

University of Louisville

ThinkIR: The University of Louisville's Institutional Repository

Electronic Theses and Dissertations

5-2011

Unsupervised and semi-supervised clustering with learnable cluster dependent kernels.

Ouiem Bchir 1978-
University of Louisville

Follow this and additional works at: <https://ir.library.louisville.edu/etd>

Recommended Citation

Bchir, Ouiem 1978-, "Unsupervised and semi-supervised clustering with learnable cluster dependent kernels." (2011). *Electronic Theses and Dissertations*. Paper 86.
<https://doi.org/10.18297/etd/86>

This Doctoral Dissertation is brought to you for free and open access by ThinkIR: The University of Louisville's Institutional Repository. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of ThinkIR: The University of Louisville's Institutional Repository. This title appears here courtesy of the author, who has retained all other copyrights. For more information, please contact thinkir@louisville.edu.

UNSUPERVISED AND SEMI-SUPERVISED CLUSTERING WITH LEARNABLE CLUSTER DEPENDENT KERNELS

By

Ouiem Bchir

B.S., E.E, National School of Engineering of Monastir, 2002

A Dissertation
Submitted to the Faculty of the
Graduate School of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

May 2011

UNSUPERVISED AND SEMI-SUPERVISED CLUSTERING WITH LEARNABLE CLUSTER DEPENDENT KERNELS

By

Ouiem Bchir

B.S., E.E, National School of Engineering of Monastir, 2002

A Dissertation Approved On

4/13/11

Date

By the following Dissertation Committee:

Frigui Hichem

Dissertation Director

Ryan Gill

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation and respect to my adviser Dr. Hichem Frigui who has supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way. I feel lucky that I had this opportunity to work in his team.

I would like to thank Dr. Lotfi Hermi for his help on the diffusion theory.

I would like to express my deepest gratitude to my parents for their love and moral support through the duration of my studies. I am also heartily grateful to my husband for his understanding and support.

Finally, I would like to dedicate this thesis to my son Rayen.

ABSTRACT

UNSUPERVISED AND SEMI-SUPERVISED CLUSTERING WITH LEARNABLE CLUSTER DEPENDENT KERNELS

Ouiem Bchir

April 13, 2011

Despite the large number of existing clustering methods, clustering remains a challenging task especially when the structure of the data does not correspond to easily separable categories, and when clusters vary in size, density and shape. Existing kernel based approaches allow to adapt a specific similarity measure in order to make the problem easier. Although good results were obtained using the Gaussian kernel function, its performance depends on the selection of the scaling parameter. Moreover, since one global parameter is used for the entire data set, it may not be possible to find one optimal scaling parameter when there are large variations between the distributions of the different clusters in the feature space.

One way to learn optimal scaling parameters is through an exhaustive search of one optimal scaling parameter for each cluster. However, this approach is not practical since it is computationally expensive especially when the data includes a large number of clusters and when the dynamic range of possible values of the scaling parameters is large. Moreover, it is not trivial to evaluate the resulting partition in order to select the optimal parameters.

To overcome this limitation, we introduce two new fuzzy relational clustering techniques that learn cluster dependent Gaussian kernels.

The first algorithm called clustering and Local Scale Learning algorithm (LSL) minimizes one objective function for both the optimal partition and for cluster dependent scaling parameters that reflect the intra-cluster characteristics of the data.

The second algorithm, called Fuzzy clustering with Learnable Cluster dependent Kernels (FLeCK) learns the scaling parameters by optimizing both the intra-cluster and the inter-cluster dissimilarities. Consequently, the learned scale parameters reflect the relative density, size, and position of each cluster with respect to the other clusters.

We also introduce semi-supervised versions of LSL and FLeCK. These algorithms generate a fuzzy partition of the data and learn the optimal kernel resolution of each cluster simultaneously. We show that the incorporation of a small set of constraints can guide the clustering process to better learn the scaling parameters and the fuzzy memberships in order to obtain a better partition of the data. In particular, we show that the partial supervision is even more useful on real high dimensional data sets where the algorithms are more susceptible to local minima.

All of the proposed algorithms are optimized iteratively by dynamically updating the partition and the scaling parameter in each iteration. This makes these algorithms simple and fast. Moreover, our algorithms are formulated to work on relational data. This makes them applicable to data where objects cannot be represented by vectors or when clusters of similar objects cannot be represented efficiently by a single prototype.

Our extensive experiments show that FLeCK and SS-FLeCK outperform existing algorithms. In particular, we show that when data include clusters with various inter-cluster and intra-cluster distances, learning cluster dependent kernel is crucial in obtaining a good partition.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
LIST OF TABLES	ix
LIST OF FIGURES	xi
I INTRODUCTION	1
A Major contributions	3
1 The clustering and Local Scale Learning algorithm	3
2 Fuzzy clustering approach with Learnable Cluster depen-	
dent kernels	3
3 The Semi-Supervised clustering and Local Scale Learning	
algorithm	4
4 The Semi-Supervised Fuzzy clustering approach with Learn-	
able Cluster dependent kernels	4
B Dissertation overview	5
II LITERATURE SURVEY	6
A Fuzzy clustering	6
1 The Fuzzy C-Means algorithm	7
2 The Gustafson-Kessel clustering algorithm	8
B Relational clustering algorithms	10
1 The Relational Fuzzy C-Means (RFCM) Algorithm	11
C Kernel based clustering	14
1 Background	14
2 The kernel K-Means algorithm	15
3 The Metric kernel Fuzzy C-Means algorithm	16
4 The Feature Space kernel Fuzzy C-Means algorithm	17

5	The kernelized Non-Euclidean Relational Fuzzy c-Means Algorithm	18
D	Spectral based clustering	20
E	Semi-supervised clustering	24
1	The Semi-supervised kernel C-means algorithm	25
2	The Semi-Supervised Spectral clustering algorithm	26
 III UNSUPERVISED RELATIONAL CLUSTERING WITH LOCAL SCALE PARAMETERS 30		
A	Introduction	30
B	Relational clustering with local scale parameters	32
1	The clustering and Local Scale Learning algorithm	32
2	Performance illustration	36
3	Limitations of LSL	44
C	Relational clustering by optimizing both the intra-cluster and inter-cluster distances	44
1	The Fuzzy clustering with learnable cluster dependent kernels algorithm	44
2	Interpretation of the learned scaling parameters	49
3	Performance illustration	53
 IV SEMI-SUPERVISED RELATIONAL CLUSTERING WITH LOCAL SCALE PARAMETERS 61		
A	Semi-Supervised relational clustering with local scaling parameter	62
1	The Semi-Supervised clustering and Local Scale Learning algorithm	62
2	Performance illustration	67
B	Semi-Supervised relational clustering optimizing both the intra-cluster and inter-cluster distances	71
1	The Semi-Supervised Fuzzy clustering with LEarnable Cluster dependent kernels algorithm	71

2	Performance illustration	75
V	EXPERIMENTS	79
A	Datasets and performance measures	79
1	Image database	79
2	Handwritten digits	81
3	Performance measures	81
B	Evaluation of the unsupervised clustering algorithms	84
1	Synthetic datasets	84
2	Application to Image database Categorization	91
3	Application to categorization of handwritten digits	95
4	Time complexity	101
5	Conclusions	101
C	Evaluation of the semi-supervised clustering algorithms	102
1	Synthetic datasets	103
2	Application to image database categorization	107
3	Application to categorization of handwritten digits	110
4	Conclusions	112
VI	CONCLUSIONS AND POTENTIAL FUTURE WORKS	113
A	Conclusions	113
B	Potential future work	116
1	Feature weighting and kernel Learning	116
2	Prototype based classifier	116
	REFERENCES	118
	CURRICULUM VITAE	126

LIST OF TABLES

1	Partition and scaling parameters learned by LSL for dataset 1 displayed in Figure 3 (a) when $K = 0.01$	39
2	Partition and scaling parameters learned by LSL for dataset 2 displayed in Figure 3 (b) when $K = 0.05$	40
3	Partition and scaling parameters learned by LSL for dataset 3 displayed in Figure 3 (c) when $K = 4$	41
4	Partition and scaling parameters learned by LSL for dataset 4 displayed in Figure 3 (d) when $K = 1.5$	42
5	Partition and scaling parameters learned by LSL for dataset 1 displayed in Figure 3 (a) when $K = 2$	43
6	Variations of σ_i with respect to the inter-cluster distance when 2 clusters have the same density and size.	51
7	Variations of σ_i for different cluster sizes and densities	52
8	Partition and scaling parameters learned by FLeCK for dataset 1 displayed in Figure 3 (a)	54
9	Partition and scaling parameters learned by FLeCK for dataset 2 displayed in Figure 3 (b)	55
10	Partition and scaling parameters learned by FLeCK for dataset 3 displayed in Figure 3 (c)	56
11	Partition and scaling parameters learned by FLeCK for dataset 4 displayed in Figure 3 (d)	57
12	Partition and scaling parameters learned by FLeCK for dataset 5 displayed in Figure 3 (a)	58
13	Partition and scaling parameters learned by SS-LSL on dataset 3 displayed in Figure 3 (c)	68

14	Partition and scaling parameters learned by SS-LSL on dataset 2 displayed in Figure 3 (b)	69
15	Partition and scaling parameters learned by SS-LSL on dataset 1 displayed in Figure 3 (a)	70
16	Partition and scaling parameters learned by SS-FLeCK on dataset 2 displayed in Figure 3 (b)	76
17	Partition and scaling parameters learned by SS-FLeCK on dataset 4 displayed in Figure 3 (d)	77
18	Sample image from each category of the COREL image database and the number of images for each category.	80
19	Variations of the sum of intra-group and inter-group distances across the five image categories of the COREL image database.	80
20	Number of samples form each category of the handwritten digits . . .	81
21	Variations of the sum of intra-group and inter-group distances across the handwritten digits categories.	82
22	Self tuning spectral clustering results on the COREL image data . .	93
23	FLeCK clustering results on the COREL image data	94
24	Self tuning spectral clustering results on the handwritten digit Data .	97
25	FLeCK clustering results on the handwritten digit Data	98
26	The scaling parameters learned by FLeCK for the clusters of the handwritten digits	99
27	The simulation time (in second) for kNERF, the self tuning spectral, and the FLeCK algorithms	101
28	Number of images used to construct the constraints versus the number of images with improved categorization	108
29	Number of digits used to construct the constraints versus the number of digits with improved categorization	111

LIST OF FIGURES

1	2-D dataset with 2 clusters with different densities	31
2	Accuracy results obtained on the dataset of Figure 1 using an extensive search of cluster dependent scaling parameters	31
3	Datasets used to illustrate the performance of LSL. Each cluster is shown by a different color.	38
4	Fuzzy memberships learned by LSL on dataset 1 (Figure 3 (a)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.	39
5	Fuzzy memberships learned by LSL on dataset 2 (Figure 3 (b)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.	40
6	Fuzzy memberships learned by LSL on dataset 3 (Figure 3 (c)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.	41
7	Fuzzy memberships learned by LSL on dataset 4 (Figure 3 (d)) with respect to (a) cluster 1, and (b) cluster 2.	42
8	Fuzzy memberships learned by LSL on dataset 5 (Figure 3 (e)) with respect to (a) cluster 1 and (b) cluster 2	43
9	LSL clustering results on dataset 1 (Figure 3 (a)) with different parameters K , (a) $K = 0.001$, (b) $K = 0.01$, (c) $K = 0.1$, (d) $K = 1$, and (e) $K = 0.05$	45
10	Three clusters with the same density and size, but different intra-cluster distances.	52
11	Fuzzy memberships learned by FLeCK on dataset 1 (Figure 3 (a)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.	54
12	Fuzzy memberships learned by FLeCK on dataset 2 (Figure 3 (b)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.	55

13	Fuzzy memberships learned by FLeCK on dataset 3 (Figure 3 (c)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.	58
14	Fuzzy memberships learned by FLeCK on dataset 5 (Figure 3 (e)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.	59
15	Fuzzy memberships learned by SS-LSL on dataset 3 (Figure 3 (a)) with respect to cluster 1 (a), cluster 2 (b), and cluster 3 (c).	68
16	Fuzzy memberships learned by SS-LSL on dataset 2 (Figure 3 (a)) with respect to cluster 1 (a), cluster 2 (b), and cluster 3 (c).	69
17	Fuzzy memberships learned by SS-LSL on dataset 5 (Figure 3 (a)) with respect to cluster 1 (a), cluster 2 (b), and cluster 3 (c).	70
18	Fuzzy memberships learned by SS-FLeCK on dataset 2 (Figure 3 (a)) with respect to cluster 1 (a), cluster 2 (b), and cluster 3 (c).	77
19	Fuzzy memberships learned by SS-FLeCK on dataset 4 (Figure 3 (a)) with respect to cluster 1 (a), and cluster 2 (b).	78
20	Results of clustering dataset 1 using (a) FCM, (b) DBSCAN, (c) GK, (d) kNERF, (e) Spectral, (f) LSL, and (f) FLeCK	85
21	Results of clustering dataset 2 using (a) FCM, (b) DBSCAN, (c) GK, (d) kNERF, (e) Spectral, (f) LSL, and (f) FLeCK	86
22	Results of clustering dataset 3 using (a) FCM, (b) DBSCAN, (c) GK, (d) kNERF, (e) Spectral, (f) LSL, and (f) FLeCK	87
23	Results of clustering dataset 4 using (a) FCM, (b) DBSCAN, (c) GK, (d) kNERF, (e) Spectral, (f) LSL, and (f) FLeCK	88
24	Results of clustering dataset 5 using (a) FCM, (b) DBSCAN, (c) GK, (d) kNERF, (e) Spectral, (f) LSL, and (f) FLeCK	89
25	Performance measures obtained on categorizing COREL dataset using FCM, DBSCAN, GK, kNERF, Spectral, LSL and FLeCK clustering approaches.	92
26	Performance measures obtained on categorizing handwritten digits dataset using FCM, DBSCAN, GK, kNERF, Spectral, LSL and FLeCK clustering approaches.	96

27	The top 10 representatives of (a) cluster 11 and (b) cluster 16 obtained by FLeCK.	100
28	The top 10 representatives of cluster 9 obtained by FLeCK.	100
29	Results of clustering dataset 2 using (a) SS-kernel-C-Means, (b) semi-supervised spectral learning, (c) semi-supervised graph clustering, (d) SS-LSL, and (e) SS-FLeCK.	104
30	Results of clustering dataset 4 using (a) SS-kernel-C-Means, (b) semi-supervised spectral learning, (c) semi-supervised graph clustering, (d) SS-LSL, and (e) SS-FLeCK.	105
31	Results of clustering dataset 5 using (a) SS-kernel-C-Means, (b) semi-supervised spectral learning, (c) semi-supervised graph clustering, (d) SS-LSL, and (e) SS-FLeCK.	106
32	The accuracy of the five semi-supervised algorithms on COREL data as the number of constraints used for partial supervision is increased.	108
33	Mean and standard deviation of the five performance measures over 20 runs of LSL and SS-LSL on the COREL dataset.	109
34	Mean and standard deviation of the five performance measures over 20 runs of FLeCK and SS-FLeCK on the COREL dataset.	109
35	The accuracy of the five semi-supervised algorithms on handwritten digits data as the number of constraints used for partial supervision is increased.	110
36	Mean and standard deviation of the five performance measures over 20 runs of LSL and SS-LSL on the handwritten digits dataset.	111
37	Mean and standard deviation of the five performance measures over 20 runs of FLeCK and SS-FLeCK on the handwritten digits dataset.	112

CHAPTER I

INTRODUCTION

Clustering consists of partitioning a dataset into subsets, so that data in each subset share some common aspects. In other words, according to a defined distance measure, objects in the same cluster should be as similar as possible and objects in different clusters should be as dissimilar as possible. Clustering has been used in many applications related to understanding and exploring the structure of the data. In particular, fuzzy clustering techniques have been shown to be suitable to describe real situations with overlapping boundaries [37].

Typically, the set of objects to be clustered could be described in two ways: object based representation, and relational based representation. While for object data representation, each object is represented by a feature vector, for the relational representation only information of how two objects are related is available. Relational data representation is more general in the sense that it can be applied when only the degree of dissimilarity between objects is available or when groups of similar objects cannot be represented efficiently by a single prototype.

Despite the large number of existing clustering methods, clustering remains a challenging task when the structure of the data does not correspond to easily separable categories, and when clusters vary in size, density and shape. kernel based approaches [52, 35, 38] can adapt a specific distance measure in order to make the problem easier. Gaussian kernel is the most common distance function. Although good results were obtained using this kernel, it relies on the selection of a scale parameter. This selection is commonly done manually. Moreover, as it is a global parameter over the entire data, it may not be appropriate when the distributions of the different clusters in the feature space exhibit large variations. Thus, in the absence of *a priori* knowledge,

the choice of a scaling parameter that can adapt to different clusters and discriminate between them is difficult. In such situations, the geometry of the data should be explored to learn local dissimilarity measures by finding intrinsic properties, such as local dimensionality, and local parametrization.

Clustering is a difficult combinatorial problem. Moreover, as a task, clustering is subjective. In fact, the dataset to be clustered may need to be grouped differently depending on the application. However, unsupervised clustering algorithms do not take into account this subjectivity, unless it can be expressed explicitly in the distance measure by selecting and weighting appropriate features or by choosing and adapting a specific dissimilarity measure. As a result, quite often, the resulting categories do not reflect the user’s expectations. Consequently, semi-supervised clustering which allows incorporating prior knowledge in the unsupervised clustering task has recently become a topic of interest. In this approach, a small amount of class labels or pairwise constraints are used to guide the unsupervised clustering process [47, 22, 43]. These pairwise constraints, called side information, specify whether two data items should be assigned to the same cluster or not. The most common approach of incorporating partial supervision information in clustering is the use of search based methods that drive the clustering algorithm towards a better categorization of the data [70, 69].

Even though semi-supervised algorithms have proved to outperform the unsupervised ones, the semi-supervision information can have a limited effect. For instance, some algorithms learn only cluster centers [62] and the supervision information is not used to learn a mapping of the data. Thus, either most of the semi-supervision information will be ignored (if the cost of violating the constraints is low) or this information will adversely affect the structure of the clusters (if the cost of violating the constraints is very high). Other semi-supervised clustering algorithms learn the centers and the covariance matrices of the clusters [24]. Even though the supervision information can be more useful in this case, it still cannot allow learning a non linear mapping of the data.

A Major contributions

The major contributions of this dissertation consists of the design, implementation, and analysis of four clustering algorithms that address the issues raised in the previous section. These algorithms partition the data, learn the scaling parameters for each cluster, and assign a fuzzy membership to each object in each cluster. The fuzzy memberships allow the algorithms to deal with overlapping clusters, and provide a richer description of the data by distinguishing between the points at the core and at boundary of the cluster. The learned scaling parameters help in identifying clusters of different size, shape and densities and, can be used in subsequent steps to provide better cluster assignment.

1 The clustering and Local Scale Learning algorithm

We introduce a new fuzzy relational clustering technique with Local Scaling parameter Learning (LSL). This approach learns the underlying cluster dependent dissimilarity measure while finding compact clusters in the given data. The learned measure is a Gaussian dissimilarity function defined with respect to each cluster that allows to control the scaling of the clusters and thus, improve the final partition. LSL minimizes one objective function for both the optimal partition and for the cluster dependent scaling parameter. This optimization is done iteratively by dynamically updating the partition and the local measure in each iteration. This make the kernel learning task takes advantages of the unlabeled data and reciprocally, the categorization task takes advantages of the local learned kernel. Moreover, as we assume that the data is available in a relational form, the proposed approach is applicable even when only the degree to which pairs of objects in the data are related is available. It is also more practical when similar objects cannot be represented efficiently by a single prototype.

2 Fuzzy clustering approach with Learnable Cluster dependent kernels

The Fuzzy clustering with Learnable Cluster dependent kernels (FLeCK) is another algorithm that we have developed that learns Gaussian kernel function with

cluster dependent scaling parameters while seeking compact clusters. Unlike the LSL which uses only the intra-cluster distances to learn the scaling parameters, FLeCK learns the scaling parameters that minimize the intra-cluster distances and maximize the inter-cluster distances simultaneously. The scaling parameter, σ_i , with respect to each cluster i is designed to distinguish and separate the objects of cluster i from the rest of the data. It reflects the relative density, size, and position of this cluster with respect to the other clusters.

To the best of our knowledge, LSL and FLeCK are the first algorithms that learn the Gaussian scaling parameter in an unsupervised way. This is a major contribution to Gaussian based clustering approaches such as kernel and spectral clustering methods that suffer from their sensitivity to this parameter.

3 The Semi-Supervised clustering and Local Scale Learning algorithm

The Semi-Supervised clustering and Local Scale Learning algorithm is an extension of the LSL. In order to guide LSL to a better partitioning of the data and avoid local minima, especially for high dimensional real world data, we incorporate prior knowledge in the unsupervised clustering task in the form of a small set of constraints on which instances should or should not reside in the same cluster.

4 The Semi-Supervised Fuzzy clustering approach with Learnable Cluster dependent kernels

The Semi-Supervised Fuzzy clustering approach with Learnable Cluster dependent kernels (SS-FLeCK) is an extension of the FLeCK algorithm. It uses side-information in the form of a small set of constraints on which instances should or should not reside in the same cluster. This is achieved by combining constraint-based methods that guide the clustering algorithm towards a better grouping of the data and local distance measure learning methods that adapt the underlying dissimilarity measure used by the clustering algorithm.

The four proposed algorithms are compared to existing algorithms using synthetic and real datasets. We provide detailed analysis of the results and justify their

better performance.

B Dissertation overview

The organization of the rest of the dissertation is as follows. In chapter 2, we outline some important background material on machine learning techniques relevant to this thesis. In chapter 3, we present our new unsupervised relational clustering approaches LSL and FLeCK. In chapter 4, we present the new semi-supervised relational clustering approaches SS-LSL and SS-FLeCK. In chapter 5, we describe the experiments conducted to validate the proposed algorithms. The performance of our approach is compared to several state-of-the-art clustering methods. Chapter 6 contains the conclusions and potential future work.

CHAPTER II

LITERATURE SURVEY

In this chapter, we outline some important background material on machine learning techniques relevant to this thesis. We first introduce fuzzy clustering. Then, we review spectral and kernel based clustering approaches. We also cover the semi-supervised clustering paradigm as well as some background on kernel and distance measure learning techniques.

A Fuzzy clustering

Clustering is an essential and very frequently performed task in pattern recognition and data mining. It can aid in a variety of tasks related to understanding and exploring the structure of large and high dimensional data. The goal of cluster analysis is to find natural groupings in a set of objects such that objects in the same cluster are as similar as possible and objects in different clusters are as dissimilar as possible.

A substantial amount of research has focused on the C-means objective function and algorithm [91], especially since it is prone to local minima in its most basic form. Recent research on C-means includes methods for initialization [73], adding local search to the algorithm [51], and generalizing the use of squared Euclidean distance to general Bregman divergences [30].

In most applications, categories are rarely well separated and boundaries are overlapping. Describing these real world situations by crisp sets does not allow the user to quantitatively distinguish between objects which are strongly associated with a particular category from those that have only a marginal association with multiple ones, particularly, along the overlapping boundaries. Fuzzy clustering methods are

good at dealing with these situations [12]. In fact, data elements can belong to more than one cluster with fuzzy membership degrees. These memberships indicate the strength of the association between that data element and a particular cluster.

Fuzzy clustering is widely used in the machine learning field. Areas of application of fuzzy cluster analysis include data analysis [17, 11], information retrieval [26, 7], image segmentation [3], and robotics [40]. One of the most widely used fuzzy clustering algorithm is the Fuzzy C-Means (FCM) Algorithm [86].

1 The Fuzzy C-Means algorithm

The FCM algorithm [86] attempts to partition a set of feature vectors $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ into C fuzzy clusters $\mathbf{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_C\}$. It assigns a membership degree $u_{ij} \in (0, 1)$, to each sample \mathbf{x}_j in each cluster i that indicates the degree to which \mathbf{x}_j belongs to cluster i . We define \mathbf{x}_j a p - dimensional column vector $\mathbf{x}_j = [\mathbf{x}_{j1}, \mathbf{x}_{j2}, \mathbf{x}_{j3}, \dots, \mathbf{x}_{jp}]^t$. Similarly, we define \mathbf{a}_i as a p - dimensional column vector $\mathbf{a}_j = [\mathbf{a}_{j1}, \mathbf{a}_{j2}, \mathbf{a}_{j3}, \dots, \mathbf{a}_{jp}]^t$

The FCM [86] aims to minimize the following objective function:

$$J = \sum_{i=1}^C \sum_{j=1}^N u_{ij}^m \|\mathbf{x}_j - \mathbf{a}_i\|^2 \quad (1)$$

subject to

$$\sum_{i=1}^C u_{ij} = 1, \quad \text{for } 1 \leq j \leq N \quad (2)$$

In (1), $m \in (1, \infty)$ is the fuzzifier that controls the fuzziness of the partition.

Minimizing the objective function in (1) subject to the constraint in (2), is a non-trivial constrained nonlinear optimization problem with continuous parameters \mathbf{a}_i and u_{ij} with no analytical solution. As a result, an alternating optimization scheme, i.e. alternatively optimizing one set of parameters while the other set of parameters are considered as fixed, is a common approach to optimize (1). It can be shown [86] that the update equations for \mathbf{a}_i and u_{ij} are

$$u_{ij} = \frac{\left(\frac{1}{\|\mathbf{x}_j - \mathbf{a}_i\|^2}\right)^{\frac{1}{m-1}}}{\sum_{q=1}^C \left(\frac{1}{\|\mathbf{x}_j - \mathbf{a}_q\|^2}\right)^{\frac{1}{m-1}}}, \quad (3)$$

and

$$\mathbf{a}_i = \frac{\sum_{j=1}^N u_{ij}^m \mathbf{x}_j}{\sum_{j=1}^N u_{ij}^m} \quad (4)$$

The steps of the FCM are outlined in algorithm 1.

Algorithm 1 The Fuzzy C-Means algorithm

Fix $m \in (1, \infty)$;

Fix the number of clusters C ;

Initialize the cluster centers \mathbf{a}_i ;

Repeat

 Update the fuzzy memberships u_{ij} using (3);

 Update the cluster centers \mathbf{a}_i using (4);

Until no change in u_{ij} .

2 The Gustafson-Kessel clustering algorithm

The FCM can work well only for spherical shaped clusters since the distances from data points to the centers of the clusters are based on the Euclidean distance. To overcome the above limitation, the Gustafson-Kessel algorithm (GK) [88] extended the Euclidean distance of the standard FCM by employing an adaptive norm. Consequently, the GK can detect clusters of different geometrical shape. The GK algorithm minimizes

$$J = \sum_{i=1}^C \sum_{j=1}^N u_{ij}^m d(\mathbf{x}_j, \mathbf{a}_i)^2 \quad (5)$$

In (5), the distance $d(\mathbf{x}_j, \mathbf{a}_i)$ is defined as

$$d(\mathbf{x}_j, \mathbf{a}_i)^2 = (\mathbf{x}_j - \mathbf{a}_i)^t \mathbf{T}_i (\mathbf{x}_j - \mathbf{a}_i) \quad (6)$$

where \mathbf{T}_i is a local norm-inducing matrix that can be adapted to the local topological structure of each cluster i .

In addition to the condition on the partition matrix \mathbf{U} in (2), the following additional constraint on the norm matrices \mathbf{T}_i is imposed

$$\det(\mathbf{T}_i) = \rho_i \quad (7)$$

The minimization of the GK objective functional is achieved by using the alternating optimization method, and results in the following update equations,

$$\mathbf{a}_i = \frac{\sum_{j=1}^N u_{ij}^m \mathbf{x}_j}{\sum_{j=1}^N u_{ij}^m}, \quad (8)$$

$$u_{ij} = \frac{\left(\frac{1}{d(\mathbf{x}_j, \mathbf{a}_i)^2} \right)^{\frac{1}{m-1}}}{\sum_{q=1}^C \left(\frac{1}{d(\mathbf{x}_j, \mathbf{a}_q)^2} \right)^{\frac{1}{m-1}}}, \quad (9)$$

$$\mathbf{T}_i = (\rho_i |\boldsymbol{\Sigma}_i|)^{1/p} \boldsymbol{\Sigma}_i^{-1}, \quad (10)$$

where

$$\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^N u_{ij}^m (\mathbf{x}_j - \mathbf{a}_i)(\mathbf{x}_j - \mathbf{a}_i)^t}{\sum_{j=1}^N u_{ij}^m}, \quad (11)$$

In (10), p is the dimensionality of the feature vector \mathbf{x}_j . The steps of the GK algorithm are outlined in algorithm 2.

The FCM algorithm has been generalized further to find clusters of different shapes [87]. For instance, in [87], Bezdek et al. proposed the Fuzzy c-Varieties (FCV) algorithm to detect structures of data when all clusters are r -dimensional linear varieties, where r is less than the dimension of the data object. This is achieved by replacing the Euclidean distance in the FCM objective function in (1) by the sum of the Euclidean distance and the scaled Euclidean distance mapped to the r principle scatter directions (r longest axes) of the data objects.

All the clustering approaches described so far are prototype based approaches that represent similar objects efficiently by a single prototype (e.g. center or center and covariance matrix). Moreover, they require an explicit expression of the feature vector of each sample.

Algorithm 2 The Gustafson–Kessel algorithm

Fix $m \in (1, \infty)$; Fix $\rho_i \in (1, \infty)$

Fix the number of clusters C ;

Initialize the Fuzzy memberships u_{ij} ;

Repeat

Update the cluster centers \mathbf{a}_i using (8);

Update the local norm-inducing matrices \mathbf{T}_i using (11) and (10);

Update the distance using (6)

Update the fuzzy memberships u_{ij} using (9)

Until no change in u_{ij} .

B Relational clustering algorithms

For relational data, only information that represents the degrees to which pairs of objects in the data are related is available. In this case, object-based algorithms, such as FCM and GK, can not be used to partition relational data. A different approach, called relational clustering, has been developed for this data. Relational clustering is more general in the sense that it is applicable to situations in which the objects to be clustered cannot be represented by numerical features. It is also more practical for situations where the distance measure does not have a closed form solution, or when groups of similar objects cannot be represented efficiently by a single prototype (e.g. center).

Although clustering of object data has been an active field of research, clustering of relational data has received much less attention. This is despite the fact that

several applications would benefit tremendously from relational clustering algorithms. For instance, in several applications, the most effective distance measures do not have a closed form expression. Thus, these measures could not be used in object-based algorithms. Examples of these measures include the earth mover distance (EMD) [72], and the integrated region matching distance (IRM) [63]. Similarly, in web data mining and web user profiling, effective distance measures take into account the URL path traversed [66], and these similarities could not be integrated into object based clustering methods.

There are several relational clustering algorithms in the literature. The well-known relational algorithm is the Sequential Agglomerative Hierarchical Non - overlapping (SHAN) algorithm [89]. When the clusters are overlapping as is the case for most real-world applications, fuzzy clustering methods are more appropriate. Examples of fuzzy relational clustering methods include Ruspini algorithm [90] and the Relational Fuzzy C-Means Algorithm (RFCM) [85]. In [85], Hathaway and al. reformulated the Fuzzy C-Means (FCM) [86] objective function to adapt it to relational data by eliminating the prototypes from the FCM objective function. They also proposed the Non Euclidean Relational Fuzzy (NERF) C-Means algorithm [80]. NERF CMeans modifies the RFCM in order to deal with non-Euclidean relational distances. The authors in [71] extended RFCM to the Relational Fuzzy C-Maximal Density estimator (RFC-MDE) algorithm. RFC-MDE is robust and learn a local density with respect to each cluster.

A more detailed description of the RFCM and NERF is provided in the following subsection.

1 The Relational Fuzzy C-Means (RFCM) Algorithm

The Fuzzy C-Means (FCM) [86] objective function, defined in (1), has been reformulated by eliminating the prototypes from its objective function in order to adapt it to relational data [85]. The resulting relational FCM (RFCM) algorithm minimizes

$$J = \sum_{i=1}^C \frac{\sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m r_{jk}}{2 \sum_{k=1}^N u_{ik}^m}, \quad (12)$$

subject to the membership constraint in (2).

Unlike the object based FCM objective function which involves dissimilarity of the objects to a cluster center, the objective function in (12) includes only the dissimilarities r_{jk} between pairs of objects. These dissimilarities could be provided, or they could be computed from the object data using

$$r_{jk} = \|\mathbf{x}_j - \mathbf{x}_k\|^2 \quad (13)$$

The minimization of the FCM and RFCM objective functions are equivalent. In [85], the update equations were derived based on the fact that the squared Euclidean distance between feature vector \mathbf{x}_k and the centroid of the i^{th} cluster i can be written in terms of the relation matrix R as

$$d_{ik}^2 = (\mathbf{R}\mathbf{v}_i)_k - \frac{\mathbf{v}_i^t \mathbf{R} \mathbf{v}_i}{2} \quad (14)$$

where $(\mathbf{R}\mathbf{v}_i)_k$ is the k^{th} entry of the the vector column $\mathbf{R}\mathbf{v}_i$. In (14), \mathbf{v}_i is the membership vector defined by

$$\mathbf{v}_i = \frac{(u_{i1}^m, \dots, u_{iN}^m)^t}{\sum_{j=1}^N u_{ij}^m} \quad (15)$$

Equation (14), which is not based on the cluster centroid explicitly, allows the computation of the distance between data points and cluster prototypes in each iteration. It uses only the relational data and the set of initial fuzzy memberships. The fuzzy memberships are updated as in the standard FCM using the implicit distance values d_{ik}^2 that have been computed using (14). That is,

$$u_{ik} = \frac{1}{\sum_{t=1}^C (d_{ik}^2 / d_{tk}^2)^{\frac{1}{m-1}}} \quad (16)$$

The RFCM objective function in (12) is then optimized by alternating between the update equations in (14) and (16), until the membership values do not change significantly between consecutive iterations.

Although the RFCM was formulated to cluster relational data, it is expected to perform in an equivalent way to the FCM only if the relation matrix \mathbf{R} is Euclidean, i.e. provided that (13) is satisfied. If this is not the case, the distances computed using (14) may be negative causing the algorithm to fail. To overcome this restriction, Hathaway and Bezdek proposed the Non Euclidean Relational Fuzzy (NERF) C-Means algorithm [80]. NERF modifies the RFCM by adding a step that uses the β -spread transform to convert a non-Euclidean matrix \mathbf{R} into an Euclidean matrix \mathbf{R}_β using

$$\mathbf{R}_\beta = \mathbf{R} + \beta (\mathbf{M} - \mathbf{I}). \quad (17)$$

In (17), β denotes a suitably chosen scalar, $\mathbf{I} \in \mathbb{R}^{N \times N}$ is the identity matrix, and \mathbf{M} is an $N \times N$ matrix whose entries are all equal to one. In [80], the authors suggested that the distances d_{ik}^2 should be checked in every iteration for negativity, which indicates a non-Euclidean relation matrix. In that case, the β -spread transform should be applied with a suitable value of β to make d_{ik}^2 positive. A lower bound for the necessary shift, $\Delta\beta$, that is needed to make the distances positives was derived in [80] to be

$$\Delta\beta = \max_{i,k} \{-2d_{ik}^2 / \|\mathbf{v}_i - \mathbf{e}_k\|^2\} \quad (18)$$

where \mathbf{e}_k denotes the k^{th} column of the identity matrix. The steps of the NERF C-Means are outlined in algorithm 3.

The relational clustering algorithm described above clusters the data in its original feature space. However, in many real applications, categories may be defined better in a transformed feature space.

Algorithm 3 The Non Euclidean Relational Fuzzy (NERF) C-Means algorithm

Fix number of clusters C and $m \in [1\infty)$;

Initialize $\beta = 0$;

Initialize the fuzzy partition matrix \mathbf{U} ;

REPEAT

 Compute \mathbf{R}^β using (17);

 Compute the membership vectors \mathbf{v}_i using (15);

 Compute the distances using (14);

IF $d_{ik}^2 < 0$ for any i, k

 Compute $\Delta\beta$ using (18);

$$d_{ik}^2 = d_{ik}^2 + (\Delta\beta/2) * \|\mathbf{v}_i - \mathbf{e}_k\|;$$

$$\beta = \beta + \Delta\beta;$$

END IF

 Update the fuzzy membership using (16);

UNTIL (fuzzy membership do not change)

C Kernel based clustering

These approaches allow a non linear mapping of the input data. They map the data into a new space in such a way that computing a linear partitioning in this feature space results in a nonlinear partitioning in the input space.

1 Background

The power of kernel learning approaches rely on their ability to produce non-linear separating hypersurfaces between clusters by performing a mapping ϕ from the input space X to a high dimensional feature space F . One of the most relevant

aspects in kernel applications is that it is possible to compute Euclidean distances in F without knowing it explicitly. This can be done using the so called distance kernel trick [75]:

$$\begin{aligned}
(\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j))^2 &= (\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j))^t \cdot (\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)) \\
&= (\Phi(\mathbf{x}_i)^t \Phi(\mathbf{x}_i)) + (\Phi(\mathbf{x}_j)^t \Phi(\mathbf{x}_j)) - 2(\Phi(\mathbf{x}_i)^t \Phi(\mathbf{x}_j)) \\
&= K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)
\end{aligned} \tag{19}$$

In (19), $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^t \Phi(\mathbf{x}_j)$ is the Mercer kernel [92]. It is a symmetric function $K : X \times X \rightarrow \mathbb{R}$ and satisfies

$$\sum_{i=1}^N \sum_{j=1}^N c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \quad \forall n \geq 2, \tag{20}$$

where $c_r \in \mathbb{R} \forall r = 1, \dots, n$. Examples of Mercer kernels include [78]

- Linear

$$K^{(l)}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^t \cdot \mathbf{x}_j \tag{21}$$

- Polynomial of degree p

$$K^{(p)}(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^t \cdot \mathbf{x}_j)^p, \quad p \in \mathbb{N} \tag{22}$$

- Gaussian

$$K^{(g)}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{(\mathbf{x}_i - \mathbf{x}_j)^2}{2\sigma^2}\right), \quad \sigma \in \mathbb{R} \tag{23}$$

2 The kernel K-Means algorithm

In general, the mapping ϕ from the input space X to a high dimensional feature space F is not known. Thus, it is not possible to compute the center of each cluster i , \mathbf{a}_i^Φ , in the feature space F in a direct way using

$$\mathbf{a}_i^\Phi = \frac{1}{|\pi_i^\Phi|} \sum_{\mathbf{x}_h \in \pi_i^\Phi} \Phi(\mathbf{x}_h). \tag{24}$$

In (24), π_i^Φ is the set of points \mathbf{x}_h belonging to cluster i . Let γ be the indicator matrix in which γ_{ih} is equal to one if \mathbf{x}_h belongs to cluster i and zero otherwise. Alternatively, the distances in the feature space could be computed using

$$\begin{aligned} \|\Phi(\mathbf{x}_j) - \mathbf{a}_i^\phi\|^2 &= \|\Phi(\mathbf{x}_j) - \frac{1}{|\pi_i^\Phi|} \sum_{\mathbf{x} \in \pi_i^\Phi} \Phi(\mathbf{x}_h)\|^2 \\ &= K_{jj} - 2 \sum_h \gamma_{ih} K_{jh} + \sum_r \sum_s \gamma_{ir} \gamma_{is} K_{rs} \end{aligned} \quad (25)$$

The kernel K-Means algorithm [5] is summarized in algorithm 4

3 The Metric kernel Fuzzy C-Means algorithm

The Metric kernel Fuzzy C-Means [45] minimizes the following objective function:

$$J^\phi = \sum_{i=1}^C \sum_{j=1}^N (u_{ij})^m \|\Phi(\mathbf{x}_j) - \Phi(\mathbf{a}_i)\|^2 \quad (26)$$

subject to the constraint in (2). In (26), $\Phi(\mathbf{a}_i)$ is the center of cluster i in the feature space, and ϕ is the mapping from the input space X to the feature space F .

Algorithm 4 The kernel K-Means algorithm

Fix the number of clusters C ;

Initialize the cluster centers in the feature space F

Repeat

For each cluster, compute distances between the data points and the clusters centers using (25)

Update the indicator matrix γ by assigning data points to the nearest cluster according to the computed distances.

Until no change in γ

Minimization of the function in (26) has been proposed only in the case of a Gaussian kernel. The reason is that in this case the derivative with respect to the \mathbf{a}_i can use the kernel trick:

$$\frac{\delta K(\mathbf{x}_j, \mathbf{a}_i)}{\delta \mathbf{v}_i} = \frac{(\mathbf{x}_j - \mathbf{a}_i)}{\sigma^2} K(\mathbf{x}_j, \mathbf{a}_i). \quad (27)$$

It can be shown [45] that the update equation for the memberships is

$$u_{ij}^{-1} = \sum_{h=1}^C \left(\frac{1 - K(\mathbf{x}_j, \mathbf{a}_i)}{1 - K(\mathbf{x}_j, \mathbf{a}_h)} \right)^{1/(m-1)}, \quad (28)$$

and for the codevectors is

$$\mathbf{v}_i = \frac{\sum_{j=1}^N (u_{ij})^m K(\mathbf{x}_j, \mathbf{a}_i) \mathbf{x}_j}{\sum_{j=1}^N (u_{ij})^m K(\mathbf{x}_j, \mathbf{a}_i)}. \quad (29)$$

The Metric kernel Fuzzy C-Means algorithm is outlined in algorithm 5.

4 The Feature Space kernel Fuzzy C-Means algorithm

The Feature Space kernel Fuzzy C-Means algorithm [57] derives the fuzzy C-means in the feature space by minimizing

$$J^\phi = \sum_{i=1}^C \sum_{j=1}^N (u_{ij})^m \left(\Phi(\mathbf{x}_j) - \mathbf{a}_i^\phi \right)^2 \quad (30)$$

subject to the membership constraint in (2). It is possible to rewrite the norm in (30) explicitly by using:

$$\begin{aligned} \mathbf{a}_i^\phi &= \frac{\sum_{j=1}^N (u_{ij})^m \Phi(\mathbf{x}_j)}{\sum_{h=1}^N (u_{ih})^m} \\ &= b_i \sum_{j=1}^N (u_{ij})^m \Phi(\mathbf{x}_j) \end{aligned} \quad (31)$$

where

$$b_i = \frac{1}{\sum_{r=1}^N (u_{ir})^m} \quad (32)$$

This trick allows the derivation of a closed form expression for the membership updating equation

$$u_{ij}^{-1} = \sum_{h=1}^C \left[\frac{K_{jj} - 2b_i \sum_{r=1}^n (u_{ir})^m K_{jr} + b_i^2 \sum_{r=1}^n \sum_{s=1}^n (u_{is})^m K_{rs}}{K_{jj} - 2b_h \sum_{r=1}^n (u_{hr})^m K_{jr} + b_h^2 \sum_{r=1}^n \sum_{s=1}^n (u_{hs})^m K_{rs}} \right]^{1/(m-1)} \quad (33)$$

The Feature Space kernel Fuzzy C-Means algorithm is outlined in algorithm 6.

5 The kernelized Non-Euclidean Relational Fuzzy c-Means Algorithm

The kernelized Non-Euclidean Relational Fuzzy C-Means (kNERF) Algorithm [25] is a kernelized version of the non-Euclidean relational fuzzy C-means algorithm. This relational algorithm complements existing (object data) kernelization of the fuzzy C-means and, has a duality relationship with the kNERF algorithm introduced in [45]. In fact, most efforts to kernalize the C-means algorithms were formulated to work directly on object data. One of the most interesting aspects of the kNERF algorithm is that it is a kernelized version that works directly on relational data and does not require explicit feature representation.

Algorithm 5 The Metric kernel Fuzzy C-Means algorithm

Fix $m \in (1, \infty)$;

Fix the number of clusters C ;

Initialize the cluster centers \mathbf{a}_i ;

Repeat

Update the fuzzy memberships u_{ij} using (28);

Update the cluster centers \mathbf{a}_i using (29);

Until no change in u_{ij} .

Algorithm 6 The Feature Space kernel Fuzzy C-Means algorithm

Fix $m \in (1, \infty)$;

Fix the number of clusters C ;

Initialize the cluster centers \mathbf{a}_i ;

Repeat

 Update the fuzzy memberships u_{ij} using (31);

 Update the cluster centers \mathbf{a}_i using (33);

Until no change in u_{ij} .

The Gaussian kernelized relational matrix $\hat{\mathbf{R}}$ is formed directly from the original relational data matrix $\mathbf{R} = [\mathbf{r}_{jk}]$ using

$$\hat{\mathbf{R}} = 1 - \exp\left(-\frac{\mathbf{r}_{jk}^2}{\sigma^2}\right) \quad (34)$$

and the non-Euclidean relational fuzzy (NERF) c-means algorithm [80] is adapted to perform clustering analysis on this kernelized matrix $\hat{\mathbf{R}}$.

Despite the existence of a large number of kernel clustering algorithms [5] (e.g., kernel K-means, kernel FCM, kernel SOM, and kernel Neural Gas), the choice of a good kernel function and the adaptation of its parameters to the data remains a challenging task. For instance, in kNERF [25], a relational Gaussian kernel is used with one global scaling parameter for the entire data. The selection of the scaling parameter is discussed in [25] but there has been no attempt to devise methods to automatically select it. Moreover, a global scaling parameter may not be appropriate when clusters are of widely differing shapes, sizes, and densities.

Another approach to data clustering is based on spectral analysis. This approach called spectral clustering, solves a similar problem as kernel methods. However, it is based on a different approach that uses information contained in the eigenvectors of a data affinity matrix to detect structure.

D Spectral based clustering

Spectral analysis has proven effective on many tasks, including information retrieval [82], image segmentation [65], word class detection [49] and data clustering [55]. It has been proven that spectral clustering can be seen as a graph cut problem where an appropriate objective function has to be optimized [18]. The core of its theory is the eigenvalue decomposition of the Laplacian matrix due to the relationship between the smallest eigenvalues of the Laplacian and the graph cut [59].

Spectral clustering is a straight forward algorithm. In the first step, a similarity graph and its corresponding weighted adjacency matrix are constructed. The fully connected graph is very often used in connection with the Gaussian similarity function. The second step of the algorithm consists of computing the Laplacian. Then the smallest eigenvectors are computed and concatenated in order to constitute the new space feature matrix. The rows of this matrix are then clustered with the C-Means algorithm [91].

The three most well-known spectral clustering algorithms include the unnormalized cut [81], the Ncut [68], and the random walk [61]. These algorithms look rather similar, apart from the fact that they use three different graph Laplacians. In all three algorithms, the principle is to change the space of the data into a new representation. The graph Laplacian properties make this change of representation useful so that clusters can be easily detected in the new representation. These approaches are summarized in algorithm 7.

Spectral clustering can be derived as an approximation to graph partitioning problem. The two most common objective functions which encode the graph partitioning cut are Ratio cut, *Ratiocut*

$$Ratiocut(\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k) = \sum_{i=1}^k \frac{cut(\mathbf{A}_i, \bar{\mathbf{A}}_i)}{|\mathbf{A}_i|} \quad (35)$$

Algorithm 7 The Spectral Clustering algorithm

Input the feature matrix X , the number of clusters C

1. Construct a similarity graph and the weighted adjacency matrix \mathbf{R}_{ij} . The fully connected graph is very often used in connection with the Gaussian similarity function

$$\mathbf{R}_{ij} = \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}} & \text{if } i \neq j \\ 0 & \text{subject to } \mathbf{x}_i \text{ feature vector } i \end{cases} \quad (36)$$

2. Compute the degree matrix \mathbf{D}

$$\mathbf{D} = \text{diag} \left(\sum_j \mathbf{R}_{ij} \right) \quad (37)$$

3. Compute the Laplacian using one of the following equations

- Unnormalized graph Laplacian

$$\mathbf{L} = \mathbf{D} - \mathbf{R} \quad (38)$$

- Symmetric normalized graph Laplacian

$$\mathbf{L}_{sym} = \mathbf{D}^{-1} \mathbf{L} \mathbf{D}^{-1} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{R} \mathbf{D}^{-\frac{1}{2}} \quad (39)$$

- Random walk normalized graph Laplacian

$$\mathbf{L}_{rw} = \mathbf{D}^{-1} \mathbf{L} = \mathbf{I} - \mathbf{D}^{-1} \mathbf{R} \quad (40)$$

4. Compute the first C eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_C$. Let \mathbf{V} be the matrix containing the vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_C$.
5. For $i=1 \dots n$, let \mathbf{y}_i be the vector corresponding to the i^{th} row of \mathbf{v} . Cluster the points $(\mathbf{y}_i)_{i=1, \dots, n}$ with the K-mean algorithm into clusters C_1, C_2, \dots, C_C .

Output Cluster A_1, \dots, A_C with $A_i = \{j / Y_j \in C_i\}$

and the Normalized cut, $Ncut$

$$Ncut(\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_k) = \sum_{i=1}^k \frac{cut(\mathbf{A}_i, \bar{\mathbf{A}}_i)}{|\mathbf{A}_i|}. \quad (41)$$

In (35) and (41), $|\mathbf{A}_i|$ is the number of elements in \mathbf{A}_i , $vol(\mathbf{A}_i)$ is the sum of the similarities in \mathbf{A}_i , and $cut(\mathbf{A}_k, \bar{\mathbf{A}}_k) = \sum_{i \in \mathbf{A}_k, j \in \bar{\mathbf{A}}_k} \mathbf{R}_{ij}$. Relaxing Ncut leads to a normalized spectral clustering while relaxing Ratocut leads to unnormalized spectral clustering [18]. Normalized spectral clustering implements both clustering objectives (minimizing inter cluster similarities and maximizing intra cluster similarities) while unnormalized clustering implements only one objective (maximizing the intra cluster similarities). The within cluster distance, $\sum_{i,j \in \mathbf{A}} \mathbf{R}_{ij}$, can be rewritten as:

$$\sum_{i,j \in \mathbf{A}} \mathbf{R}_{ij} = \sum_{i \in \mathbf{A}, j \in \mathbf{A} \cup \bar{\mathbf{A}}} \mathbf{R}_{ij} - \sum_{i \in \mathbf{A}, j \in \bar{\mathbf{A}}} \mathbf{R}_{ij} \quad (42)$$

$$= vol(\mathbf{A}) - cut(\mathbf{A}, \bar{\mathbf{A}}) \quad (43)$$

Thus, the within cluster similarity is maximized if $cut(\mathbf{A}, \bar{\mathbf{A}})$ is small and if $vol(\mathbf{A})$ is large. This is part of the objective function of the normalized cut. However, for the case of Ratocut, the objective function is to maximize $|\mathbf{A}|$ and $|\bar{\mathbf{A}}|$ instead of $vol(\mathbf{A})$ and $vol(\bar{\mathbf{A}})$.

Spectral clustering could also be seen as a random walk technique. In fact, the transition probability of jumping from vertex V_i to vertex V_j in one step is equal to $\frac{\mathbf{R}_{ij}}{\mathbf{D}_i}$. Moreover, the transition matrix of the random walk is expressed as $\mathbf{P} = \mathbf{D}^{-1}\mathbf{R}$. Thus, \mathbf{L}_{rw} and \mathbf{P} are related:

$$\mathbf{L}_{rw} = \mathbf{I} - \mathbf{P}. \quad (44)$$

Ncut is also connected to the random walk, since $Ncut(\mathbf{A}, \mathbf{B}) = \mathbf{P}(\mathbf{x}_1 \in \mathbf{B} | \mathbf{x}_0 \in \mathbf{A})$ [18].

Spectral clustering can also be viewed as a kernel K-Means algorithm. In [33], the authors show that there is an equivalence between kernel K-Means and the

spectral clustering. For instance, in the case of the normalized cut, the correspondence with the objective function of C-Means restricts the kernel to be

$$\mathbf{K} = \alpha \mathbf{D}^{-1} + \mathbf{D}^{-1} \mathbf{A} \mathbf{D}^{-1}, \quad (45)$$

where α is a positive coefficient that guarantees the positive definiteness of K .

Another clustering approach, called diffusion clustering is highly related to spectral clustering. In fact, the random walk Laplacian, $\mathbf{D}^{-1} \mathbf{L}$, can be interpreted as a stochastic matrix representing random walk on the graph [61]. Then, the diffusion map, which is a mapping between the original space and the first k eigenvectors, can be defined as $\Psi(\mathbf{x}) = (\lambda_1 \psi_1^t, \lambda_2 \psi_2^t, \dots, \lambda_C \psi_C^t)$, where ψ_i^t are the right eigenvectors of the graph Laplacian and λ_i are the corresponding eigenvalues.

Some reference papers have compared the performance of different spectral algorithms [61]. The results suggest that some algorithms are more stable, have desirable properties, and have superior clustering results. In particular, reference paper [18] states that random walk is equivalent to diffusion clustering. Moreover, in [18], the authors show that normalized spectral clustering (Ncut) is better than the unnormalized one (Ratiocut) and that random walk is better than symmetric spectral clustering because \mathbf{L}_{sym} may lead to undesired artifacts on the eigenvectors [18].

Spectral algorithms can be very effective for clustering. However, as in kernel clustering, the choice of the Gaussian parameter σ is crucial for this approach. In fact, spectral clustering is quite sensitive to the changes in the similarity graph and to the choice of its parameters. Other limitations of spectral clustering include their sensitivity in the presence of background noise and multiscaled data. For the latter case, it was recommended to replace the uniform σ with a location dependent scale $\sigma(\mathbf{x}_j)$ [42]. In particular, in [42], the authors proposed to calculate a local scaling parameter σ_j for each data point \mathbf{x}_j and defined the similarity between a pair of points \mathbf{x}_j and \mathbf{x}_k as

$$\mathbf{S}_{jk} = \exp \left(-\frac{\text{dist}(\mathbf{x}_j, \mathbf{x}_k)^2}{\sigma_j \sigma_k} \right) \quad (46)$$

The local scale σ_j is defined in [42] as the distance between point \mathbf{x}_j and its P^{th} neighbor. Nevertheless, this approach is still dependent on the selection of the neighborhood parameter P which is fixed empirically for each dataset. Moreover, if \mathbf{x}_j is a noise point, adapting σ_j to this point will distort this fact.

Clustering is a difficult optimization task. The problem is more challenging for large and high dimensional data. One possible solution to alleviate this problem is to use partial supervision to guide the search process and narrow the space of possible solutions. Recently, semi-supervised clustering has emerged as a new research direction in machine learning to improve the performance of unsupervised learning using some supervision information.

E Semi-supervised clustering

In addition to the pairwise distance information used by unsupervised relational clustering, in many cases partial information is available in the form of cluster labels for few data samples or of pairs of samples that should or should not be assigned to the same cluster. The available knowledge is too far from being representative of a target classification of the items, so that supervised learning is not an option. This limited form of supervision can be used to guide the clustering process. The resulting approach is called semi-supervised clustering. Previous studies have empirically demonstrated the values of partial supervision in diverse domains, such as clustering [47], video surveillance [22], and text classification [43].

The most common way of incorporating supervision information in the unsupervised clustering process is by using constrained search based methods. In these methods, the clustering algorithm is modified to integrate a set of constraints. These constraints can guide the search for a more effective grouping of the data and avoid local minima. These modifications could be done by providing the algorithm with a better initialization, e.g., by performing a transitive closure [48]. They could also be done by integrating a penalty or a reward term in the cost function of the algorithm [70]. Another way of incorporating the constraints in search based approaches is by satisfying the constraints in the assignment step of the clustering process [69].

1 The Semi-supervised kernel C-means algorithm

In [27], a relational semi-supervised kernel clustering algorithm (SS-kernel-CMeans) was proposed. It minimizes the following objective function

$$J = \sum_{i=1}^C \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \pi_i} \frac{\sum_{j=1}^N \sum_{k=1}^N K_{jk}}{2|\pi_i|} - \sum_{i=1}^C \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in M_l} \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \pi_i} \frac{2w_{jk}}{|\pi_i|} + \sum_{i=1}^C \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in C_l} \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \pi_i} \frac{2w_{jk}}{|\pi_i|}. \quad (47)$$

In (47), M_l is a set of must link constraints such that $(\mathbf{x}_j, \mathbf{x}_k) \in M_l$ implies that objects \mathbf{x}_j and \mathbf{x}_k must be assigned to the same cluster. Similarly, C_l a set of cannot link constraints such that $(\mathbf{x}_j, \mathbf{x}_k) \in C_l$ implies that objects \mathbf{x}_j and \mathbf{x}_k cannot be assigned to the same cluster.

In (47), π_i is the set of points belonging to cluster i , $|\pi_i|$ its corresponding cardinality, and N and C are the number of points and the number of clusters respectively, and K_{jk} is the relational Gaussian kernel function. Let \mathbf{W} be the constraint matrix such that

$$\mathbf{W}_{ij} = \begin{cases} -w_{ij} & \text{if } (\mathbf{x}_j, \mathbf{x}_k) \in C_l \\ w_{ij} & \text{if } (\mathbf{x}_j, \mathbf{x}_k) \in M_l \\ 0 & \text{otherwise} \end{cases} \quad (48)$$

Let \mathbf{z}_i be an indicator vector for cluster i . This vector is of length N and $\mathbf{z}_i(j) = 0$ if \mathbf{x}_j is not in cluster i , and 1 otherwise. The objective in (47) can be written as follow

$$J = \sum_{i=1}^C \sum_{(\mathbf{x}_j, \mathbf{x}_k) \in \pi_i} \frac{\sum_{j=1}^N \sum_{k=1}^N \mathbf{z}_j^t (\mathbf{K}_{jk} + 2\mathbf{W}_{jk}) \mathbf{z}_k}{\mathbf{z}_j \mathbf{z}_k^t} \quad (49)$$

By setting $\hat{\mathbf{K}} = \mathbf{K}_{jk} + 2\mathbf{W}_{jk}$, the authors in [27] showed that the SS-kernel-CMeans objective function in (49) is equivalent to the kernel k-means one. The steps of the Semi-supervised kernel C-means are outlined in algorithm 8

Algorithm 8 The Semi-supervised kernel C-means algorithm

Input:

K: input similarity matrix,

W: constraint penalty matrix,

C: number of clusters

Output:

$\{\pi_i\}$: final partitioning of the points

Repeat

Form the matrix $\hat{\mathbf{K}} = \mathbf{K} + 2\mathbf{W}$.

For each cluster, compute distances between the data points and the clusters centers using (25)

Update the indicator matrix \mathbf{z}_i by assigning data points to the nearest cluster according to the computed distances.

Until no change in \mathbf{z}_i

The Semi-supervised kernel C-means has several drawbacks. First, since it is crisp version, it is sensitive to initialization and cannot deal with overlapping boundaries. Second, no method has been suggested to automate the selection of the scaling parameter and this parameter has to be set manually. Moreover, it is not clear how to generate the constraints weights \mathbf{W}_{jk} .

2 The Semi-Supervised Spectral clustering algorithm

While spectral algorithms have been very useful in unsupervised learning clustering, little work has been done in developing semi-supervised spectral clustering. One spectral approach to semi-supervised clustering is the spectral learning algorithm [44]. In this approach, the authors present a simple extension to the spectral learn-

ing algorithm to take advantage of available supervision information. This algorithm does not have an explicit underlying objective function. It simply injects the pairwise constraints into the affinity matrix before clustering. In fact, since for most similarity functions, the maximum pairwise similarity value is 1, and the minimum similarity is 0, the authors assigned 1 to the affinity matrix entries corresponding to a *Must-Link* pairs of points and 0 to the affinity matrix entries corresponding to a *Cannot-Link* pairs of points. The Semi-Supervised spectral learning steps are outlined in algorithm 9.

Algorithm 9 The Semi-Supervised spectral learning algorithm

Input the feature matrix X , the number of clusters C

1. Construct a similarity graph and the weighted adjacency matrix \mathbf{R}_{ij} . The fully connected graph is very often used in connection with the Gaussian similarity function

$$\mathbf{R}_{ij} = \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}} & \text{if } i \neq j \\ 0 & \text{subject to } \mathbf{x}_i \text{ feature vector } i \end{cases} \quad (50)$$

2. Assign 1 to the affinity matrix entries corresponding to a *Must-Link* pairs of points and 0 to the affinity matrix entries corresponding to a *Cannot-Link* pairs of points.

2. Compute the degree matrix D

$$D = \text{diag} \left(\sum_j \mathbf{R}_{ij} \right) \quad (51)$$

3. Compute the Laplacian;
4. Compute the first C eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_C$. Let V be the matrix containing the vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_C$.
7. For $i=1 \dots n$, let \mathbf{y}_i be the vector corresponding to the i^{th} row of V . Cluster the points $(\mathbf{y}_i)_{i=1, \dots, n}$ with the K-mean algorithm into clusters C_1, C_2, \dots, C_C .

Output Cluster A_1, \dots, A_C with $A_i = \{j/Y_j \in C_i\}$

Algorithm 10 The Semi-Supervised graph clustering algorithm

Input the feature matrix X , the number of clusters C

1. Construct a similarity graph and the weighted adjacency matrix R_{ij} . The fully connected graph is very often used in connection with the Gaussian similarity function

$$\mathbf{R}_{ij} = \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}} & \text{if } i \neq j \\ 0 & \text{subject to } \mathbf{x}_i \text{ feature vector } i \end{cases} \quad (52)$$

2. Compute the degree matrix D

$$\mathbf{D} = \text{diag} \left(\sum_j \mathbf{R}_{ij} \right) \quad (53)$$

3. Compute the Laplacian.
4. Construct the constraint matrix \mathbf{W} such that \mathbf{W}_{ij} is $-w_{ij}$ for a cannot link, w_{ij} for a must link, and 0 otherwise.
5. Compute the matrix

$$\mathbf{L} = \mathbf{L} + \mathbf{W} \quad (54)$$

6. Compute the first C eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_C$. Let V be the matrix containing the vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_C$.
7. For $i=1 \dots n$, let \mathbf{y}_i be the vector corresponding to the i^{th} row of \mathbf{v} . Cluster the points $(\mathbf{y}_i)_{i=1, \dots, n}$ with the K-mean algorithm into clusters C_1, C_2, \dots, C_C .

Output Cluster A_1, \dots, A_C with $A_i = \{j / Y_j \in C_i\}$

Recently, the authors in [41] considered a semi-supervised formulation of the normalized cut objective that had a spectral algorithm associated with it. In this work, only *Must-Link* constraints are considered in the formulation, and penalty weights for constraint violations are not considered. Moreover, it solves an expensive constrained eigen-decomposition problem.

The recent theoretical connection between weighted kernel k-means and several

graph clustering objectives has lead to algorithms for optimizing the semi-supervised graph clustering algorithm [1]. The steps of the semi-supervised graph clustering algorithm are outlined in algorithm 10.

CHAPTER III

UNSUPERVISED RELATIONAL CLUSTERING WITH LOCAL SCALE PARAMETERS

A Introduction

Recently, relational clustering has become an active field of research due to its ability to use the adjacency structure of the data and avoid dealing with a prefixed shape of clusters. Relational clustering can be seen as kernel based approaches since a kernel function can be thought of as a pairwise dissimilarity function. The choice of such a kernel function allows the mapping of the input data into a new space in such a way that computing nonlinear partitioning in the input space can reduce to a simple partitioning in the feature space.

One of the most common dissimilarity function, due to its analytical proprieties, is the Gaussian kernel function. Although good results were obtained using this kernel, generally, its performance depends on the selection of the scaling parameter σ . This selection is commonly done by trying several values. Moreover, since one global parameter is used for the entire dataset, it may not be possible to find one optimal σ when there are large variations between the distributions of the different clusters in the feature space.

In Figure 1, we use a simple example to motivate the advantage of cluster dependent kernel resolution σ_i compared to one global σ . As this data contains two clusters, we construct a Gaussian kernel with respect to each cluster using a different scaling parameter. In particular, we run kNERF [25] multiple times using all pairwise combinations of $\sigma_i = 0.001, 0.01, 0.02, 0.03 \dots, 0.2$.

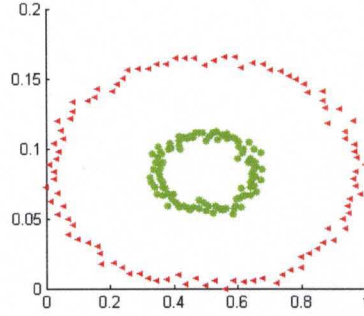


Figure 1. 2-D dataset with 2 clusters with different densities

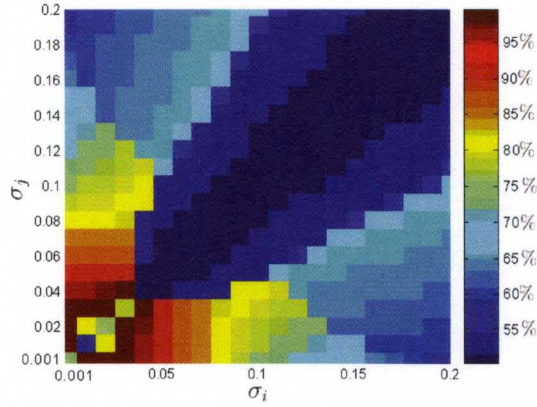


Figure 2. Accuracy results obtained on the dataset of Figure 1 using an extensive search of cluster dependent scaling parameters

Figure 2 shows the accuracy results as heat map obtained using several combinations of σ_1 and σ_2 . The colors toward “red” indicate a high accuracy and colors toward “blue” indicate a low accuracy. We can see from Figure 2 that the clustering results corresponding to the specific cases where $\sigma_1 = \sigma_2$ (i.e. along the diagonal) cannot achieve accuracy better than 80%. On the other hand, for different combinations of σ (such as $\sigma_i = 0.001$ and $\sigma_j = 0.03$), an accuracy of 100% can be achieved. This simple example confirms that one global scaling parameter cannot deal effectively with this data and a cluster dependent scaling parameter σ_i is more appropriate.

One way to learn optimal scaling parameters is to try several combination (as

illustrated in the above example), evaluate each partition by using some validity measure [20], and identify the optimal partition. However, this exhaustive search of one scaling parameter with respect to each cluster is not practical. It is computationally expensive and increases significantly with the number of clusters and the range of possible values of σ_i .

In this chapter, we introduce new fuzzy relational clustering techniques that learns cluster dependent σ_i in an efficient way through optimization of an objective function. The proposed algorithms learn the underlying cluster dependent dissimilarity measure while finding compact clusters in the given data. The learned measure is a local dissimilarity based on the Gaussian kernel function, which is highly related to the heat kernel equation [58].

B Relational clustering with local scale parameters

1 The clustering and Local Scale Learning algorithm

In the following, we assume that $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ is a set of N data points to be partitioned into C clusters. We also assume that $\mathbf{R} = [\mathbf{r}_{jk}]$ is a relational matrix where \mathbf{r}_{jk} represents the degree to which pairs of objects \mathbf{x}_j and \mathbf{x}_k are related. The matrix \mathbf{R} could be given or it could be constructed from the features of the objects. Each object \mathbf{x}_j belongs to cluster i with a fuzzy membership u_{ij} that satisfies

$$0 \leq u_{ij} \leq 1, \text{ and } \sum_{i=1}^C u_{ij} = 1, \text{ for } i, j \in \{1, \dots, N\}. \quad (55)$$

To simplify notation, in the rest of the thesis, we will drop the square from \mathbf{r} and σ and use \mathbf{r} to denote \mathbf{r}^2 and σ to denote σ^2 . That is, our kernel distance would be defined as $\mathbf{D}_{jk}^i = 1 - \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right)$ instead of $\mathbf{D}_{jk}^i = 1 - \exp\left(-\frac{\mathbf{r}_{jk}^2}{\sigma_i^2}\right)$.

The clustering and Local Scale Learning algorithm (LSL) minimizes

$$J = \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \left(1 - \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right)\right) - \sum_{i=1}^C \frac{K_1}{\sigma_i^2} \quad (56)$$

subject to the membership constraint in (55).

In (56), $m \in (1, \infty)$ is the fuzzifier. The term $u_{ij}^m u_{ik}^m$ can be regarded as the likelihood that two points \mathbf{x}_j and \mathbf{x}_k belong to the same cluster i . We use β_{jk}^i to denote this

term. That is, we let

$$\beta_{jk}^i = u_{ij}^m u_{ik}^m. \quad (57)$$

The LSL algorithm is based on minimizing a joint objective function with two terms. The first term seeks compact clusters using a local relational distance, \mathbf{D}_{jk}^i , with respect to each cluster i . This distance is defined as

$$\mathbf{D}_{jk}^i = 1 - \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) \quad (58)$$

In (58), \mathbf{D}_{jk}^i is based on the Gaussian kernel function and is intimately related to the heat flow [58]. In fact, locally, the heat kernel is approximately equal to the Gaussian, i.e.,

$$\mathbf{H}_\sigma(\mathbf{x}, \mathbf{y}) \approx (4\pi\sigma)^{-\frac{p}{2}} \exp\left(-\frac{d(\mathbf{x}, \mathbf{y})}{4\sigma}\right) \quad (59)$$

when the squared distance between \mathbf{x} and \mathbf{y} , $d(\mathbf{x}, \mathbf{y})$, is sufficiently small [58].

In (58), the scaling parameter σ_i controls the rate of decay of \mathbf{D}_{jk}^i as a function of the distance between \mathbf{x}_j and \mathbf{x}_k with respect to cluster i . Using a cluster dependent scaling parameter σ_i allows LSL to deal with the large variations in the feature space between the distributions and the geometric characteristics of the different clusters.

The second term in (56) is a regularization term to avoid the trivial solution where all the scaling parameters σ_i are infinitely large. In fact, without this term, minimizing (56) with respect to σ_i gives the trivial solution of a very large σ_i that merges all points into a single cluster. Another trivial solution that minimizes the objective function in (56) is when one of σ_i is zero. In order to keep the derivation simple, we do not introduce another regulation term. We simply assume that σ_i is not null. Later in this section, we will discuss how to handle this case when it happens.

The goal of the proposed LSL algorithm is to learn the C clusters, the scaling parameters σ_i of each cluster, and the membership values u_{ij} , of each sample \mathbf{x}_j in each cluster i . We achieve this by optimizing the objective function in (56) with respect to σ_i and u_{ij} . In order to optimize (56) with respect to u_{ij} , we use the relational dual of the fuzzy C-mean algorithm formulated by Hathaway et al. [85]. It has been proven in [85] that the Euclidean distance $d_{ik}^2 = \|\mathbf{x}_k - \mathbf{c}_i\|^2$, from feature \mathbf{x}_k

to the center of the i^{th} cluster, \mathbf{c}_i , can be written in terms of the relational matrix \mathbf{D}^i as

$$d_{ik}^2 = (\mathbf{D}^i \mathbf{v}_i)_k - \frac{\mathbf{v}_i^t \mathbf{D}^i \mathbf{v}_i}{2}, \quad (60)$$

where \mathbf{v}_i is the membership vector of all N samples in cluster i defined by

$$\mathbf{v}_i = \frac{(u_{i1}^m, \dots, u_{iN}^m)^t}{\sum_{j=1}^N u_{ij}^m} \quad (61)$$

Using the implicit distance values, d_{ik}^2 , the objective function in (56) could be rewritten as

$$J = \sum_{i=1}^C \sum_{j=1}^N u_{ij}^m \cdot d_{ij}^2 - \sum_{i=1}^C \frac{K_1}{\sigma_i^2} \quad (62)$$

To optimize J with respect to u_{ij} subject to (55), we use the Lagrange multiplier technique and obtain

$$J = \sum_{i=1}^C \sum_{j=1}^N u_{ij}^m \cdot d_{ij}^2 - \sum_{i=1}^C \frac{K_1}{\sigma_i^2} - \sum_{j=1}^N \lambda_j \left(\sum_{i=1}^C u_{ij} - 1 \right) \quad (63)$$

By setting the gradient of J to zero, we obtain

$$\frac{\delta J}{\delta \lambda} = \sum_{i=1}^C u_{ij} - 1 = 0 \quad (64)$$

and

$$\frac{\delta J}{\delta u_{ij}} = m u_{ij}^{m-1} d_{ij}^2 - \lambda = 0 \quad (65)$$

Solving (65) for u_{ij} yields

$$u_{ij} = \left(\frac{\lambda}{m d_{ij}^2} \right)^{\frac{1}{(m-1)}} \quad (66)$$

Substituting (66) back into (64), we obtain

$$\sum_{i=1}^C u_{ij} = \left(\frac{\lambda}{m} \right)^{\frac{1}{m-1}} \sum_{i=1}^C \left(\frac{1}{d_{ij}^2} \right)^{\frac{1}{m-1}} = 1 \quad (67)$$

Thus,

$$\left(\frac{\lambda}{m}\right)^{\frac{1}{m-1}} = \frac{1}{\sum_{i=1}^C \left(\frac{1}{d_{ij}^2}\right)^{\frac{1}{m-1}}}. \quad (68)$$

Substituting this expression back in (66), we obtain

$$u_{ij} = \frac{\left(\frac{1}{d_{ij}^2}\right)^{\frac{1}{m-1}}}{\sum_{t=1}^C \left(\frac{1}{d_{tj}^2}\right)^{\frac{1}{m-1}}}. \quad (69)$$

Simplifying (69), we obtain the following update equation

$$u_{ij} = \frac{1}{\sum_{t=1}^C (d_{ij}^2/d_{tj}^2)^{\frac{1}{m-1}}}. \quad (70)$$

We notice from equations (60) and (70) that the expression of u_{ij} does not depend on any notion of cluster prototype (e.g center). In fact, it depends only on the relational matrix \mathbf{R} , and the normalized membership vector \mathbf{v} .

In the objective function in (56), the resolution of the different clusters σ_i are independent of each other. Thus, in order to optimize (56) with respect to σ_i , we can reduce the optimization problem to C independent problems. That is, we convert the objective function in (56) to the following C simpler functions

$$J^i = \sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \left(1 - \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right)\right) - \frac{K_1}{\sigma_i^2} \quad (71)$$

for $i = 1, \dots, C$. The optimal update equation of the scaling parameters σ_i can be obtained using the Lagrange method by solving

$$\frac{\partial J^i}{\partial \sigma_i} = -\sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \frac{\mathbf{r}_{jk}}{\sigma_i^2} \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) + \frac{2K_1}{\sigma_i^3} = 0 \quad (72)$$

Using the duality between the Gaussian similarity and the heat flow function, it has been shown in [58] that

$$\sum_{\substack{j \\ \mathbf{r}_{jk} < \epsilon}} \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) = \frac{N}{(\pi\sigma_i)^{-\frac{p}{2}}} \quad (73)$$

when ϵ is sufficiently small. In (73), p is the dimension of the manifold. In the worst case, the point located in the neighborhood of a point j within the radius ϵ are distant from j with at most ϵ . As ϵ is sufficiently small, we can assume that the distances \mathbf{r}_{jk} are almost constant in the neighborhood of j . Thus, (73) can be rewritten as

$$\exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) = \frac{N}{|\mathcal{N}| (\pi\sigma_i)^{-\frac{p}{2}}} \quad (74)$$

where $|\mathcal{N}|$ is the cardinality of the neighborhood of j . Substituting (74) in (72) gives

$$\frac{\partial J^i}{\partial \sigma_i} = -\sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \frac{N \cdot \mathbf{r}_{jk}}{2\pi^{2-\frac{p}{2}} |\mathcal{N}| (\sigma_i)^{2-\frac{p}{2}}} + \frac{2K_1}{\sigma_i^3} \quad (75)$$

Setting (75) to zero and solving for σ_i , we obtain

$$\sigma_i = \left(\frac{K}{\sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \mathbf{r}_{jk}} \right)^{\frac{2}{2+p}} \quad (76)$$

where

$$K = K_1 \frac{2\pi^{2-\frac{p}{2}} |\mathcal{N}|}{N} \quad (77)$$

We should recall here that our assumption about a non null σ_i is reasonable. In fact, the update equation in (76) shows that σ_i is zero only when the distance \mathbf{r}_{jk} between two points, \mathbf{x}_j and \mathbf{x}_k , that belong to cluster i (non zero fuzzy memberships u_{ij} and u_{ik}) is infinitely large. This scenario is very unlikely.

We notice from (76) and (58) that for each cluster i , σ_i controls the rate of decay of \mathbf{D}_{jk}^i with respect to the distance \mathbf{r}_{jk} . In fact, σ_i is inversely proportional to the intra-cluster distances with respect to each cluster i . Thus, when the intra-cluster dissimilarity is small, σ_i is large allowing the pairwise distances over the same cluster to be smaller and thus obtain a more compact cluster. On the other hand, when the intra-cluster dissimilarity is high, σ_i is small to prevent points which are not highly similar from being mapped to the same location. According to (76), σ_i can also be seen as the average time to move between points in cluster i .

The resulting LSL approach is outlined in algorithm 11.

2 Performance illustration

To illustrate the ability of LSL to learn appropriate local scaling parameters and cluster the data simultaneously, we use it to partition synthetic 2D datasets. We should mention here that, for the purpose of visualizing the results, we use feature based and 2-dimensional data. Relational dissimilarity matrices are obtained by computing the Euclidean distance between these feature vectors. We use 5 datasets that

include categories of different shapes with unbalanced sizes and densities. Figure 3 displays the 5 synthetic datasets. Each cluster is displayed with a different color.

For the five datasets, we set the number of clusters C to the true one (see Figure 3), the fuzzifier m to 1.1, and the maximum number of iterations to 100. As LSL requires the specification of one parameter K , we repeat the clustering process with $K = [0.001, 0.01, 0.05, 0.1, 0.5, 1, 1.5, 2, 4, 8, 10]$ and select the best results. The matrix of fuzzy memberships is initialized randomly.

Algorithm 11 The LSL algorithm

Fix number of clusters C , $K \in]0 \infty)$, and $m \in [1 \infty)$;

Initialize the fuzzy partition matrix U ;

Initialize the scaling parameters σ_i ;

REPEAT

 Compute the dissimilarity D^i for all the clusters using (58) ;

 Compute the membership vectors \mathbf{v}_i using (61);

 Compute the distances using (60);

 Update the fuzzy membership using (70);

 Update the scaling parameter σ_i using (76);

UNTIL (fuzzy membership do not change or maximum number of iteration is reached)

Table 1 displays the clustering results and the scaling parameters learned by LSL on dataset 1. First, we notice that the estimated cluster dependent kernel parameters reflect the geometry of the data. More specifically, they are related to the densities of the clusters. In fact, the smallest scaling parameter is found for cluster 3 which is the least dense cluster, and σ_1 and σ_2 are comparable. This is due to the fact that cluster 1 and cluster 2 have comparable geometric characteristics. By learning appropriate scaling parameters, LSL partitions this data correctly. Figure 4 displays the fuzzy memberships with respect to each cluster. We can notice that

most membership values tend to be binary (either 0 or 1).

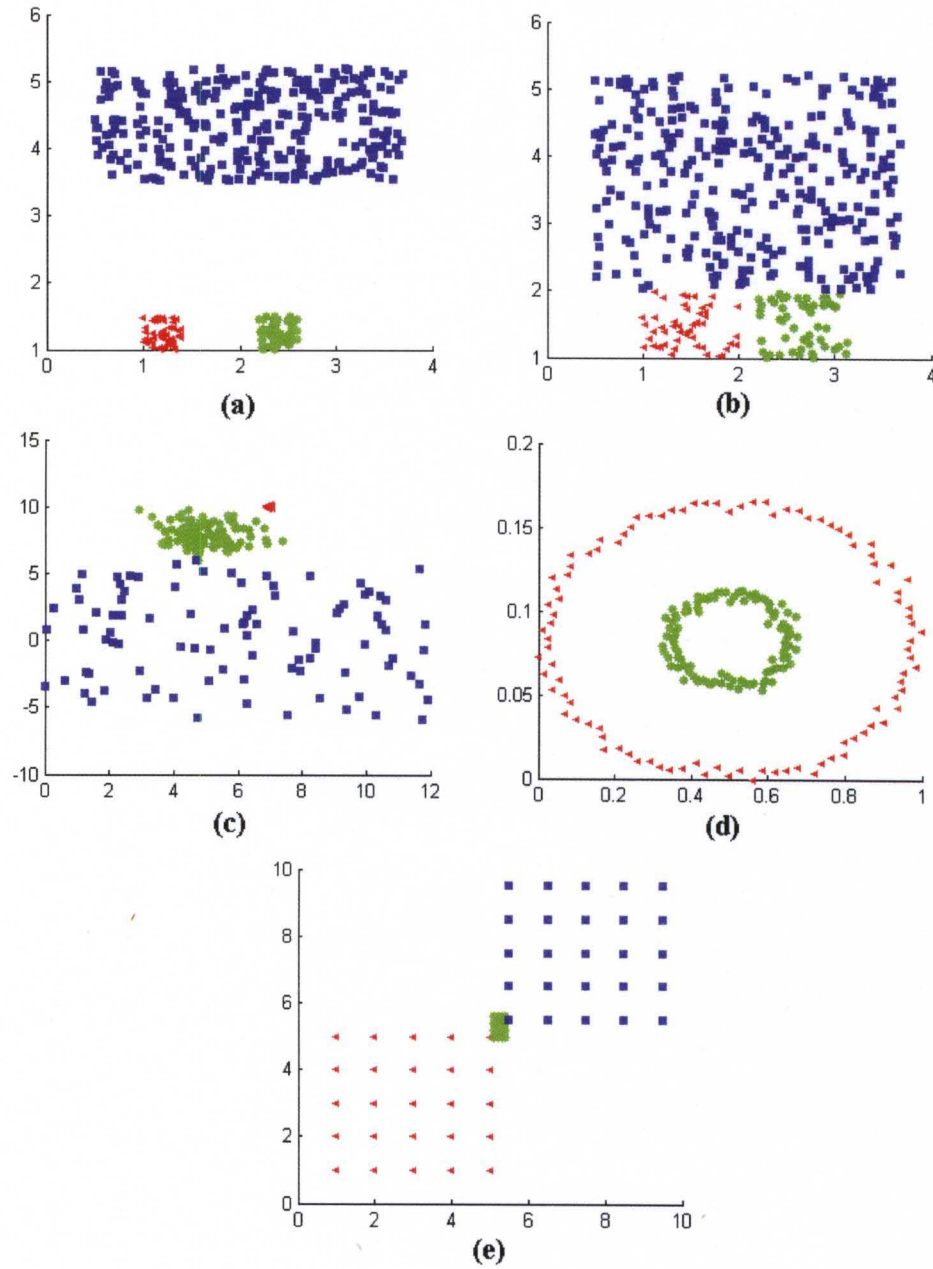


Figure 3. Datasets used to illustrate the performance of LSL. Each cluster is shown by a different color.

This means that the learned scaling parameters have mapped the data to well separated clusters in the feature space.

TABLE 1

Partition and scaling parameters learned by LSL for dataset 1 displayed in Figure 3 (a) when $K = 0.01$

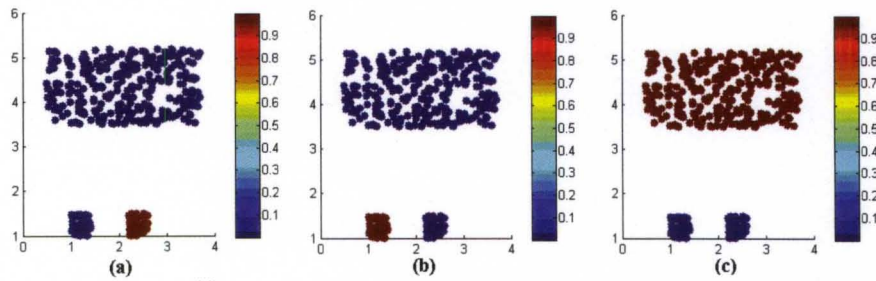
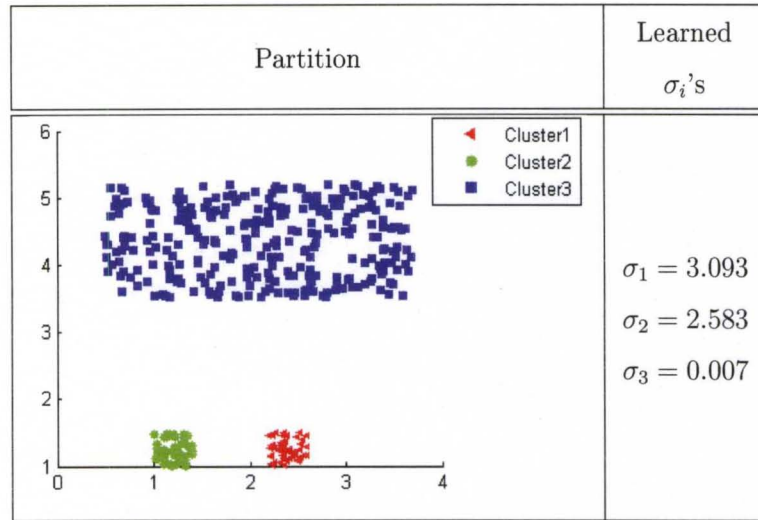


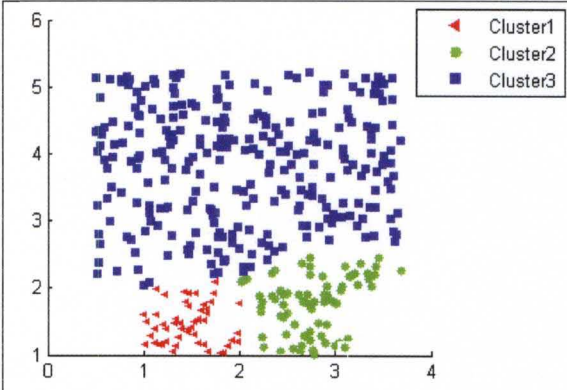
Figure 4. Fuzzy memberships learned by LSL on dataset 1 (Figure 3 (a)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

The partitioning of dataset 2 is reported in Table 2. For this example, the three clusters have the same density. As a result, LSL learns three scaling parameters that are comparable. However, some points along the cluster boundaries are not correctly

categorized. In fact, as shown in Figure 5, the returned fuzzy memberships of cluster 2 and cluster 3 are too fuzzy (between 0.3 and 0.33). This is due to the characteristic of this dataset where the boundaries are not well defined.

TABLE 2

Partition and scaling parameters learned by LSL for dataset 2 displayed in Figure 3 (b) when $K = 0.05$

Partition	Learned σ_i 's
	$\sigma_1 = 0.039$ $\sigma_2 = 0.041$ $\sigma_3 = 0.041$

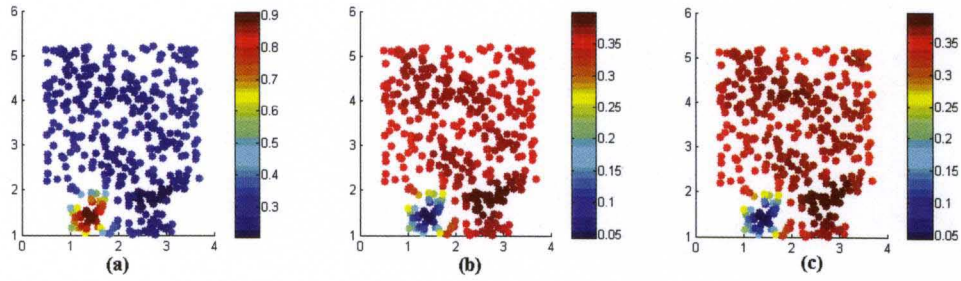
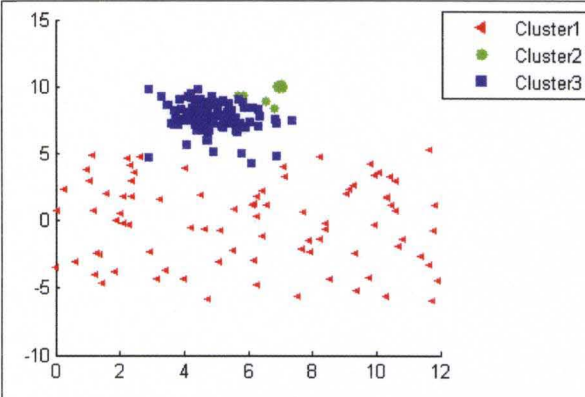


Figure 5. Fuzzy memberships learned by LSL on dataset 2 (Figure 3 (b)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

TABLE 3

Partition and scaling parameters learned by LSL for dataset 3 displayed in Figure 3 (c) when $K = 4$.

Partition	Learned σ_i 's
	$\sigma_1 = 2.887$ $\sigma_2 = 53.33$ $\sigma_3 = 25.60$

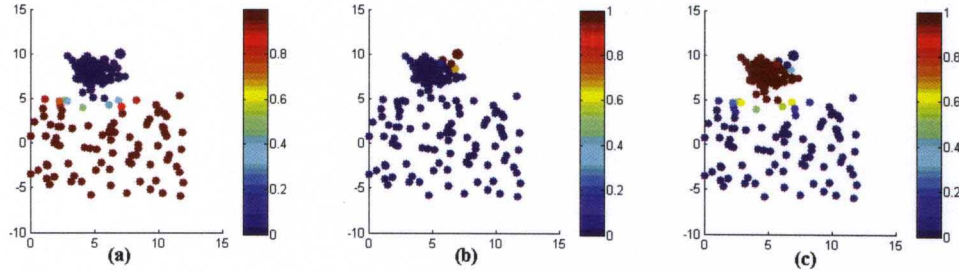


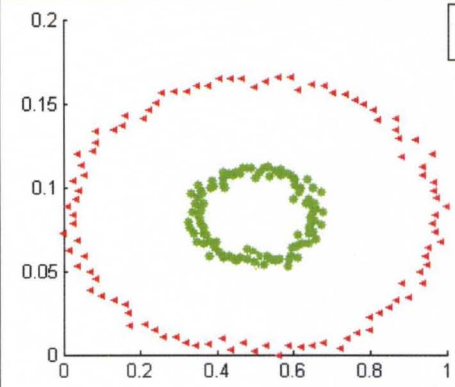
Figure 6. Fuzzy memberships learned by LSL on dataset 3 (Figure 3 (c)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

A similar analysis on the learned scaling parameters for dataset 3 is conducted and reported in Table 3. As it can be seen, the sparse cluster, cluster 1, has the smallest scaling parameter ($\sigma_1 = 2.887$). Cluster 3 which is less sparse than cluster 1 has smaller σ , ($\sigma_3 = 25.6$), and the densest cluster, cluster 2, has the highest σ ($\sigma_2 = 53.33$). From Table 3, we observe that some boundary points are not categorized correctly. However, as it can be seen in Figure 6, the fuzzy membership of these points

is around 0.5. Thus, the learned memberships can be used to identify these boundary points.

TABLE 4

Partition and scaling parameters learned by LSL for dataset 4 displayed in Figure 3 (d) when $K = 1.5$

Partition	Learned σ_i 's
	$\sigma_1 = 0.014$ $\sigma_2 = 0.029$

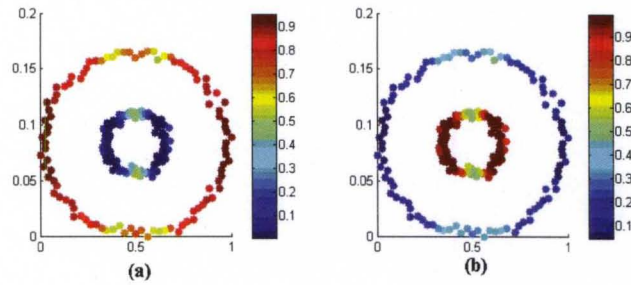


Figure 7. Fuzzy memberships learned by LSL on dataset 4 (Figure 3 (d)) with respect to (a) cluster 1, and (b) cluster 2.

Although it is hard to define the notion of density on dataset 4 (Table 4), we can notice that the two learned scaling parameters ($\sigma_1 = 0.0014$ and $\sigma_2 = 0.029$) are meaningful. In fact, as cluster 2 is slightly denser than cluster 1, σ_2 is slightly higher

than σ_1 . We should mention here that the learned scaling parameters are equal to the ones found by extensive search in section III-A. This shows the efficiency of LSL in learning the scaling parameters.

Figure 7 shows that some points have fuzzy memberships around 0.5, indicating that they are close to both clusters in the mapped feature space. This is an inherent limitation of the LSL since it implicitly uses the Euclidean distance to map the data

TABLE 5

Partition and scaling parameters learned by LSL for dataset 1 displayed in Figure 3 (a) when $K = 2$.

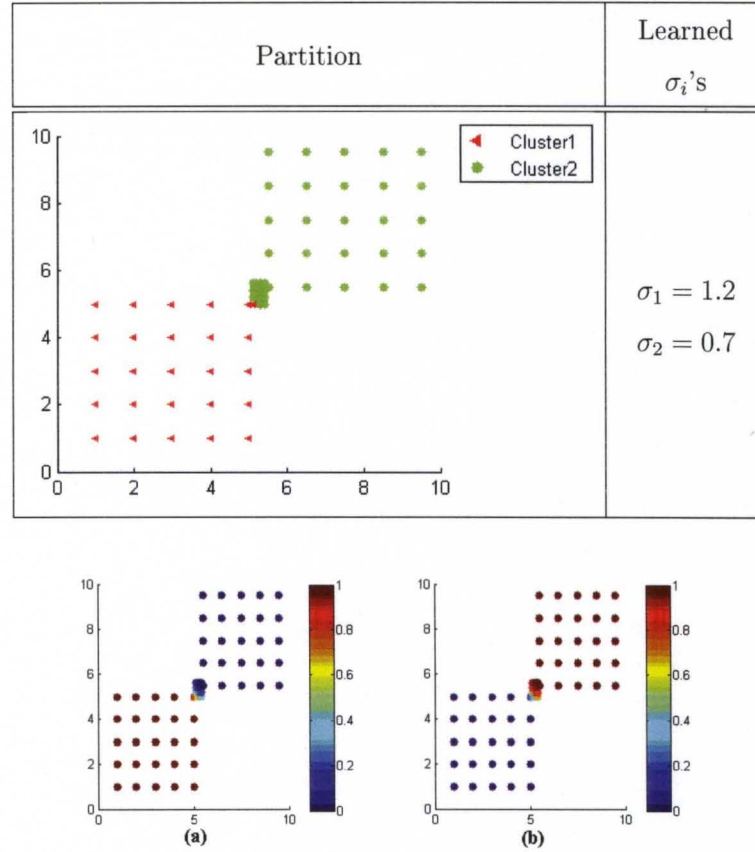


Figure 8. Fuzzy memberships learned by LSL on dataset 5 (Figure 3 (e)) with respect to (a) cluster 1 and (b) cluster 2

As displayed in Table 5, LSL cannot categorize dataset 5 correctly. This is due to the fact that it optimizes only the intra cluster dissimilarity. In fact, since the intra cluster dissimilarity of the sparse cluster is smaller than its inter cluster dissimilarity with the denser one, the two cluster are merged together by the LSL algorithm. This constitute one drawback of the LSL approach. In fact, this is an inherited limitation of all clustering algorithms that minimize the intra-cluster distances without considering the inter-cluster distances.

3 Limitations of LSL

We can notice from the update equation (76) that σ_i depends on one parameter, K . This parameter is a function of the balancing constant K_1 and the characteristics of the dataset (Eq. 77). Although, LSL learns C scaling parameters with the specification of one parameter, this is still a limitation of this approach. In fact, the choice of the parameter K can have a significant impact on the final partition. For instance, if we consider the case of dataset 1 (Figure 3 (a)), the good result displayed in Table 1 is obtained for $K = 0.05$. However, if we consider other different values of K as in Figure 9, the clustering results are not meaningful.

In order to address this drawback, in the next section, we propose a new approach that avoids the need to specify K by exploring the inter cluster dissimilarities in addition to the intra cluster dissimilarities.

C Relational clustering by optimizing both the intra-cluster and inter-cluster distances

1 The Fuzzy clustering with learnable cluster dependent kernels algorithm

Although, the LSL algorithm learns C scaling parameters that reflect the geometric characteristics of the dataset, we have shown in the previous section that it has two limitations. First, as it minimizes only the intra-cluster dissimilarity, the scaling parameter reflects only the intra-cluster characteristics of the data. Thus,

LSL may not categorize data with small inter-cluster distances correctly. Second, its performance depends on the specification of one parameter, K .

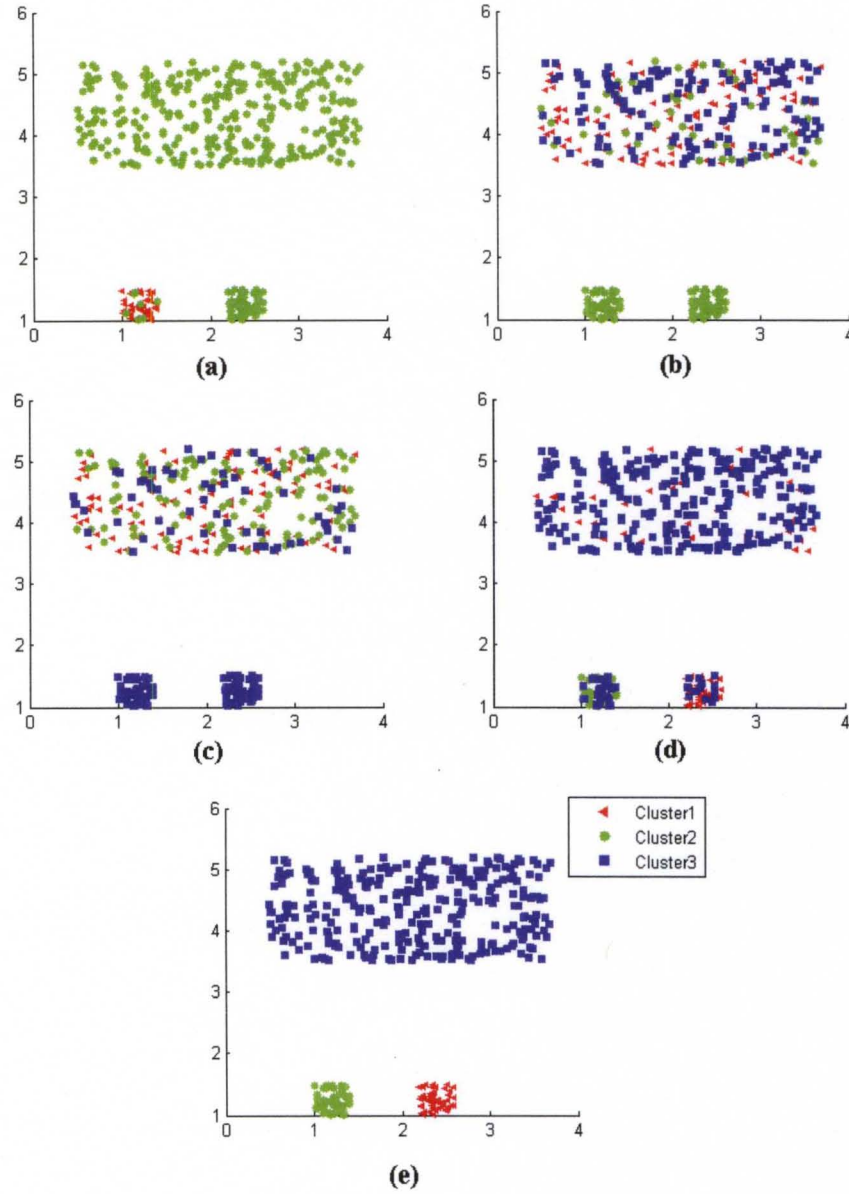


Figure 9. LSL clustering results on dataset 1 (Figure 3 (a)) with different parameters K , (a) $K = 0.001$, (b) $K = 0.01$, (c) $K = 0.1$, (d) $K = 1$, and (e) $K = 0.05$.

To address the above limitations, we propose an alternative approach to learn

local kernels by minimizing the intra-cluster distances and maximizing the intra-cluster distances simultaneously. This is consistent with the basic definition of cluster analysis: seeks to partition data into groups by minimizing within-group dissimilarities or by maximizing between-group dissimilarities. Thus, if we consider the Gaussian dissimilarity function as defined in (58), the optimal choice of the scaling parameter is the one that allows high intra-cluster dissimilarity and low inter-cluster dissimilarity. The proposed algorithm, called Fuzzy clustering with learnable cluster dependent kernels (FLeCK), learns a scaling parameter with respect to each cluster that allows it to distinguish and separate the cluster's objects from the rest of the data. By considering one cluster at a time, FLeCK learns a scaling parameter σ_i from the intra-cluster characteristics of cluster i and its relative dissimilarities with the remaining objects on the dataset.

Using the same notation defined in subsection III-B-1, the FLeCK algorithm minimizes the intra-cluster dissimilarity

$$J^{intra} = \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \left(1 - \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i} \right) \right) - \sum_{i=1}^C \frac{K}{\sigma_i}. \quad (78)$$

That is, we replace the LSL objective function in (56) with the objective function in (78). The first term in (78) that seeks clusters that have compact local relational distances is kept the same as in (58). However, the second term is slightly different. In fact, it is $\sum_{i=1}^C \frac{K}{\sigma_i}$ instead of $\sum_{i=1}^C \frac{K}{\sigma_i^2}$ in order to keep the optimization simple. It still aims to avoid the trivial solution where all the scaling parameters σ_i are infinitely large. Another trivial solution that minimizes the objective function in (78) is when one of σ_i is zero. In order to keep the derivation simple, we do not introduce another regulation term. We simply assume that σ_i is not null. Later in this section, we will discuss how to handle this case when it happens.

In FLeCK, instead of treating K as a constant, we propose to determine the optimal K by simultaneously minimizing the intra-cluster dissimilarity and maximizing the inter-cluster dissimilarity. Thus, in addition to minimizing the objective function in (78), we maximize

$$J^{inter} = \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N \alpha_{jk}^i \left(1 - \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i} \right) \right) + \sum_{i=1}^C \frac{K}{\sigma_i} \quad (79)$$

The term α_{jk}^i in (79) refers to the likelihood that two points \mathbf{x}_j and \mathbf{x}_k do not belong to the same cluster i . It can be defined using the fuzzy memberships of the two points as

$$\alpha_{jk}^i = u_{ij}^m (1 - u_{ik}^m) + u_{ik}^m (1 - u_{ij}^m). \quad (80)$$

Similar to the second term in (78), the second term in (79) is a regularization term to avoid the trivial solution where all the scaling parameters σ_i are infinitely large.

The objective of FLeCK is to learn the C clusters, the scaling parameters σ_i of each cluster, and the membership values u_{ij} , of each sample \mathbf{x}_j in each cluster i that optimize (78) and (79). In order to optimize these functions with respect to σ_i , we assume that σ_i 's are independent from each other and reduce the optimization problem to C independent problems. That is, we convert the set of objective functions in (78) and in (79) to the following C simpler set of functions

$$\min J_i^{intra} = \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \left(1 - \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i} \right) \right) - \frac{K}{\sigma_i}, \quad (81)$$

and

$$\max J_i^{inter} = \sum_{j=1}^N \sum_{k=1}^N \alpha_{jk}^i \left(1 - \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i} \right) \right) + \frac{K}{\sigma_i} \quad (82)$$

for $i = 1, \dots, C$. In (81), β_{jk}^i is the likelihood that two points \mathbf{x}_j and \mathbf{x}_k belong to the same cluster i and is defined as in (57).

To obtain an update equation for the scaling parameters σ_i , we first set

$$\frac{\partial J_i^{intra}}{\partial \sigma_i} = - \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \frac{\mathbf{r}_{jk}}{\sigma_i^2} \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i} \right) + \frac{K}{\sigma_i^2} = 0 \quad (83)$$

and solve for K . We obtain

$$K = \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \mathbf{r}_{jk} \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right). \quad (84)$$

Substituting (84) back in (82) gives

$$J_i^{inter} = \sum_{j=1}^N \sum_{k=1}^N \alpha_{jk}^i \left(1 - \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right)\right) + \frac{1}{\sigma_i} \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \mathbf{r}_{jk} \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) \quad (85)$$

Taking the derivative of (85) with respect to σ_i , we obtain

$$\begin{aligned} \frac{\partial J_i^{inter}}{\partial \sigma_i} &= -\frac{1}{\sigma_i^2} \sum_{j=1}^N \sum_{k=1}^N \alpha_{jk}^i \mathbf{r}_{jk} \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) \\ &\quad - \frac{1}{\sigma_i^2} \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \mathbf{r}_{jk} \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) \\ &\quad + \frac{1}{\sigma_i^3} \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \mathbf{r}_{jk}^2 \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) \end{aligned} \quad (86)$$

In the LSL approach (refer to III-B), we used the Heat kernel approximation (Eq. 74) in order to derive the update equation for σ_i . This approximation requires the specification of a neighborhood parameter. Moreover, as we are optimizing both the intra-cluster and inter-cluster distances, the update equation is more complex to drive. Instead, to keep the algorithm simple, we use a different approximation. We assume that the values of σ_i do not change significantly from one iteration ($t-1$) to the next one (t), and set (86) to zero, we obtain

$$\sigma_i^{(t)} = \frac{Q_1^i}{Q_2^i} \quad (87)$$

where

$$Q_1^i = \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \mathbf{r}_{jk}^2 \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i^{(t-1)}}\right) \quad (88)$$

and

$$Q_2^i = \sum_{j=1}^N \sum_{k=1}^N \left(\alpha_{jk}^i \mathbf{r}_{jk} \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i^{(t-1)}} \right) + \beta_{jk}^i \mathbf{r}_{jk} \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i^{(t-1)}} \right) \right) \quad (89)$$

As mentioned earlier, in theory, a trivial solution that optimizes the objective functions in (78) and in (79) is to have one of σ_i 's equal to zero. We notice from equation (87) that this happens when Q_1^i is equal to zero or when Q_2^i tends toward infinity. The first situation happens when a cluster has at most one point (i.e. all β_{jk}^i are zero). If this occurs at any given iteration, we simply discard the cluster and update the number of clusters. The second situation occurs when \mathbf{r}_{jk} approaches infinity. This scenario is not possible as discussed in subsection III-B-1 for the LSL algorithm.

Optimization of (78) and (79) with respect to u_{ij} is not trivial and does not lead to a closed form expression. To keep the computation simple, we optimize only (78) with respect to u_{ij} , use the relational dual of the fuzzy C-means algorithm formulated by Hathaway et al. [85], and obtain the same update equation as in (70).

The resulting FLeCK approach is outlined in algorithm 12.

2 Interpretation of the learned scaling parameters

The expression of the scaling parameter defined by equation (87) is different from the one learned by the LSL algorithm (Eq. (76)). It depends on both the inter-cluster and the intra-cluster distances. One possible interpretation of this expression is that σ_i is related to the rate of the dissimilarity change. In fact, according to Fick's Second Law of Diffusion [28], the time rate of concentration change is related to the second derivative of the concentration gradient through the diffusion coefficient γ , i.e.,

$$\frac{\partial C_d}{\partial t} = \gamma \frac{\partial^2 C_d}{\partial^2 d} \quad (90)$$

where C_d is the concentration at a distance d at the time t . In our case,

$$C_d = D_{jk}^i = 1 - \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) \quad (91)$$

and

$$t = r_{jk} \quad (92)$$

which gives

$$\frac{\partial D_{jk}^i}{\partial t} = \gamma_{jk}^i \cdot \mathbf{r}_{jk}^2 \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) \quad (93)$$

Setting the diffusion constant γ_{jk}^i to $\frac{\beta_{jk}^i}{Q_2}$ and summing over all the points belonging to the same cluster, we obtain an expression of the scaling parameter σ_i with respect to each cluster i (defined in (87)) that matches the definition of the time rate of the dissimilarity change within the same cluster.

Algorithm 12 The FLeCK algorithm

Fix number of clusters C and $m \in [1\infty)$;

Initialize the fuzzy partition matrix \mathbf{U} ;

Initialize the scaling parameter σ_i to 1;

REPEAT

 Compute the dissimilarity D^i for all clusters using (58) ;

 Compute the membership vectors \mathbf{v}_i using (61);

 Compute the distances using (60);

 Update the fuzzy membership using (70);

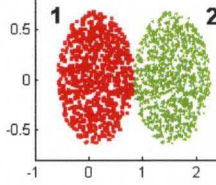
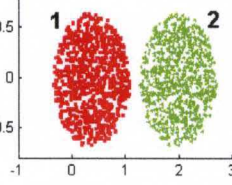
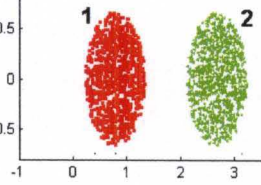
 Discard empty clusters and update the number of clusters;

 Update the scaling parameter σ_i using (87);

UNTIL (fuzzy membership do not change)

TABLE 6

Variations of σ_i with respect to the inter-cluster distance when 2 clusters have the same density and size.

Datasets															
Scaling Parameters	<table><tr><td>σ_1</td><td>σ_2</td></tr><tr><td>1.2</td><td>1.2</td></tr></table>	σ_1	σ_2	1.2	1.2	<table><tr><td>σ_1</td><td>σ_2</td></tr><tr><td>1.4</td><td>1.4</td></tr></table>	σ_1	σ_2	1.4	1.4	<table><tr><td>σ_1</td><td>σ_2</td></tr><tr><td>3</td><td>3</td></tr></table>	σ_1	σ_2	3	3
σ_1	σ_2														
1.2	1.2														
σ_1	σ_2														
1.4	1.4														
σ_1	σ_2														
3	3														

To provide an empirical interpretation of the scaling parameters learned by FLeCK, we run few experiments using synthetically generated datasets. In the first experiment, we use a dataset that has two clusters with the same shape, density, and number of points. We only vary the inter-cluster distances and examine the scaling parameters learned by FLeCK. Table 6 shows the datasets and the learned σ_i for each cluster. As it can be seen, when all parameters are fixed, σ_i gets larger as the inter-cluster distance increases. In fact, σ_i is designed to allow high dissimilarity between each given point and other points in its cluster but low dissimilarity between points in different clusters. That is, when the inter-cluster distance is larger, FLeCK assigns higher values of σ_i to allow for larger intra-cluster dissimilarity. However, as the clusters are moved closer to each other, FLeCK assigns lower σ_i to avoid small inter-cluster dissimilarity.

In the second experiment, we use a similar dataset, shown in Figure 10, that has 3 clusters with different relative arrangement. Clusters 1 and 3 are close to each other but far from cluster 2. Consequently, FLeCK learns two similar scaling parameter for clusters 1 and 3 ($\sigma_1 = 2.5$ and $\sigma_3 = 2.6$) and a larger parameter ($\sigma_2 = 3.5$) for cluster 2. In fact, σ_1 should not be too large so that points from cluster 3 cannot have high degrees of dissimilarity. A similar analysis applies to cluster 3. Cluster 2, on the

other hand, is far away from the other clusters. Thus, σ_2 can be larger (to increase the within cluster dissimilarity) without the risk of including points from the other clusters.

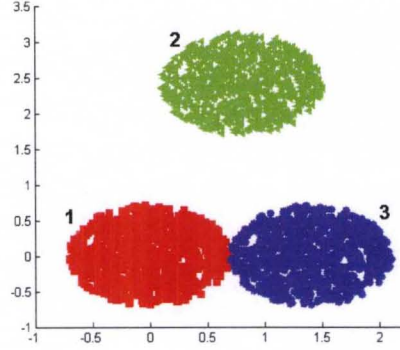


Figure 10. Three clusters with the same density and size, but different intra-cluster distances.

TABLE 7

Variations of σ_i for different cluster sizes and densities

Datasets															
Scaling Parameters	<table><tr><td>σ_1</td><td>σ_2</td></tr><tr><td>1.4</td><td>1.4</td></tr></table>	σ_1	σ_2	1.4	1.4	<table><tr><td>σ_1</td><td>σ_2</td></tr><tr><td>3.9</td><td>1</td></tr></table>	σ_1	σ_2	3.9	1	<table><tr><td>σ_1</td><td>σ_2</td></tr><tr><td>5.7</td><td>23</td></tr></table>	σ_1	σ_2	5.7	23
σ_1	σ_2														
1.4	1.4														
σ_1	σ_2														
3.9	1														
σ_1	σ_2														
5.7	23														

In the third experiment, we vary the cluster densities and sizes and examine the values of σ_i learned by FLeCK. The different datasets and the learned σ_i are reported in Table 7. As it can be seen, for the second dataset, the relatively larger cluster gets assigned a larger σ . Similar intuitive results would be obtained if σ_i 's

were learned using the variance of each cluster. However, these variances would be the same regardless of the intra-cluster distance. FLeCK, on the other hand, would assign different values as the intra-cluster distance varies. For the third dataset in this experiment, FLeCK assigns smaller σ to the cluster with larger density.

We should emphasize that, in addition to identifying more meaningful clusters, the learned σ_i 's can be used in subsequent steps to provide better cluster assignment. For instance, for point \mathbf{x}_k in the second dataset in Table 7, FLeCK would assign a higher membership in cluster 1 than cluster 2. This is not the case when the Euclidean distance is used since \mathbf{x}_k is spatially closer to cluster 2. Using a covariance matrix induced distance, such as the GK algorithm [88], would provide results similar to those obtained by FLeCK. However, the latter approach is restricted to data represented by feature vectors. Similarly for \mathbf{x}_k in the third dataset in Table 7, FLeCK would assign a higher membership in cluster 2 than cluster 1. This intuitive result may not be obtained with the Euclidean distance or even with a covariance matrix induced distance.

3 Performance illustration

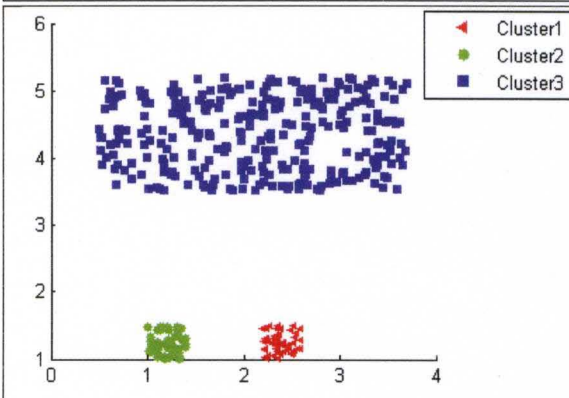
To illustrate the ability of FLeCK to learn appropriate local scaling parameters and cluster the data simultaneously, we use it to categorize synthetic 2D datasets. We use the same datasets as those used to illustrate the LSL algorithm (Figure 3), and the same parameters and initialization.

Table 8 displays the partition and the scaling parameters learned by FLeCK on dataset 1. The learned parameters reflect the relative position of each cluster with respect to the others. For this data, cluster 1 and cluster 2 are close to each other, and quite distant from cluster 3. Consequently, FLeCK learns a large scaling parameter ($\sigma_3 = 106.6$) for cluster 3. This large σ_3 allows points that are spatially dispersed to be grouped into one cluster without including points from the other clusters. On the other hand, FLeCK learns smaller σ_i 's for the two small and dense clusters ($\sigma_1 = 24.94$ and $\sigma_2 = 30.17$). These small σ_1 and σ_2 prevent points that are not spatially very close from being assigned to these clusters. The fuzzy memberships

of all points in the three clusters are displayed in Figure 11. As it can be seen, most of the points belong to one of the clusters with a membership larger than 0.8. This indicates that the mapping using the learned σ_i 's makes the three clusters well separated in the feature space.

TABLE 8

Partition and scaling parameters learned by FLeCK for dataset 1 displayed in Figure 3 (a)

Partition	Learned σ_i 's
	$\sigma_1 = 24.94$ $\sigma_2 = 30.17$ $\sigma_3 = 106.6$

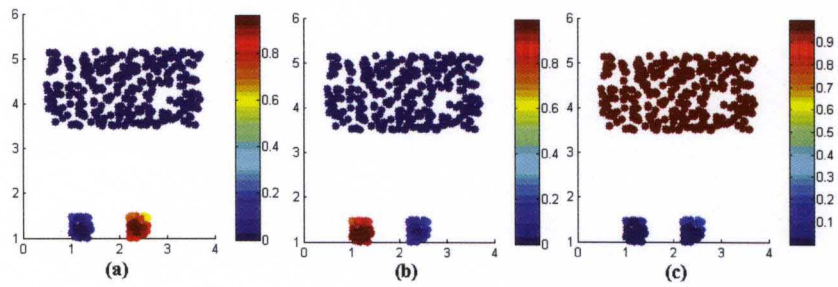


Figure 11. Fuzzy memberships learned by FLeCK on dataset 1 (Figure 3 (a)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

TABLE 9

Partition and scaling parameters learned by FLeCK for dataset 2 displayed in Figure 3 (b)

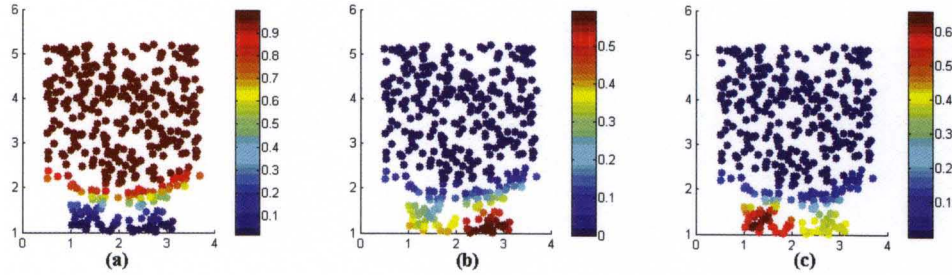
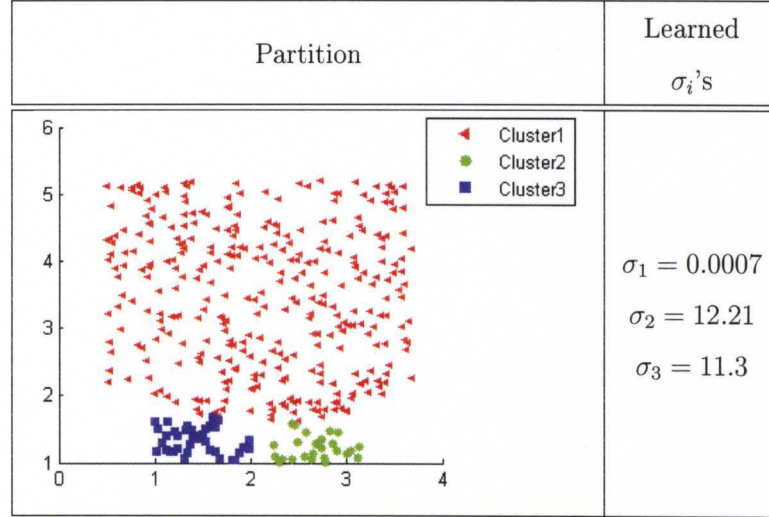


Figure 12. Fuzzy memberships learned by FLeCK on dataset 2 (Figure 3 (b)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

The second dataset, shown in Figure 3 (b), contains two clusters that have the same shape, density, and size, and a third cluster that is larger. The partition and the scaling parameters are displayed in Table 9. As it can be seen, FLeCK was able to partition this data correctly. This was possible due to the learned scaling parameters for each cluster. For instance, cluster 1 has the smallest scaling parameter ($\sigma_1 = 0.0007$). This reflects the fact that points within this cluster are dispersed and

some of them are spatially closer to cluster 2 and cluster 3 than other points within this cluster. This small σ_1 ensures that cluster 1 does not include points from the other clusters. FLeCK learns relatively larger scaling parameters for cluster 2 and cluster 3 ($\sigma_2 = 12.21$ and $\sigma_3 = 11.3$). In fact, these two parameters can be relatively larger (to maximize the intra-cluster dissimilarity) without including points from the other cluster. We can also notice that FLeCK performs better than LSL on this dataset (refer to Table 9 and Table 2), and handles the boundary points better. We also notice from Figure 12 that the points lying at the boundaries separating the three clusters have fuzzy memberships in the 0.5 range, thus reflecting the geometric characteristic of this dataset.

TABLE 10

Partition and scaling parameters learned by FLeCK for dataset 3 displayed in Figure 3 (c)

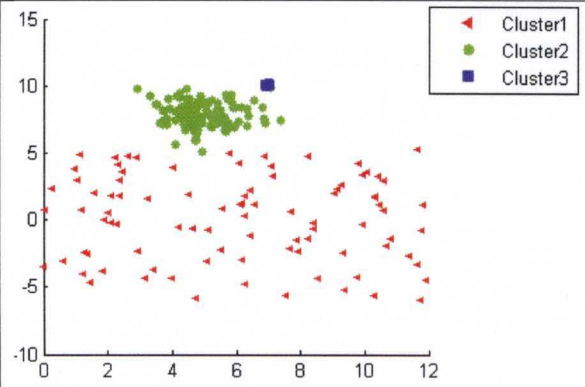
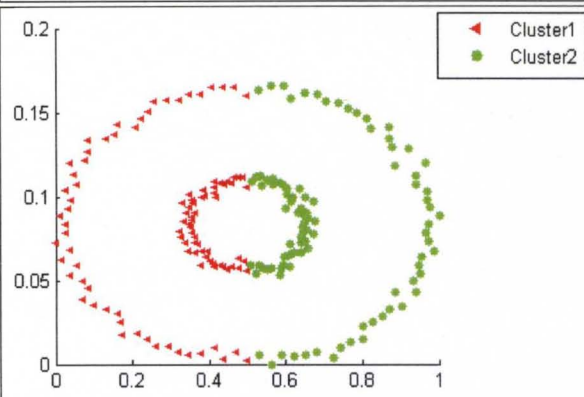
Partition	Learned σ_i 's
	$\sigma_1 = 109.92$ $\sigma_2 = 18.29$ $\sigma_3 = 2.44$

Table 10 reports a similar analysis conducted on dataset 3. As it can be seen, the sparse cluster (cluster 1) has the largest scaling parameter ($\sigma_1 = 109.92$). This large σ_1 allows points that are spatially dispersed to be grouped into one cluster. On the other hand, the small and dense cluster (cluster 3) has the small scaling parameter ($\sigma_3 = 2.44$). This smallest σ_2 prevents points that are not spatially very close from being assigned to this cluster. As in the previous example, FLeCK has outperformed

LSL especially at the cluster's boundaries (refer to Table 3). This good clustering result is also reflected by the fuzzy memberships of all points in the three clusters as shown in Figure 13.

TABLE 11

Partition and scaling parameters learned by FLeCK for dataset 4 displayed in Figure 3 (d)

Partition	Learned σ_i 's
	$\sigma_1 = 2.12$ $\sigma_2 = 2.10$

The clustering results returned by FLeCK on dataset 4 (Table 11) are not meaningful. In fact, as dataset 4 is constituted of two co-centric ovals, it does not correspond to the standard way of perceiving the intra-cluster and the inter-cluster dissimilarities. That is why the scale parameters learned by FLeCK fail to deal with the geometric characteristics of this dataset.

Table 12 reports the results obtained by partitioning dataset 5 using FLeCK. We should recall that LSL was not able to partition this data correctly (refer to Table 5). FLeCK, on the other hand, was able to partition this data correctly because it takes into account both the intra cluster and the inter cluster dissimilarities. The scaling parameters learned by FLeCK reflect the geometric characteristics of this dataset. As it can be seen, since this dataset contains two clusters that have the same shape, density, and size, and a third cluster that is denser and much smaller,

FLeCK learns a small scaling parameter for cluster 1 ($\sigma_1 = 3.33$) and the same scaling parameter for clusters 2 and 3 ($\sigma_2 = 22.36$ and $\sigma_3 = 22.36$).

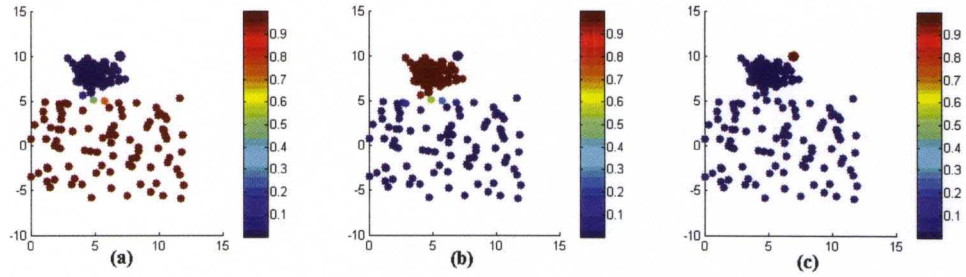


Figure 13. Fuzzy memberships learned by FLeCK on dataset 3 (Figure 3 (c)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

TABLE 12

Partition and scaling parameters learned by FLeCK for dataset 5 displayed in Figure 3 (a)

Partition	Learned σ_i 's
	$\sigma_1 = 3.33$ $\sigma_2 = 22.36$ $\sigma_3 = 22.36$

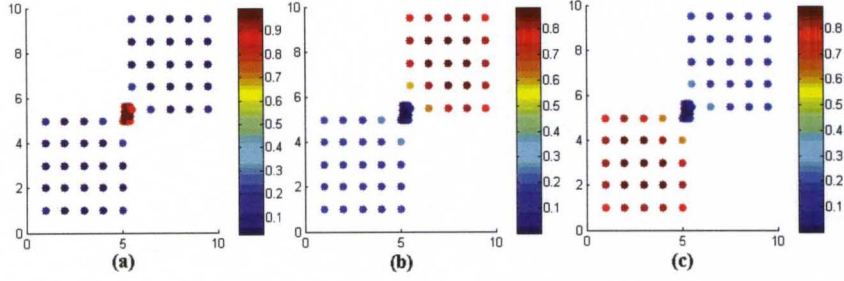


Figure 14. Fuzzy memberships learned by FLeCK on dataset 5 (Figure 3 (e)) with respect to (a) cluster 1, (b) cluster 2, and (c) cluster 3.

In this chapter, we presented two approaches that learn local Gaussian kernel for each cluster and cluster the data simultaneously. We showed that both approaches give satisfactory clustering results on 2D datasets. Moreover, we showed that the learned scaling parameters and the fuzzy memberships returned by LSL and FLeCK are meaningful and reflect the geometric characteristic of the data.

The first algorithm, LSL, minimizes the intra-cluster distances only, while the second algorithm, FLeCK, optimizes both the intra-cluster and the inter-cluster dissimilarities. Thus, the scaling parameters learned by FLeCK contain more information than the one learned by LSL. In fact, these parameters are not only influenced by the intra-cluster distances, but also by the relative cluster positions, densities and sizes. This allows a better description of the data and consequently, a better partition of the data.

To derive the update equation for σ_i , LSL uses the Heat flow approximation. While this assumption allowed LSL to deal with clusters with irregular shapes (e.g. co-centric ovals in Table 4), it requires the specification of a parameter K . This parameter which is a function of the regulation term K_1 and the local geometric characteristics of the data, can affect the clustering results.

To overcome the need to specify K , FLeCK was formulated to optimize both the intra-cluster and inter-cluster distances. Also, instead of using the heat flow approximation while deriving the update equations, it uses a different approximation that assumes that the cluster's scaling parameters do not vary significantly from one

iteration to another. While this assumption has made the derivation simpler and the algorithm parameter free, it also made FLeCK unable to deal with clusters of arbitrary shapes where the notion of intra-cluster and inter-cluster distances are not well defined.

The LSL and FLeCK objective functions have several parameters and their optimization is prone to several local minima and is sensitive to initialization. This problem is more acute for high dimensional dataset. Thus, if a small amount of prior knowledge is available, it can be used to guide the clustering algorithms to avoid most local minima and obtain a better partition. In the next chapter, we present semi-supervised versions of these algorithms that use a small amount of side information.

CHAPTER IV

SEMI-SUPERVISED RELATIONAL CLUSTERING WITH LOCAL SCALE PARAMETERS

Clustering is a difficult combinatorial problem that is susceptible to local minima, especially for high dimensional real world data. Incorporating prior knowledge in the unsupervised learning task, in order to guide the clustering process has attracted considerable interest among researchers in the data mining and machine learning communities. This prior knowledge is usually available in the form of hints, constraints, or labels. Supervision in the form of constraints is more practical than providing class labels. This is because in many real world applications, the true class labels may not be known, and it is much easier to specify whether pairs of points should belong to the same or to different clusters. In fact, pairwise constraints occur naturally in many domains.

In this chapter, we present two semi-supervised clustering approaches: the Semi-Supervised clustering and Local Scale Learning algorithm (SS-LSL), and the Semi-Supervised Fuzzy clustering with LEarnable Cluster dependent kernels (SS-FLeCK). As is common for most semi-supervised clustering algorithms, we assume that for both algorithms we have pairwise "*Should-Link*" constraints (pairs of points that should belong to the same cluster) and "*Should not-Link*" constraints (pairs of points that should belong to different clusters) provided with the input.

We should note here that, in standard semi-supervised clustering [9, 47], the above constraints are referred to as "*Must-Link*" and "*Cannot-Link*" because they are crisp, treated equally important, and enforced during the optimization. In our formulation, the pairwise constraints used in SS-LSL and SS-FLeCK are soft reflecting the uncertainty associated with *a priori* knowledge about the pair of points that should or

should not belong to the same cluster. Thus, they can be viewed as recommendations.

Let $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ be a set of N data points and let $\mathbf{R} = [\mathbf{r}_{jk}]$ be a relational matrix where \mathbf{r}_{jk} represent the distance between \mathbf{x}_j and \mathbf{x}_k . We assume that some partial information is available and let \mathbf{Sl} be the indicator matrix for the set of "Should-Link" pairs of constraints such that $\mathbf{Sl}(j, k) = 1$ means that \mathbf{x}_j and \mathbf{x}_k should be assigned to the same cluster and 0 otherwise. Similarly, let \mathbf{SNl} be the indicator matrix for the set of "Should not-Link" pairs such that $\mathbf{SNl}(j, k) = 1$ means that \mathbf{x}_j and \mathbf{x}_k should not be assigned to the same cluster and 0 otherwise.

A Semi-Supervised relational clustering with local scaling parameter

1 The Semi-Supervised clustering and Local Scale Learning algorithm

The Semi-Supervised Local Scaling Learning (SS-LSL) minimizes the following multi-term objective function:

$$\begin{aligned}
J &= \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \left(1 - \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) \right) \\
&- w \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \mathbf{Sl}(j, k) \\
&+ w \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \mathbf{SNl}(j, k) \\
&- \sum_{i=1}^C \frac{K_1}{\sigma_i^2}
\end{aligned} \tag{94}$$

subject to

$$0 \leq u_{ij} \leq 1, \text{ and } \sum_{i=1}^C u_{ij} = 1, \text{ for } j \in \{1, \dots, N\}. \tag{95}$$

It is an extension of the LSL algorithm that incorporates partial supervision in the same way as in [6]. As in the LSL objective function, the first term in (94) seeks compact clusters, and the last term, $\sum_{i=1}^C \frac{K_1}{\sigma_i^2}$, is a regularization term to avoid the trivial solution where all σ_i are infinitely large.

The second term in (94), is a reward term for satisfying "*Should-Link*" constraints. It is constructed in such a way that the reward between nearby "*Should-Link*" points is higher than that between distant ones. In fact, it is weighted by the term $(u_{ij}^m u_{ik}^m)$ which measures the extent to which \mathbf{x}_j and \mathbf{x}_k belong to the same cluster i . If this term is high and $(\mathbf{x}_i, \mathbf{x}_j)$ are supposed to be "*Should-Link*" then the learned kernel for this cluster is appropriate and the reward should be larger to allow the adjustment. A similar analysis could be conducted with respect to the third term, which is a penalty for violating "*Should not-Link*" constraints. It is constructed in such a way that the penalty between nearby "*Should not-Link*" points is higher than distant ones. It is also weighted by the term $(u_{ij}^m u_{ik}^m)$. If this term is large and $(\mathbf{x}_i, \mathbf{x}_j)$ are supposed to be "*Should not-Link*" then the kernel for this cluster is not appropriate and the penalty should be larger to allow the adjustment.

In (94), the weight $w \in (0, 1)$ provides a way of specifying the relative importance of the "*Should-Link*" and "*Should not-Link*" constraints compared to the sum of inter-cluster distances. In our approach, we fix it as the ratio of the number of constraints to the total number of points.

In order to optimize (94) with respect to σ_i , we assume that σ_i 's are independent from each other and reduce the optimization problem to C independent problems. That is, we convert the objective function in (94) to the following C simpler set of functions

$$\begin{aligned}
J_i = & \sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \left(1 - \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i} \right) \right) \\
& - w \sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \mathbf{Sl}(j, k) \\
& + w \sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \mathbf{SNl}(j, k) \\
& - \frac{K}{\sigma_i^2}
\end{aligned} \tag{96}$$

for $i = 1 \dots C$. As the reward and penalty terms do not depend on the scaling parameters σ_i explicitly, setting the derivative of J_i with respect to σ_i gives the same

update equation for σ_i as the LSL algorithm, i.e, equation (76).

In order to optimize (98) with respect to u_{ij} , we rewrite the objective function in (94) as

$$J = \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \left(1 - \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) - w\mathbf{Sl}(j, k) + w\mathbf{SNl}(j, k) \right) - \sum_{i=1}^C \frac{K}{\sigma_i^2} \quad (97)$$

or simply as

$$J = \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N u_{ij}^m u_{ik}^m \hat{D}_{jk}^i - \sum_{i=1}^C \frac{K_1}{\sigma_i^2} \quad (98)$$

where

$$\hat{D}_{jk}^i = D_{jk}^i - w\mathbf{Sl}(j, k) + w\mathbf{SNl}(j, k) \quad (99)$$

In (99), D_{jk}^i is the distance between \mathbf{x}_j and \mathbf{x}_k using the scaling parameter of cluster i . This is the same distance used within LSL algorithm (refer to Eq. (58)). \hat{D}_{jk}^i can be regarded as the "effective distance" that takes into account the satisfaction and violation of the constraints. For instance, if the pair of points $(\mathbf{x}_i, \mathbf{x}_j)$ are supposed to be "*Should-Link*" then

$$\begin{cases} \mathbf{Sl}(j, k) = 1 \\ \mathbf{SNl}(j, k) = 0 \end{cases}$$

In this case, the effective distance \hat{D}_{jk}^i reduces to

$$\hat{D}_{jk}^i = D_{jk}^i - w. \quad (100)$$

In other words the actual distance is reduced to help in keeping these points within the same cluster and thus, maintaining the satisfaction of the constraints.

Similarly, if a pair of points $(\mathbf{x}_i, \mathbf{x}_j)$ are supposed to be "*Should not-Link*" then

$$\begin{cases} \mathbf{Sl}(j, k) = 0 \\ \mathbf{SNl}(j, k) = 1 \end{cases}$$

Thus, the effective distance $\hat{\mathbf{D}}_{jk}^i$ defined by (99) becomes

$$\hat{\mathbf{D}}_{jk}^i = \mathbf{D}_{jk}^i + w \quad (101)$$

That is, the actual distance is increased to help in preventing these points from being assigned to the same cluster.

Following the same steps used to derive the update equations for u_{ij} in LSL, it can be shown that optimization of J w.r.t u_{ij} yields

$$u_{ij} = \frac{1}{\sum_{t=1}^C (d_{ij}^2/d_{tj}^2)^{\frac{1}{m-1}}}. \quad (102)$$

where

$$d_{ik}^2 = \left(\hat{\mathbf{D}}^i \mathbf{v}_i \right)_k - \frac{\mathbf{v}_i^t \hat{\mathbf{D}}^i \mathbf{v}_i}{2}, \quad (103)$$

and

$$\mathbf{v}_i = \frac{(u_{i1}^m, \dots, u_{iN}^m)^t}{\sum_{j=1}^N u_{ij}^m} \quad (104)$$

We should note that it is possible for the effective distances $\hat{\mathbf{D}}_{jk}^i$ to be negative depending on the constant w . To overcome this, we use the β -spread transform [80] to convert the non-Euclidean distance $\hat{\mathbf{D}}$ into Euclidean distance $\hat{\mathbf{D}}^{\beta_1}$ as follows

$$\hat{\mathbf{D}}^{\beta_1} = \hat{\mathbf{D}} + \beta_1 \cdot (\mathbf{M} - \mathbf{I}) \quad (105)$$

In (105), β_1 is a suitably chosen scalar, $\mathbf{I} \in \mathbb{R}^{N \times N}$ is the identity matrix and $\mathbf{M} \in \mathbb{R}^{N \times N}$ has all its entries equal to one. The lower bound by which its necessary to shift β_1 to make the distances positives is derived in [80] to be

$$\Delta\beta_1 = \max_{i,k} \left\{ -2d_{ik}^2 / \|\mathbf{v}_i - \mathbf{e}_k\|^2 \right\}, \quad (106)$$

where \mathbf{e}_k denotes the k^{th} column of the identity matrix. The SS-LSL algorithm is summarized in algorithm 13.

Algorithm 13 The SS-LSL algorithm

Fix number of clusters C and $m \in]1, \infty[$;

Initialize $\beta_1 = 0$;

Initialize the fuzzy partition matrix \mathbf{U} ;

Initialize the scaling parameter σ_i to 1;

Create \mathbf{Sl} and \mathbf{SNl} pairwise constraints;

REPEAT

 Compute the dissimilarity \mathbf{D} for all clusters using (58) ;

 Compute the dissimilarity $\hat{\mathbf{D}}^i$ for all clusters using (99) ;

 Compute $\hat{\mathbf{D}}^{\beta_1}$ using (105);

 Compute the membership vectors \mathbf{v}_i using (104);

 Compute the distances using (103);

IF $d_{ik}^2 < 0$ for any i, k

 Compute $\Delta\beta_1$ using (106);

$$d_{ik}^2 = d_{ik}^2 + (\Delta\beta_1/2) * \|\mathbf{v}_i - \mathbf{e}_k\|;$$

$$\beta_1 = \beta_1 + \Delta\beta;$$

END IF

 Update the fuzzy membership using (102);

 Update the scaling parameter σ_i using (76);

UNTIL (fuzzy membership do not change)

Partial supervision information can also come from users feedback. In contrast to relevance feedback which asks users to label retrieved information, providing pairwise constraints does not necessarily require users to have prior knowledge or experience with the dataset. Typically, the system identifies the most ambiguous pairs

of samples and presents them to the user. The user then provides the constraint information as a feedback. In order to maximize the utility of the limited supervised data available in a semi-supervised setting, pairwise constraints should be, if possible, actively selected as maximally informative ones rather than chosen at random. This would imply that fewer constraints will be required to significantly improve the clustering accuracy. To this end, pairwise constraints are selected among points lying at the clusters boundaries.

2 Performance illustration

To illustrate the ability of SS-LSL to learn appropriate local scaling parameters and cluster the data simultaneously, we use it to categorize synthetic 2D datasets. We use the three datasets where the unsupervised LSL did not perform well. These are dataset 2 (Figure 3(b)), dataset 3 (Figure 3(c)), and dataset 5 (Figure 3(b)). dataset 1 and dataset 2 are not considered because LSL have partitioned them correctly, and, thus, no further enhancement is possible. We use the same experiment setting as in section III-B.

In order to construct the set of “*Should-Link*” and “*Should-Not Link*” constraints, we randomly select few points that are at the boundary of each cluster. For each dataset, we select 2% of the total number of points to construct the set of “*Should-Link*” and “*Should-Not Link*”. Pairs of selected points that belong to the same cluster (using the ground truth) constitute the “*Should Link*” set, Sl . Similarly, pairs of points belonging to different clusters constitute the “*Should not-Link*” set, SNl . In addition, we use the transitive closure [48] to obtain more pairwise constraints from the available ones.

Table 14 reports the results of dataset 2, where the three clusters have the same density. First, we notice that the three learned scaling parameters are very close to each other. Second, we notice that the points at the boundaries of cluster 2 and cluster 3 are better categorized than for the LSL approach (refer to Table 2). Moreover, in terms of the learned fuzzy memberships, we can see from Figure (16), that cluster 2 and cluster 3 are better characterized, compared with the LSL

results (Figure 5). Thus, the pairwise constraints have helped the clustering process to obtain a better partition.

TABLE 13

Partition and scaling parameters learned by SS-LSL on dataset 3 displayed in Figure 3 (c)

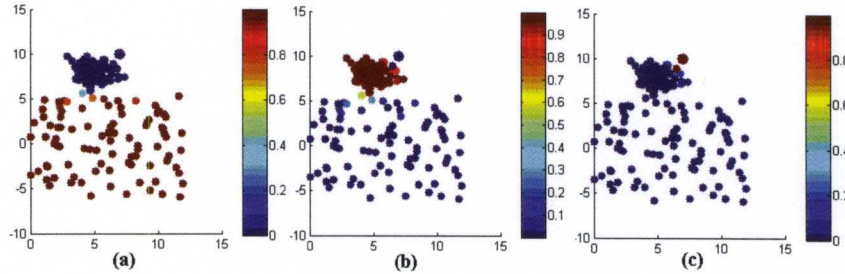
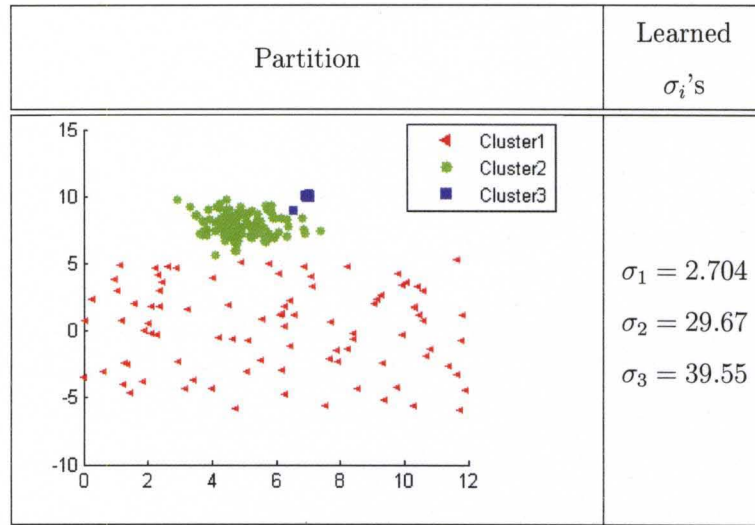


Figure 15. Fuzzy memberships learned by SS-LSL on dataset 3 (Figure 3 (a)) with respect to cluster 1 (a), cluster 2 (b), and cluster 3 (c).

Table 15 displays the results of applying SS-LSL on dataset 3. Compared to the LSL results (refer to Table 3), this is a better partition. In fact, the points lying at the boundaries separating the different clusters are better categorized. This is also reflected in the learned fuzzy memberships as displayed in Figure 15. This

enhancement is due to the few constraints that guided the algorithm in learning better kernel parameters.

TABLE 14

Partition and scaling parameters learned by SS-LSL on dataset 2 displayed in Figure 3 (b)

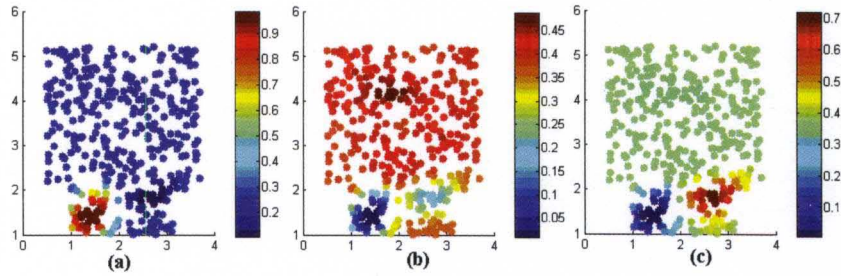
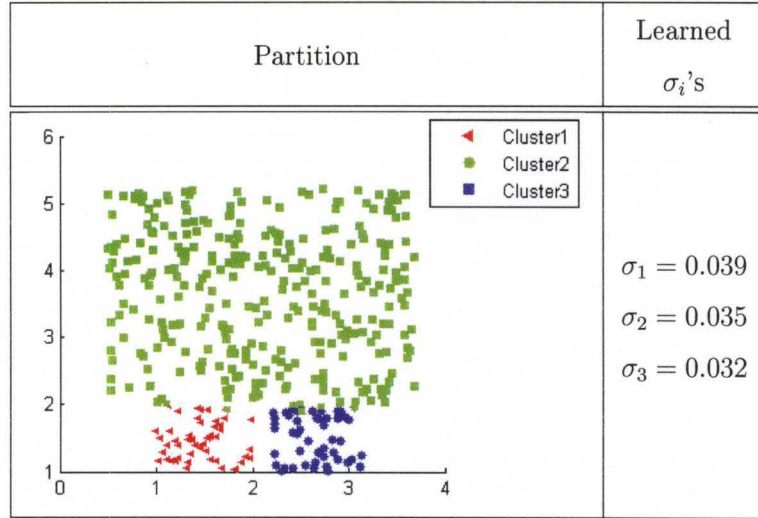


Figure 16. Fuzzy memberships learned by SS-LSL on dataset 2 (Figure 3 (a)) with respect to cluster 1 (a), cluster 2 (b), and cluster 3 (c).

TABLE 15

Partition and scaling parameters learned by SS-LSL on dataset 1 displayed in Figure 3 (a)

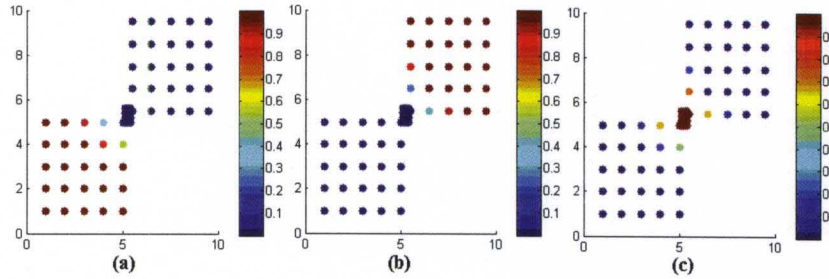
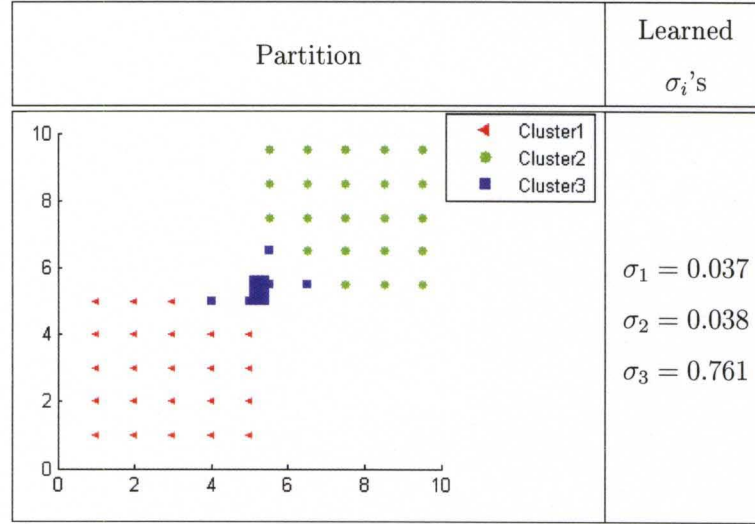


Figure 17. Fuzzy memberships learned by SS-LSL on dataset 5 (Figure 3 (a)) with respect to cluster 1 (a), cluster 2 (b), and cluster 3 (c).

As reported in Table 3, dataset 5 was a hard example for LSL that was not partitioned correctly. Using 2% of the data for partial supervision, the SS-LSL was able to provide satisfactory clustering results on this dataset as shown in Table 15. Moreover, the learned scaling parameters reflect the structure of the data as the sparse clusters (cluster 1 and cluster 3) have smaller scaling parameters than the densest one (cluster 2).

B Semi-Supervised relational clustering optimizing both the intra-cluster and inter-cluster distances

1 The Semi-Supervised Fuzzy clustering with LEarnable Cluster dependent kernels algorithm

The Semi-supervised Fuzzy clustering with LEarnable Cluster dependent kernels (SS-FLeCK) is an extension of the FLeCK algorithm described in section III-B-C that incorporates partial supervision information. It attempts to satisfy a set of “*Should-Link*” and “*Should-Not Link*” constraints while minimizing the intra-cluster dissimilarity

$$\begin{aligned}
J^{intra} = & \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \left(1 - \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i} \right) \right) \\
& - w \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \mathbf{Sl}(j, k) \\
& + w \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \mathbf{SNl}(j, k) \\
& - \sum_{i=1}^C \frac{K}{\sigma_i}
\end{aligned} \tag{107}$$

and maximizing the inter-cluster dissimilarity

$$J^{inter} = \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N \alpha_{jk}^i \left(1 - \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i} \right) \right) + \sum_{i=1}^C \frac{K}{\sigma_i}. \tag{108}$$

In (108), β_{jk}^i is the likelihood that two points \mathbf{x}_j and \mathbf{x}_k belong to the same cluster i as defined in (57), and α_{jk}^i is the likelihood that two points \mathbf{x}_j and \mathbf{x}_k do not belong to the same cluster i as defined in equation (80).

The first task of SS-FLeCK is based on minimizing a joint objective function with multiple terms. The first term in (107) seeks compact clusters as for the FLeCK approach. Similarly, the regularization term $\left(\sum_{i=1}^C \frac{K}{\sigma_i} \right)$ in (107) which avoids the trivial solution of infinitely large σ_i is kept the same as in (78).

The second and the third terms are the reward and penalty terms respectively. They are constructed in the same way as for SS-LSL. In particular, the second term in (107), is a reward term for satisfying "*Should-Link*" constraints. It is constructed in a such a way that the reward between nearby "*Should-Link*" points is higher than that between distant ones. It is weighted by the term β_{jk}^i which measures the extent to which \mathbf{x}_j and \mathbf{x}_k belong to the same cluster i . If this term is high and $(\mathbf{x}_i, \mathbf{x}_j)$ are supposed to be "*Should-Link*" then the learned kernel for this cluster is appropriate and the reward should be larger to allow the adjustment. A similar analysis could be conducted with respect to the third term, which is a penalty for violating "*Should not-Link*" constraints. It is constructed in such a way that the penalty between nearby "*Should not-Link*" points is higher than distant points one. It is also weighted by the term β_{jk}^i . If this term is large and $(\mathbf{x}_i, \mathbf{x}_j)$ are supposed to be "*Should not-Link*" then the kernel for this cluster is not appropriate and the penalty should be larger to allow the adjustment.

In (107), the weight $w \in (0, 1)$ provides a way of specifying the relative importance of the "*Should-Link*" and "*Should not-Link*" constraints compared to the sum of inter-cluster distances. As for SS-LSL algorithm, we fix it as the ratio of the total number of constraints to the number of points.

The second objective function J^{inter} in (108), is related to the inter-cluster dissimilarities. It is the same as its corresponding one for the FLeCK algorithm (see Eq. (79)).

The proposed SS-FLeCK algorithm attempts to optimize (107) and (108) by learning the C clusters, the scaling parameter, σ_i , of each cluster, and the membership values, u_{ij} , of each sample \mathbf{x}_j in each cluster i .

In order to optimize (107) and (108) with respect to σ_i , we assume that σ_i 's are independent from each others and reduce the optimization problem to C independent problems. That is, we convert the set of objective functions in (107) and (108) to the following C simpler set of functions

$$\begin{aligned}
J_i^{intra} &= \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \left(1 - \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i} \right) \right) \\
&- w \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \mathbf{Sl}(j, k) \\
&+ w \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \mathbf{SNl}(j, k) \\
&- \frac{K}{\sigma_i}
\end{aligned} \tag{109}$$

and

$$J_i^{inter} = \sum_{j=1}^N \sum_{k=1}^N \alpha_{jk}^i \left(1 - \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i} \right) \right) + \frac{K}{\sigma_i} \tag{110}$$

for $i = 1, \dots, C$. To obtain an update equation for the scaling parameters, we first set the derivative of J_i^{intra} with respect to σ_i to zero and solve for K . Then, we substitute the expression of K back in (110). Assuming that the values of σ_i do not change significantly from one iteration $(t-1)$ to the next one (t) and setting the derivative of J_i^{inter} to zero, we obtain

$$\sigma_i^{(t)} = \frac{Q_1^i}{Q_2^i} \tag{111}$$

where

$$Q_1^i = \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \mathbf{r}_{jk}^2 \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i^{(t-1)}} \right) \tag{112}$$

and

$$\begin{aligned}
Q_2^i &= \sum_{j=1}^N \sum_{k=1}^N \alpha_{jk}^i \mathbf{r}_{jk} \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i^{(t-1)}} \right) \\
&+ \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \mathbf{r}_{jk} \exp \left(-\frac{\mathbf{r}_{jk}}{\sigma_i^{(t-1)}} \right)
\end{aligned} \tag{113}$$

We notice that the expression of the scaling parameter defined by equation (111) is the the same as for FLeCK algorithm (Eq. 87). In fact, σ_i 's are independent from the "*Should-Link*" and "*Should not-Link*" constraints terms. Thus, optimizing (107) and (108) with respect to σ_i , is the same as optimizing (78) and (79) with respect to σ_i .

Optimization of (107) and (108) with respect to u_{ij} is not trivial and does not lead to a closed form expression. To keep the computation simple, we optimize only (107) with respect to u_{ij} . First, we rewrite the objective function in (107) as

$$\begin{aligned} J^{intra} &= \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \left(1 - \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) - w\mathbf{Sl}(j, k) + w\mathbf{SNl}(j, k) \right) \\ &- \sum_{i=1}^C \frac{K}{\sigma_i} \end{aligned} \quad (114)$$

or simply as

$$J^{intra} = \sum_{i=1}^C \sum_{j=1}^N \sum_{k=1}^N \beta_{jk}^i \hat{D}_{jk}^i - \sum_{i=1}^C \frac{K}{\sigma_i} \quad (115)$$

where

$$\hat{D}_{jk}^i = 1 - \exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) - w\mathbf{Sl}(j, k) + w\mathbf{SNl}(j, k) \quad (116)$$

The "effective" distance, $\hat{\mathbf{D}}_{jk}^i$, has the same expression and interpretation as the distance in SS-LSL (refer to subsection IV-A-1). As the term $\sum_{i=1}^C \frac{K}{\sigma_i^2}$ does not depend on u_{ij} , optimizing (115) with respect to u_{ij} is the same as optimizing (98) with respect to u_{ij} . Thus, as for SS-FLeCK, we substitute \mathbf{D} by $\hat{\mathbf{D}}$ in (103), and obtain the same u_{ij} update equation as in (102).

The resulting SS-FLeCK algorithm is summarized in algorithm 14.

Algorithm 14 The SS-FLeCK algorithm

Fix number of clusters C and $m \in [1\infty)$;
Initialize $\beta_1 = 0$;
Initialize the fuzzy partition matrix \mathbf{U} ;
Initialize the scaling parameter σ_i to 1;
Create \mathbf{Sl} and \mathbf{SNl} pairwise constraints;

REPEAT

 Compute the dissimilarity \mathbf{D} for all clusters using (58) ;
 Compute the dissimilarity $\hat{\mathbf{D}}^i$ for all clusters using (116) ;
 Compute $\hat{\mathbf{D}}^{\beta_1}$ using (105);
 Compute the membership vectors \mathbf{v}_i using (104);
 Compute the distances using (103);

IF $d_{ik}^2 < 0$ for any i, k

 Compute $\Delta\beta_1$ using (106);
 $dist_{ik}^2 = dist_{ik}^2 + (\Delta\beta_1/2) * \|v_i - e_k\|$;
 $\beta_1 = \beta_1 + \Delta\beta$;

END IF

 Update the fuzzy membership using (102);
 Update the scaling parameter σ_i using (111);

UNTIL (fuzzy membership do not change)

2 Performance illustration

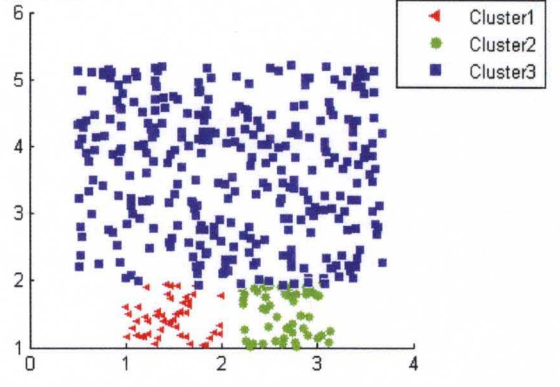
To illustrate the ability of SS-FLeCK to use partial supervision information to improve the results, we use it to categorize the two synthetic datasets where the unsupervised FLeCK did not perform well. The two datasets are displayed in Figure

3 (b) and Figure 3 (c). We use the same experimental setting as outlined in subsection IV-A-2.

In order to construct the set of “*Should-Link*” and “*Should-Not Link*” constraints, we randomly select few points that are at the boundaries of each cluster. For each dataset, we select 2% of the total number of points to construct the set of “*Should-Link*” and “*Should-Not Link*”. Pairs of selected points that belong to the same cluster (using the ground truth) constitute “*Should Link*” set, *Sl*. Similarly, pairs of points belonging to different clusters constitute “*Should not-Link*” set, *SNl*. In addition, we use the transitive closure [48] to obtain more pairwise constraints from the available ones.

TABLE 16

Partition and scaling parameters learned by SS-FLeCK on dataset 2 displayed in Figure 3 (b)

Partition	Learned σ_i 's
	$\sigma_1 = 5.102$ $\sigma_2 = 0.079$ $\sigma_3 = 4.515$

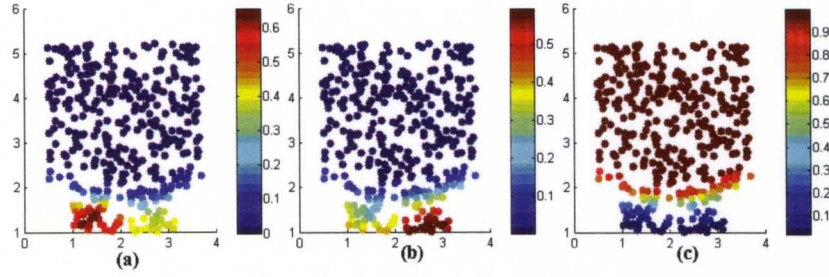


Figure 18. Fuzzy memberships learned by SS-FLeCK on dataset 2 (Figure 3 (a)) with respect to cluster 1 (a), cluster 2 (b), and cluster 3 (c).

Table 16 reports the results of dataset 2, where the three clusters have the same density. We notice that the points at the boundaries of the three clusters are better categorized than for the FLeCK approach (refer to Table 9). Moreover, in terms of the learned fuzzy memberships, we can see from Figure 18, that three clusters are better characterized, compared with the FLeCK results (Figure 12). Thus, the pairwise constraints have helped the clustering process to obtain a better partition.

TABLE 17

Partition and scaling parameters learned by SS-FLeCK on dataset 4 displayed in Figure 3 (d)

Partition	Learned σ_i 's
	$\sigma_1 = 4.04$ $\sigma_2 = 24.02$

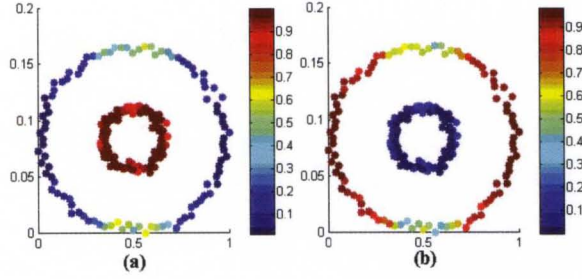


Figure 19. Fuzzy memberships learned by SS-FLeCK on dataset 4 (Figure 3 (a)) with respect to cluster 1 (a), and cluster 2 (b).

As reported in Table 11, dataset 4 was a hard example for FLeCK that was not partitioned correctly. Using 2% of the data for partial supervision, the SS-FLeCK was able to provide a satisfactory clustering results on this dataset as shown in Table 17. Moreover, the learned scaling parameters reflect the structure of the data as the sparse cluster (cluster 2) has the largest scaling parameter ($\sigma_2 = 24.02$). This large σ_2 allows points that are spatially dispersed to be grouped into one cluster. On the other hand, the small and dense cluster (cluster 1) has the smallest scaling parameter ($\sigma_1 = 4.04$). This is also reflected in the learned fuzzy memberships as displayed in Figure 19.

In this chapter, we presented two semi-supervised algorithms that learn multiple kernels and the fuzzy partitioning of the data simultaneously, guided by a small amount of pairwise constraints. We have shown that the incorporation of these constraints have guided the clustering process to better learn the scaling parameters and the fuzzy memberships that reflect the structure of the data. Consequently, a better partition of the data can be obtained. These pairwise constraints can be even more useful on real high dimensional datasets where the algorithm is more susceptible to local minima. Results on real and high-dimensional datasets will be reported in the next chapter and compared to other similar algorithms.

CHAPTER V

EXPERIMENTS

The objective of the proposed algorithms (LSL, FLeCK, SS-FLeCK, and SS-LSL) is to partition the data and learn a kernel resolution for each cluster. In this chapter, we assess the performance of the proposed approaches and compare their performances to different clustering algorithms. First, we compare these algorithms using the same five datasets used to illustrate LSL and FLeCK (refer to Figure 3). Then, in order to illustrate the ability of the proposed algorithms to learn local kernels and to cluster dissimilarity measure derived from real and high dimensional data, we use it to categorize a subset of the COREL image database and the handwritten digits database. A description of these datasets and the performance measures used to compare the different algorithms is provided in the following subsections.

A Datasets and performance measures

1 Image database

We use a subset of 450 color images from the COREL image collection. This subset includes five categories. Table 18 displays a sample image from each category along with the number of images for each category. The categories and the images within each one are selected to have different sizes, intra-group, and inter-group variations. Table 19 displays the sum of the intra - group and inter - group distances. For instance, some categories such as “Butterfly” and “Antelope” have the smallest sum of intra - group distances while category “Garden”, and “Bus” have the largest sum of intra - group distances. Similarly, category “Butterfly” has small sum of inter-category distances, while other categories such as “Beaches”, “Garden”, and “Buses” have large sum of inter-category distances.

TABLE 18

Sample image from each category of the COREL image database and the number of images for each category.






Category	Antelope	Beaches	Butterflies	Garden	Buses
Sample image					
No. of images	100	100	50	100	100

TABLE 19

Variations of the sum of intra-group and inter-group distances across the five image categories of the COREL image database.

	Antelope	Beach	Butterfly	Garden	Bus
Antelope	6558	10895	5853	9282	11059
Beach	10895	7829	5725	10032	11635
Butterfly	5853	5725	1841	6312	5969
Garden	9282	10032	6312	7483	11404
Bus	11059	11635	5969	11404	7358

To compute the distance and compare the content of the images, we use two standard MPEG-7 descriptors [53]. The first one is a 32 bin Scalable Color Descriptor (SCD). This descriptor extracts a quantized HSV color histogram from an RGB image. The probability values of each bin are calculated and indexed. The resulting histogram is then transformed, via a discrete Haar transformation, and non uniformly quantized and offset. The resulting array of values is then sorted and used as a feature descriptor.

The second MPEG-7 feature is the edge histogram descriptor (EHD). The EHD represents the frequency and directionality of the edges within the image. First, simple edge detector operators are used to identify edges and group them into five categories:

vertical, horizontal, 45-degree diagonal, 135-degree diagonal, and non-directional edge types. Then, local, global, and semi-local edge histograms are generated. The EHD feature is represented by a 150-dim vector.

2 Handwritten digits

The second real data that we use is a subset of 1166 handwritten digits from the Pen-Based Recognition of handwritten digits collection [76] available at the UCI Machine Learning repository. The categories within this data have different sizes, intra-group, and inter-group variations. Table 20 displays the number of samples from each category, and Table 21 displays the sum of the intra - group and inter - group distances.

The handwritten samples were collected using a pressure sensitive tablet that sends the x and y coordinates of the pen at fixed time intervals. The raw data is processed so that each digit is represented by 8 (x,y) coordinates resulting in a 16-D feature vector.

TABLE 20

Number of samples from each category of the handwritten digits

Category	0	1	2	3	4	5	6	7	8	9
No. of samples	128	131	107	122	108	119	114	117	109	111

3 Performance measures

To assess the performance of the proposed approaches and compare their performances to different clustering algorithms, we assume that the ground truth is known and we compute the partition accuracy. First, each cluster is assigned a label based on the majority of the true labels of its elements. Then, the correct classification rate of each cluster is computed. The overall accuracy of the partition is computed as the average of the individual clusters rates weighted by the clusters cardinality.

TABLE 21

Variations of the sum of intra-group and inter-group distances across the handwritten digits categories.

	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'
'0'	6046	12421	10352	10626	7714	9969	8813	10082	7964	9705
'1'	12421	7018	5879	7604	7868	9544	9351	7046	9534	7895
'2'	10352	5879	2622	7129	6613	7429	6928	4860	6991	7600
'3'	10626	7604	7129	3040	6983	8272	7552	6990	9114	5436
'4'	7714	7868	6613	6983	3902	7670	6377	7145	7389	5739
'5'	9969	9544	7429	8272	7670	7381	7555	7939	7578	8051
'6'	8813	9351	6928	7552	6377	7555	3165	8023	7117	6990
'7'	10082	7046	4860	6990	7145	7939	8023	4825	7233	7870
'8'	7964	9534	6991	9114	7389	7578	7117	7233	5411	8631
'9'	9705	7895	7600	5436	5739	8051	6990	7870	8631	4209

Another way of evaluating the performance of a clustering algorithm is by comparing its partition matrix U to the ground truth partition matrix $U^{(T)}$. Since the best one-to-one mapping of the clusters needs to be identified by finding the best permutation of the rows of the U matrices, such comparison is not trivial. An efficient way of comparing the partition matrices indirectly is by using the coincidence matrix, also called cluster connectivity matrix [60]. The coincidence matrices $\psi^{(1)}$ and $\psi^{(2)}$ of two partition matrices $U^{(1)}$ and $U^{(2)}$ are defined as

$$\psi = [\psi_{ij}], \quad 1 \leq j, k \leq N, \quad (117)$$

where

$$\psi_{jk} = \sum_{i=1}^C \mu_{ij} \mu_{ik}. \quad (118)$$

Using the two coincidence matrices, a 2×2 contingency Table

$$\begin{bmatrix} N_{SS} & N_{SD} \\ N_{DS} & N_{DD} \end{bmatrix}$$

is defined where

$$N_{SS}(\psi^{(1)}, \psi^{(2)}) = \sum_{j=2}^N \sum_{k=1}^{j-1} \psi_{jk}^{(1)} \psi_{jk}^{(2)}, \quad (119)$$

$$N_{SD}(\psi^{(1)}, \psi^{(2)}) = \sum_{j=2}^N \sum_{k=1}^{j-1} \psi_{jk}^{(1)} (1 - \psi_{jk}^{(2)}), \quad (120)$$

$$N_{DS}(\psi^{(1)}, \psi^{(2)}) = \sum_{j=2}^N \sum_{k=1}^{j-1} (1 - \psi_{jk}^{(1)}) \psi_{jk}^{(2)}, \quad (121)$$

$$N_{DD}(\psi^{(1)}, \psi^{(2)}) = \sum_{j=2}^N \sum_{k=1}^{j-1} (1 - \psi_{jk}^{(1)}) (1 - \psi_{jk}^{(2)}). \quad (122)$$

In the above, the indices S and D stand for Same cluster and Different clusters respectively.

Using the contingency Table, many measures can be computed to compare two partitions [20]. In this paper, we use the Rand statistics Q_{Rand} , Jaccard coefficient $Q_{Jaccard}$, Folkes-Mallows index Q_{FM} , and the Hubert index Q_{Hubert} . These indices are defined as:

$$Q_{Rand}(\psi^{(cl)}, \psi^{(T)}) = \frac{N_{SS} + N_{DD}}{N_{..}} \quad (123)$$

$$Q_{Jaccard}(\psi^{(cl)}, \psi^{(T)}) = \frac{N_{SS}}{N_{SS} + N_{SD} + N_{DS}} \quad (124)$$

$$Q_{FM}(\psi^{(cl)}, \psi^{(T)}) = \frac{N_{SS}}{\sqrt{(N_{SS} + N_{SD})(N_{SS} + N_{DS})}} \quad (125)$$

$$Q_{Hubert}(\psi^{(cl)}, \psi^{(T)}) = \frac{N_{..}N_{SS} - N_{S.}N_{.S}}{\sqrt{N_{.S}N_{S.}N_{.D}N_{D.}}} \quad (126)$$

These measures compare each generated partition $U^{(cl)}$ to the ground truth partition $U^{(T)}$. For the ground truth and for non-fuzzy clustering algorithms, we use crisp membership $u_{ij} \in \{0, 1\}$. All of the above measures provide larger values when the two partitions are more similar.

B Evaluation of the unsupervised clustering algorithms

In this section, we compare our unsupervised clustering algorithms, LSL and FLeCK, to those obtained using the FCM [86], DBSCAN [77], GK [88], kNERF [25], and self tuning spectral clustering [42]. For the relational approaches, the Euclidean distance is used to compute the pairwise distances. To assess the performance of the different clustering algorithms and compare them, we use the ground truth, and the five performance measures described in subsection V-A-3.

1 Synthetic datasets

In subsection III-B-2 and subsection III-C-3, we have illustrated how LSL and FLeCK learn local kernels and cluster dissimilarity measures on the five 2D datasets displayed in Figure 3. We also provided an interpretation of the learned scaling parameters. In order to better assess the performance of LSL and FLeCK on these datasets, we compare their clustering results with the clustering approaches mentioned above.

For all algorithms, we set the number of clusters C to the true one (see Figure 3), the fuzzifier m to 1.1, and the maximum number of iterations to 100. As LSL requires the specification of one parameter K , we use $K = [0.001, 0.01, 0.05, 0.1, 0.5, 1, 1.5, 2, 4, 8, 10]$ and select the best results. For DBSCAN and self tuning spectral, we tune the neighborhood parameter from 1 to 20 by an increment of 1. For kNERF, we tune the scaling parameter between 0.01 and 100 with a step of 0.1. The matrix of fuzzy memberships is initialized randomly.

Figure 20 displays the clustering results of the considered algorithms on dataset 1. As it can be seen, the FCM (Figure 20(a)), GK (Figure 20(b)) and kNERF (Figure 20(d)) were not able to categorize this dataset correctly. Although, kNERF is based on a non linear mapping of the input data, it was not able to categorize this data correctly. This is due mainly to the fact that one global scaling parameter is not sufficient when the input data includes clusters with different local statistics.

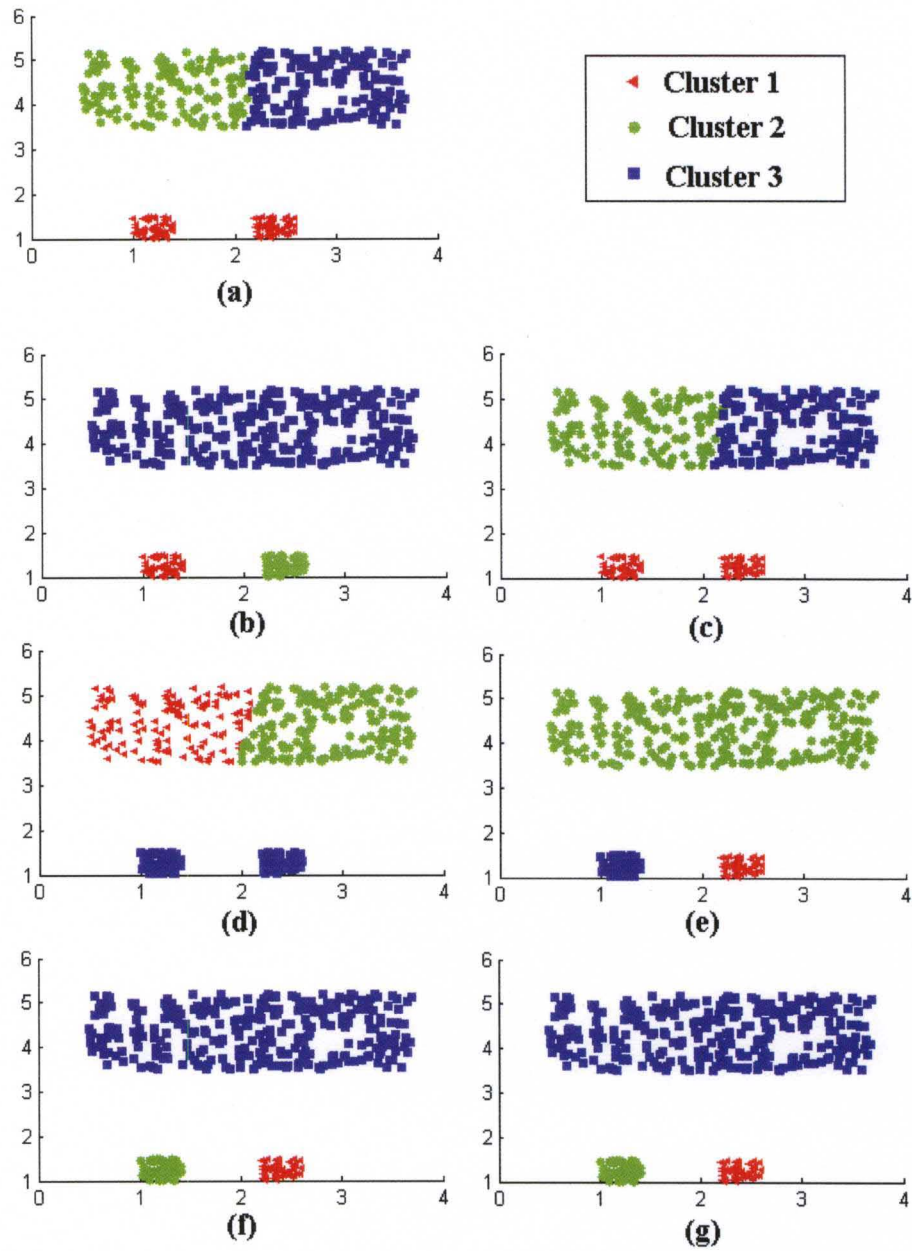


Figure 20. Results of clustering dataset 1 using (a) FCM, (b) DBSCAN, (c) GK, (d) kNERF, (e) Spectral, (f) LSL, and (f) FLeCK

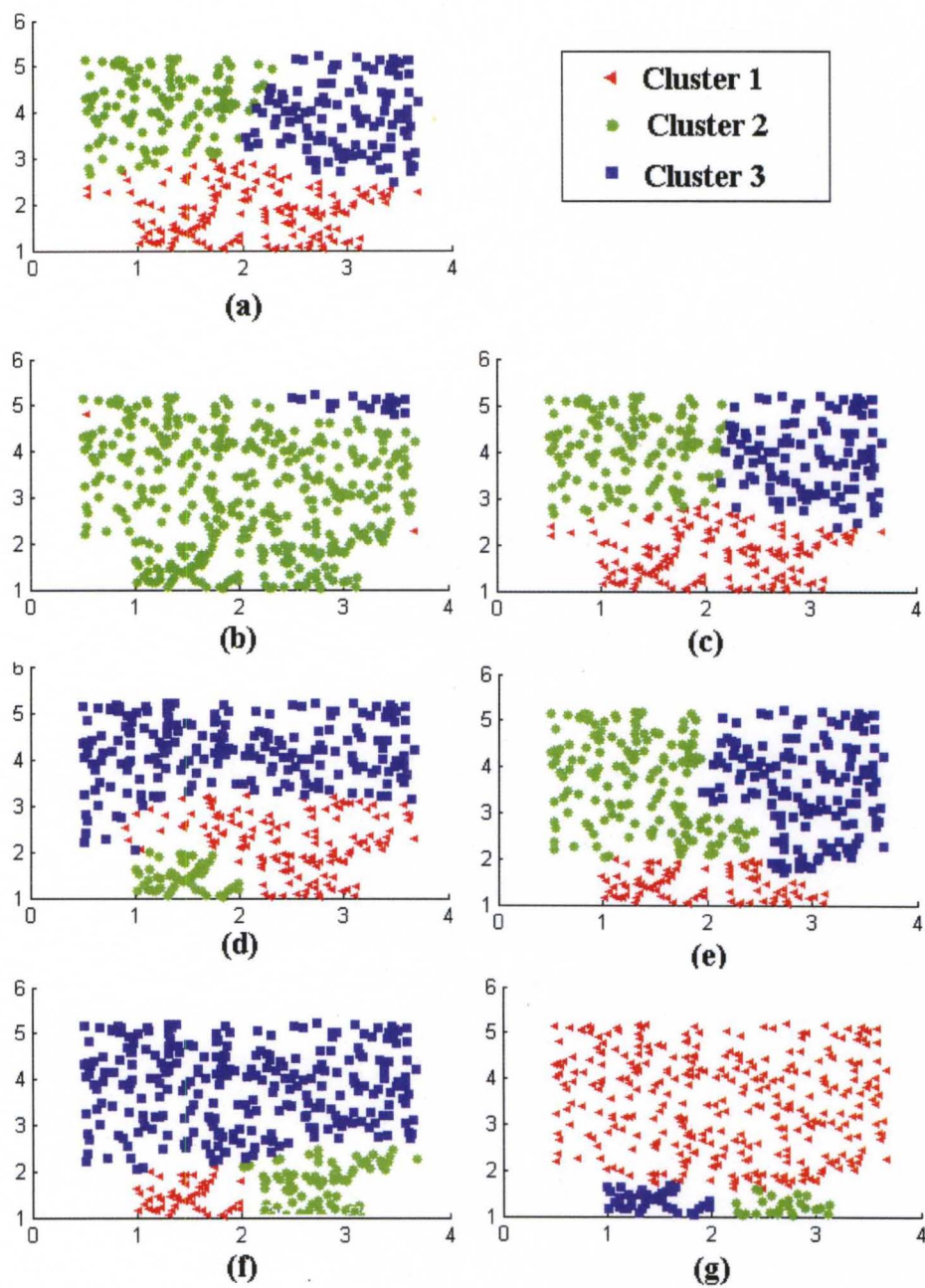


Figure 21. Results of clustering dataset 2 using (a) FCM, (b) DBSCAN, (c) GK, (d) kNERF, (e) Spectral, (f) LSL, and (f) FLeCK

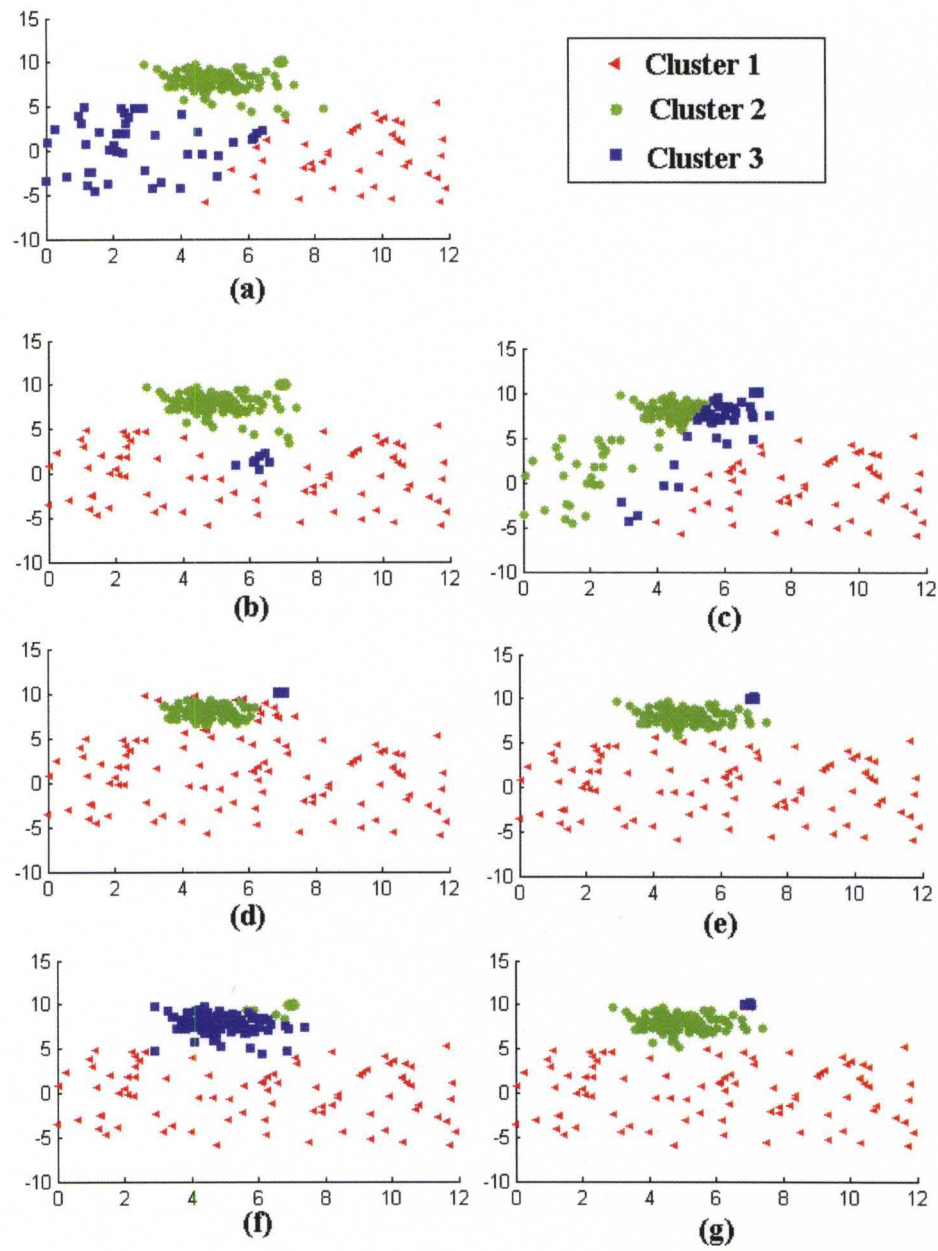


Figure 22. Results of clustering dataset 3 using (a) FCM, (b) DBSCAN, (c) GK, (d) kNERF, (e) Spectral, (f) LSL, and (f) FLeCK

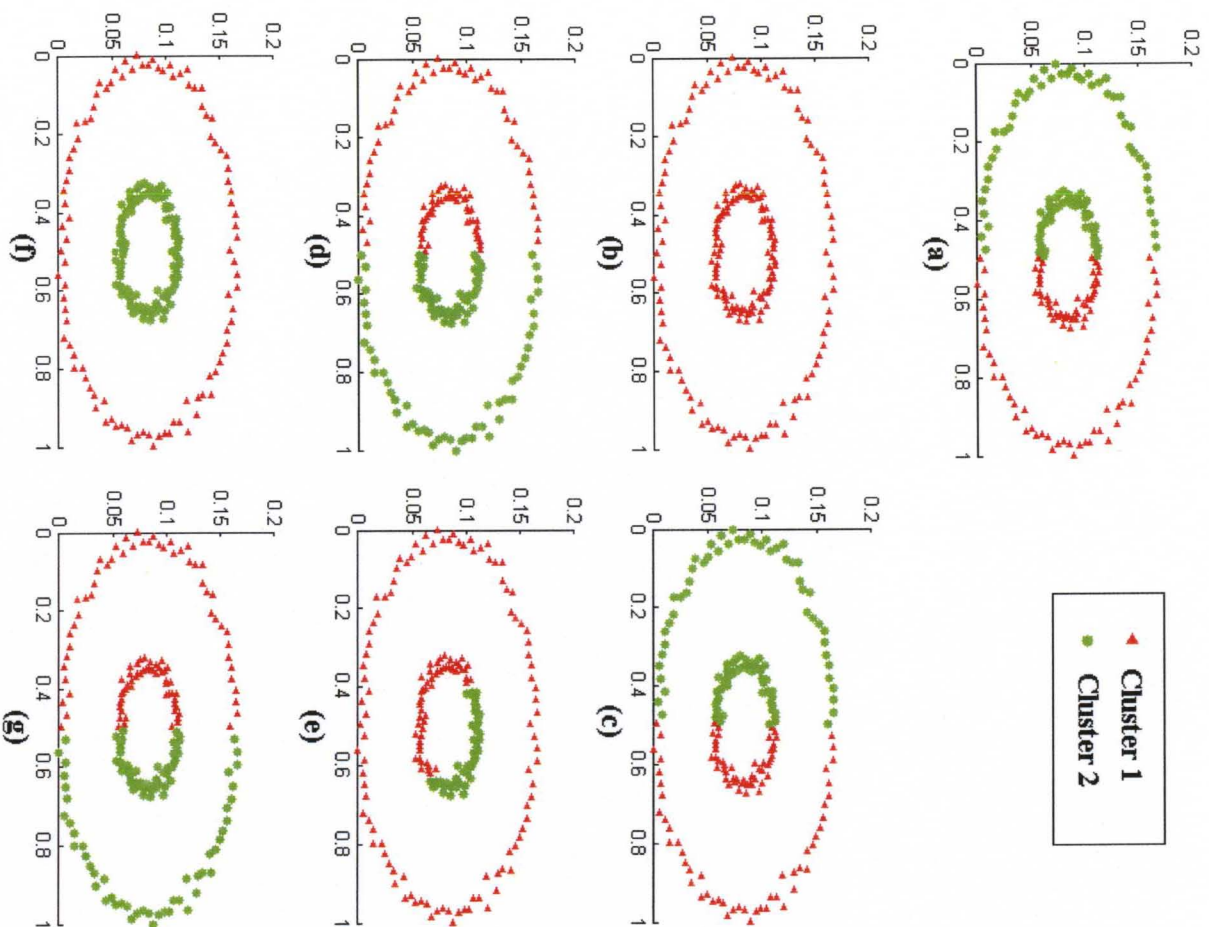


Figure 23. Results of clustering dataset 4 using (a) FCM, (b) DBSCAN, (c) GK, (d) k-NN, (e) Spectral, (f) ISL, and (g) FLeCK

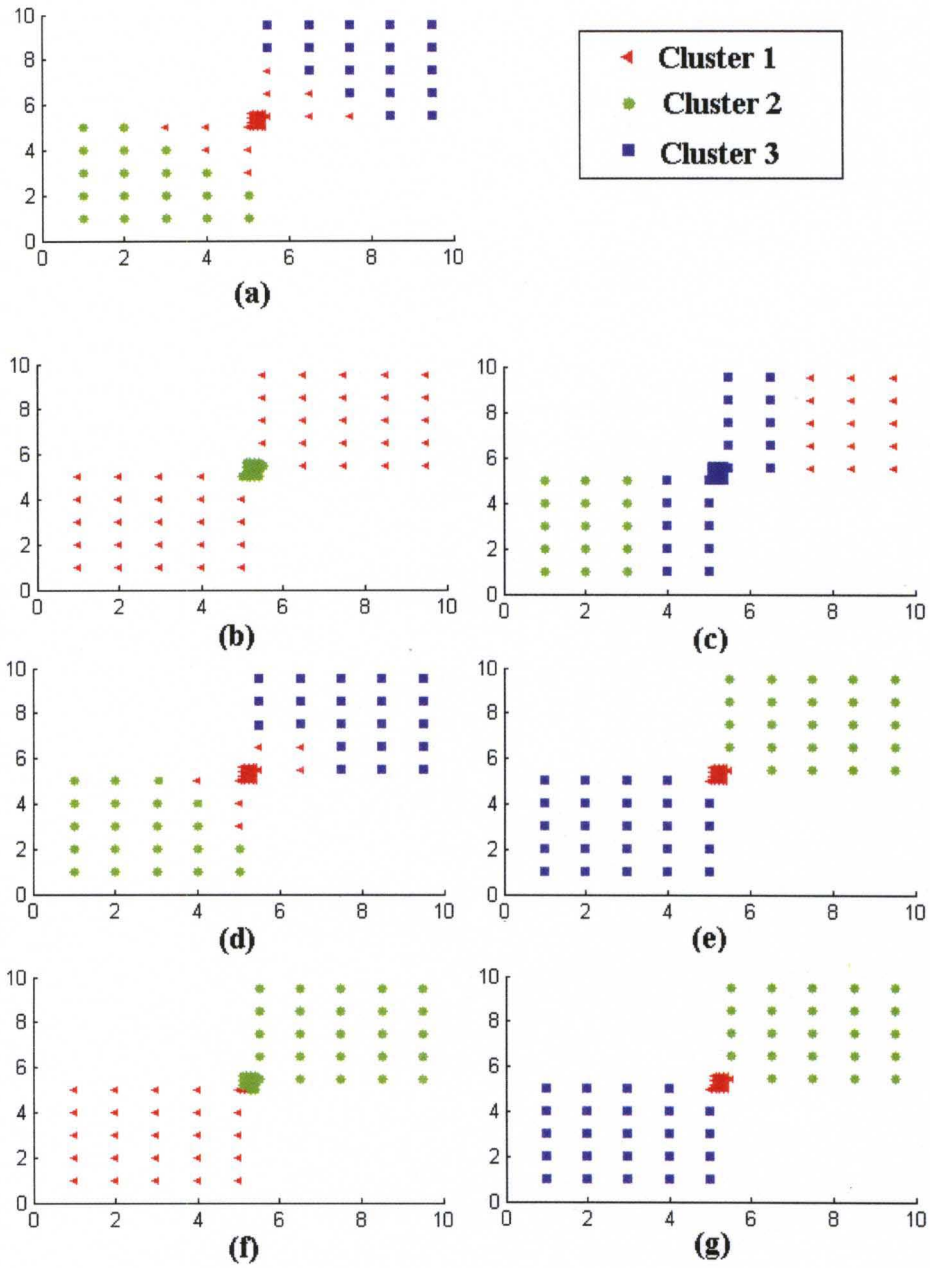


Figure 24. Results of clustering dataset 5 using (a) FCM, (b) DBSCAN, (c) GK, (d) kNERF, (e) Spectral, (f) LSL, and (g) FLeCK

DBSCAN and self tuning spectral were able to categorize dataset 1 correctly after tuning the neighboring parameters. Similarly, LSL categorized dataset 1 cor-

rectly by tuning the K parameter. On the other hand, FLeCK was able to categorize this dataset without any tuning.

Figure 21 displays the clustering results on dataset 2. As it can be seen, neither FCM, DBSCAN, GK, kNERF, or Spectral clustering were able to categorize this data correctly. On the other hand, LSL and FLeCK learned local exponential mapping of the data and were able to partition this data correctly. This example illustrates the inability of existing algorithm to partition data that includes clusters with large intra-cluster and small inter-cluster distances. LSL was able to partition this data correctly by trying several values of K . On the other hand, FLeCK was designed to optimize both criteria simultaneously and was able to partition it correctly without fixing any parameters.

Figure 22 displays the clustering results of dataset 3. Since this dataset contains three clusters of different sizes and densities that are close to each other, widely used clustering algorithms such as FCM [86], DBSCAN [77] and GK [88] are not able to categorize it successfully. On the other hand, kNERF and LSL provide meaningful partition of this dataset. They have categorized few points that are at the clusters boundaries incorrectly. On the other hand, spectral clustering and FLeCK categorized dataset 3 correctly. The self tuning spectral was able to do this by tuning the neighborhood within a user specified range to find local scaling parameters. On the other hand, FLeCK without tuning any parameters learns the local scaling parameters and performs clustering simultaneously.

dataset 4 is constituted of two concentric ovals. It does not correspond to the classical way of perceiving intra cluster and inter cluster distances. As a result, only LSL was able to categorize this data correctly (Figure 23 (f)) after tuning the parameter K .

The fifth synthetic data (Figure 3 (e)) contains two clusters that have the same shape, density, and size, and a third cluster that is denser and much smaller. As shown in Figure 24, only self tuning spectral and FLeCK were able to categorize this dataset correctly. This is due to the fact that this data has intra-cluster distances that are much larger than some inter cluster distances. LSL and FLeCK use local

Gaussian kernel and optimize both the intra-cluster and inter-cluster distances.

2 Application to Image database Categorization

To illustrate the ability of LSL and FLeCK to learn local kernels and to cluster dissimilarity measure derived from real and high dimensional data, we use it to categorize a subset of COREL image database that is described in subsection V-A-1. For object based algorithms (FCM and GK), the feature vectors described in subsection V-A-1 are concatenated to construct a 182 dimensional feature vector. For the relational approaches, the Euclidean distance is used to compute the distance between images with respect to each feature. The relational dissimilarity matrix is the sum of these distances. For all algorithms, we fix the number of clusters to 5 (since we have 5 categories). For the fuzzy algorithms, we fix the fuzzifier m to 1.1. As in the previous examples, we use $K = [0.001, 0.01, 0.05, 0.1, 0.5, 1, 1.5, 2, 4, 8, 10]$ for LSL approach. For DBSCAN and self tuning spectral, we tune the neighborhood parameter from 1 to 20 by an increment of 1. For kNERF, we tune the scaling parameter between 0.01 and 100 with a step of 0.1. Since we should not use the ground truth to select the optimal parameter, for each algorithm, we select the parameter that gives the minimum intra-cluster dissimilarity.

To quantify the performance of the different clustering algorithms and compare them, we assume that the ground truth is known, and we evaluate the performance of the algorithms by using the accuracy, Rand statistics, Q_{Rand} , Jaccard coefficient, $Q_{jaccard}$, Folkes-Mallows index, Q_{FM} , and Hubert index, Q_{Hubert} as described in subsection V-A-3.

Figure 25 compares the performance of the seven considered algorithms. As it can be seen, object-based algorithms (FCM and GK) can not partition this data correctly. In fact, one prototype (center or center and covariance) is not sufficient for sparse high dimensional spaces.

Moreover, kNERF, self tuning spectral and LSL were not able to categorize this image data base as good as FLeCK. In fact, these algorithms are sensitive to the choice of their respective parameters. However, for high dimensional data it is not

trivial to visualize the clustering result in order to choose the parameter that gives the best partition. Moreover, since we should not use the ground truth to compute the performance measures for the purpose of parameter selection, we selected the parameter that gives the minimum intra-cluster dissimilarity. Therefore, the choice of the parameters is not optimal in this case. Thus, although LSL was able to categorize complex data like the co-centric ovals, it cannot partition real world data where categories have generally cloud shapes. Figure 25 shows that FLeCK outperforms all of the other methods with respect to all five measures without tuning any parameter.

Since the self tuning spectral has the closest performance to FLeCK, we will use the partitions of these two algorithms to compare and analyze the results in more details. Tables 22 and 23 summarize the partitions obtained by the self tuning spectral and FLeCK algorithms. For each algorithm, we display one sample image from each cluster, the number of images assigned to the cluster, and the most representative images (i.e. images closest to the cluster center for spectral algorithm and images with high memberships for FLeCK).

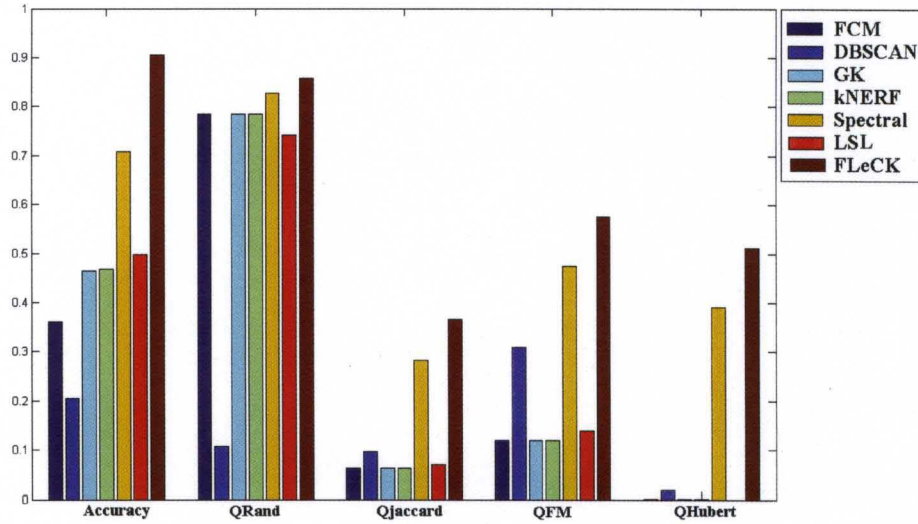


Figure 25. Performance measures obtained on categorizing COREL dataset using FCM, DBSCAN, GK, kNERF, Spectral, LSL and FLeCK clustering approaches.

TABLE 22

Self tuning spectral clustering results on the COREL image data






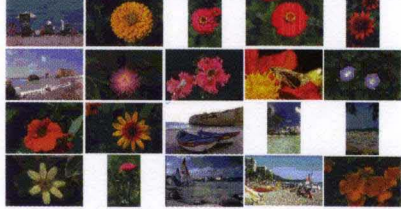





Cluster	Representative Image	No. images	Most representative images
1		79	
2		97	
3		74	
4		133	
5		67	

TABLE 23

FLeCK clustering results on the COREL image data

Cluster	Representative Image	σ_i 's	No. images	Most representative images
1		$\sigma_1 = 11.07$	46	
2		$\sigma_2 = 23.94$	105	
3		$\sigma_3 = 26.72$	90	
4		$\sigma_4 = 35.16$	110	
5		$\sigma_5 = 28.45$	99	

As it can be seen, the self tuning spectral has combined several images from different categories into the same cluster. For instance, cluster 3 contains images from

“Beach” and “Garden” categories, and cluster 4 contains images from “Butterfly” and “Antelope” categories. FLeCK on the other hand, was able to partition most of the data correctly because it adapted a scaling parameter to each cluster. The learned scaling parameters are reported in Table 23 for each cluster. As it can be seen, cluster 4 (“Garden”) has a relatively large scaling parameter ($\sigma_4 = 35.16$). In fact, referring to the inter-group and the intra-group distances in Table 19, we observe that this category has larger intra and inter category distances than the other ones. Categories “Antelope”, “Beach”, and “Bus”, on the other hand, have relatively smaller inter-cluster distances than “Garden” with the “Butterfly” category. Consequently, FLeCK learns smaller scaling parameters ($\sigma_2 = 23.94$, $\sigma_3 = 26.72$, and $\sigma_5 = 28.45$) for the clusters representing these categories. We should note here that σ_2 , σ_3 , and σ_5 could not be as large as σ_4 because their categories have smaller inter-category distance with the “Butterfly” category (refer to Table 19). In fact, a larger σ_2 (“Antelope” cluster) may result in including images of “Butterflies”.

Cluster 1 (“Butterflies”) has the smallest scaling parameter. From Table 19, we notice that category “Butterfly” has the smallest intra-category distances. That is, images within this category form the densest region in the feature space. Moreover, these images are relatively close to images from the “Antelope”, “Beach”, and “Bus” categories. FLeCK was able to identify this cluster correctly by learning a relatively much smaller scaling parameter ($\sigma_1 = 11.07$). A smaller σ prevents images that are spatially close but, not within the dense region, from being assigned to this cluster.

3 Application to categorization of handwritten digits

In this experiment, we compare the performance of the considered clustering algorithms when used to categorize the handwritten digits database which is described in subsection V-A-2. We use this data to illustrate the abilities of LSL and FLeCK to learn local kernels and to cluster dissimilarity measure derived from real and high dimensional data.

We use the same experimental setting as in subsection V-B- 2. That is, for all algorithms, we use the Euclidean distance and we fix the number of clusters to 16.

For the fuzzy algorithms, we fix the fuzzifier m to 1.1. For self tuning spectral, we tune the neighborhood parameter from 1 to 20 by an increment of 1. For kNERF, we tune the scaling parameter between 0.01 and 100 with a step of 0.1. We use $K=[0.001, 0.01, 0.05, 0.1, 0.5, 1, 1.5, 2, 4, 8, 10]$ for LSL approach. Since we should not use the ground truth to select the optimal parameter, for each algorithm, we select the parameter that gives the partition with the minimum sum of intra-cluster dissimilarities.

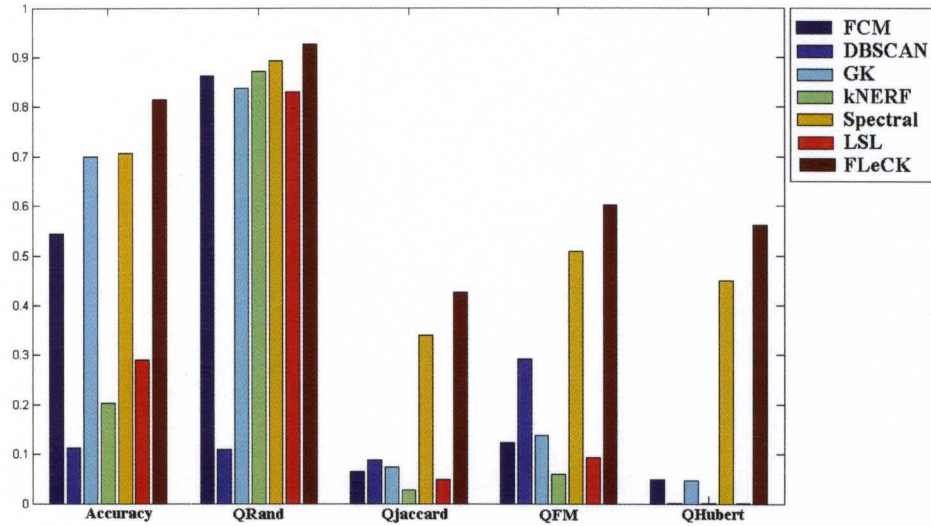


Figure 26. Performance measures obtained on categorizing handwritten digits dataset using FCM, DBSCAN, GK, kNERF, Spectral, LSL and FLeCK clustering approaches.

We compare our clustering results to those obtained using FCM [86], DBSCAN [77], GK [88], kNERF [25], and self tuning spectral [42]. To quantify the performance of the different clustering algorithms and compare them, we assume that the ground truth is known, and we evaluate the performance of the algorithms by using the accuracy, Rand statistics, $QRand$, Jaccard coefficient, $Qjaccard$, Folkes-Mallows index, QFM , and Hubert index, $QHubert$, [20] as described in subsection V-A-3.

Figure 26 compares the performance of the considered algorithms. As it can be seen, prototype based clustering (FCM and GK) are not able to categorize this data correctly. In fact, prototype based clustering approaches cannot deal with high di-

mensional data. Similarly, kNERF, self tuning spectral and LSL were not able to categorize this dataset correctly. In fact, as for the COREL dataset (see subsection V-B-2), they did not give good clustering on high dimensional real data because it is not trivial to fix their respective tuning parameters.

As it can be seen from Figure 26, FLeCK outperforms the other methods. Since the self tuning spectral has the closest performance measure to FLeCK, we will use the partitions of these two algorithms to compare and analyze the results in more details.

TABLE 24

Self tuning spectral clustering results on the handwritten digit Data

		True class										
		'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	Total
Clusters	1	0	7	0	4	0	0	0	7	0	3	21
	2	0	1	0	0	0	0	0	44	0	0	45
	3	0	0	0	0	37	0	0	0	0	7	44
	4	0	0	0	0	65	0	0	0	0	0	65
	5	14	0	0	0	0	0	0	0	0	0	14
	6	0	19	0	0	0	0	0	0	0	0	19
	7	0	0	0	0	0	0	108	0	0	0	108
	8	50	0	0	0	0	0	0	0	0	0	50
	9	1	1	0	1	6	59	6	14	0	27	112
	10	0	5	58	0	0	0	0	0	0	0	63
	11	0	0	0	0	0	0		41	0	0	41
	12	0	15	0	0	0	0	0	0	0	0	15
	13	63	0	0	1	0	57	0	0	109	70	300
	14	0	51	49	0	0	0	0	1	0	0	101
	15	0	0	0	115	0	3	0	0	0	4	122
	16	0	32	0	1	0	0	0	10	0	0	43

TABLE 25

FLeCK clustering results on the handwritten digit Data

		True class										
		'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	Total
Clusters	1	0	0	0	0	0	0	0	0	41	0	41
	2	0	55	59	0	0	0	0	1	1	0	156
	3	45	0	0	0	0	0	0	0	1	0	46
	4	9	0	0	0	0	0	0	0	0	0	9
	5	45	0	0	0	0	0	0	0	0	0	45
	6	0	0	0	0	0	0	0	0	59	1	60
	7	0	0	0	0	0	0	105	1	0	0	106
	8	0	0	0	1	0	2	0	0	0	76	79
	9	0	1	0	116	0	34	0	12	0	14	177
	10	0	19	0	0	5	26	8	0	1	1	60
	11	0	0	0	0	68	0	0	0	0	3	71
	12	0	54	0	5	0	0	0	18	0	8	85
	13	0	0	0	0	0	57	0	0	5	0	62
	14	0	2	8	0	0	0	0	85	0	0	95
	15	29	0	0	0	0	0	1	0	1	0	31
	16	0	0	0	0	35	0	0	0	0	8	43

The self tuning spectral did relatively better than the other existing approaches. However, as it can be seen from Table 24, it has created one big cluster that contains 300 elements from categories '0', '6', '8' and '9'. Moreover, although category '1' was split in five clusters (cluster 1, 6, 12, 14, and 16), it was not well segregated in clusters 1 and 14. This may be due to the fact that adapting one σ to each sample distorted the structure of the data.

TABLE 26

The scaling parameters learned by FLeCK for the clusters of the handwritten digits

Clusters	No. of samples	σ_i
1	41	16.4
2	156	49.2
3	46	20.3
4	9	3.4
5	45	23.6
6	60	18.5
7	106	29.3
8	79	27.4
9	177	50.7
10	60	17.3
11	71	17.4
12	85	27.8
13	62	31.7
14	95	25.6
15	31	11.2
16	43	10.11

FLeCK on the other hand, was able to partition the data better than self tuning spectral. Referring to Table 25, some digits have been categorized using more than one cluster (clusters 0, 4, 5 and 8). This is due to the different writing styles. For instance, digit '4' has been categorized using two clusters (cluster 11 and 16). Figure 27 displays the top 10 representatives of these two clusters (corresponding to the highest memberships). We can notice that each cluster contains one style of writing the digit '4'. On the other hand, FLeCK categorized other digits using one cluster (clusters 1, 2, 3, 6, 7, and 9). These are categories with more consistent writing style.

Figure 28 displays the top 10 representatives of one such cluster representing digit 3.

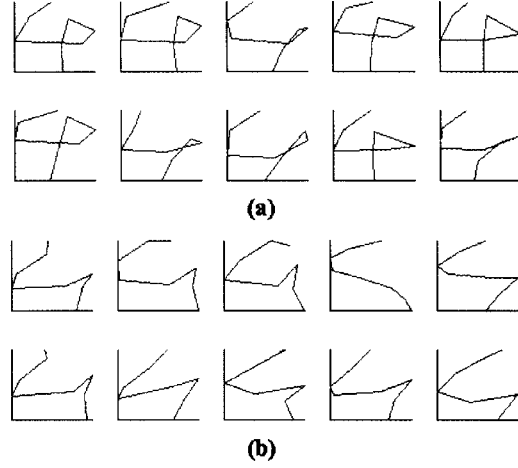


Figure 27. The top 10 representatives of (a) cluster 11 and (b) cluster 16 obtained by FLeCK.

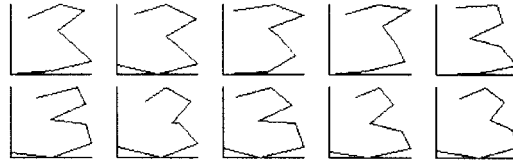


Figure 28. The top 10 representatives of cluster 9 obtained by FLeCK.

FLeCK learned a scaling parameter for each cluster. These parameters are reported in Table 26. For instance, cluster 4 is the smallest cluster and contains only 9 digits. These digits form the densest region in the feature space (small sum of intra cluster distances) and are relatively close to other digits categories (small sum of inter cluster distances). Consequently, FLeCK learns a small scaling parameter ($\sigma_4 = 3.4$) for this cluster. This Small σ prevent digits that are spatially close but not within the dense region, from being assigned to this clusters. On the other hand, FLeCK leaned a relatively larger scaling parameter for cluster 9 ($\sigma_9 = 50.65$) representing the digit 3. This is due to the fact that this cluster includes a large number of samples with large intra-cluster distances (without being split into small clusters).

4 Time complexity

TABLE 27

The simulation time (in second) for kNERF, the self tuning spectral, and the FLeCK algorithms

	No. of points	No. of clusters	Spectral	FLeCK
dataset 1	400	3	20.93 s	11.38s
dataset 2	400	3	21.03 s	11.27s
dataset 3	293	3	11.83 s	6.34 s
dataset 4	200	2	3.81 s	2.61 s
dataset 5	66	3	0.49 s	0.31 s
COREL data	450	5	49.68 s	16 s
digit data	1166	16	1323.05 s	144.20 s

Since the self tuning spectral has the closest performance measure to FLeCK, we access the performance of FLeCK, and self tuning spectral [42] by comparing their performance in terms of computational time. As we can notice from Table 27, FLeCK is computationally less expensive than self tuning spectral. This becomes more significant as the size of the dataset increases. In fact, though self tuning spectral can produce high-quality clustering on small datasets, it has limited applicability to large-scale problems due to its computational complexity of $O(N^3)$ where N is the number of data points [33].

5 Conclusions

The experimental results presented in this section demonstrate the effectiveness of our proposed clustering algorithms. In addition to giving better clustering results than the other clustering algorithms, FLeCK presents the advantage of auto-learning

of the scaling parameter. It does not need to tune any parameters. This is an important advantage since it may not be evident to even choose a range of possible values. In fact, in the 2 D experiments, we have selected the parameters for self tuning spectral based on visualizing the results and selecting the best one. Although this can be performed on 2D dataset, it is not possible for high dimensional dataset where visualization may not be trivial and the ground truth is not known.

C Evaluation of the semi-supervised clustering algorithms

In this section, we compare the semi-supervised versions of LSL and FLeCK (SS-LSL and SS-FLeCK) clustering results to those obtained by the most related semi-supervised clustering approaches. Namely, we compare our results to those obtained using the semi-supervised kernel C-means (SS-kernel-C-Means) [27], the Semi-Supervised Spectral Learning algorithm [44] and the Semi-supervised graph clustering algorithm [1]. The five considered approaches are relational algorithms, and the Euclidean distance is used to compute the relational dissimilarities for all of them. To assess the performance of the different clustering algorithms and compare them, we use the ground truth, and the same five performance measures described in subsection V-A-3.

Ideally, the supervision information should be provided by the user (in terms of feedback) in an interactive mode. However, to carry out an objective experiment (for the purpose of algorithm evaluation), we automate the process of constraints selection and attempt to simulate the user’s feedback. We assume that boundary points are the most informative points, and that these points should be selected for the supervision information. First, we ran the considered algorithms without supervision for few iterations. Then, we identify 2% of boundary points (points with low fuzzy membership values in all clusters). Next, we use the ground truth of these points to construct must-link and cannot-link constraints.

1 Synthetic datasets

We have shown in subsection IV-A-2 and subsection IV-B-2 how SS-LSL and SS-FLeCK have enhanced LSL and FLeCK clustering results on 2D datasets by using a small amount of side information. In order, to better assess the performance of SS-LSL and SS-FLeCK on these datasets, we compare their clustering results with other clustering approaches as mentioned above. We use the three synthetic datasets where the unsupervised algorithms (LSL and FLeCK) did not perform well. The three datasets are displayed in Figure 3 (b), 3 (d), and Figure 3 (e).

Figure 29 displays the clustering results on dataset 2. As it can be seen, neither SS-kernel-C-Means (Figure 29 (a)), semi-supervised spectral (Figure 29 (b)) or semi-supervised graph clustering (Figure 29 (c)) were able to categorize this dataset correctly. The main reason is that these three algorithms use a global scaling parameter and thus, they are not able to deal with the local characteristics of this data even when partial supervision information is used to guide them.

As mentioned in subsection IV-C-2, the points at the boundaries of cluster 2 and cluster 3 are better categorized by LSL approach (refer to Table 2). Similarly, the overlapping boundaries between the three clusters are better characterized using SS-FLeCK (refer to Table 9). Thus, the partial supervision guided effectively the LSL and FLeCK towards a better categorization of the data.

For dataset 4, and as it can be seen in Figure 30, the SS-kernel-C-Means (Figure 30 (a)), semi-supervised spectral (Figure 30 (b)), and semi-supervised graph (Figure 30 (c)) were not able to partition this data correctly. This confirms that the problem is not likely to be caused by bad initialization or by getting trapped in a local minima, but because a global scaling parameter is not appropriate when the data has different local statistics. On the other hand, the partial supervision has guided SS-FLeCK to partition dataset 4 correctly (Figure 30 (e)).

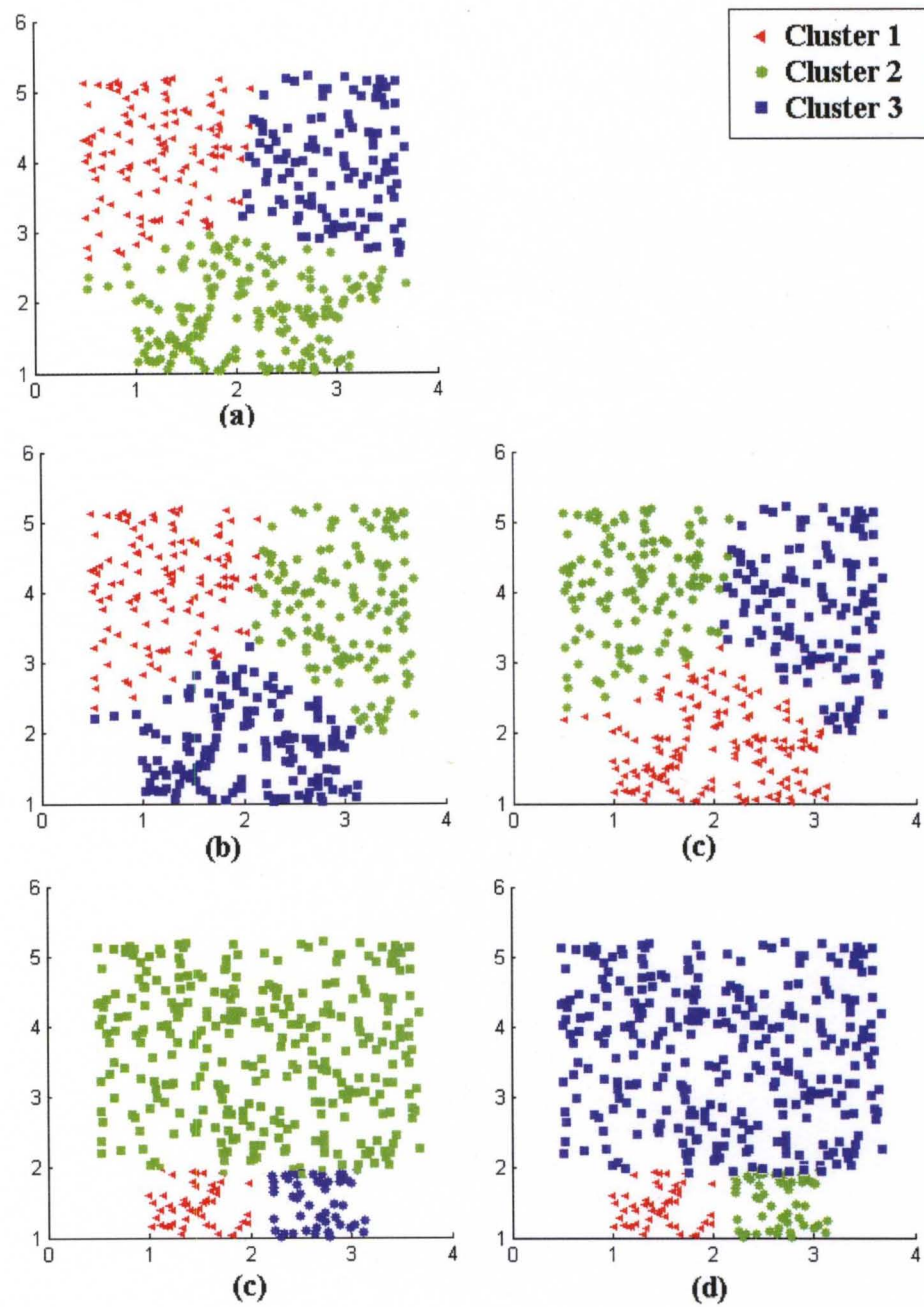
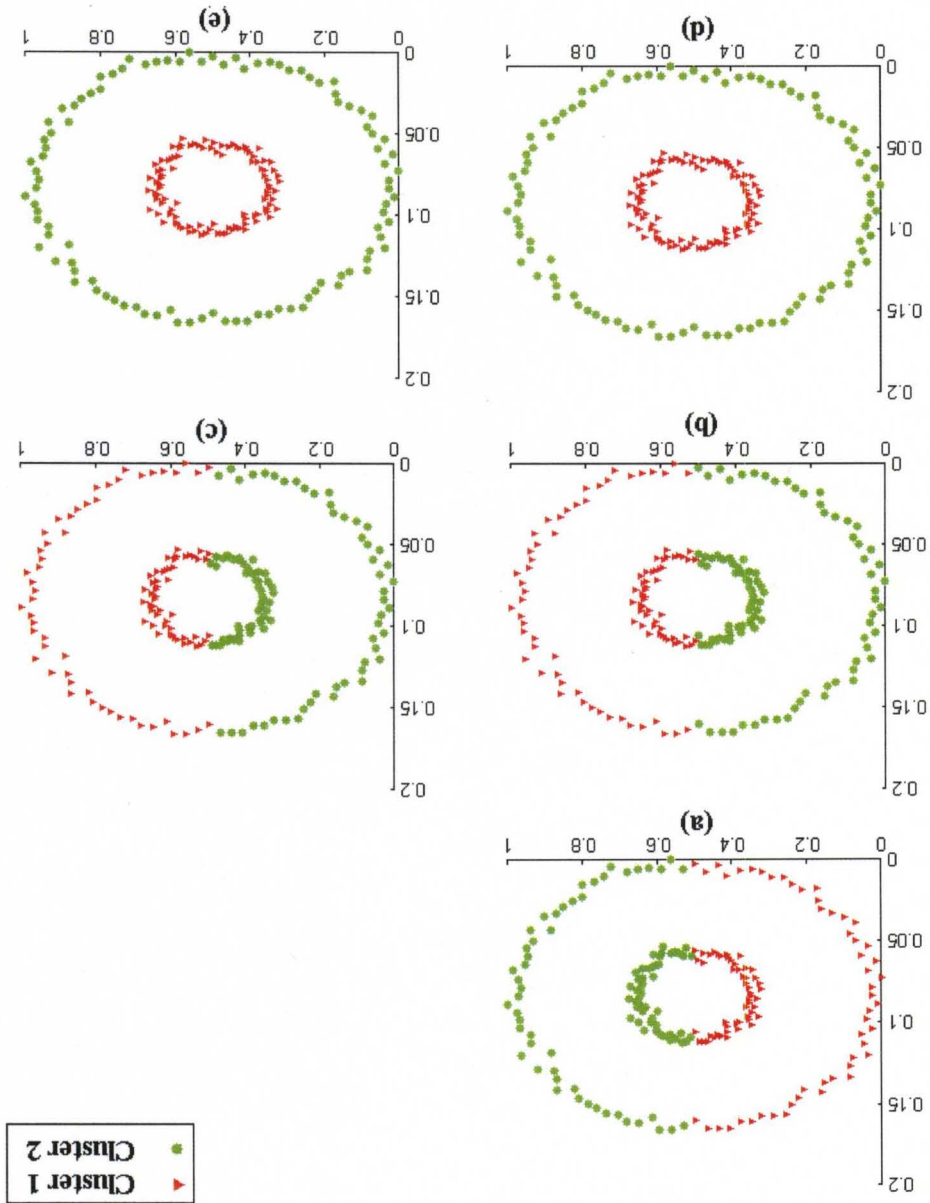


Figure 29. Results of clustering dataset 2 using (a) SS-kernel-C-Means, (b) semi-supervised spectral learning, (c) semi-supervised graph clustering, (d) SS-LSL, and (e) SS-FLeCK.

Figure 30. Results of clustering dataset 4 using (a) SS-kernel-C-Means, (b) semi-supervised spectral learning, (c) semi-supervised graph clustering, (d) SS-LSL, and (e) SS-FLeCK.



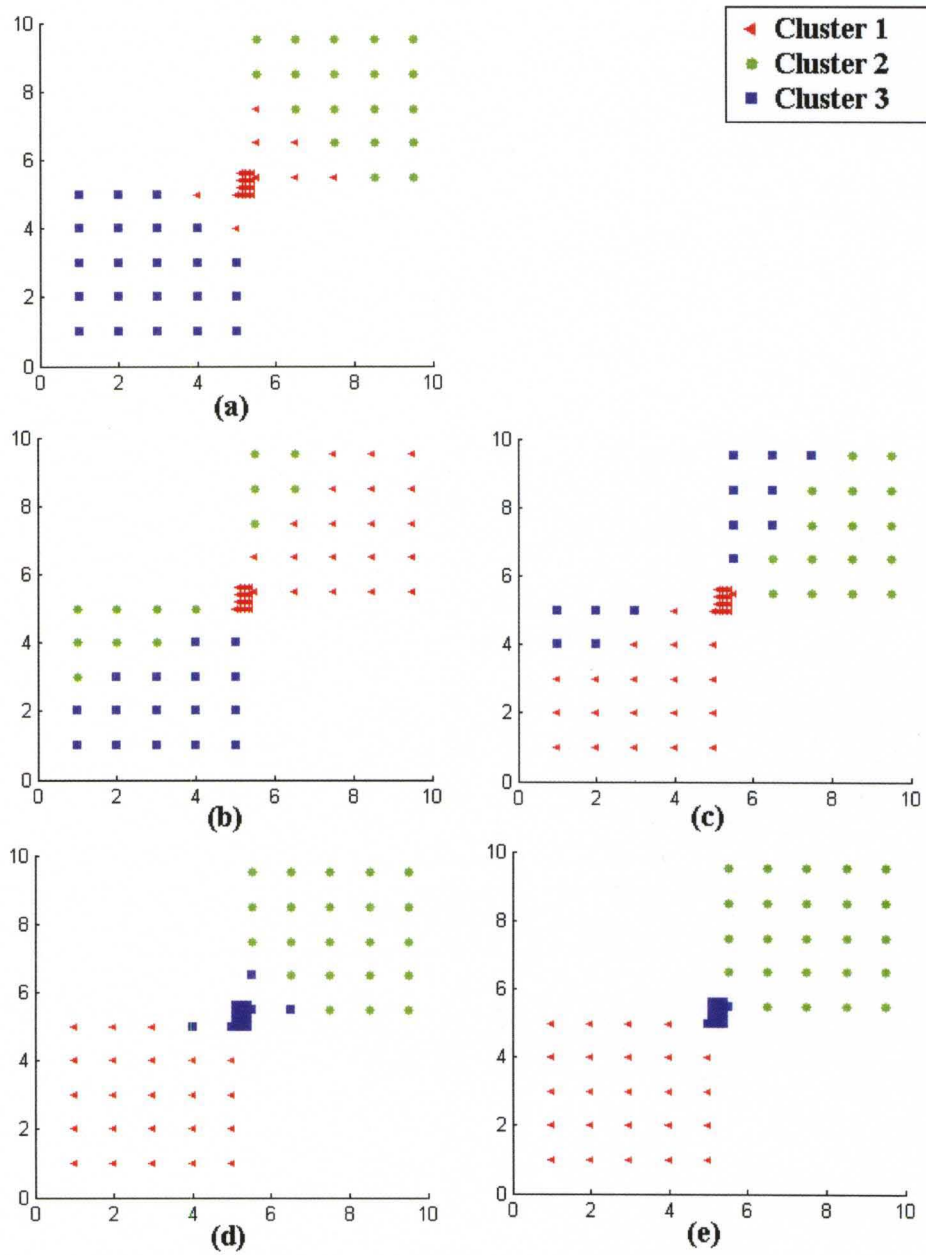


Figure 31. Results of clustering dataset 5 using (a) SS-kernel-C-Means, (b) semi-supervised spectral learning, (c) semi-supervised graph clustering, (d) SS-LSL, and (e) SS-FLeCK.

Figure 31 displays the clustering results on dataset 5 using the five clustering

algorithms. As it can be seen, SS-kernel-C-Means (Figure 31 (a)), semi-supervised spectral learning (Figure 31 (b)), and the semi-supervised graph clustering (Figure 31 (c)) were not able to partition this dataset. As with previous examples, this is also due to the one global scaling parameter used by these algorithms. The partial supervision on the other hand has guided SS-LSL to a better partition of the data (Figure 31 (d)).

2 Application to image database categorization

To evaluate the performance of SS-LSL and SS-FLeCK on the COREL dataset, we compare their results to those obtained using the semi-supervised kernel C-means (SS-kernel-C-Means) [27], the Semi-Supervised Spectral Learning algorithm [44] and the Semi-supervised graph clustering algorithm [1]. We first perform the unsupervised versions of the five considered algorithms. Then, we run their corresponding semi-supervised versions 10 times and incrementally add 1% of the total number of images as pairwise constraints each time. The performance of the five algorithms, measured in terms of accuracy rate, on the multiple runs is shown in Figure 32. As it can be seen, all these algorithms improve in performance when supervision is increased. However, the SS-FLeCK has the best performance for this dataset. This Figure also shows that a small amount of supervision has allowed SS-LSL to improve significantly its performance. In fact, when the percentage of data samples used for supervision is equal to 10 %, SS-LSL outperforms Semi-Supervised spectral learning and Semi-Supervised graph clustering, and has the same performance as SS-kernel-C-Means.

Table 28 compares the number of images used to construct the constraints and the number of images with improved categorization when the constraint ratio is equal to 2%. We note that the improvement is much higher than the number of images used to construct the constraints. That is, the 2% of supervision information has guided the algorithm to categorize more images correctly.

As mentioned earlier, the partial supervision information is used to guide the clustering algorithm and can make them less sensitive to initialization. To illustrate this point, we compare the performance of the unsupervised and semi-supervised

algorithm (with 2% of supervision) for 20 runs with different initializations. We use the performance measures defined in subsection V-A-3.

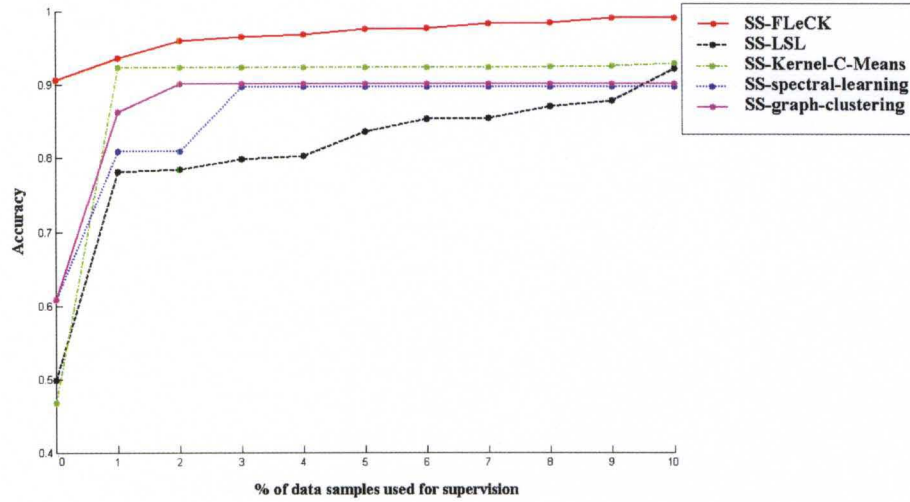


Figure 32. The accuracy of the five semi-supervised algorithms on COREL data as the number of constraints used for partial supervision is increased.

TABLE 28

Number of images used to construct the constraints versus the number of images with improved categorization

Algorithm	Number of images used to construct the constraints	Number of images with improved categorization
SS-LSL	9	58
SS-FLeCK	9	18

Figure 33 compares the mean and standard deviation of the five performance measures obtained on the COREL dataset using LSL and SS-LSL. Similarly, Figure 34 compares the mean and standard deviation of these performance measures using FLeCK and SS-FLeCK. First, we note that the semi-supervised algorithms

outperform the unsupervised version since their performance mean is always larger. Second, the standard deviation of the performance of the semi-supervised algorithm is much smaller than that of the unsupervised version. This indicates that the partial supervision makes the algorithm less sensitive to initialization.

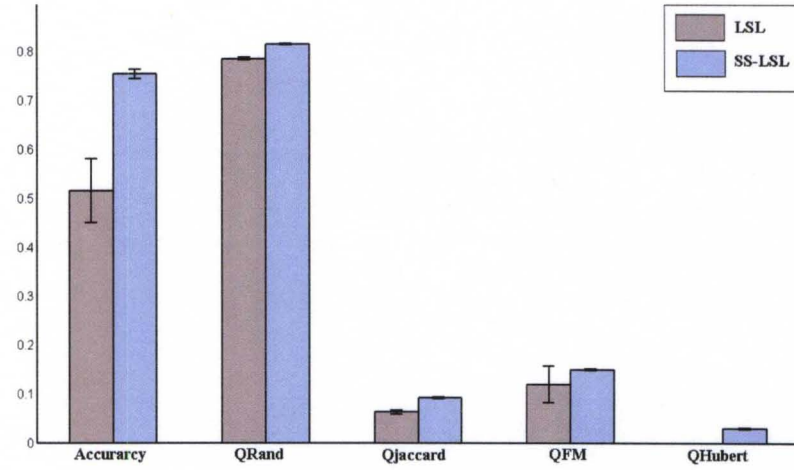


Figure 33. Mean and standard deviation of the five performance measures over 20 runs of LSL and SS-LSL on the COREL dataset.

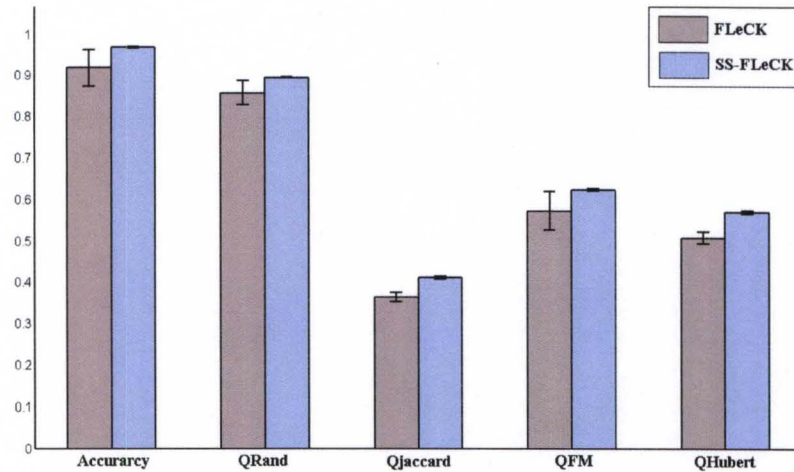


Figure 34. Mean and standard deviation of the five performance measures over 20 runs of FLeCK and SS-FLeCK on the COREL dataset.

3 Application to categorization of handwritten digits

Figure 35 reports the accuracy of the five considered algorithms as we increase the amount of supervision information. As expected, the accuracy of all algorithm improve as we increase the number of constraints. Moreover, as it can be seem, SS-LSL outperforms SS-kernel-C-Means, SS-spectral learning and SS-graph clustering when the percentage of data sample used for supervision is above 2%. We also note that SS-FLeCK outperforms all the other approaches.

Table 29 reports the number of digits used to construct the constraints versus the number of digits with improved categorization when the ratio of constraints is set to 2%. As it can be seen, the number digits with improved categorization is much larger than the number of digits used to construct the constraints. This indicates that the partial supervision has guided the algorithm to a more global optima.

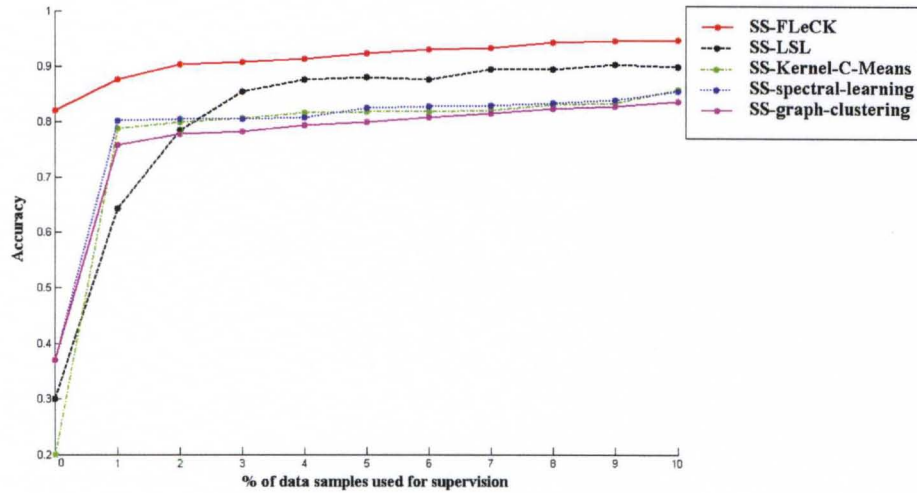


Figure 35. The accuracy of the five semi-supervised algorithms on handwritten digits data as the number of constraints used for partial supervision is increased.

TABLE 29

Number of digits used to construct the constraints versus the number of digits with improved categorization

Algorithm	Number of digits used to construct the constraints	Number of digits with improved categorization
SS-LSL	23	109
SS-FLeCK	23	58

Figure 36 and Figure 37 report the mean and standard deviation of the performance measures of LSL versus SS-LSL and FLeCK versus SS-FLeCK, respectively, over 20 runs with different initializations. As with the image database, we note that in addition to giving better performance measures than their respective unsupervised clustering versions, the SS-LSL and SS-FLeCK are less sensitive to initialization and their final partition is more consistent.

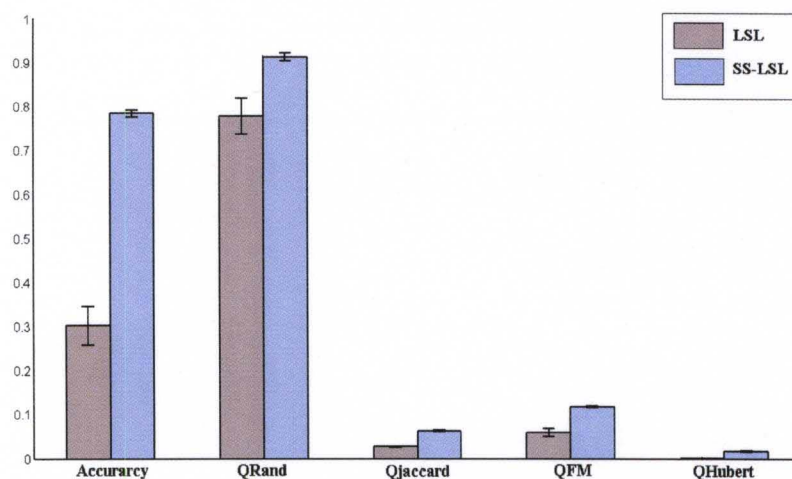


Figure 36. Mean and standard deviation of the five performance measures over 20 runs of LSL and SS-LSL on the handwritten digits dataset.

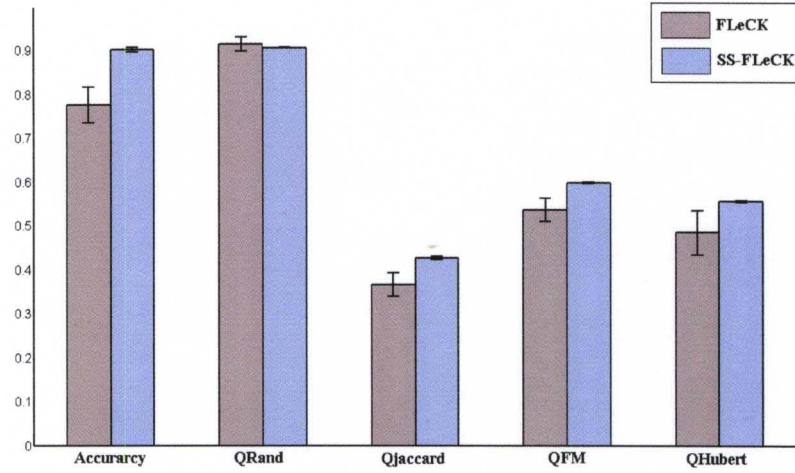


Figure 37. Mean and standard deviation of the five performance measures over 20 runs of FLeCK and SS-FLeCK on the handwritten digits dataset.

4 Conclusions

In this section, we have illustrated the clustering performance of SS-LSL and SS-FLeCK on synthetic 2D datasets. We have shown that SS-LSL and SS-FLeCK outperform other related semi-supervised clustering algorithms on 2D synthetic data. In particular, we have shown that a small subset of constraints can guide the algorithm towards a more optimal partition, and thus, improving the categorization of many more samples. We have also shown that the partial supervision can make the algorithm less sensitive to initialization and local minima.

CHAPTER VI

CONCLUSIONS AND POTENTIAL FUTURE WORKS

A Conclusions

Clustering is a challenging task especially when the structure of the data does not correspond to easily separable categories, and when clusters vary in size, density and shape. Existing kernel based approaches allow the use of a specific similarity measure in order to make the problem easier. The choice of such a kernel function allows the mapping of the input data into a new space in such a way that computing a simple partitioning in this feature space results in a nonlinear partitioning in the input space. One of the most common dissimilarity functions, due to its analytical properties, is the Gaussian kernel function. Although good results were obtained using this kernel, generally, its performance depends on the selection of the scaling parameter σ . This selection is commonly done by trying several values. Moreover, since one global parameter is used for the entire dataset, it may not be possible to find one optimal σ when there are large variations between the distributions of the different clusters in the feature space.

One way to learn optimal scaling parameters is to try several combinations, evaluate each partition using some validity measure, and identify the optimal partition. However, this exhaustive search of one scaling parameter with respect to each cluster is not practical. It is computationally expensive and increases significantly with the number of clusters and the range of possible values of σ_i . Moreover, it may not be possible to quantify the optimal partition.

In this thesis, we addressed this limitation and proposed two new fuzzy relational clustering techniques that learn cluster dependent σ_i in an efficient way: the clustering and Local Scale Learning algorithm (LSL) and the Fuzzy clustering ap-

proach with Learnable Cluster dependent Kernels (FLeCK).

The clustering and Local Scale Learning algorithm (LSL) learns the underlying cluster dependent dissimilarity measure while finding compact clusters in the given dataset. The learned measure is a Gaussian dissimilarity function defined with respect to each cluster that allows to control the scaling of the clusters and thus, improve the final partition. We minimize one objective function for both the optimal partition and for the cluster dependent scaling parameter. This optimization is done iteratively by dynamically updating the partition and the local measure in each iteration. This make the kernel learning task takes advantages of the unlabeled data and reciprocally, the categorization task takes advantages of the local learned kernel. Moreover, LSL is formulated to work on relational data. This makes it applicable even when clusters of similar objects cannot be represented efficiently by a single prototype. It is also more practical when similar objects cannot be represented efficiently by a single prototype.

To derive the update equation for σ_i , LSL uses the heat flow approximation. While this assumption allowed LSL to deal with clusters with irregular shapes, LSL still requires the specification of a parameter K . This parameter which is a function of the regularization term K_1 and the local geometric characteristics of the data, can affect the clustering results. To overcome the need to specify K , we proposed a second algorithm, called the Fuzzy clustering approach with Learnable Cluster dependent Kernels (FLeCK) that optimizes both the intra-cluster and inter-cluster distances. Instead of using the heat flow approximation while deriving the update equations, FLeCK uses a different approximation that assumes that the cluster's scaling parameters do not vary significantly from one iteration to another. The scaling parameter, σ_i , with respect to each cluster i is designed to distinguish and separate the objects of cluster i from the rest of the data. It reflects the relative density, size, and position of this cluster with respect to the other clusters.

To the best of our knowledge, LSL and FLeCK are the first algorithms that learn the Gaussian scaling parameter in an unsupervised way. This is a major contributions to Gaussian based clustering approaches such as kernel and spectral clustering methods that suffer from their sensitivity to this parameter.

We have illustrated the clustering performance of LSL and FLeCK on synthetic 2D datasets and on high dimensional real data. Our experimental results on 2-D datasets have demonstrated the effectiveness of LSL and FLeCK. In addition, we showed that the learned scaling parameters and the fuzzy memberships returned by LSL and FLeCK are meaningful and reflect the geometric characteristic of the data. We have showed that the scaling parameters learned by FLeCK are not only influenced by the intra-cluster distances, but also by the relative cluster positions, densities and sizes. This allows a better description of the data and consequently, a better partition of the data.

Our experiments on real and high dimensional data have indicated that LSL may not perform well on high dimensional data. One reason for this suboptimal behavior is the difficulty in specifying the parameter K . Another reason is related to the inherent limitation of all clustering algorithms that minimize the intra-cluster distances without considering the inter-cluster distances. FLeCK was designed to overcome these limitations and has proved to outperform other algorithms. In addition, FLeCK presents the advantage of auto-learning of the scaling parameter.

Both the LSL and FLeCK algorithms minimize complex objective functions that are prone to several local minima and sensitive to initialization. This problem is more acute for high dimensional datasets. Thus, if a small amount of prior knowledge is available, it can be used to guide the clustering algorithms to avoid most local minima and obtain a better partition. In the second part of this thesis, we presented two semi-supervised clustering approaches: the Semi-Supervised clustering and Local Scale Learning algorithm (SS-LSL), and the Semi-Supervised Fuzzy clustering with Learnable Cluster dependent Kernels (SS-FLeCK). We assume that for both algorithms we have a set of pairwise *"Should-Link"* constraints (pairs of points that should belong to the same cluster) and a set of *"Should not-Link"* constraints (pairs of points that should belong to different clusters).

We have illustrated the clustering performance of SS-LSL and SS-FLeCK on synthetic 2D datasets and on high dimensional real data. We have shown that SS-LSL and SS-FLeCK outperform other related semi-supervised clustering algorithms.

In particular, we have shown that using a small subset of constraints can guide the algorithm towards a more optimal partition, and thus, improving the categorization of many more samples. We have also shown that the partial supervision can make the algorithm less sensitive to initialization and local minima.

B Potential future work

1 Feature weighting and kernel Learning

Since the influence of the features is generally not equally important in the definition of the category to which similar patterns belong and application dependent, the problem of selecting the best subset of features or attributes constitutes a way of integrating domain expertise in the clustering algorithms. Most feature weighting and selection methods assume that feature relevance is invariant over the task's domain. As a result, they learn a single set of weights for the entire dataset. This assumption can degrade the performance of the learning system when the dataset is made of different categories or classes. One possible solution is to perform clustering and feature discrimination simultaneously. This can be achieved by integrating feature weighting in the process of learning Gaussian kernels. The resulting algorithm would partition the data into clusters, and learns a scaling parameter simultaneously. The learned scaled parameter would be feature dependent with respect to each cluster.

2 Prototype based classifier

The k-Nearest neighbor classifier (kNN) is used for many pattern recognition applications where the underlying probability distribution of the data is unknown a priori. Traditional kNN stores all the known data points as labeled prototypes. This makes the algorithm computationally prohibitive for very large database, due to the limitation of computer storage and the cost of searching for the nearest neighbors of an input vector. To overcome the above challenges several techniques have been proposed. Most of them can reduce the searching time for the nearest neighbors but do not decrease the storage requirements. One way of reducing both the search

time and the storage requirements is by using clustering algorithm to summarize the data and identify a subset of prototypes. This task is not trivial, especially when dealing with high dimensional datasets. In fact, Euclidean distance measures in high dimensional space are measured across volume. However, volume increases exponentially as dimensionality increases, and points tend to become equidistant. One way to overcome this limitation is to use simultaneous clustering and kernel learning to summarize the large set of training samples by few representative prototypes. Each prototype is a cluster that is identified by the clustering approach and would have its own learned distance measure. This region dependent distances would be explored by the classifier to retrieve more relevant neighbors and improve its accuracy.

REFERENCES

- [1] B. Kulis, S. Basu, I. Dhillon and R. Mooney. "Semi-supervised graph clustering: a kernel approach", *Machine Learning*, Vol. 74, pp. 1-22, 2009.
- [2] R. Chatpatanasiri, T. Korsrilabutr, P. Tangchanachaianan, and B. Kijsirikul. "On kernelization of supervised Mahalanobis distance learners", *ArXiv*, 2008.
- [3] S. Chatzis and T. A. Varvarigou, "A Fuzzy Clustering Approach Toward Hidden Markov Random Field Models for Enhanced Spatially Constrained Image Segmentation", *IEEE T. Fuzzy Systems*, vol. 16, 2008.
- [4] J. V. Davis and I. S. Dhillon. "Structured metric learning for high dimensional problems" , *KDD*, pp 195-203, 2008.
- [5] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. "A survey of kernel and spectral methods for clustering", *Pattern Recognition*, pp 176-190, 2008.
- [6] H. Frigui and C. Hwang, "Semi-supervised Clustering and Aggregation of Relational Data", *ISCC*, 2008.
- [7] H. Frigui and C. Hwang, "Fuzzy Clustering and Aggregation of Relational Data With Instance-Level Constraints", *IEEE T. Fuzzy Systems*, vol. 16, 2008.
- [8] A. Globerson, S. Roweis. "Metric learning by collapsing classes", *NIPS*, pp. 451-458, 2006.
- [9] S.C.H. Hoi, W. L. S. Chang . "Semi Supervised Distance Metric Learning for Collaborative Image Retrieval", *CVPR*, 2008.
- [10] P. Jain, B. Kulis, and K. Grauman. "Fast image search for learned metrics", *CVPR*, 2008.

- [11] W. Pedrycz and A. Amato and V. D. Lecce and V. Piuri, "Fuzzy Clustering With Partial Supervision in Organization and Classification of Digital Images", *IEEE TFS*, vol. 16, 2008.
- [12] G. Raju , T. Binu, T. Sonam, T. Kumar. "Fuzzy Clustering Methods in Data Mining: A Comparative Case Analysis," *icacte*, pp.489-493, 2008
- [13] M. Slaney, K. Q. Weinberger, and W. White. "Learning a metric for music similarity", *ISMIR*, 2008.
- [14] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. "Information-theoretic metric learning". *ICML*, pp. 209-216, 2007.
- [15] H. Frigui, C. Hwang, and F. Chung-Hoon Rhee, "Clustering and aggregation of relational data with applications to image database categorization". *Pattern Recognition*, 2007.
- [16] B. Hammer, A. Hasenfuss," Relational Neural Gas", *KI* , 2007.
- [17] V. Loia and W. Pedrycz and S. Senatore, "Semantic Web Content Analysis: A Study in Proximity-Based Collaborative Clustering", *IEEE TFS*, vol. 15, 2007
- [18] Luxburg, "U. A tutorial on spectral clustering", *Statistics and Computing* , 2007.
- [19] L. Torresani, K. C. Lee. "Large margin component analysis", *NIPS*, pp. 505-512,2007.
- [20] C. Borgelt and R. kruse, "Finding the number of fuzzy clusters by resampling", *IEEE Conf. on Fuzzy Systems*, 2006.
- [21] K. Weinberger, J. Blitzer, L. Saul. "Distance metric learning for large margin nearest neighbor classification", *NIPS*, pp. 1473-1480, 2006.
- [22] R. Yan, J. Zhang, J. Yang, & A. Hauptmann. "A discriminative learning framework with pairwise constraints for video object classification", *IEEE TPAMI*, pp. 578-593, 2006.

- [23] A. Globerson and S. T. Roweis. "Metric learning by collapsing classes", *NIPS*, 2005.
- [24] N. Grira, M. Crucianu, and N. Boujemaa. "Semi-supervised fuzzy clustering with pairwise-constrained competitive agglomeration", *IEEE Conf on Fuzzy Systems*, pp. 867-872, 2005.
- [25] R.J. Hathaway, J.M. Huband, and J.C.Bezdek, "A Kernelized Non-Euclidean Relational Fuzzy c- Means Algorithm", *FUZZ-IEEE*, 2005.
- [26] Yih-Jen Horng and Shyi-Ming Chen and Yu-Chuan Chang and Chia-Hoang Lee, "A new method for fuzzy information retrieval based on fuzzy hierarchical clustering and fuzzy inference techniques", *IEEE T. Fuzzy Systems*, vol 13, 2005.
- [27] Kulis, B., Basu, S., Dhillon, I., and Mooney, R. 2005. "Semi-supervised graph clustering: a kernel approach", *ICML*, 2005.
- [28] J. Philibert, "One and a Half Century of Diffusion: Fick, Einstein, before and beyond Diffusion Fundamentals", 2005.
- [29] K. Q. Weinberger, J. Blitzer, and L. K. Saul. "Distance metric learning for large margin nearest neighbor classification", *NIPS*, 2005.
- [30] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. "Clustering with Bregman divergences", *SIAM*, pp. 234-245, 2004.
- [31] S. Basu, M. Bilenko, R. Mooney. "A probabilistic framework for semi-supervised clustering", *KDD* , 2004
- [32] T. De Bie, J. Suykens, and B. De Moor. "Learning from general label constraints", *IAPR*, pp. 671-679, 2004.
- [33] I. Dhillon and Y. Guan and B. Kulis, "Kernel k-means: spectral clustering and normalized cuts", *ACM SIGKDD*, 2004.
- [34] I. Dhillon, Y. Guan, and B. Kulis. "Kernel k-means, spectral clustering and normalized cuts", *KDD*, 2004.

- [35] R. Inokuchi and S. Miyamoto, "LVQ clustering and SOM using a kernel function", *IEEE Conf. on Fuzzy Systems*, 2004.
- [36] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. "Neighbourhood component analysis", *NIPS*, 2004.
- [37] F. Klawonn, "Fuzzy Clustering: Insights and a New Approach", *Mathware & Soft Computing* 11, 2004.
- [38] A. K. Qinand and P. N. Suganthan, "Kernel neural gas algorithms with application to cluster analysis", *ICPR*, 2004.
- [39] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng. "Online and batch learning of pseudo-metrics", *ICML*, 2004.
- [40] P. Xiaoping Liu and M. Q.-H. Meng, "Online data-driven fuzzy clustering with applications to real-time robotic tracking", *IEEE T. Fuzzy Systems*, pp. 516-523, 2004.
- [41] S. Yu, J. Shi, "Segmentation given partial grouping constraints", *PAMI*, pp. 173-183, 2004.
- [42] L. Zelnik-Manor, P. Perona, "Self-Tuning spectral clustering", *NIPS*, 2004.
- [43] S. Basu, A. Banerjee, and R. J. Mooney. "Active semi-supervision for pairwise constrained clustering", *ICML*, 2003.
- [44] D. Kamvar, et. al., "Spectral Learning", *IJCAI*, 2003.
- [45] Z.-D. Wu, W.-X. Xie, and J.-P. Yu, "Fuzzy c-means clustering algorithm based on kernel method", *ICCIMA*, pp. 49-54, 2003.
- [46] Z. H. Zhang, J. T. Kwok, D. Y. Yeung. "Parametric distance metric learning with label information", *IJCAI*, pp. 1450-1452, 2003.
- [47] E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," *NIPS*, 2003.

- [48] S. Basu, A. Banerjee, and R. J. Mooney. "Semisupervised clustering by seeding", *ICML*, 2002.
- [49] C. Brew and S. Schulte im Walde. "Spectral clustering for german verbs", *IEMNLP*, 2002.
- [50] R. N. Dave and S. Sen. "Robust fuzzy clustering of relational data", *IEEE TFS*, vol. 10, pp. 713-727, 2002.
- [51] I. S. Dhillon, Y. Guan, and J. Kogan. "Iterative clustering of high dimensional text data augmented by local search", *ICDM*, 2002.
- [52] M. Girolami. "Mercer kernel based clustering in feature space", *Neural Networks*, pp. 780-784, 2002.
- [53] B. S. Manjunath, P. Salembier, and T. Sikora, "Introduction to MPEG 7: Multimedia content description language", *John Wiley*, 2002.
- [54] M. J. Er, S. Wu, J. Lu and H. L. Toh, "Face Recognition With Radial Basis Function (RBF) Neural Networks", *Neural Networks*, 2002.
- [55] A. Y. Ng, M. I. Jordan, and Y. Weiss. "On spectral clustering: Analysis and an algorithm", *NIPS*, 2002.
- [56] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. J. Russell. "Distance metric learning with application to clustering with side-information", *NIPS*, pp. 505-512, 2002.
- [57] D.-Q. Zhang and S.-C. Chen, "Fuzzy clustering using kernel method", *ICCA*, 2002.
- [58] M. Belkin and P. Niyogi, "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering", *NIPS*, 2001.
- [59] N. Cristianini, J. S. Taylor, A. Elisseeff, and J. S. Kandola. "On kernel-target alignment", *NIPS*, pp. 367-373, 2001.
- [60] E. Levine and E. Domany, "Unsupervised estimation of cluster validity using resampling", *Neural Computation*, pp. 2573-2593, 2001.

- [61] M. Meila and J. Shi. "A random walks view of spectral segmentation", *IWAIS*, 2001.
- [62] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. "Constrained k-means clustering with background knowledge", *ICML*, pp. 577-584, 2001.
- [63] J. Ze Wang, J. Li, and G. Wiederhold. "SIMPLiCity: Semantics-sensitive integrated matching for picture Libraries", *PAMI*, pp. 947-963, 2001.
- [64] H. Frigui, O. Nasraoui. "Simultaneous clustering and attribute discrimination Fuzzy Systems", *fuzzy systems*, pp. 158 - 163, 2000.
- [65] M. Meila, J. Shi, "Learning segmentation by random walks", *NIPS*, pp. 873-879, 2000.
- [66] O. Nasraoui, H. Frigui, R. Krishnapuram, and A. Joshi. "Extracting web user profiles using relational competitive fuzzy clustering", *International journal on artificial Intelligence Tools*, pp. 509-526, 2000.
- [67] S. Roweis and L. Saul. "Nonlinear dimensionality reduction by locally linear embedding", *Science*, pp. 2323-2326, 2000.
- [68] J. Shi, J. Malik, "Normalized cuts and image segmentation", *PAMI*, pp. 888-905, 2000.
- [69] K. Wagstaff and C. Cardie. "Clustering with instance-level constraints", *ICML*, pp. 1103-1110, 2000.
- [70] A. Demiriz, K. Bennett, and M. Embrechts. "Semi-supervised clustering using genetic algorithms", *IESTANN*, pp. 809-814, 1999.
- [71] O. Nasraoui, R. Krishnapuram and A. Joshi. "Relational clustering based on a new robust estimator with application to web mining", *IWWW*, 1999.
- [72] Yossi Rubner. "Perceptual metrics for image database navigation", *Stanford University, Stanford*, 1999.

- [73] P. Bradley and U. Fayyad. "Refining initial points for k-means clustering", *ICML*, pp. 91-99, 1998.
- [74] P. Perona and W. T. Freeman, "A Factorization Approach to Grouping", *ECCV*, 1998.
- [75] B. Scholkopf, A.J. Smola, and K. R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem", *Neural Computation*, 1998.
- [76] F. Alimoglu, E. Alpaydin, "Methods of Combining Multiple Classifiers Based on Different Representations for Pen-based Handwriting Recognition," *TAINN*, 1996.
- [77] M. Ester, H. P. Kriegel, J. Sander, X. Xu. "A density-based algorithm for discovering clusters in large spatial databases with noise", *KDD*, 1996.
- [78] V.N. Vapnik. "The Nature of Statistical Learning Theory", *Springer*, NY, USA, 1995.
- [79] T. Cox and M. Cox. "Multidimensional Scaling", *Chapman & Hall*, London, 1994.
- [80] R. J. Hathaway and J. C. Bezdek, "NerF c-means: Non-Euclidean relational fuzzy clustering", *Pattern Recognition*, pp.429-437, 1994.
- [81] L. Hagen, L., A. B. Kahng. "New spectral methods for ratio cut partitioning and clustering", *CADICS*, pp. 1074-1085, 1992.
- [82] S. C. Deerwester, S. T. Dumais, T. K. Landauer, George W. Furnas, and Richard A. Harshman. "Indexing by latent semantic analysis", *JASIS*, pp. 391-407, 1990.
- [83] K. Fukunaga. "Introduction to Statistical Pattern Recognition", *Elsevier*, 1990.
- [84] Gath, and A. B. Geva. "Unsupervised optimal fuzzy clustering", *PAMI*, pp.773-781, 1989.
- [85] R. J Hathaway, J.W. davenport, and J,C Bezdek, "Relational duals of the c-means algorithms", *Pattern Recognition*,1989.
- [86] J. bezdek, "Pattern Recognition with fuzzy objective function algorithm", *Plenum Press*, New york, 1981.

- [87] J. Bezdek, C. Coray, R. Gunderson, and J. Watson, "Detection and characterization of cluster substructure", *SIAM*, pp. 339-357, 1981.
- [88] E. Gustafson and W. Kessel, "Fuzzy clustering with a fuzzy covariance matrix", *IEEE CDC*, 1979.
- [89] P.H. Sneath and R.R. Sokal "Numerical taxonomy", *Freeman, San Francisco*, 1973.
- [90] E. Ruspini. "Numerical methods for fuzzy clustering", *Information science*, pp 319-350, 1970.
- [91] J. B. MacQueen. "Some Methods for classification and Analysis of Multivariate Observations", *5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, pp. 281-297, 1967
- [92] N. Aronszajn. "Theory of reproducing kernels", *AMS*, pp. 337-404, 1950.

CURRICULUM VITAE

NAME: Ouiem Bchir

ADDRESS: Department of Computer Engineering and Computer Science
University of Louisville
Louisville, KY 40292

DOB: Monastir, Tunisia - July 14, 1978

EDUCATION: M.S., Automatic and signal processing
Passed the examinations of elaborated studies, June 2005
National School of Engineering of Tunis,
Tunis, Tunisia

B.S., Electrical Engineering
with Highest Honors and Presidential award, June 2002
National School of Engineering of Monastir,
Monastir, Tunisia

Associate Degree in Mathematics and Physics, June 1999
Preparatoty Engineering Studies in Mathematics and Physics,
Tunis, Tunisia

HONORS AND

AWARDS: • 08/2006 - Graduate Research Assistant ship,
University of Louisville, US

- 08/2002 - Presidential award,
Tunis, Tunisia
- 07/1997 - Outstanding Student scholarship
for preparatory engineering studies,
Tunis, Tunisia

JOURNAL

- PUBLICATIONS:
1. A. Zare, **O. Bchir**, H. Frigui, and P. Gader "Piece-wise Convex Multiple Model Endmember Detection",
IEEE TGRS, *under review*
 2. **O. Bchir**, **H. Frigui** "Fuzzy clustering with learnable cluster dependent kernels"
IEEE TFS, *under review*

CONFERENCE

- PUBLICATIONS:
1. **O. Bchir**, **H. Frigui** "Fuzzy clustering with learnable cluster dependent kernels"
FUZZ-IEEE 2011, *under review*
 2. **O. Bchir**, H. Frigui "Fuzzy relational kernel clustering with local scaling parameter learning",
IEEE MLSP 2010.
 3. **O. Bchir**, H. Frigui, A. Zare, and P. Gader " Multiple Model Endmember detection Based on Spectral and Spatial Information",
IEEE WHISPERS 2010
 4. A. Zare, **O. Bchir**, H. Frigui, and P. Gader "Spatially Smooth Piece-Wise Convex Endmember Detection",
IEEE WHISPERS 2010
 5. A. Zare, **O. Bchir**, H. Frigui, and P. Gader "A comparison of deterministic and probalistic approaches to endmember detection", IEEE WHISPERS 2010