Advances in Intelligent Systems and Computing 245

Van-Nam Huynh · Thierry Denœux
Dang Hung Tran · Anh Cuong Le
Son Bao Pham *Editors*

Knowledge and Systems Engineering

Proceedings of the Fifth International Conference KSE 2013, Volume 2

Springer

Advances in Intelligent Systems and Computing

Volume 245

Series Editor

Janusz Kacprzyk, Warsaw, Poland

For further volumes:
http://www.springer.com/series/11156

Van-Nam Huynh · Thierry Denœux
Dang Hung Tran · Anh Cuong Le
Son Bao Pham
Editors

Knowledge and Systems Engineering

Proceedings of the Fifth International Conference KSE 2013, Volume 2

Springer

Editors
Van-Nam Huynh
School of Knowledge Science
Japan Advanced Institute of Science and Technology
Ishikawa
Japan

Thierry Denœux
Universite de Technologie de Compiegne
Compiegne Cedex
France

Dang Hung Tran
Faculty of Information Technology
Hanoi National University of Education
Hanoi
Vietnam

Anh Cuong Le
Faculty of Information Technology
University of Engineering and Technology - VNU Hanoi
Hanoi
Vietnam

Son Bao Pham
Faculty of Information Technology
University of Engineering and Technology - VNU Hanoi
Hanoi
Vietnam

ISSN 2194-5357 ISSN 2194-5365 (electronic)
ISBN 978-3-319-02820-0 ISBN 978-3-319-02821-7 (eBook)
DOI 10.1007/978-3-319-02821-7
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013950935

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains papers presented at the Fifth International Conference on Knowledge and Systems Engineering (KSE 2013), which was held in Hanoi, Vietnam, during 17–19 October, 2013. The conference was jointly organized by Hanoi National University of Education and the University of Engineering and Technology, Vietnam National University. The principal aim of KSE Conference is to bring together researchers, academics, practitioners and students in order to not only share research results and practical applications but also to foster collaboration in research and education in Knowledge and Systems Engineering.

This year we received a total of 124 submissions. Each of which was peer reviewed by at least two members of the Program Committee. Finally, 68 papers were chosen for presentation at KSE 2013 and publication in the proceedings. Besides the main track, the conference featured six special sessions focusing on specific topics of interest as well as included one workshop, two tutorials and three invited speeches. The kind cooperation of Yasuo Kudo, Tetsuya Murai, Yasunori Endo, Sadaaki Miyamoto, Akira Shimazu, Minh L. Nguyen, Tzung-Pei Hong, Bay Vo, Bac H. Le, Benjamin Quost, Sébastien Destercke, Marie-Hélène Abel, Claude Moulin, Marie-Christine Ho Ba Tho, Sabine Bensamoun, Tien-Tuan Dao, Lam Thu Bui and Tran Dinh Khang in organizing these special sessions and workshop is highly appreciated.

As a follow-up of the Conference, two special issues of the Journal of *Data & Knowledge Engineering* and *International Journal of Approximate Reasoning* will be organized to publish a small number of extended papers selected from the Conference as well as other relevant contributions received in response to subsequent calls. These journal submissions will go through a fresh round of reviews in accordance with the journals' guidelines.

We would like to express our appreciation to all the members of the Program Committee for their support and cooperation in this publication. We would also like to thank Janusz Kacprzyk (Series Editor) and Thomas Ditzinger (Senior Editor, Engineering/Applied Sciences) for their support and cooperation in this publication.

Last, but not the least, we wish to thank all the authors and participants for their contributions and fruitful discussions that made this conference a success.

Hanoi, Vietnam
October 2013

Van-Nam Huynh
Thierry Denœux
Dang Hung Tran
Anh Cuong Le
Son Bao Pham

Organization

Honorary Chairs

Van Minh Nguyen – Hanoi National University of Education, Vietnam
Ngoc Binh Nguyen – VNU University of Engineering and Technology, Vietnam

General Chairs

Cam Ha Ho – Hanoi National University of Education, Vietnam
Anh Cuong Le – VNU University of Engineering and Technology, Vietnam

Program Chairs

Van-Nam Huynh – Japan Advanced Institute of Science and Technology, Japan
Thierry Denœux – Université de Technologie de Compiègne, France
Dang Hung Tran – Hanoi National University of Education, Vietnam

Program Committee

Akira Shimazu, Japan
Azeddine Beghdadi, France
Son Bao Pham, Vietnam
Benjamin Quost, France
Bernadette Bouchon-Meunier, France
Binh Thanh Huynh, Vietnam
Bay Vo, Vietnam
Cao H, Tru, Vietnam
Churn-Jung Liau, Taiwan
Dinh Dien, Vietnam
Claude Moulin, France
Cuong Nguyen, Vietnam
Dritan Nace, France
Duc Tran, USA
Duc Dung Nguyen, Vietnam
Enrique Herrera-Viedma, Spain
Gabriele Kern-Isberner, Germany
Hiromitsu Hattori, Japan
Hoang Truong, Vietnam
Hung V. Dang, Vietnam
Hung Son Nguyen, Poland
Jean Daniel Zucker, France

Jérôme Lang, France
Jing Liu, China
Jiuyong Li, Australia
Jonathan Lawry, UK
Kenji Satou, Japan
Lam T. Bui, Vietnam
Bac H. Le, Vietnam
Loannis Parissis, France
Marie-Helene Abel, France
Martin Steffen, Norway
Masahiro Inuiguchi, Japan
Michel Riveill, France
Mina Ryoke, Japan
Minh-Dung Phan, Thailand
Mitsuru Ikeda, Japan
Minh L. Nguyen, Japan
Noboru Takagi, Japan
Peter Whigham, New Zealand
Phayung Meesad, Thailand
Quang-Huy Nguyen, France
Quang Uy Nguyen, Ireland
Sabine Bensamoun, France
Sadaaki Miyamoto, Japan
Serge Stinckwich, France
Sébastien Destercke, France
Si Quang Le, UK
Son Doan, USA
Tien-Tuan Dao, France
Tetsuya Murai, Japan
Thanh Binh Nguyen, Vietnam
Thanh Tri Nguyen, Vietnam
Thanh-Thuy Nguyen, Vietnam
The Duy Bui, Vietnam
The Loc Nguyen, Vietnam
Thomas Huynh, USA
Tho Hoan Pham, Vietnam
Thepchai Supnithi, Thailand
The Dung Luong, Vietnam
Tran Dinh Khang, Vietnam
Tsutomu Fujinami, Japan
Tzung-Pei Hong, Taiwan
Vladik Kreinovich, USA
Xiaoshan Li, Macau
Xuan Hoai Nguyen, Vietnam
Xuan-Hieu Phan, Vietnam
Yasuo Kudo, Japan

Contents

Part I
Workshop Invited Talks

The Place of Causal Analysis in the Analysis of Simulation Data

Ladislav Hluch

Abstract. This talk briefly reviews selected basic concepts and principles of structural approach to causal analysis, and outlines how they could be harnessed for analyzing and summarizing the data from simulations of complex dynamic systems, and for exploratory analysis of simulation models through machine learning. We illustrate the proposed method in the context of human behaviour modeling on a sample scenario from the EDA project A-0938-RT-GC EUSAS. The method revolves around the twin concepts of a causal partition of a variable of interest, and a causal summary of a simulation run. We broadly define a causal summary as a partition of the significant values of the analyzed variables (in our case the simulated motives fear and anger of human beings) into separate contributions by various causing factors, such as social influence or external events. We demonstrate that such causal summaries can be processed by machine learning techniques (e.g. clustering and classification) and facilitate meaningful interpretations of the emergent behaviours of complex agent-based models.

Acknowledgement. This work was supported by the European Defence Agency project A-0938-RT-GC EUSAS, by the Slovak Research and Development Agency under the contract No. APVV-0233-10, and by the project VEGA No. 2/0054/12.

Ladislav Hluch
Institute of Informatics, Slovak Academy of Sciences

V.-N. Huynh et al. (eds.), *Knowledge and Systems Engineering, Volume 2*,
Advances in Intelligent Systems and Computing 245,
DOI: 10.1007/978-3-319-02821-7_1,

Evolutionary Computation in the Real World: Successes and Challenges

Graham Kendall

Abstract. Evolutionary Computation has the potential to address many problems which may seem intractable to some of the methodologies that are available today. After briefly describing what evolutionary computation is (and what it is not), I will outline some of the success stories before moving onto the challenges we face in having these algorithms adopted by the industrial community at large.Some of the areas I will draw upon include Checkers and Chess, Scheduling and Timetabling, Hyper-heuristics and Meta-heuristics, as well some other problems drawn from the Operational Research literature.

Graham Kendall
The University of Nottingham Malaysia Campus,
Selangor Darul Ehsan, Malaysia

V.-N. Huynh et al. (eds.), *Knowledge and Systems Engineering, Volume 2*,
Advances in Intelligent Systems and Computing 245,
DOI: 10.1007/978-3-319-02821-7_2,

Part II
KSE 2013 Special Sessions and Workshop

A Method of Two-Stage Clustering with Constraints Using Agglomerative Hierarchical Algorithm and One-Pass *k*-Means++

Yusuke Tamura, Nobuhiro Obara, and Sadaaki Miyamoto

Abstract. The aim of this paper is to propose a two-stage method of clustering in which the first stage uses one-pass *k*-means++ and the second stage uses an agglomerative hierarchical algorithm. This method outperforms a foregoing two-stage algorithm by replacing the ordinary one-pass *k*-means by one-pass *k*-means++ in the first stage. Pairwise constraints are also taken into consideration in order to improve its performance. Effectiveness of the proposed method is shown by numerical examples.

1 Introduction

Clustering techniques [7, 9] has recently been becoming more and more popular, as huge data on the web should be handled. Such data are frequently unclassified in contrast to those in traditional pattern classification problems where most data have classification labels [5]. Not only methods of unsupervised classification but also those of semi-supervised classification [6] and constrained clustering [2, 3] have been developed to handle such data.

Clustering techniques in general can be divided into two categories of hierarchical clustering and non-hierarchical clustering. Best-known methods in the first category are agglomerative hierarchical clustering, while that in the second category is the method of *k*-means [8]. Most methods of semi-supervised classification and constrained clustering are non-hierarchical, but agglomerative hierarchical clustering is at least as useful as non-hierarchical techniques in various applications. A drawback in agglomerative hierarchical clustering is that larger computation is needed when compared with simple non-hierarchical methods such as the *k*-means.

Yusuke Tamura · Nobuhiro Obara
Master's Program in Risk Engineering, University of Tsukuba, Ibaraki 305-8573, Japan

Sadaaki Miyamoto
Department of Risk Engineering, University of Tsukuba, Ibaraki 305-8573, Japan
e-mail: miyamoto@risk.tsukuba.ac.jp

V.-N. Huynh et al. (eds.), *Knowledge and Systems Engineering, Volume 2*,
Advances in Intelligent Systems and Computing 245,
DOI: 10.1007/978-3-319-02821-7_3,

Here is a question: how can we develop a method of agglomerative hierarchical clustering that can handle large amount of data with semi-supervision or constraints? We have partly answered this question by developing a method of agglomerative hierarchical clustering in which pairwise constraints can be handled using penalties in the agglomerative clustering algorithm [11]. Moreover a two-stage clustering has been suggested in which the first-stage uses k-means and the second stage is a class of agglomerative hierarchical clustering [10]. However, performance of the two-stage algorithm should still be improved.

In this paper we introduce a variation of the algorithm presented in [10]. In short, we use one-pass k-means++[1] in the first stage and show an improved two stage clustering algorithm with pairwise constraints. Several numerical examples are shown to observe the usefulness of the proposed method.

The rest of this paper is organized as follows. Section 2 provides preliminaries, then Section 3 shows the two-stage algorithm herein. Section 4 shows effectiveness and efficiency of the proposed algorithm using a number of numerical examples. Finally, Section 5 concludes the paper.

2 Preliminary Consideration

We begin with notations. Let the set of objects be $X = \{x_1, \cdots, x_n\}$. Each object x_k is a point in the p-dimensional Euclidean space $\boldsymbol{R}^p$: $x_i = (x_{i1}, \cdots, x_{ip}) \in \boldsymbol{R}^p$

Clusters are denoted by $G_1, G_2, \cdots, G_C$, and the collection of clusters is given by $\mathscr{G} = \{G_1, G_2, \cdots, G_C\}$. Clusters are partition of X:

$$\bigcup_{i=1}^{C} G_i = X,\ G_i \cap G_j = \emptyset\ (i \neq j) \tag{1}$$

2.1 Agglomerative Hierarchical Clustering

Assume that $d(G, G')$ is a dissimilarity measure defined between two clusters; calculation formula of $d(G, G')$ will be given after the following general algorithm of agglomerative hierarchical clustering, abbreviated **AHC** in which **AHC 1** and **AHC 2** are the steps of this algorithm.

AHC1: Let initial clusters given by objects.
$G_i = \{x_i\}, (i = 1, \cdots, n)$
$C = n$, (C is the number of clusters and n is the number of objects)
Calculate $d(G, G')$ for all pairs $G, G' \in \mathscr{G} = \{G_1, G_2, \cdots, G_C\}$.
AHC2: Merge the pair of clusters of minimum dissimilarity:

$$d(G_q, G_r) = \arg \min_{G, G' \in \mathscr{G}} d(G, G') \tag{2}$$

Add $\hat{G} = G_q \cup G_r$ to $\mathscr{G}$ and remove G_q, G_r from $\mathscr{G}$.
$C = C - 1$.
If $C = 1$, then output the process of merge of clusters as a dendrogram and stop.
AHC3: Calculate $d(\hat{G}, G')$ for $\hat{G}$ and all other $G' \in \mathscr{G}$. go to **AHC2**.

We assume that the dissimilarity between two objects is given by the squared Euclidean distance:

$$d(x_k, x_l) = \|x_k - x_l\|^2 = \sum_{j=1}^{p} (x_{kj} - x_{lj})^2.$$

Moreover the centroid method is used here, which calculate $d(\hat{G}, G')$ as follows.

Centroid method:

Let $M(G)$ be the centroid (the center of gravity) of G:

$$M(G) = (M_1(G), \cdots, M_p(G))^T,$$

where

$$M_j(G) = \frac{1}{|G|} \sum_{x_k \in G} x_{kj}, \ (j = 1, \cdots, p) \tag{3}$$

and let

$$d(G, G') = \|M(G) - M(G')\|^2 \tag{4}$$

2.2 *k*-Means and *k*-Means++

The method of k-means repeats the calculation of centroids of clusters and nearest centroid allocation of each object until convergence [4]. It has been known that the result is strongly dependent on the choice of initial values.

The method of k-means++ [1] improves such dependence on initial clusters by using probabilistic selection of initial centers. To describe k-means++, let v_i be the i-th cluster center and $D(x)$ be the Euclidean distance between object x and the already selected centers nearest to x. The algorithm is as follows [1].

1a: Let the first cluster center v_1 be a randomly selected object from X.
1b: Let a new center v_i be selected from X with probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$.
1c: Repeat **1b** until k cluster centers are selected.
2: Carry out the ordinary k-means algorithm.

Step **1b** is called "D^2 weighting", whereby a new cluster center that have larger distance from already selected centers will have larger probability to be selected.

2.3 Pairwise Constraints

Two sets ML and CL of constraints are used in constrained clustering [2, 3]. A set $ML = \{(x_i, x_j)\} \subset X \times X$ consists of *must-link* pairs so that x_i and x_j should be in a same cluster, while another set $CL = \{(x_k, x_l)\} \subset X \times X$ consists of *cannot-link* pairs so that x_i and x_j should be in different clusters. ML and SL are assumed to be symmetric in the sense that if $(x_i, x_j) \in ML$ then $(x_j, x_i) \in ML$, and if $(x_k, x_l) \in CL$ then $(x_l, x_k) \in CL$.

Note that ML is regarded as an undirected graph in which nodes are objects appeared in ML, and an undirected edge is $(x_i, x_j) \in ML$.

Introduction of the pairwise constraints to k-means has been done by Wagstaff et al. [12]. The developed algorithm is called COP k-means.

3 A Two-Stage Algorithm

A two-stage algorithm of clustering for large-scale data is proposed, in which the first stage uses one-pass k-means++ to have a medium number of cluster centers and the second stage uses the centroid method. Pairwise constraints are taken into account in both stages.

3.1 One-Pass COP k-Means++

One pass k-means implies that the algorithm does not iterate the calculation of the centroid and the nearest center allocation: it first generates initial cluster centers, then each object is allocated to the cluster of the nearest center. After the allocation, new cluster centers are calculated as the centroids (3). Then the algorithm stops without further iteration.

Pairwise Constraints in the First Stage

Moreover the one-pass algorithm must take pairwise constraints into account. ML (must-link) is handled as the initial set of objects, as ML defines a connected components of a graph. Then the centroid of the connected components is used instead of the objects in the components. On the other hand, CL (cannot-link) is handled in the algorithm.

Thus the algorithm in the first stage is called one-pass COP k-means++, which is as follows.

One-Pass COP k-means++ in the first stage

1: Let initial clusters be generated by using the D^2 weighting.
2: Each object $x \in X$ is allocated to the cluster of the nearest center that does not break the given pairwise constraints CL. If x cannot be allocated to any cluster due to the constraints, stop with flag **FAILURE**.
3: Cluster centers are updated as the centroids (3).

4: Stop. (Note that this step is replaced by 'repeat steps 2 and 3 until convergence' if the one-pass condition is removed.)
End of One-Pass COP k-means++.

3.2 Agglomerative Algorithm in the Second Stage

Information of the centroids $M(G_i)$ and the number of elements $|G_i|$ in cluster G_i $(i = 1,2,\ldots,c)$ is passed to the second stage. Note that information concerning every object $x \in X$ is not required to generate clusters by AHC.

Different sets of $M(G_i)$ are obtained from the first stage. To have better clusters in the second stage, a number of different trials of the first stage are made and those centroids with the minimum value of

$$J = \sum_{i=1}^{C} \sum_{x \in G_i} \|x - M(G_i)\|^2 \tag{5}$$

is taken for the second stage.

Pairwise Constraints in the Second Stage

Although must-link constraints is already handled in the first stage, cannot-link constraints still exist in the second stage. Hence CL is handled by a penalty term in the following algorithm.

Penalized Agglomerative Hierarchical Clustering Algorithm (P-AHC)

P-AHC1: For initial clusters derived from the first stage, calculate $d(G,G')$ for all $G, G' \in \mathscr{G}$.
P-AHC2:

$$d(G_q,G_r) = \arg \min_{G,G' \in \mathscr{G}} \{d(G,G') + \sum_{x_k \in G, x_l \in G'} \omega_{kl}\}$$

using the penalty term with ω_{kl}:
if $(x_k,x_l) \in CL$, $\omega_{kl} > 0$; if $(x_k,x_l) \notin CL$, $\omega_{kl} = 0$.
Let $\bar{G} = G_q \cup G_r$.
Add $\bar{G}$ to $\mathscr{G}$ and delete G_q, G_r from $\mathscr{G}$.
$C = C - 1$. If $C = 1$, stop.
P-AHC3: Calculate $d(\bar{G},G')$ for all other $G' \in \mathscr{G}$. Go to **P-AHC2**.

Note that ω is taken to be sufficient large, i.e., we assume hard constraints.

4 Numerical Examples

Two data sets were used for evaluating the present method with other methods already proposed elsewhere. One is an artificial data set on the plane, while the second is a real data set from a data repository [1].

As for the methods, the following abbreviated symbols are used:

- PAHC: penalized AHC algorithm;
- COPKPP: one-pass COP k-means++ ;
- COPK: ordinary one-pass COP k-means ;
- COPKPP(n): one-pass COP k-means++ with n different initial values;
- COPK(n): one-pass COP k-means with n different initial values.

The computation environment is as follows.

CPU: Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz - 3.60GHz
Memory: 8.00 GB
OS: Windows 7 Professional 64bit
Programming Language: C

Two Circles

First data is shown In Fig. 1. The objective is to separate the outer circle having 700 points and the inner circle with 9,300 points. Note that the two clusters are 'unbalanced' in the sense that the numbers of objects are very different.

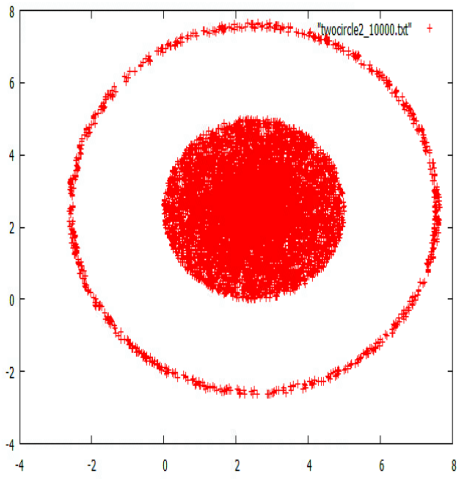

Fig. 1 Data of 'two circles'

Shuttle Data Set

The Shuttle data set downloaded from [1] has 9 dimensions that can be divided into seven classes. About 80% of points belong to Class 1. We divide this data set into two clusters: one cluster is Class 1 and another cluster should be other six classes, since to detect small six clusters in 20% of points and one large cluster of 80% of points directly is generally a difficult task.

Evaluation Criteria

The evaluation has been done using three criteria: objective function values, the Rand index, and the run time.

Note that *CL* alone is used and *ML* is not used here, since *ML* was found to be not useful when compared with *CL* by preliminary tests on these data sets.

Pairs of objects in *CL* were randomly selected from the data set: one object from a cluster and another object from another cluster. For artificial data set the number in *CL* varies from 0 to 50; for the Shuttle data the number in *CL* varies from 0 to 500. The number of trials $n = 100$ (the number of trials in the first stage is 100) or $n = 10$ were used.

4.1 Evaluation by Objective Function Value

The averages of objective function values *J* are plotted in Figs. 2 and 3, respectively for the artificial data and the Shuttle data.

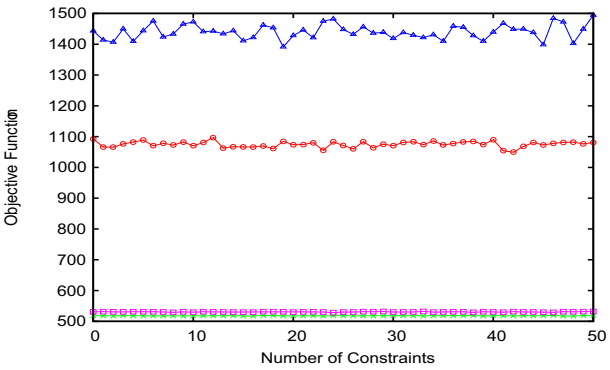

Fig. 2 Objective function values with *CL* for artificial data. Red circles are for COPK(100)-PAHC. Green × are for COPKPP(100)-PAHC. Blue triangles are for COPK(10)-PAHC. Pink squares are for COPKPP(10)-PAHC.

From these figures it is clear that COPKPP-PAHC has less values of the objective function than COPK-PAHC.

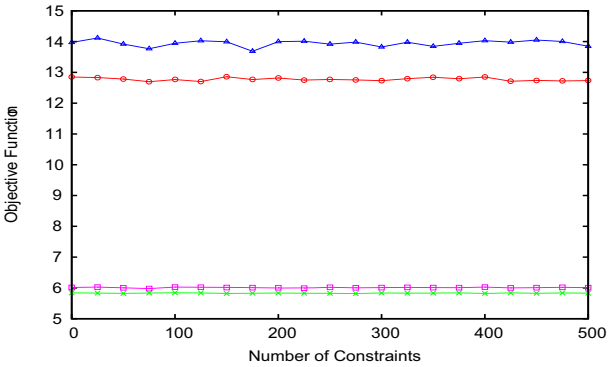

Fig. 3 Objective function values with *CL* for the Shuttle data. Red circles are for COPK(100)-PAHC. Green × are for COPKPP(100)-PAHC. Blue triangles are for COPK(10)-PAHC. Pink squares are for COPKPP(10)-PAHC.

4.2 Evaluation by RandIndex

The Rand index has been used as a standard index to measure precision of classification [12]:

$$Rand(P_1, P_2) = \frac{|C_a| + |C_b|}{{}_nC_2} \tag{6}$$

where P_1 and P_2 means the precise classification and the actually obtained classification. $|C_a|$ is the number of pairs of objects in C_a such that a pair in C_a is in the same precise class and at the same time in the same cluster obtained by the experiment; $|C_b|$ is the number of pairs of objects in C_b such that a pair in C_a is in different precise classes and at the same time in different clusters obtained by the

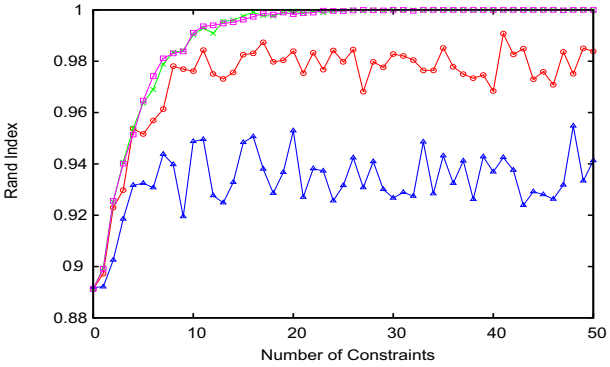

Fig. 4 Rand index values with *CL* for artificial data. Red circles are for COPK(100)-PAHC. Green × are for COPKPP(100)-PAHC. Blue triangles are for COPK(10)-PAHC. Pink squares are for COPKPP(10)-PAHC.

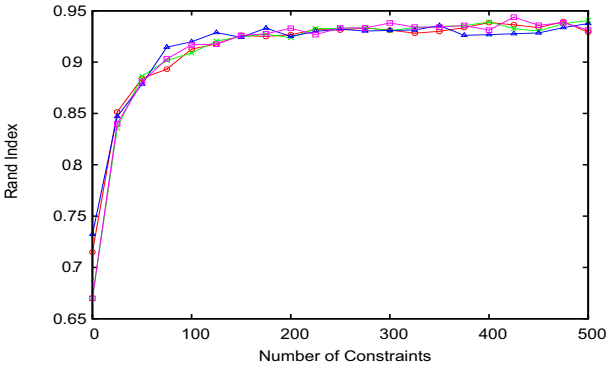

Fig. 5 Rand index values with *CL* for the Shuttle data. Red circles are for COPK(100)-PAHC. Green × are for COPKPP(100)-PAHC. Blue triangles are for COPK(10)-PAHC. Pink squares are for COPKPP(10)-PAHC.

experiment. If the resulting clusters precisely coincide with the precise classes, then $Rand(P_1, P_2) = 1$, and vice versa.

The Rand index with $n = 100$ has been calculated and the results are shown in Figs. 4 and 5, respectively for the artificial data and the Shuttle data. The former figure shows advantage of COPKPP, while the effect of K-means++ is not clear in the second example.

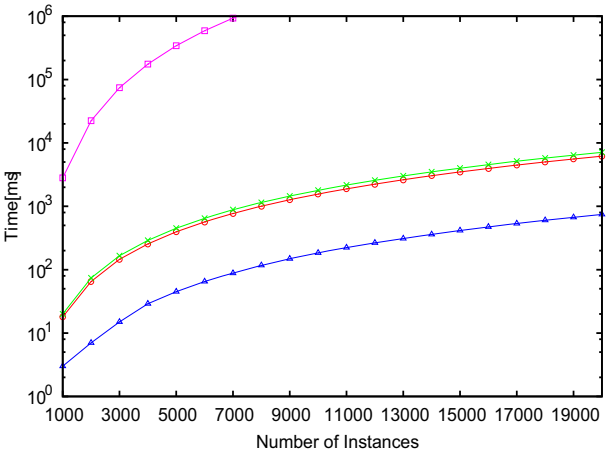

Fig. 6 Relation between the number of objects in artificial data and the CPU time. Red circles are for COPK(100)-PAHC. Green × are for COPKPP(100)-PAHC. Blue triangles are for COPKPP(10)-PAHC. Pink squares are for PAHC.

4.3 Evaluation by CPU Time

How total CPU time varies by using one-pass COP k-means++ or one-pass COP k-means was investigated. The used methods were COPK(100)-PAHCCCOPKPP(100)-PAHCCCOPKPP(10)-PAHCC and PAHC (without the first stage). Ten trials with n objects and their average CPU time was measured with $n = 1,000 - 20,000$. In the first stage the number of objects was reduced to 1% and the second stage AHC was carried out. The result is shown in Fig. 6.

Fig. 6 shows that CPU time was reduced to 0.1% by introducing the two-stage method. When COPK(100)-PAHC and COPKPP(100)-PAHC are comparted, the latter needs more time, but the difference is not notable.

5 Conclusion

This paper proposed a two-stage algorithm in which the first stage uses one-pass k-means++ and the second stage uses the centroid method of agglomerative hierarchical clustering. Pairwise constraints were moreover introduced in the algorithm. It has been shown by numerical examples that one-pass k-means++ is effective when compared with one-pass k-means in the first stage. Thus the dependence on initial values was greatly improved. Moreover the use of cannot-links was effective in the numerical examples. This inclination is in accordance with other studies, e.g., [11].

The two-stage procedure could handle relatively large-scale data sets. However, more tests on larger real data should be done as a future work in order to show the usefulness of the proposed method in a variety of applications.

Acknowledgment. The authors greatly appreciate anonymous reviewers for their useful comments. This study has partially been supported by the Grant-in-Aid for Scientific Research, JSPS, Japan, No.23500269.

References

1. Arthur, D., Vassilvitskii, S.: k-means++: The Advantages of Careful Seeding. In: Proc. of SODA 2007, pp. 1027–1035 (2007)
2. Basu, S., Bilenko, M., Mooney, R.J.: A Probabilistic Framework for Semi-Supervised Clustering. In: Proc. of the Tenth ACM SIGKDD (KDD 2004), pp. 59–68 (2004)
3. Basu, S., Davidson, I., Wagstaff, K.L. (eds.): Constrained Clustering. CRC Press (2009)
4. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum, New York (1981)
5. Bishop, C.: Pattern Recognition and Machine Learning. Springer (2006)
6. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press (2006)
7. Everitt, B.S.: Cluster Analysis, 3rd edn., Arnold (1993)
8. MacQueen, J.B.: Some methods of classification and analysis of multivariate observations. In: Proc. of 5th Berkeley Symposium on Math. Stat. and Prob., pp. 281–297 (1967)
9. Miyamoto, S.: Introduction to Cluster Analysis. Morikita-shuppan (1999) (in Japanese)

10. Obara, N., Miyamoto, C.S.: A Method of Two-Stage Clustering with Constraints Using Agglomerative Hierarchical Algorithm and One-Pass *K*-Means. In: Proc. of SCIS-ISIS 2012, pp. 1540–1544 (2012)
11. Terami, A., Miyamoto, S.: Constrained Agglomerative Hierarchical Clustering Algorithms with Penalties. In: Proc. of FUZZ-IEEE 2011, pp. 422–427 (2011)
12. Wagstaff, N., Cardie, C., Rogers, S., Schroedl, S.: Constrained K-means Clustering with Background Knowledge. In: Proc. of ICML 2001, pp. 577–584 (2001)
13. `http://archive.ics.uci.edu/ml/`

An Algorithm Combining Spectral Clustering and DBSCAN for Core Points

So Miyahara, Yoshiyuki Komazaki, and Sadaaki Miyamoto

Abstract. The method of spectral clustering is based on the graph Laplacian, and outputs good results for well-separated groups of points even when they have nonlinear boundaries. However, it is generally difficult to classify a large amount of data by this technique because computational complexity is large. We propose an algorithm using the concept of core points in DBSCAN. This algorithm first applies DBSCAN for core points and performs spectral clustering for each cluster obtained from the first step. Simulation examples are used to show performance of the proposed algorithm.

1 Introduction

Many researchers are now working on analysis of huge data on the web. In accordance with this, many methods of data analysis have been developed. Data clustering is not exceptional: nowadays a variety of new algorithms of clustering is being applied to large-scale data sets. Special attention has been paid to spectral clustering [4, 2, 3] which is based on a weighted graph model and uses the graph Laplacian. It has been known that this method works well even when clusters have strongly nonlinear boundaries between clusters, as far as they are well-separated.

In spite of its usefulness, the spectral clustering has a drawback: it has a relatively large computation when compared with a simple algorithm of the K-means [4, 5]. The latter can be applied to huge data, since the algorithm is very simple, but the former uses eigenvalues and eigenvectors which needs much more computation.

This paper proposes a method combining the spectral clustering and the idea in a simple graph-theoretical method based on DBSCAN [6]. The both methods are

So Miyahara · Yoshiyuki Komazaki
Master's Program in Risk Engineering, University of Tsukuba, Ibaraki 305-8573, Japan

Sadaaki Miyamoto
Department of Risk Engineering, University of Tsukuba, Ibaraki 305-8573, Japan
e-mail: miyamoto@risk.tsukuba.ac.jp

V.-N. Huynh et al. (eds.), *Knowledge and Systems Engineering, Volume 2*,
Advances in Intelligent Systems and Computing 245,
DOI: 10.1007/978-3-319-02821-7_4,

well-known, but their combination with a simple modification leads a new algorithm. A related study has been done by Yan et al. [7] in which K-means is first used and the centers from K-means are clustered using the spectral clustering. The present study is different from [7], since the original objects are made into clusters by the spectral clustering by the method herein, whereas the K-means centers are clustered in [7]. A key point is that only core-points are used for clustering, and other 'noise points' are allocated to clusters using a simple technique of supervised classification. Moreover, these two methods of the spectral clustering and DBSCAN has a common theoretical feature that is useful for reducing computation, and hence the combination proposed here has a theoretical basis, as we will see later. Such a feature cannot be found between K-means and the spectral clustering.

The rest of this paper is organized as follows. Section 2 gives preliminaries, and then Section 3 proposes a new algorithm using the spectral clustering and DBSCAN for core points. Section 4 shows illustrative examples and a real example. Finally, Section 5 concludes the paper.

2 Preliminary Consideration

This section discusses the well-known methods of the spectral clustering and DBSCAN.

2.1 Spectral Clustering

The spectral clustering, written as SC here, uses a partition of a graph of objects $D = \{1,2,\ldots,n\}$ for clustering. The optimality of the partition is discussed in [3] but omitted here.

Assume that the number of clusters is fixed and given by c. A similarity matrix $S = (s_{ij})$ is generated using a dissimilarity $d(i,j)$ between i and j. We assume that $d(i,j)$ is the Euclidean distance in this paper, although many other dissimilarity can also be used for the same purpose.

$$S = [s_{ij}],\ s_{ij} = \exp\left(-\frac{d(i,j)}{(2\sigma^2)}\right)$$

where σ is a positive constant. When the ε-neighborhood graph should be used, then those s_{ij} with $d(i,j) > \varepsilon$ should be set to zero. We then calculate

$$D = diag(d_1,\cdots,d_n),\ d_i = \sum_{j=1}^{n} s_{ij}$$

and the graph Laplacian L:

$$L = D^{-\frac{1}{2}}(D-S)D^{-\frac{1}{2}}$$

Minimum c eigenvalues are taken and the corresponding eigenvectors are assumed to be $u_1, \cdots, u_c$. A matrix

$$U = (u_1, \cdots, u_c)$$

is then defined. Each component of the eigenvalues has correspondence to an object. Then K-means clustering of each rows with c clusters will give the results of clustering by SC [3]. Concretely, suppose row vectors of U are $\boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_n^\top$: $U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)^\top$, then K-means algorithm is applied to objects $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$, where $\boldsymbol{u}_j$ $(j = 1, \ldots, n)$ is a c-vector [3].

2.2 DBSCAN-CORE

DBSCAN proposed by Ester et al. [6] generates clustering based on density of objects using two parameters Eps and MinPts. For given Eps and MinPts, the Eps-neighborhood of $p \in D$ is given by

$$N_{\mathrm{Eps}}(p) = \{q \in D \mid d(p,q) \leq \mathrm{Eps}\}$$

When an object p satisfies $|N_{\mathrm{Eps}}(p)| \geq \mathrm{MinPts}$, then p is called a core-point (note: $|N_{\mathrm{Eps}}(p)|$ is the number of elements in $N_{\mathrm{Eps}}(p)$).

If the next two conditions are satisfied, then p is called *directly density-reachable from q*:

1. $p \in N_{\mathrm{Eps}}(q)$, and
2. $|N_{\mathrm{Eps}}(q)| \geq \mathrm{MinPts}$ (q is a core-point).

A variation of the DBSCAN algorithm used here starts from a core-point called seed, and then collects all *core points* that are directly density-reachable from the seed. Then they form a cluster. Then the algorithm repeats the same procedure until no more cluster is obtained. The remaining objects are left unclassified. In other words, this algorithm searches the connected components of the graph generated from core points with the edges of direct reachability, and defines clusters as the connected components.

This algorithm is simpler than the original DBSCAN in that only core-points are made into clusters, while non-core points are included in clusters by the original DBSCAN. Therefore the present algorithm is called DBSCAN-CORE in this paper. Specifically, The set D is first divided into C of core points and N of non-core points:

$$D = C \cup N, \qquad C \cap N = \emptyset.$$

Clusters $C_1, \ldots, C_l$ generated by DBSCAN-CORE is a partition of C:

$$\bigcup_{i=1}^{l} C_i = C, \qquad C_i \cap C_j = \emptyset \ \ (i \neq j).$$

How to decide appropriate values of the parameters is given in [6], but omitted here.

3 Combining DBSCAN-CORE and Spectral Clustering

A method proposed here first generates clusters of core-points using DBSCAN-CORE and then each clusters are subdivided by the spectral clustering. We assume that Eps-neighborhood graph is used for the both method, i.e., the same value of Eps is applied: $s_{ij} = 0$ iff $d(i, j) \geq Eps$ in the spectral clustering and N_{Eps} is used for DBSCAN-CORE.

We then have the next proposition.

Proposition 1. *Let $G_1, \ldots, G_K$ be clusters of set C of core-points generated by the spectral clustering. Then, for arbitrary G_i, there exists C_j such that $G_i \subseteq C_j$.*

The proof is based on the fact that no cluster by the spectral clustering connects different connected components of graph C [3].

Note that DBSCAN-CORE has a fast algorithm similar to generation of spanning trees. Thus the complexity is $O(n)$, which is less than the complexity of the spectral clustering. We hence have the following simple algorithm combining DBSCAN-CORE and the spectral clustering.

Algorithm DBSCAN-CORE-SC:

1. Define core points and carry out DBSCAN-CORE. Let $C_1, \ldots, C_l$ be clusters of C.
2. Generate subclusters of C_i for all $i = 1, 2, \ldots, l$ by the spectral clustering.

3.1 Clusters of Data Set D

The above procedure generates clusters of C, the set of core points, but the non-core points will remain as noises. When we wish to classify noises to one of the clusters of C, a simple supervised classification algorithm can be used. A typical algorithm is the k nearest neighbor method (kNN) [4]: Let $x \in N$ should be allocated to some cluster. Suppose $y_1, \ldots, y_k \in C$ be k nearest neighbors of x in C. Then the class h is determined by the following:

$$h = \arg \max_{1 \leq j \leq l} |\{y_1, \ldots, y_k\} \cap C_j|.$$

When $k = 1$, the above is reduced to the nearest neighbor allocation:

$$h = \arg \min_{1 \leq j \leq l} d(x, C_j),$$

where $d(x, C_j) = \min_{y \in C_j} d(x, y)$. The nearest neighbor allocation is used for numerical examples below.

We thus have an algorithm to generate clusters of D by first generating clusters of C using DBSCAN-CORE-SC and then allocate other points. We moreover use a particular option that only those points in $N_{\mathrm{Eps}}(q)$ for some core point q should be allocated using kNN, but those points $p' \notin N_{\mathrm{Eps}}(q')$ for all $q' \in C$ should be left as *noise points*. This algorithm is called DBSCAN-CORE-SC-kNN in this paper.

3.2 Other Related Algorithms

Although we propose DBSCAN-CORE-SC and DBSCAN-CORE-SC-*k*NN here, there are other algorithms that should be compared with the proposed algorithms.

Algorithm SC-CORE

Step 1. Select core points by the same procedure as the DBSCAN-CORE.
Step 2. Generate clusters by the spectral clustering for the core points without using DBSCAN-CORE.

End of SC-CORE

Thus SC-CORE generates clusters of C. Accordingly, we can define SC-CORE-*k*NN by using the *k*NN after applying SC-CORE.

4 Numerical Examples

Algorithms of DBSCANCSCCSC-CORE-*k*NNCDBSCAN-CORE-SC-*k*NN, and SC-CORE-*k*NN have been done by using the following computational environment.

- Hardware: Deginnos Series
- OS: Ubuntu 12.10 i64 bit OSj
- CPU: Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz
- Memory: 16.00 GB
- Language: Python 2.7
- Eigenvalue solver: linalg.eig in Numpy library

In order to reduce the effect of initial values in the K-means used in the spectral clustering, 50 trials with different random initial values were used and the clusters with minimum objective function values were selected.

The used parameters were the same values for all methods: The nearest neighbor allocation: $k = 1$ and the neighborhood graph with $\sigma = 1.0$ were used. Eps were determined by using the sorted 4-dist graph given in [6]. Thus MinPts $= 4$. The value of Eps is thus different according to the examples. First example uses Eps $= 0.0015$, Eps $= 0.0006$ for the second, and Eps $= 0.18$ for the third.

Noise points in the following figures are shown by black $\ast$, while clusters are shown by $+$ and $\bigcirc$ with different colors.

4.1 Results for Artificial Data Sets

Two artificial data sets on the plane were used. First data shown in Fig. 1 called *test data 1* has $2,650$ objects with 100 noise points. Second data shown in Fig. 2 called *test data 2* has $5,030$ objects with 50 noise points. Figures 3 and 4 show the results from SC-CORE-*k*NN and DBSCAN-CORE-SC-*k*NN for test data 1, respectively; Figures 5 and 6 show the results from SC-CORE-*k*NN and DBSCAN-CORE-SC-*k*NN for test data 2, respectively.

In the both examples DBSCAN-CORE divided the set of core points into two clusters: upper cluster and lower cluster in test data 1 and inner cluster and outer cluster in test data 2.

CPU times for SC, SC-CORE-*k*NN, and DBSCAN-CORE-SC-*k*NN are compared in Table 1 (Note that the time for preprocessing to calculate similarity values is not included in Table 1 and Table 3). The four figures show that good and same clusters are obtained by the two methods, and Table shows that run time is effectively reduced by the proposed method.

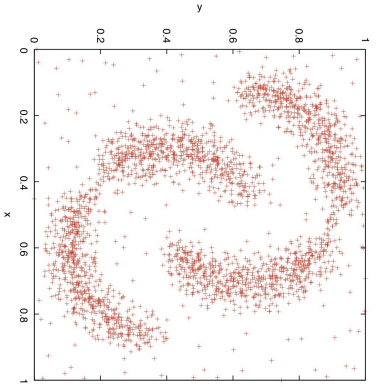

Fig. 1 Test data 1

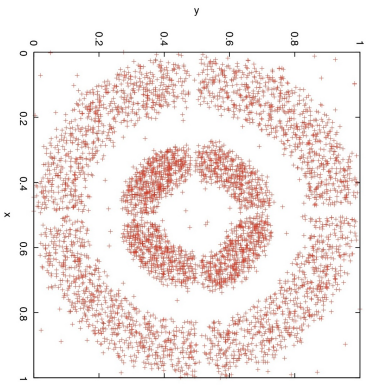

Fig. 2 Test data 2

Table 1 CPU time for artificial data with different methods

	Time(s)	
Method	test data1	test data2
SC	85.99019	510.94347
SC-CORE-*k*NN	84.04765	495.55304
DBSCAN-CORE-SC-*k*NN	29.05077	179.54790

4.2 The Iris Data Set

The well-known *iris* data has been handled by the different methods. As shown in Table 2, the same classification results were obtained from the different methods of SC-CORE-*k*NN and DBSCAN-CORE-SC-*k*NN. DBSCAN-CORE generated two well-separated clusters in *iris*. Then SC generated two subclusters from the larger cluster by DBSCAN-CORE.

The CPU time is again reduced by using DBSCAN-CORE-SC-*k*NN, as shown in Table 3.

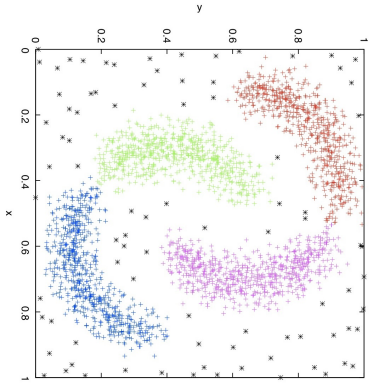

Fig. 3 Clusters generated by SC-CORE-kNN for test data 1

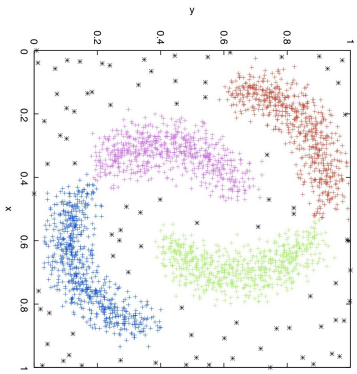

Fig. 4 Clusters generated by DBSCAN-CORE-SC-kNN for test data 1

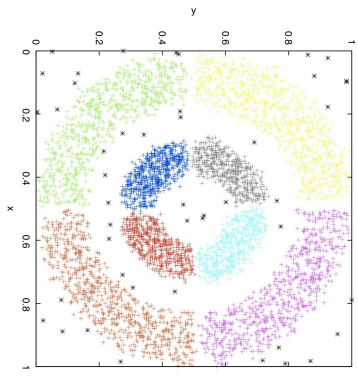

Fig. 5 Clusters generated by SC-CORE-kNN for test data 2

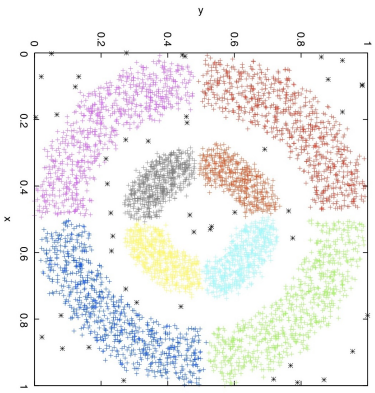

Fig. 6 Clusters generated by DBSCAN-CORE-SC-kNN for test data 2

Table 2 The results for iris data from different methods, where the Rand index is used

Method	Rand Index
HCM	0.87374
SC (complete graph)	0.87373
SC (ε-neighborhood graph)	0.85682
SC-CORE-kNN	0.85682
DBSCAN-CORE-SC-kNN	0.85682

Table 3 CPU time for *iris* data with different methods

Method	Time[s]
SC	0.36957
SC-CORE-*k*NN	0.35769
DBSCAN-CORE-SC-*k*NN	0.20951

5 Conclusion

The combination of DBSCAN with core points alone and the spectral clustering has been discussed. This combination is not an ad hoc technique, but has a methodological consistency shown in Proposition 1. The numerical results show effectiveness and efficiency of the proposed method. In the numerical examples, the values of the parameters greatly affects the results, and hence how good values of the parameters can be found should be an important subject of future study.

A fundamental problem is that no definite method to determine the number of clusters beforehand in DBSCAN-CORE-SC proposed here, which needs further research. More experiments for huge amount of real data and evaluation of the results should also be done.

Acknowledgment. The authors greatly appreciate anonymous reviewers for their useful comments. This study has partially been supported by the Grant-in-Aid for Scientific Research, JSPS, Japan, No.23500269.

References

1. Shi, J., Malik, J.: Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
2. Ng, A.Y., Jordan, M.I., Weiss, Y.: On Spectral Clustering: Analysis and an Algorithm. In: Advances in Neural Information Processing System, pp. 849–856 (2001)
3. von Luxburg, U.: A Tutorial on Spectral Clustering. Statistics and Computing 17(4), 395–416 (2007)
4. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. Wiley, Chichester (1973)
5. Miyamoto, S., Ichihashi, H., Honda, K.: Algorithms for Fuzzy Clustering. Springer, Berlin (2008)
6. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226–231 (1996)
7. Yan, D., Huang, L., Jordan, M.I.: Fast Approximate Spectral Clustering. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 907–916 (2009)

Relational Fuzzy c-Means and Kernel Fuzzy c-Means Using a Quadratic Programming-Based Object-Wise β-Spread Transformation

Yuchi Kanzawa

Abstract. Clustering methods of relational data are often based on the assumption that a given set of relational data is Euclidean, and kernelized clustering methods are often based on the assumption that a given kernel is positive semidefinite. In practice, non-Euclidean relational data and an indefinite kernel may arise, and a β-spread transformation was proposed for such cases, which modified a given set of relational data or a given a kernel Gram matrix such that the modified β value is common to all objects.

In this paper, we propose a quadratic programming-based object-wise β-spread transformation for use in both relational and kernelized fuzzy c-means clustering. The proposed system retains the given data better than conventional methods, and numerical examples show that our method is efficient for both relational and kernel fuzzy c-means.

1 Introduction

Fuzzy c-means (FCM) [2] is a well-known clustering method for vectorial data. In contrast, relational fuzzy c-means (RFCM) [2] clusters relational data. However, RFCM is not always able to cluster non-Euclidean relational data, because the membership cannot always be calculated. To overcome this limitation, a non-Euclidean RFCM (NERFCM) has been proposed [3]. NERFCM modifies the given data so that the memberships can be calculated, and this modification is called a β-spread transformation.

In order to cluster data with nonlinear borders, an algorithm that converts the original pattern space to a higher-dimensional feature space has been proposed [4]. This algorithm, known as kernel FCM (K-FCM), uses a nonlinear transformation defined by kernel functions in the support vector machine (SVM) [5]. In kernel

Yuchi Kanzawa
Shibaura Institute of Technology, Koto 135-8548 Tokyo, Japan
e-mail: kanzawa@sic.shibaura-it.ac.jp

V.-N. Huynh et al. (eds.), *Knowledge and Systems Engineering, Volume 2*,
Advances in Intelligent Systems and Computing 245,
DOI: 10.1007/978-3-319-02821-7_5,

data analysis, it is not necessary to know the explicit mapping of the feature space; however, its inner product must be known. Despite this, an explicit mapping has been reported and this was used to describe the appearance of clusters in a high-dimensional space [6], [7].

K-FCM fails for indefinite kernel matrices when the magnitude of the negative eigenvalues is extremely large, because the memberships cannot be calculated if the dissimilarity between a datum and a cluster center is updated to become a negative value. Although indefinite kernel matrices can be transformed to positive-definite ones by subtracting the minimal eigenvalue from their diagonal components, or by replacing negative eigenvalues with 0, these procedures result in over-transformation of the matrix. Although the clustering can still be executed, the risk is that the memberships can become extremely fuzzy and worsen the clustering result. Therefore, an indefinite-kernel FCM (IK-FCM) method has been developed [8]; this adopts a β-spread transformation and is similar to the derivation of NERFCM from RFCM.

In the conventional β-spread transformation for NERFCM or IK-FCM, the modified β value is common to all objects in the given relational data matrix or kernel Gram matrix. In this paper, we propose that a different value is added to each object in the given matrices. We refer to this as an object-wise β-spread transformation, and it allows clustering to be performed while retaining the original relational data matrix or kernel Gram matrix to the maximum possible extent. Because β is vector valued, we cannot determine its minimal value such that the dissimilarities between elements in the data set and cluster centers would be non-negative. Hence, we consider determining this vector for the case where the dissimilarities are non-negative, minimizing the squared Frobenius norms of the difference between the original matrix and the object-wise β-spread transformed matrix, which can be achieved by solving a quadratic programming problem. The proposed methods retain the given data better than previous methods, and so we expect them to produce better clustering results. Numerical examples show that this is the case.

The remainder of this paper is organized as follows. In Section 2, we introduce some conventional FCM methods. In Section 3, we propose two clustering algorithms: RFCM using a quadratic programming-based object-wise β-spread transformation (qO-NERFCM) and K-FCM using a quadratic programming-based object-wise β-spread transformation (qO-IK-FCM). In Section 4, we present some numerical examples, and conclude this paper in Section 5.

2 Preliminaries

In this section, we introduce RFCM, NERFCM, K-FCM, and IK-FCM. RFCM and K-FCM provide the basic methodology for NERFCM and IK-FCM, which apply a β-spread transformation to non-Euclidean relational data and indefinite kernel Gram matrices, respectively.

2.1 RFCM and NERFCM

For a given data set $X = \{x_k \mid k \in \{1,\dots,N\}\}$, the dissimilarity $R_{k,j}$ between x_k and x_j is given. Here, R is a matrix whose (k,j)-th element is $R_{k,j}$. Let C denote the cluster number. The goal of RFCM and NERFCM is obtaining the membership by which the datum x_k belongs to the i-th cluster, denoted by $u_{i,k}$, from R. $u \in \mathbb{R}^{C\times N}$ is referred to as the partition matrix.

RFCM is obtained by solving the optimization problem

$$\text{minimize}_u \sum_{i=1}^{C} \frac{\sum_{k=1}^{N}\sum_{j=1}^{N} u_{i,k}^m u_{j,k}^m R_{k,j}}{2\sum_{t=1}^{N} u_{i,t}^m}, \tag{1}$$

$$\text{subject to} \sum_{i=1}^{C} u_{i,k} = 1, \tag{2}$$

where $m > 1$ is a fuzzifier parameter. The RFCM procedure is as follows.

1

STEP 1. Fix $m > 1$ and assume an initial partition matrix u.

STEP 2. Update $v_i \in \mathbb{R}^N$ as

$$v_i = \left(u_{i,1}^m, \cdots, u_{i,N}^m\right)^{\mathsf{T}} / \sum_{k=1}^{N} u_{i,k}^m. \tag{3}$$

STEP 3. Update $d_{i,k}$ as

$$d_{i,k} = (Rv_i)_k - v_i^{\mathsf{T}} R v_i / 2. \tag{4}$$

STEP 4. Update the membership as

$$u_{i,k} = 1 / \sum_{j=1}^{C} \left(d_{i,k}/d_{j,k}\right)^{1/(m-1)}. \tag{5}$$

STEP 5. If the stopping criterion is satisfied, terminate this algorithm. Otherwise, return to STEP 2.

We say that a matrix $R \in \mathbb{R}^{N\times N}$ is *Euclidean* if there exists a set of points $\{y_1,\cdots,y_N\} \in \mathbb{R}^{N-1}$ such that $R_{k,j} = \|y_k - y_j\|_2^2$, and *non-Euclidean* if no such set of points exists. R is Euclidean if and only if HRH is negative semi-definite for $H = E - \mathbf{1}\mathbf{1}^{\mathsf{T}}/N$, where E is the N-dimensional unit matrix, and $\mathbf{1}$ is an N-dimensional vector whose elements are all 1. For a non-Euclidean R, RFCM only works when the positive eigenvalues of HRH are not particularly large. However, RFCM fails for a non-Euclidean R when the positive eigenvalues of HRH are extremely large because the membership cannot be calculated after the value of $d_{i,k}$ is updated to a negative value.

In order to overcome this limitation, the following modification of R, called the β-spread transformation, has been developed [3]:

$$R_\beta = R + \beta(\mathbf{1}\mathbf{1}^{\mathsf{T}} - E), \tag{6}$$

where β is a positive scalar value. With this β-spread transformation, NERFCM is given by the following algorithm.

1

STEP 1. Fix $m > 1$ and assume an initial partition matrix u. Set $\beta = 0$.
STEP 2. Execute STEP 2 of Algorithm 1.
STEP 3. Update $d_{i,k}$ as

$$d_{i,k} = \left(R_\beta v_i\right)_k - v_i^{\mathsf{T}} R_\beta v_i / 2. \tag{7}$$

STEP 4. If $d_{i,k} < 0$, update $\Delta\beta, d_{i,k}$, and β as

$$\Delta\beta = \max\{-2d_{i,k}/\|e_k - v_i\|^2\}, \tag{8}$$
$$d_{i,k} \leftarrow d_{i,k} + \Delta\beta/2\|e_k - v_i\|^2, \tag{9}$$
$$\beta \leftarrow \beta + \Delta\beta. \tag{10}$$

STEP 5. Execute STEP 4 of Algorithm 1.
STEP 6. If the stopping criterion is satisfied, terminate this algorithm. Otherwise, return to STEP 2.

Another option for tackling non-Euclidean relational data is to apply RFCM to a set of Euclidean relational data R', that has been modified from R in the following ways. The first R' is obtained by:

$$R'_{k,j} = K'_{k,k} - 2K'_{k,j} + K'_{j,j}, \tag{11}$$

where K' is the positive semi-definite matrix obtained from $K = -(1/2)HRH$ by subtracting the scaled identity matrix with its minimal eigenvalue if it is negative, that is,

$$K' = K - \lambda_{\min} E \quad (\lambda_{\min} < 0), \tag{12}$$

where $\lambda_{\min}$ is the minimal eigenvalue of K. In this paper, we refer to this revision as "diagonal shift" (DS), and its application to RFCM as RFCM-DS. The second R' is obtained by Eq. (11), when K' is the positive semi-definite matrix formed from $K = -(1/2)HRH$ by setting all the negative eigenvalues to zero. We refer to this modification as "nearest positive semi-definite" (nPSD), and thus, its application to RFCM is denoted as RFCM-nPSD.

In the NERFCM algorithm, β is adaptively determined at STEP 4; hence, the modification from R to R_β is suppressed to a minimum such that the algorithm execution can continue, whereas DS and nPSD may cause an over-transformation, only allowing the execution of RFCM. Indeed, it has been reported that RFCM-DS causes the memberships to become extremely fuzzy [3].

2.2 K-FCM and IK-FCM

For a given data set $X = \{x_k \mid k \in \{1,\ldots,N\}\}$, K-FCM assumes that the kernel matrix $K \in \mathbb{R}^{N\times N}$ is given. Let $\mathbb{H}$ be a higher-dimensional feature space, $\Phi : X \to \mathbb{H}$ be a map from the data set X to the feature space $\mathbb{H}$, and $W = \{W_i \in \mathbb{H} \mid i \in \{1,\cdots,C\}\}$ be a set of cluster centers in the feature space.

K-FCM is obtained by solving the following optimization problem:

$$\text{minimize}_{u,W} \sum_{i=1}^{C} \sum_{k=1}^{N} u_{i,k}^m \|\Phi(x_k) - W_i\|_{\mathbb{H}}^2 \tag{13}$$

subject to Eq. (2). Generally, Φ cannot be given explicitly, so the K-FCM algorithm assumes that a kernel function $\mathscr{K} : x \times x \to \mathbb{R}$ is given. This function describes the inner product value of the pairs of elements in the data set of the feature space as $\mathscr{K}(x_k, x_j) = \langle \Phi(x_k), \Phi(x_j) \rangle$. However, it can be interpreted that Φ is given explicitly by allowing $\mathbb{H} = \mathbb{R}^N$, $\Phi(x_k) = e_k$, where e_k is the N-dimensional unit vector whose ℓ-th element is the Kronecker delta $\delta_{k,\ell}$, and by introducing $K \in \mathbb{R}^{N \times N}$ such that

$$K_{k,j} = \langle \Phi(x_k), \Phi(x_j) \rangle. \tag{14}$$

According to this discussion, K-FCM is given as follows.

1

STEP 1. Fix $m > 1$. Assume a kernel matrix $K \in \mathbb{R}^{N \times N}$ and an initial partition matrix u.

STEP 2. Update cluster centers as

$$W_i = \left(u_{i,1}^m, \cdots, u_{i,N}^m\right)^{\mathsf{T}} / \sum_{k=1}^{N} u_{i,k}^m. \tag{15}$$

STEP 3. Update the dissimilarity between each element in the data set and the cluster center as

$$d_{i,k} = (e_k - W_i)^{\mathsf{T}} K (e_k - W_i). \tag{16}$$

STEP 4. Update the membership as

$$u_{i,k} = 1 / \sum_{j=1}^{C} \left(d_{i,k}/d_{j,k}\right)^{1/(m-1)} \tag{17}$$

STEP 5. If (u, d, W) converge, terminate this algorithm. Otherwise, return to STEP 2.

K-FCM is constructed based on Eq. (14), i.e., K is positive semi-definite. Even so, K is sometimes introduced without the existence of Φ being guaranteed. In this case, K is not always positive semi-definite. Similar to RFCM, K-FCM works for an indefinite K when the magnitude of negative eigenvalues is not particularly large. However, K-FCM fails for indefinite K when the magnitude of negative eigenvalues is extremely large, because the memberships cannot be calculated after the dissimilarity between a datum and a cluster center is updated as a negative value. In order to overcome this limitation, the following β-spread transformation of K has been developed [8]:

$$K_\beta = K + \beta E. \tag{18}$$

With this β-spread transformation, IK-FCM is given by the following algorithm.

1

STEP 1. Fix $m > 1$ for K-FCM. Assume a kernel matrix $K \in \mathbb{R}^{N\times N}$ and an initial partition matrix u. Set $\beta = 0$ and $K_0 = K$.

STEP 2. Execute STEP 2 of Algorithm 1.

STEP 3. Update $d_{i,k}$ as

$$d_{i,k} = (e_k - W_i)^{\mathsf{T}} K_\beta (e_k - W_i). \tag{19}$$

STEP 4. If $d_{i,k} < 0$, update $\Delta\beta, d_{i,k}$, β, and K_β as:

$$\Delta\beta = \max\{-d_{i,k}/\|e_k - W_i\|_2^2\}, \tag{20}$$
$$d_{i,k} \leftarrow d_{i,k} + \Delta\beta\|e_k - W_i\|^2, \tag{21}$$
$$\beta \leftarrow \beta + \Delta\beta, \tag{22}$$
$$K_\beta \leftarrow K_\beta + \Delta\beta E. \tag{23}$$

STEP 5. Execute STEP 4 of Algorithm 1.

STEP 6. If the stopping criterion is satisfied, terminate this algorithm. Otherwise, return to STEP 2.

Another option for handling indefinite kernel data is to apply K-FCM to a positive semi-definite matrix K', which is modified from K in the following two ways. The first K' is obtained from K by adding the scaled identity matrix with its minimal eigenvalue if it is negative, that is,

$$K' = K + \lambda_{\min} E \quad (\lambda_{\min} < 0), \tag{24}$$

where $\lambda_{\min}$ is the minimal eigenvalue of K. As for RFCM, we refer to this revision as "diagonal shift" (DS), and its application to K-FCM is thus K-FCM-DS. The second K' is obtained from K by setting all the negative eigenvalues to zero, and thus K-FCM becomes K-FCM-nPSD.

In the IK-FCM algorithm, β is adaptively determined at STEP 4; hence, the modification from K to K_β is suppressed to a minimum such that the algorithm execution can continue, whereas DS and nPSD may cause an over-transformation, only allowing the execution of K-FCM.

3 Quadratic Programming-Based Object-Wise β-Spread Fuzzy Clustering

3.1 Concept of the Proposed Algorithms

In the conventional β-spread transformation given by Eq. (6) for NERFCM or Eq. (18) for IK-FCM, the modified β value is common to all objects in the given relational data matrix or kernel Gram matrix. In this paper, we propose that a different value is added to each object in the given matrices. We refer to this as an object-wise β-spread transformation, and it allows clustering to be performed while retaining the original relational data matrix or kernel Gram matrix to the maximum possible extent. The object-wise β-spread transformation for RFCM is

$$R_\beta = R + \frac{1}{2}\boldsymbol{\beta}\mathbf{1}^\mathsf{T} + \frac{1}{2}\mathbf{1}\boldsymbol{\beta}^\mathsf{T} - \mathrm{diag}(\boldsymbol{\beta}), \tag{25}$$

and that for K-FCM is

$$K_\beta = K + \mathrm{diag}(\boldsymbol{\beta}), \tag{26}$$

where $\boldsymbol{\beta} \in \mathbb{R}_+^N$. If all the elements of $\boldsymbol{\beta}$ are the same, then the object-wise β-spread transformation is identical to that in NERFCM and IK-FCM.

Because $\boldsymbol{\beta}$ is vector valued, we cannot determine its minimal value such that the dissimilarities between elements in the data set and cluster centers would be non-negative. Therefore, we consider determining $\boldsymbol{\beta}$ for the case where the dissimilarities are non-negative, minimizing the squared Frobenius norms $\|R_\beta - R\|_\mathrm{F}^2$ and $\|K_\beta - K\|_\mathrm{F}^2$, which can be achieved by solving a quadratic programming problem.

3.2 RFCM Using a Quadratic Programming-Based Object-Wise $\boldsymbol{\beta}$-Spread Transformation

Using RFCM with an object-wise β-spread transformation, the following condition must be satisfied in order for the dissimilarities between the elements in the data set and cluster centers to be non-negative:

$$-\frac{1}{2}(e_k - v_i)^\mathsf{T} R_\beta (e_k - v_i) \geq 0 \tag{27}$$

$$\begin{aligned}\Leftrightarrow & -\frac{1}{2}(e_k - v_i)^\mathsf{T} R_0 (e_k - v_i) - \frac{1}{4}(e_k - v_i)^\mathsf{T}\boldsymbol{\beta}\mathbf{1}^\mathsf{T}(e_k - v_i) \\ & - \frac{1}{4}(e_k - v_i)^\mathsf{T}\mathbf{1}\boldsymbol{\beta}^\mathsf{T}(e_k - v_i) \\ & + \frac{1}{2}(e_k - v_i)^\mathsf{T}\mathrm{diag}(\boldsymbol{\beta})(e_k - v_i) \geq 0\end{aligned} \tag{28}$$

$$\Leftrightarrow d_{i,k} + \frac{1}{2}\sum_{\ell=1}^{N}(e_k^{(\ell)} - v_i^{(\ell)})^2\boldsymbol{\beta}_\ell \geq 0, \tag{29}$$

where $e_k^{(\ell)}$ and $v_i^{(\ell)}$ are the ℓ-th element of e_k and v_i, respectively. Under this condition, the value of $\boldsymbol{\beta}$ that minimizes $\|R_\beta - R\|_\mathrm{F}^2$ can be obtained by solving the following quadratic programming problem:

$$\text{minimize}_{\boldsymbol{\beta}} \frac{1}{2}\boldsymbol{\beta}^\mathsf{T} A \boldsymbol{\beta} \tag{30}$$

$$\text{subject to } d_{i,k} + \frac{1}{2}\sum_{\ell=1}^{N}(e_k^{(\ell)} - v_i^{(\ell)})^2\boldsymbol{\beta}_\ell \geq 0 \quad (k \in \{1,\cdots,N\}, i \in \{1,\cdots,C\}), \tag{31}$$

where

$$A_{k,j} = \begin{cases} N-1 & (k=j), \\ 1 & (k \neq j). \end{cases} \tag{32}$$

Using the obtained value of $\boldsymbol{\beta}$, we can describe the dissimilarity between the datum x_k and the cluster center v_i as

$$d_{i,k}(\boldsymbol{\beta}) = d_{i,k}(0) + \frac{1}{2}\sum_{\ell=1}^{N}(e_k^{(\ell)} - v_i^{(\ell)})^2\boldsymbol{\beta}_\ell. \tag{33}$$

If we set a tentative value of $\boldsymbol{\beta}$, and obtain the modified value of $\boldsymbol{\beta}+\Delta\boldsymbol{\beta}$ satisfying the above constraint, we need only solve the following quadratic programming problem for $\Delta\boldsymbol{\beta}$.

$$\text{minimize}_{\Delta\boldsymbol{\beta}}\frac{1}{2}\Delta\boldsymbol{\beta}^{\mathsf{T}}A\Delta\boldsymbol{\beta} \tag{34}$$

$$\text{subject to } d_{i,k}(\boldsymbol{\beta}) + \frac{1}{2}\sum_{\ell=1}^{N}(e_k^{(\ell)} - v_i^{(\ell)})^2\Delta\boldsymbol{\beta}_\ell \geq 0$$
$$(k \in \{1,\cdots,N\}, i \in \{1,\cdots,C\}). \tag{35}$$

Hence, we set $\boldsymbol{\beta}$ to 0 at the beginning of the algorithm and then modify $\boldsymbol{\beta}$ by the value of $\Delta\boldsymbol{\beta}$ obtained from the above programming problem, provided that at least one of dissimilarities between a datum and a cluster center is non-negative while the algorithm execution continues. On the basis of the above, we modify the NERFCM algorithm to the following quadratic programming-based object-wise β-spread NERFCM (qO-NERFCM).

1

STEP 1. Fix $m > 1$ and assume an initial partition matrix u. Set $\boldsymbol{\beta} = \Delta\boldsymbol{\beta} = 0$.

STEP 2. Update the cluster center $v_i \in \mathbb{R}^N$ as

$$v_i = \left(u_{i,1}^m, \cdots, u_{i,N}^m\right)^{\mathsf{T}} / \sum_{k=1}^{N} u_{i,k}^m. \tag{36}$$

STEP 3. Update the dissimilarity between data and cluster centers $d_{i,k}$ as

$$d_{i,k} = \left(R_\beta v_i\right)_k - v_i^{\mathsf{T}} R_\beta v_i/2. \tag{37}$$

STEP 4. If $d_{i,k} < 0$, solve the quadratic programming problem for $\Delta\boldsymbol{\beta}$

$$\text{minimize}_{\boldsymbol{\beta}}\frac{1}{2}\Delta\boldsymbol{\beta}^{\mathsf{T}}A\Delta\boldsymbol{\beta} \tag{38}$$

$$\text{subject to } d_{i,k}(\boldsymbol{\beta}) - \frac{1}{2}\sum_{\ell=1}^{N}(e_k^{(\ell)} - v_i^{(\ell)})^2\Delta\boldsymbol{\beta}_\ell \geq 0$$
$$(k \in \{1,\cdots,N\}, i \in \{1,\cdots,C\}) \tag{39}$$

and update $d_{i,k}$ and $\boldsymbol{\beta}$ as

$$d_{i,k} \leftarrow d_{i,k} + \frac{1}{2}\|e_k - v_i\|_{\boldsymbol{\beta}}^2, \tag{40}$$

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \Delta\boldsymbol{\beta}. \tag{41}$$

STEP 5. Update the membership $u_{i,k}$ as

$$u_{i,k} = 1/\sum_{j=1}^{C}(d_{i,k}/d_{j,k})^{1/(m-1)}. \tag{42}$$

STEP 6. If the stopping criterion is satisfied, terminate this algorithm. Otherwise, return to STEP 2.

Determining $\Delta\boldsymbol{\beta}$ in conventional NERFCM is identical to solving the quadratic programming problem given by Eqs. (38) and (39) with the additional constraint $\boldsymbol{\beta}_k = \boldsymbol{\beta}_j \quad (k \neq j)$, because the objective function $\boldsymbol{\beta}^{\mathsf{T}} A \boldsymbol{\beta}$ becomes $\frac{1}{2}\mathbf{1}^{\mathsf{T}} A \mathbf{1} \beta^2$, resulting in the expression given in Eq. (8). The constraints for β in Eqs. (30), (31) are more relaxed in qO-NERFCM than in conventional NERFCM, and hence qO-NERFCM achieves a lower objective function value than conventional NERFCM.

3.3 K-FCM Using Quadratic Programming-Based Object-Wise β-Spread

Using a quadratic programming-based object-wise β-spread transformation in K-FCM, the following condition must be satisfied in order for the dissimilarities between data and cluster centers to be non-negative:

$$(e_k - v_i)^{\mathsf{T}} K_{\boldsymbol{\beta}} (e_k - v_i) \geq 0 \tag{43}$$

$$\Leftrightarrow (e_k - v_i)^{\mathsf{T}} K_0 (e_k - v_i) + (e_k - v_i)^{\mathsf{T}} \mathrm{diag}(\boldsymbol{\beta}) (e_k - v_i) \geq 0 \tag{44}$$

$$\Leftrightarrow d_{i,k} + \sum_{\ell=1}^{N} (e_k^{(\ell)} - v_i^{(\ell)})^2 \boldsymbol{\beta}_\ell \geq 0. \tag{45}$$

Under this condition, the value of $\boldsymbol{\beta}$ that minimizes $\|K_\beta - K\|_{\mathrm{F}}^2$ can be obtained by solving the following quadratic programming problem.

$$\text{minimize}_{\boldsymbol{\beta}} \boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{\beta} \tag{46}$$

$$\text{subject to } d_{i,k} + \sum_{\ell=1}^{N} (e_k^{(\ell)} - v_i^{(\ell)})^2 \boldsymbol{\beta}_\ell \geq 0 \quad (k \in \{1, \cdots, N\}, i \in \{1, \cdots, C\}) \tag{47}$$

Using the obtained value of $\boldsymbol{\beta}$, we can describe the dissimilarity between the datum x_k and the cluster center v_i as

$$d_{i,k}(\boldsymbol{\beta}) = d_{i,k}(0) + \sum_{\ell=1}^{N} (e_k^{(\ell)} - v_i^{(\ell)})^2 \boldsymbol{\beta}_\ell \tag{48}$$

If we set a tentative value of $\boldsymbol{\beta}$, and obtain the modified value of $\boldsymbol{\beta} + \Delta\boldsymbol{\beta}$ satisfying the above constraint, we need only solve the following quadratic programming problem for $\Delta\boldsymbol{\beta}$.

$$\text{minimize}_{\Delta\boldsymbol{\beta}} \Delta\boldsymbol{\beta}^{\mathsf{T}} \Delta\boldsymbol{\beta} \tag{49}$$

$$\text{subject to } d_{i,k}(\boldsymbol{\beta}) + \frac{1}{2} \sum_{\ell=1}^{N} (e_k^{(\ell)} - v_i^{(\ell)})^2 \Delta\boldsymbol{\beta}_\ell \geq 0 \quad (k \in \{1, \cdots, N\}, i \in \{1, \cdots, C\}) \tag{50}$$

Hence, we set $\boldsymbol{\beta}$ to 0 at the beginning of the algorithm and then modify $\boldsymbol{\beta}$ using the value of $\Delta\boldsymbol{\beta}$ obtained from the above programming problem, provided that at least