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *Ann. Stat., 36*, 587–613.

Huang Y., Liu H., and Zhou S. (2014). Quadratic regularization projected Barzilai-Borwein method for nonnegative matrix factorization *Data Min Knowl Disc.* DOI: 10.1007/s10618-014-0390-x

Hubert L. and Arabie P. (1985). Comparing partitions. J CLASSIF. 2, 193–218.

Jain A.K., Murty M.N. and Flynn P.J. (1999). Data clustering: A review. ACM Computing Surveys 31(3), 264–323.

Kim H. and Park H. (2007). Sparse non-negative matrix factorizations via alternating non-negativityconstrained least squares for microarray data analysis. *Bioinformatics*, *23* , 1495–1502.

Kim H. and Park H. (2008). Nonnegative Matrix Factorization Based on Alternating Non-negativity-constrained Least Squares and the Active Set Method. *SIAM Journal on Matrix Analysis and Applications*, *30(2)*, 713-730.

Kim H. and Park H. (2011). Fast nonnegative matrix factorization: an active-set-like method and comparisons. *SIAM Journal on Scientific Computing (SISC)*, *33(6)*, 3261–3281.

Kim D., Sra S., and Dhillon I.S. (2007). Fast newton-type methods for the least squares non-negative matrix approximation problem. *in Proceedings of the 2007 SIAM International Conference on Data Mining.*

Knight, K. and Fu, W. (2000). Asymptotics for lasso-type estimators. *Ann. Stat., 28*, 1356–1378.

Krause, N. and Singer, Y. (2004). Leveraging the margin more carefully. *Proceedings of the 21st International Conference on Machine Learning ICML 2004*. Banff, Alberta, Canada.

Kreutz-Delgado K., Murray J.F., Rao B.D., Engan K., Lee T., and Sejnowski T.J. (2003). Dictionary learning algorithms for sparse representation. Neur. Comput., 15(2), 349–396.

Kuhn H.W. and Kuenne R.E. (1962). An Efficient Algorithm for the Numerical Solution of the Generalized Weber Problem in Spatial Economics. Journal of Regional Science 4, 21–34.

Kumar M. and Patel N.R. (2007). Clustering data with measurement errors. Computational Statistics & Data Analysis 51, 6084–6101.

Le, H.M., Le Thi H.A., and Nguyen, M.C. (2013). DCA based algorithms for feature selection in Semi-Supervised Support Vector Machines. *Machine Learning and Data Mining in Pattern Recognition*, Petra Perner (Ed), LNAI 7988, 528–542.

Le, H.M., Le Thi, H.A., Pham Dinh, T. and Huynh, V.N. (2013). Block clustering based on difference of convex functions (DC) programming and DC algorithms. *Neural Computation, 25(10)*, 2776–807.

Le, H.M., Le Thi, H.A., Nguyen, M.C. (2015). Sparse Semi-Supervised Support Vector Machines by DC Programming and DCA. Neurocomputing, 153 (4), 62–76.

M. Le Hoai and M. T. Ta. Dc programming and dca for solving minimum sum-of-squares clustering using weighted dissimilarity measures. Transaction Computational Collective Intelligence, 13:113–131, 2014.

H. A. Le Thi (web site). Dc programming and dca. http://www.lita.univ-lorraine.fr/~lethi/index.php/dca.html. Accessed in July 2015.

H. A. Le Thi. (1994). Analyse num´erique des algorithmes de l'optimisation DC. Approches locale et globale. Codes et simulations numériques en grande dimension. Applications. Thèse de doctorat, Université de Rouen.

Le Thi, H.A. (1997). Contribution à l'optimisation non convexe et l'optimisation globale: Théorie, Algorithmes et Applications. Habilitation à Diriger des Recherches, Université de Rouen.

H. A. Le Thi, T. Pham Dinh, and M. Le Dung. Exact penalty in d.c. programming. Vietnam Journal of Mathematics, 27(2):169–178, 1999.

Le Thi, H.A. (2000). An efficient algorithm for globally minimizing a quadratic function under convex quadratic constraints. *Mathematical Programming, 87(3)*, 401-426.

Le Thi, H.A. (2012). A new approximation for the ℓ_0-norm. Research Report LITA EA 3097, University of Lorraine.

Le Thi, H.A., Belghiti, T., and Pham Dinh, T. (2006). A new efficient algorithm based on DC programming and DCA for Clustering. *Journal of Global Optimization, 37*, 593–608.

Le Thi, H.A., Le H.M., Nguyen, V.V., and Pham Dinh, T. (2008). A dc programming approach for feature selection in support vector machines learning. *Journal of Advances in Data Analysis and Classification, 2*, 259–278.

Le Thi, H.A., LE, H.M., and Pham Dinh, T. (2007a). Optimization based DC programming and DCA for Hierarchical Clustering. *European Journal of Operational Research, 183(3)*, 1067–1085.

Le Thi, H.A., Nguyen, T.P., and Pham Dinh, T. (2007b). A continuous approach for solving the concave cost supply problem by combining DCA and B&B techniques. *European Journal of Operational Research, 183*, 1001–1012.

Le Thi H.A., Le Hoai M. and Pham Dinh T. (2007c). Fuzzy clustering based on nonconvex optimisation approaches using difference of convex (DC) functions algorithms. Journal of Advances in Data Analysis and Classification 2, 1–20.

Le Thi, H.A., Huynh, V.N., and Pham Dinh, T. (2009). Convergence Analysis of DC Algorithms for DC programming with subanalytic data. Research Report, National Institute for Applied Sciences, Rouen 2009 http://www.optimization-online.org/DB_HTML/2013/08/3996.html.

Le Thi, H.A., Nguyen, V.V., and Ouchani, S. (2009). Gene Selection for Cancer Classification Using DCA. *Journal of Fonctiers of Computer Science and Technology, 3(6)*, 62–72.

Le Thi, H.A. and Pham Dinh, T. (1997). Solving a class of linearly constrained indefinite quadratic problems by DC algorithms. *Journal of Global Optimization, 11(3)*, 253–285.

Le Thi, H.A. and Pham Dinh, T. (1998). A branch-and-bound method via D.C. optimization algorithm and ellipsoidal technique for box constrained nonconvex quadratic programming problems. *Journal of Global Optimization, 13*, 171–206.

H. A. Le Thi and T. Pham Dinh. Dc programming approach for solving the multidimensional scaling problem. Nonconvex Optimizations and Its Applications: Special Issue From Local to Global Optimization, pages 231–276, 2001.

Le Thi, H.A. and Pham Dinh, T. (2002). DC Programming: Theory, Algorithms and Applications. The State of the Art (28 pages). *Proceedings of The First International Workshop on Global Constrained Optimization and Constraint Satisfaction (Cocos' 02)*, Valbonne-Sophia Antipolis, France, October 2-4.

Le Thi, H.A. and Pham Dinh, T.(2003). Large Scale Molecular Optimization From Distance Matrices by a D.C. Optimization Approach. *SIAM Journal on Optimization, 4(1)*, 77–116.

Le Thi H.A. and 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.

Le Thi, H.A. and Pham Dinh, T. (2008). A continuous approach for the concave cost supply problem via DC Programming and DCA. *Discrete Applied Mathematics, 156*, 325–338.

Le Thi, H.A. and Pham Dinh, T. (2009). Minimum Sum-of-Squares Clustering by DC Programming and DCA In D.-S. Huang et al. (Eds.): ICIC 2009, LNAI 5755, pp. 327–340.

H. A. Le Thi, T. Pham Dinh, and V. N. Huynh. Exact penalty techniques in dc programming. Journal of Global Optimization, pages 1–27, 2011.

Le Thi, H.A., Pham Dinh, T. (2013). DC programming approaches for Distance Geometry problems. in "Distance Geometry: Theory, Methods and Applications", Mucherino, A; Lavor, C; Liberti, L.; Maculan, N. (Eds), Springer, 225–290.

Le Thi, H.A., Pham Dinh, T. and Nguyen Van, T. (2002). Combination between Local and Global methods for solving an Optimization problem over the Efficient set. *European Journal of Operational Research, 142*, 258–270.

Le Thi, H.A. and Moeini, M. (2012). Long-Short Portfolio Optimization Under Cardinality Constraints by Difference of Convex Functions Algorithm. *Journal of Optimization Theory & Applications*, DOI 10.1007/s10957-012-0197-0 October 2012, 27 pages.

Le Thi, H.A. & Nguyen M.C. (2013). Efficient algorithms for Feature Selection in Multi-class Support Vector Machine. *Advanced Computational Methods for Knowledge Engineering*, Studies in Computational Intelligence 479, Springer.

Le Thi H.A., Huynh, V.N., and Pham Dinh, T. (2012a). Exact Penalty and Error Bounds in DC Programming. *Journal ofGlobal Optimization, 52(3)*, 509–535.

Le Thi H.A., Moeini, M., Pham Dinh, T., and Joaquim, J. (2012b). A DC programming approach for solving the symmetric eigenvalue complementarity problem. *Computational Optimization and Applications, 51(3)*, 1097–1117.

Le Thi H.A., Tran, D.Q., and Pham Dinh, T. (2012c). A DC programming approach for a class of bilevel programming problems and its application in Portfolio Selection. *Numerical Algebra, Control and Optimization (NACO), 1*, 167–185.

Le Thi, H.A., Le, H.M., Pham Dinh, T. & Huynh, V.N. (2013a). Binary classification via spherical separator by DC programming and DCA. *Journal of Global Optimization*, 56:4, 1393–1407.

Le Thi, H.A., Nguyen, T.B.T & Le H.M. (2013b). Sparse signal recovery by Difference of Convex functions Algorithms. *Lecture Notes in Computer Science*, ISBN 978-3-642-36542-3, 387–397.

Le Thi, H.A., Vo, X.T.. & Pham Dinh, T. (2013c). Robust Feature Selection for SVMs under Uncertain Data. *in Advances in Data Mining. Applications and Theoretical Aspects, LNCS 7987*, pp. 151-165.

Le Thi, H.A., Le, H.M. & Pham Dinh, T. (2014a). Feature Selection in machine learning: an exact penalty approach using a Difference of Convex function Algorithm. *Machine Learning (published online 04.07.14)*, DOI: 10.1007/s10994-014-5455-y.

Le Thi H.A., Pham Dinh, T., and Vo X.T. (2014b). DC programming and DCA for Nonnegative Matrix Factorization. *In D. Hwang et al. (Eds): ICCCI 2014, LNAI 8733*, pp. 573–582.

Le Thi, H.A., Le, H.M., Pham Dinh, T. & Lauer, F. (2014c). A DC programming algorithm for switched linear regression. *IEEE Transactions on Automatic Control*, 99, forthcoming.

Le Thi, H.A., Huynh, V.N. & Pham Dinh, T. (2014d). DC Programming and DCA for solving general DC Programs. Proceedings of 2nd International Conference on Computer Science, Applied Mathematics and Applications(ICCSAMA 2014), in Press, (35 pages).

Le Thi, H.A., Le, H.M., Pham Dinh, T. (2014e). New and efficient DCA based algorithms for minimum sum-of-squares clustering. Pattern Recognition 47(1), 388–401.

Le Thi, H.A., Vo, X.T., Pham Dinh, T. (2014f). Feature Selection for linear SVMs under Uncertain Data: Robust optimization based on Difference of Convex functions Algorithms. Neural Networks 59, 36–50.

Le Thi, H.A., Nguyen, M.C., Pham Dinh, T. (2014g). A DC programming approach for finding Communities in networks. Neural Computation, 26(12), 2827–2854.

Le Thi, H.A., Pham Dinh, T., Le, H.M., and Vo, X.T. (2015). DC approximation approaches for sparse optimization. Eur. J. Oper. Res. 244(1), 26–46.

Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2007). Efficient sparse coding algorithms. Advances in Neural Information Processing Systems, 19, 801–808.

Lee S.D., Kao B., and Cheng R. (2007). Reducing UK-means to K-means. in Proceedings of the Seventh IEEE International Conference on Data Mining Workshops(ICDMW '07), pp. 483–488.

Lee D.D. & Seung H.S. (1997). Unsupervised learning by convex and conic coding. *Proceedings of the Conference on Neural Information Processing Systems*, *9*, 515–521.

Lee D.D. and Seung H.S. (1999). Learning the Parts of Objects by Nonnegative Matrix Factorization. *Nature*, *401*, 788–791.

Lee D.D. and Seung H.S. (2001). Algorithms for Non-negetive matrix factorization. *Advances in Neural Information Processing Systems*, *13*, 556–562.

Li S.Z., Hou X., Zhang H. and Cheng Q. (2001). Learning spatially localized parts-based representations. In Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Vol. I, pp. 207–212, Hawaii, USA.

Lin C.-J. (2007a). On the convergence of multiplicative update algorithm for non-negetive matrix factorization. *IEEE Transactions on Neural Networks.*

Lin C.-J. (2007b). Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, *19*, 2756–2779.

Liu, Y. & Shen, X. (2006a). Multicategoryψ-Learning. *Journal of the American Statistical Association*, 101, 500–509.

Liu, Y. & Zheng, Y.F. (2006b). FS SFS: A novel feature selection method for support vector machines. *Pattern Recognition, 39(7)*, 1333–1345.

Liu Y. and Wu Y. (2007). Variable selection via a combination of the L0 and L1 penalties. *J Comput Graph Stat, 16(4)*, 782–798.

Liu W., Zheng N. and Lu X. (2003). Non-negative matrix factorization for visual coding. IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03), pp. III - 293–6.

MacQueen J.B. (1967). Some methods for classification and analysis of multivariate observations. in Proc. Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297.

Mairal, J., Elad, M., and Sapiro, G. (2008). Sparse representation for color image restoration. IEEE Transactions on Image Processing, 17(1):53–69.

Mairal, J., Bach, F., Ponce, J. and Sapiro, G. (2010). Online learning for matrix factorization and sparse coding. Journal of Machine Learning Research 11,19–60.

Mairal, J., Bach, F., and Ponce, J. (2014). Sparse Modeling for Image and Vision Processing. Foundations and Trends in Computer Graphics and Vision 8(2-3), 85–283.

Mallat, S. & Zhang, Z. (1993). Matching pursuit in a time-frequency dictionary. *IEEE Trans. Signal Processing*, 41(12), 3397–3415.

Mallat, S. (1999). A wavelet tour of signal processing (second edition). Academic Press, New York.

Mangasarian, O.L. (1996). Machine learning via polyhedral concave minimization. in "Applied Mathematics and Parallel Computing – Festschrift for Klaus Ritter", H. Fischer, B. Riedmueller, S. Schaeffler, editors, Physica-Verlag, Germany, 175—188.

Michelot C. (1986). A finite algorithm for finding the projection of a point onto the canonical simiplex of Rn. *J. Optim. Theory Appl.*, *50(1)*, 195–200.

Mohri, M.,& Medina, A.M. (2014). Learning Theory and Algorithms for Revenue Optimization in Second-Price Auctions with Reserve. *Proceeding ICML'14 Proceedings of the 31th international conference on Machine learning*,

Natarajan, B.K. (1995). Sparse approximate solutions to linear systems. *SIAM J. Comp.*, 24, 227–234.

Neumann, J., Schnörr G., Steidl, G. (2005). Combined SVM-based feature selection and classification. *Machine Learning*, 61(1-3), 129–150.

Niu, Y.S., Pham Dinh, T., Le Thi H.A. & Judice, J. (2012). Efficient DC Programming Approaches for Asymmetric Eigenvalue Complementarity Problem. *Software Optimization Methods and Software, 28(4).*

Ngai W., Kao B., Chui C., Cheng R., Chau M. and Yip K.Y. (2006). Efficient Clustering of Uncertain Data. in Proceedings of the Sixth International Conference on Data Mining (ICDM'06), pp. 436–445.

Nguyen T.B.T., Le Thi H.A., Le H.M., Vo X.T. (2015). DC approximation approach for L0–minimization in compressed sensing. in H.A. Le Thi et al. (Eds.), ICCSAMA 2015: Proceedings of 3rd International Conference on Computer Science, Applied Mathematics and Applications, AISC 358, pp. 37–48, Springer 2015.

Olshausen, B.A. and Field, D.J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature, 381:607–609.

Olshausen, B.A. and Field, D.J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? Vision Research 37,3311–3325.

Ong, C.S & Le Thi H.A. (2013). Learning sparse classifiers with Difference of Convex functions Algorithms. *Optimization Methods and Software*, 28:4, 830–854.

Paatero P. and Tapper U. (1994). Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, *5*, 111–126.

Pant R., Trafalis T.B., and Barker K. (2011). Support Vector Machine Classification of Uncertain and Imbalanced Data using Robust Optimization. *in Proceedings of the 15th WSEAS international conference on Computers*, 369–374.

Pati, Y.C., Rezaifar, R. & Krishnaprasa, P.S. (1993). Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. *27th Asilomar Conf. on Signals, Systems and Comput.*, Nov. 1993.

Pauca V.P., Piper J., and Plemmons R.J. (2006). Nonnegative Matrix Factorization for Spectral Data Analysis. *Linear Algebra and its Applications*, *416*, 29–47.

Peleg, D. and Meir, R. (2008). A bilinear formulation for vector sparsity optimization. *Signal Processing, 8(2)*, 375–389.

J. Peng, Y. Xiay (2005). A cutting algorithm for the minimum sum-of-squared error clustering. in: Proceedings of the 2005 SIAM International Conference on Data Mining.

Pham Dinh T. (1986). Algorithms for solving a class of non convex optimization problems. Methods of subgradients, volume 129 of North-Holland Mathematics Studies. Elsevier Science Publishers.

Pham Dinh T. and Le Thi H.A. (1997). Convex analysis approach to DC programming: Theory, algorithms and applications. *Acta Math. Vietnamica*, *22(1)*, 289–357.

Pham Dinh T. & Le Thi H.A. (1998). A DC optimization algorithms for solving the trust-region subproblem. *SIAM J Optim*, *8(2)*, 476–505.

Pham Dinh, T., Nguyen Canh, N. & Le Thi, H.A. (2010). An efficient combination of DCA and B&B using DC/SDP relaxation for globally solving binary quadratic programs. *Journal of Global Optimization*, 48:4, 595–632.

Pham Dinh, T. & Le Thi, H.A. (2014). Recent Advances in DC Programming and DCA. *Transactions on Computational Collective Intelligence*, 8342, 1-37.

D. N. Phan, M. C. Nguyen, and H. A. Le Thi. A dc programming approach for sparse linear discriminant analysis. In Advanced Computational Methods for Knowledge Engineering, volume 282 of Advances in Intelligent Systems and Computing, pages 65–74. 2014. ISBN 978-3-319-06568-7.

M. Protter and M. Elad. (2009). Image sequence denoising via sparse and redundant representations. IEEE Transactions on Image Processing, 18(1):27–35.

Rao, B.D. & Kreutz-Delgado, K. (1999). An affine scaling methodology for best basis selection. *IEEE Trans. Signal Processing*, 47, 87–200.

Rao, B.D., Engan, K., Cotter, S.F., Palmer. J., & KreutzDelgado, K. (2003). Subset selection in noise based on diversity measure minimization. *IEEE Trans. Signal Processing*, 51(3), 760–770.

Rinaldi, F. (2009). Mathematical Programming Methods for minimizing the zero norm over polyhedral sets. PhD thesis in Operations Research at Sapienza University of Rome.

Rinaldi, F., Schoen, F. & Sciandrone, M. (2010). Concave programming for minimizing the zero-norm over polyhedral sets. *Comput. Optim. Appl.*, 46(3), 467–486.

Rockafellar, R.T (1970). Convex Analysis. Princeton: Princeton University.

Rubinstein R., Zibulevsky M., and Elad M. (2010). Double Sparsity: Learning Sparse Dictionaries for Sparse Signal Approximation. IEEE TRANSACTIONS ON SIGNAL PROCESSING, 58(3), 1553–1564.

Sandler R. and Lindenbaum M. (2011). Nonnegative Matrix Factorization with Earth Mover's Distance Metric for Image Analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence, 33(8)*, 1590–1602.

Schmidt, M., Fung, G. & Rosales, G. (2007). Fast Optimization Methods for L1 Regularization: A Comparative Study and Two New Approaches. *Proceedings of Machine Learning: ECML 2007*, Lecture Notes in Computer Science, 4701, 286–297.

Shahnaz, F., Berry, M.W., Langville, A.N., Pauca, V.P., & Plemmons, R.J. (2006). Document clustering using nonnegative matrix factorization. *Information Processing and Management*, *42*, 373–386.

Sherali H.D. and Desai J. (2005). A global optimization RLT-based approach for solving the hard clustering problem. Journal of Global Optimization 32, 281–306.

Shivaswamy P.K., Bhattacharyya C., and Smola A.J. (2006). Second order cone programming approaches for handling missing and uncertain data. *J Mach Learn Res, 7*, 1238–1314.

Skretting, K. and Engan, K. (2010). Recursive least squares dictionary learning algorithm. Signal Processing, IEEE Transactions on 58(4),2121–2130.

Sriperumbudur, B.K., Torres, D.A. & Lanckriet, R.G. (2007). Sparse eigen methods by D.C. programming. *Proceeding ICML '07, Proceedings of the 24th international conference on Machine learning*, 831-838.

Takhar, D., Laska, J.N., Wakin, M.B., Duarte, M.F., Baron, D., Sarvotham, S., Kelly, K.F. & Baraniuk, R.G. (2006). A New Compressive Imaging Camera Architecture using Optical-Domain Compression. *Computational Imaging IV at IS&T/SPIE Electronic Imaging*, San Jose, California, January 2006.

Tan, M., Wang, L. & Tsang, I.W. (2010). Learning sparse svm for feature selection on very high dimensional datasets. *ICML 2010*.

Thiao, M., Pham Dinh, T. & Le Thi, H.A. (2008). DC Programming approach for solving a class of Nonconvex Programs dealing with zero-norm. *Modelling, Computation and Optimization in Information Systems ans Management Science,* CCIS 14, 348–357, Springer-Verlag.

Thiao M., Pham Dinh T., & Le Thi H.A. (2010). A DC programming approach for Sparse Eigenvalue Problem. *Proceedings of the 27th International Conference on Machine Learning, ICML 2010*, 1063–1070.

Thurau C., Kersting K., Wahabzada M., & Bauckhage C. (2011). Convex non-negative matrix factorization for massive datasets. *Knowl Inf Syst*, *29*,457–478.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Roy. Stat. Soc.*, 46, 431–439.

J. F. Toland. Direct calculation of the information matrix via the EM algorithm. Journal of Mathematical Analysis and Applications, 66:399–415, 1978.

Trombettoni, G., Araya, I., Neveu, B., Chabert, G. (2011). Inner regions and interval linearizations for global optimization. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, 99–104.

J. B. H. Urruty. Generalized differentiability duality and optimization for problem dealing with differences of convex functions. Lecture Notes in Economics and Mathematical Systems, volume, 256:260–277, 1986.

Van Loan C.F. (2000). The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, *123(1-2)*, 85–100.

Vavasis S.A. (2009). On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, *20*, 1364–1377.

Vinod H.D. (1969). Integer programming and the theory of grouping. Journal of the American Statistical Association 64, 506–519.

Weston, J., Elisseeff, Scholkopf, A.B. & Tipping, M. (2003). Use of the Zero-Norm with Linear Models and Kernel Methods. *Journal of Machine Learning Research*, 3, 1439–1461.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T. & Vapnik, V. (2001). Feature selection for SVMs. In Neural Information Processing Systems, Cambridge, MA, 2001. MIT Press.

Wu Y. and Liu Y. (2007). Robust truncated-hinge-loss support vector machines. *J Am Stat Assoc, 102(479)*, 974–893.

Wu K.-L. and Yang M.-S. (2002). Alternative c-means clustering algorithms. Pattern Recognition 35(10), 2267–2278.

Xavier A.E. and Xavier V.L. (2011). Solving the minimum sum-of-squares clustering problem by hyperbolic smoothing and partition into boundary and gravitational regions. Pattern Recognition 44, 70–77.

Liming Yang and Laisheng Wang. A class of semi-supervised support vector machines by dc programming. Adv. Data Analysis and Classification, 7(4):417–433, 2013.

J. Yang, J. Wright, T. S. Huang, and Y. Ma. (2010). Image super-resolution via sparse representation. IEEE Transactions on Image Processing, 19(11):2861–2873.

Alan L. Yuille and Anand Rangarajan. The concave-convex procedure. Neural Computation, 15(4):915–936, 2003.

Zhang C.H. (2010). Nearly unbiased variable selection under minimax concave penalty. *Ann Stat 38(2)*, 894–942.

Zhang, T. (2009). Some sharp performance bounds for least squares regression with regularization. *Ann. Statist.*, 37, 2109–2144.

Zhang, H.H, Ahn, J., Lin, X. & Park, C. (2006). Gene selection using support vector machines with non-convex penalty. *Bioinformatics*, 2(1), 88–95.

Zhu, J., Rosset, S., Hastie, T. & Tibshirani, R. (2004). 1-norm support vector machines. In S. Thrun, L.Saul, & B. Scholkopf (Eds.), *Adv. neur. inf. proc. sys.*, 16 Cambridge, MA: MIT Press.

Zou, H. (2006). The adaptive lasso and its oracle properties. *J. Amer. Stat. Ass.*, 101, 1418–1429.

Zou, H. & Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *Ann. Statist.*, 36(4), 1509–1533.