2011 IEEE International Conference on Fuzzy Systems
June 27-30, 2011, Taipei, Taiwan

Fuzzy Clustering With Learnable Cluster Dependent Kernels

Ouiem Bchir
CECS department
Multimedia Research Laboratory
University of Louisville
Louisville, KY, 14208
Email: ouiem.bchir@gmail.com

Hichem Frigui
CECS department
Multimedia Research Laboratory
University of Louisville
Louisville, KY, 14208
Email: h.frigui@louisville.edu

***Abstract*—We propose a new relational clustering approach, called Fuzzy clustering with Learnable Cluster dependent Kernels (FLeCK), that learns multiple kernels while seeking compact clusters. A Gaussian kernel is learned with respect to each cluster. It reflects the relative density, size, and position of the cluster with respect to the other clusters. These kernels are learned by optimizing both the intra-cluster and the inter-cluster similarities. Moreover, FLeCK is formulated to work on relational data. This makes it applicable to data where objects cannot be represented by vectors or when clusters of similar objects cannot be represented efficiently by a single prototype. The experiments show that FLeCK outperforms several other algorithms. In particular, we show that when data include clusters with various inter and intra cluster distances, learning cluster dependent kernel is crucial in obtaining a good partition.**

***Index Terms*—Fuzzy clustering, kernel learning, Gaussian kernel.**

I. Introduction

Clustering consists of partitioning a data set 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 [1].

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 (dis)similarity 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 [2], [3], [4] allow to adapt a specific similarity measure in order to make the problem easier. Gaussian kernel is the most common similarity 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 apriori 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 similarity measures by finding intrinsic properties, such as local dimensionality, and local parametrization.

In this work, we introduce a new fuzzy relational clustering technique. The proposed algorithm, called Fuzzy clustering with Learnable Cluster dependent Kernels (FLeCK), learns the underlying cluster dependent dissimilarity measure while finding compact clusters. The learned measure is a local dissimilarity based on the Gaussian kernel function defined with respect to each cluster. In particular, we propose to learn a scaling parameter σ_i with respect to each cluster i. This scaling parameter is designed to distinguish and separate the objects of cluster i from the rest of the data. We achieve this by considering one cluster at a time, and learning a scaling parameter that maximizes the intra-cluster similarity and minimizes the inter-cluster similarity.

The organization of the rest of this paper is as follows. In section 2, we highlight related clustering algorithms. In section 3, we present our algorithm. In section 4, we describe the experiments conducted to validate the proposed algorithm. Finally, section 5 contains the conclusions and potential future work.

II. Related works

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 overlap-

978-1-4244-7317-5/11/$26.00 ©2011 IEEE

ping boundaries. Fuzzy clustering methods are good at dealing with these situations [5]. 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. One of the most widely used fuzzy clustering algorithms is the Fuzzy C-Means (FCM) Algorithm [6] and its relational dual, Relational Fuzzy C-means algorithm (RFCM) [7].

Although FCM based approaches are quite efficient, they are not suitable for all types of data, and cannot handle clusters of different sizes and densities effectively [8]. Another major drawback of FCM based approaches is that they cannot separate clusters that are non-linearly separable in the input space. Two recent approaches have emerged for tackling such a problem. They allow mapping of 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 [2], [3], [4], [9]. These are mainly approaches based on kernel and spectral clustering.

Kernel methods allow the incorporation of a priori knowledge. This knowledge is implicit and is integrated in the learning process through the use of a kernel function. However, despite the existence of a large number of kernel clustering algorithm (e.g., kernel K-means [2], [10], kernel FCM [11], [12], [13], kernel SOM [3], [14] and kernel Neural Gas [4]), the choice of a good kernel function and the adaptation of its parameters to the data remains a challenging task. For instance, in the Kernelized Non-Euclidean Relational Fuzzy c-Means Algorithm (kNERF) [15], a relational Gaussian kernel is used with one global scaling parameter for all the data. The selection of the scaling parameter is discussed 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.

Spectral clustering algorithm methods solve a similar problem as kernel methods, but use a different approach. Its theory arises from the graph cut concept [9]. As for kNERF, spectral clustering use the Gaussian similarity function in order to model the local neighborhood relationships between data points. Thus, it has the same challenging task of determining and adapting the scaling parameter to the data set.

The Gaussian similarity function, used in kernel and spectral clustering, is defined as

$$\mathbf{S}_{jk} = exp\left(-\frac{dist\left(\mathbf{x}_j, \mathbf{x}_k\right)}{\sigma^2}\right) \tag{1}$$

where $dist$ measures the dissimilarity between patterns and σ controls the rate of decay of S. The scaling parameter σ is commonly set manually.

Ng et al. [16] suggested selecting σ that provides the least distorted clusters by running the algorithm for different values of σ. However, the range of values to be tested still has to be set manually. Moreover, a global σ for mapping all the data may not provide a good clustering when the input data includes clusters with different local statistics.

In [17], 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 as

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

The selection of the local scale σ_j is defined in [17] 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 data set. Moreover, if $\mathbf{x}_j$ is a noise point, adapting σ_j to this point will distort this fact.

To the best of our knowledge, no automatic way to select the optimal scaling parameter and adapt it to each cluster has been proposed even though the performance of Gaussian kernel based approaches is highly dependent on this parameter. In this paper, we propose a clustering algorithm, called Fuzzy clustering with Learnable Cluster dependent Kernels (FLeCK), that learns the underlying cluster dependent scaling parameters while finding compact clusters.

III. Fuzzy clustering with learnable cluster dependent kernels

Considering the Gaussian similarity function as defined in (1), the optimal choice of the scaling parameter is the one that allows to segregate and distinguish between the clusters. Thus, this scaling parameter has to ensure a high intra-cluster similarity and low inter-cluster similarity. A very large value of σ would minimize the intra-cluster similarity. However, this choice would also minimize the inter-cluster similarity and merge all the objects into a single cluster. One possible approach to avoid this trivial solution is to seek scaling parameters that minimize the intra-cluster similarities and maximize the intra-cluster similarities simultaneously.

Let $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ be a set of N data points and let $\mathbf{R} = [\mathbf{r_{jk}}]$ be a relational matrix where r_{jk} represent the degrees to which pairs of objects are similar. The FLeCK algorithm minimizes the intra-cluster similarity

$$\begin{aligned} J^{intra} &= \sum_{i=1}^{C}\sum_{j=1}^{N}\sum_{k=1}^{N}\beta_{jk}^{i}\left(1 - exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right)\right) \\ &- \sum_{i=1}^{C}\frac{K}{\sigma_i} \end{aligned} \tag{3}$$

and maximizes the inter-cluster similarity

$$J^{inter} = \sum_{i=1}^{C}\sum_{j=1}^{N}\sum_{k=1}^{N}\alpha_{jk}^{i} exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) - \sum_{i=1}^{C}\frac{K}{\sigma_i} \tag{4}$$

In (3) and (4), C is the number of clusters. The constant K provides a way of specifying the relative importance of the regularization term compared to the sum of intra-cluster and inter-cluster distances. When K is zero, minimizing (3) with respect to σ_i gives the trivial solution of a very large σ_i that merges all points into a single cluster. Instead of treating K as a constant, we propose to determine the optimal K

by simultaneously minimizing the intra-cluster similarity and maximizing the inter-cluster similarity.

The term β^i_{jk} in (3) is the likelihood that two points $\mathbf{x}_j$ and $\mathbf{x}_k$ belong to the same cluster i and is defined as

$$\beta^i_{jk} = u^m_{ij} u^m_{ik} \tag{5}$$

where u_{ij} is the fuzzy membership of $\mathbf{x}_j$ in cluster i and satisfies the constraints

$$0 \leq u_{ij} \leq 1 \;\; and \;\; \sum_{i=1}^{C} u_{ij} = 1, \; for \; j \in \{1, \ldots N\}. \tag{6}$$

Similarly, the term α^i_{jk} in (4) refers to the likelihood that two points $\mathbf{x}_j$ and $\mathbf{x}_k$ do not belong to the same cluster i. It can also be defined using the fuzzy memberships of the two points using

$$\alpha^i_{jk} = u^m_{ij}\left(1 - u^m_{ik}\right) + u^m_{ik}\left(1 - u^m_{ij}\right). \tag{7}$$

In (5) and (7), $m \in (1, \infty)$ is a constant that determines the level of cluster fuzziness. A larger m results in fuzzier clusters. Typically, m is set to 2 [18].

The first term in (3) seeks clusters that have compact local relational distances

$$\mathbf{D}^i_{jk} = 1 - exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) \tag{8}$$

with respect to each cluster i. The scaling parameter, σ_i, controls the rate of decay of $\mathbf{D}^i_{jk}$. We should note here that our approach learns a cluster dependent σ_i in order to deal with the large variations between the distributions and the geometric characteristics of each cluster. The second term in (3) and (4) is a regularization term to avoid the trivial solution where all σ_i are infinitely large.

The proposed FLeCK algorithm attempts to optimize (3) and (4) 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 (3) and (4) 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 (3) and (4) to the following C simpler set of functions

$$J^{intra}_i = \sum_{j=1}^{N}\sum_{k=1}^{N} \beta^i_{jk}\left(1 - exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right)\right) - \frac{K}{\sigma_i}, \tag{9}$$

and

$$J^{inter}_i = \sum_{j=1}^{N}\sum_{k=1}^{N} \alpha^i_{jk} exp\left(-\frac{\mathbf{r}_{jk}}{\sigma_i}\right) - \frac{K}{\sigma_i} \tag{10}$$

for $i = 1, \ldots, C$. To obtain an update equation for the scaling parameters, we first set the derivative of J^{intra}_i with respect to σ_i to zero and solve for K. Then, we substitute the expression of K back in (10). 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^{inter}_i to zero, we obtain

$$\sigma^{(t)}_i = \frac{Q^i_1}{Q^i_2} \tag{11}$$

where

$$Q^i_1 = \sum_{j=1}^{N}\sum_{k=1}^{N} \beta^i_{jk}\mathbf{r}^2_{jk} exp\left(-\frac{\mathbf{r}_{jk}}{\sigma^{(t-1)}_i}\right) \tag{12}$$

and

$$\begin{aligned} Q^i_2 &= \sum_{j=1}^{N}\sum_{k=1}^{N} \alpha^i_{jk}\mathbf{r}_{jk} exp\left(-\frac{\mathbf{r}_{jk}}{\sigma^{(t-1)}_i}\right) \\ &+ \sum_{j=1}^{N}\sum_{k=1}^{N} \beta^i_{jk}\mathbf{r}_{jk} exp\left(-\frac{\mathbf{r}_{jk}}{\sigma^{(t-1)}_i}\right) \end{aligned} \tag{13}$$

The expression of the scaling parameter defined by equation (11) depends on both the inter-cluster and the intra-cluster distances.

Optimization of (3) and (4) 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 (3) with respect to u_{ij}. We achieve this by using the relational dual of the fuzzy C-means algorithm formulated by Hathaway et al. [7]. As it has been proved in [7], the Euclidean distance, $d^2_{ik} = \|\mathbf{x}_k - \mathbf{c}_i\|^2$, from feature $\mathbf{x}_k$ to the i^{th} cluster, $\mathbf{c}_i$, can be written in terms of the relational matrix $\mathbf{D}^i$ as

$$d^2_{ik} = \left(\mathbf{D}^i\mathbf{v}_i\right)_k - \frac{\mathbf{v}^t_i\mathbf{D}^i\mathbf{v}_i}{2}. \tag{14}$$

In (14), $\mathbf{v}_i$ is the membership vector of all N samples in cluster i defined by

$$\mathbf{v}_i = \frac{(u^m_{i1}, \ldots, u^m_{iN})^t}{\sum_{j=1}^{N} u^m_{ij}}. \tag{15}$$

Using Lagrange multiplier technique, it can be shown that optimization of (3) with respect to u_{ij}, subject to (6), yields the following membership update equation

$$u_{ij} = \frac{1}{\sum_{t=1}^{C}\left(d^2_{ij}/d^2_{tj}\right)^{\frac{1}{m-1}}}. \tag{16}$$

The resulting FLeCK algorithm is summarized below.

Algorithm 1 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 (8) ;
  Compute the membership vectors $\mathbf{v}_i$ using (15);
  Compute the distances using (14);
  Update the fuzzy membership using (16);
  Update the scaling parameter σ_i using (11);
UNTIL (fuzzy membership do not change)

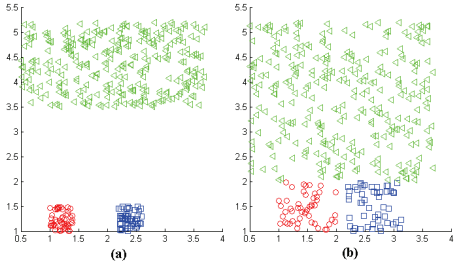

Fig. 1. 2D datasets. Each cluster is shown by a different color.

IV. Experimental Results

The performance of the proposed algorithm is illustrated and compared to kNERF [15] and Self tuning spectral [17]. We also illustrate the advantage of learning cluster dependent σ_i's compared to one global σ.

A. Synthetic data sets

To illustrate the ability of FLeCK to learn appropriate local scaling parameters and cluster the data simultaneously, we use it to categorize synthetic data sets. We use 2 data sets that include categories of different sizes and densities. 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. Results on real and high-dimensional data sets will be reported in the next section. Figure 1 displays the 2 synthetic data sets where each cluster is displayed with a different color.

For kNERF, and Self tuning spectral, we repeat the clustering with different parameters and report the best results. In particular, 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 0.001 and 10 with a step of 0.01.

Figure 2 displays the clustering results of the three algorithms on data set 1. As shown in figure 2 (b), self tuning spectral was able to categorize it correctly after tuning the scaling parameter. Although, similar to spectral clustering kNERF is based on a non linear mapping of the input data, it was not able to categorize this data correctly (Fig. 2 (a)). 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. FLeCK and self tuning spectral provide local exponential mapping of the data and were able to partition this data correctly. The main difference between these approaches, as explained in section III, is that FLeCK learns the local scaling parameters and performs clustering simultaneously, while the self tuning spectral tunes the neighborhood within a user specified range to find local scaling parameters.

The scaling parameters learned by FLeCK reflect the relative position of each cluster with respect to the other clusters. For this data, cluster 1 and cluster 2 are close to each other, and cluster 3 is quite distant from both of them. Consequently, FLeCK learns a large scaling parameter $(\sigma_3 = 106.64)$ for

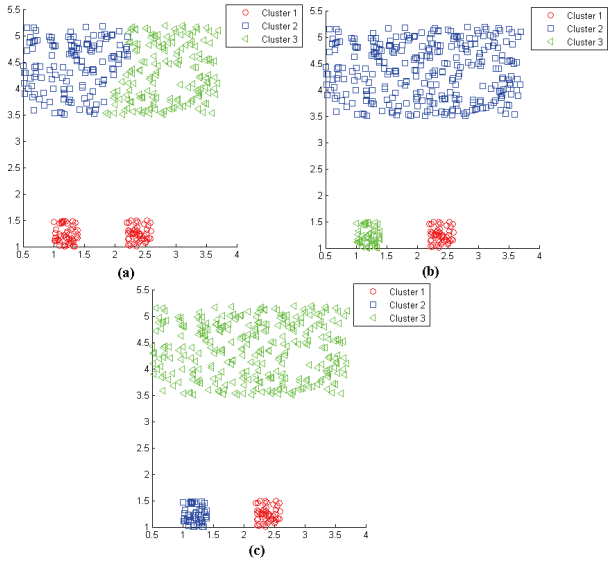

Fig. 2. Results of clustering the data in fig. 1 (a) using (a) kNERF, (b) Spectral, and (c) FLeCK

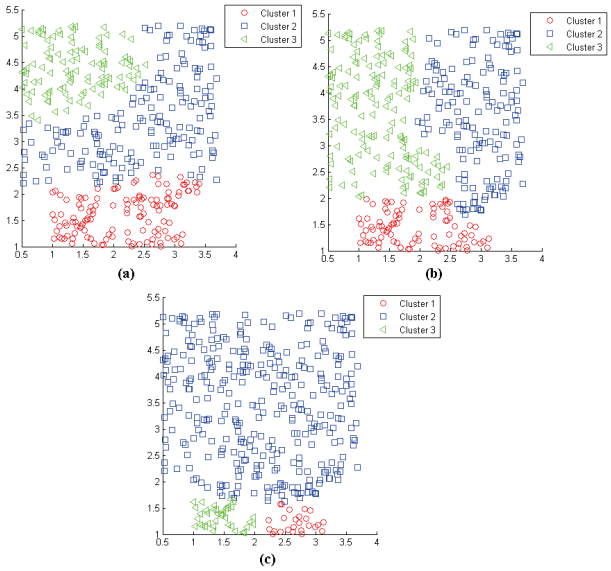

Fig. 3. Results of clustering the data in fig. 1 (b) using (a) kNERF, (b) Spectral, and (c) FLeCK

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 second data set, shown in figure 1 (b), contains two clusters that have the same shape, density, and size, and a third cluster that is larger. The clustering results of the three algorithms on this data are shown in figure 3. As it can be seen, only FLeCK was able to partition this data correctly. This was possible due to the learned scaling parameters for each cluster.

TABLE I
NUMBER OF SAMPLES FORM EACH CATEGORY

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

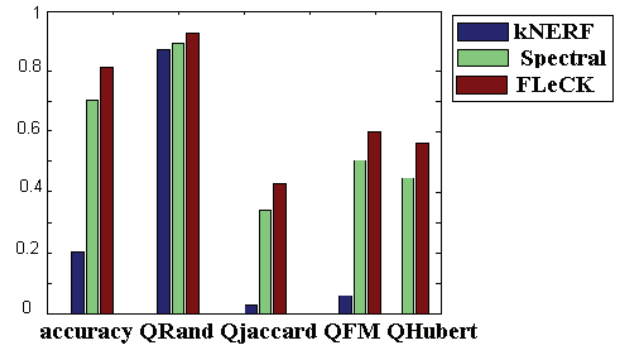

Fig. 4. Performance measures obtained on Handwritten Digits Data set using kNERF, Spectral and FLeCK clustering approaches.

For instance, cluster 2 has the smallest scaling parameter ($\sigma_2 = 0.0007$). This reflects the fact that points within this cluster are dispersed and some of them are closer to cluster 1 and cluster 3 than other points within this cluster. This small σ_2 ensures that cluster 2 does not include points from the other clusters. FLeCK learns a relatively bigger scaling parameters for cluster 1 and cluster 3 ($\sigma_1 = 12.21$ *and* $\sigma_3 = 11.3$). In fact, these two scaling parameters can be relatively larger (to maximize the intra-cluster similarity) without including points from the other cluster.

Besides giving a better clustering results than kNERF and self tuning spectral, FLeCK has two more main advantages over these two algorithms. First, it does not require the specification of the scaling parameter or the neighborhood parameter. Instead, it learns a cluster dependent scaling parameter in a completely unsupervised manner. Second, the learned scaling parameters provide a better description of the data and can be explored in subsequent learning steps.

B. Real data set

To illustrate the ability of FLeCK to learn local kernels and to cluster relational data derived from real and high dimensional data, we use it to categorize the Handwritten Digits database. We use a subset of 1166 handwritten digits from the Pen-Based Recognition of Handwritten Digits collection [19] available at the UCI Machine Learning data set. 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 I displays the number of samples form each category.

We compare our clustering results to those obtained using the kNERF [15], and self tuning spectral clustering [17]. To quantify the performance of the different clustering algorithms and compare them, we assume that the ground truth is known,

TABLE II
KNERF CLUSTERING RESULTS ON THE HANDWRITTEN DIGIT DATA

		True class									
		'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'
Clusters	1	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	1	0	0	0	0	0	0
	3	0	0	0	103	0	3	0	0	0	2
	4 - 13	0	0	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	1
	15	0	0	0	0	0	0	0	0	0	0
	16	128	131	107	18	108	116	114	117	109	108

TABLE III
SELF TUNING SPECTRAL CLUSTERING RESULTS ON THE HANDWRITTEN DIGIT DATA

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

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]. All these measures provide larger values when the two partitions are more similar.

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

TABLE IV
FLECK CLUSTERING RESULTS ON THE HANDWRITTEN DIGIT DATA

		True class									
		'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'
Clusters	1	0	0	0	0	0	0	0	0	41	0
	2	0	55	59	0	0	0	0	1	1	0
	3	45	0	0	0	0	0	0	0	1	0
	4	9	0	0	0	0	0	0	0	0	0
	5	45	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	0	0	0	59	1
	7	0	0	0	0	0	0	105	1	0	0
	8	0	0	0	1	0	2	0	0	0	76
	9	0	1	0	116	0	34	0	12	0	14
	10	0	19	0	0	5	26	8	0	1	1
	11	0	0	0	0	68	0	0	0	0	3
	12	0	54	0	5	0	0	0	18	0	8
	13	0	0	0	0	0	57	0	0	5	0
	14	0	2	8	0	0	0	0	85	0	0
	15	29	0	0	0	0	0	1	0	1	0
	16	0	0	0	0	35	0	0	0	0	8

TABLE V
THE SCALING PARAMETERS LEARNED BY FLECK, NO. OF SAMPLES PER CLUSTER, AND THE INTRA AND INTER CLUSTER DISTANCES

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

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 similarity.

Figure 4 compares the performance of the three considered algorithms. As it can be seen, FLeCK outperforms the other methods with respect to all five measures. Although, kNERF and self tuning spectral use a similar Gaussian kernel, they were not able to cluster this data properly. This is due to their sensitivity to the choice of the scaling parameter.

As it can be seen from table II, kNERF has succeeded to segregate only the digit "3" (cluster 3). In fact, almost all the other elements of the data set are gathered in a big cluster (cluster 16) and the remaining clusters are empty. This is due to the fact that one global σ for mapping all the data can not provide the correct partition when the input data includes clusters with different local statistics.

The self tuning spectral did relatively better than kNERF because it uses a local scaling parameter that is defined as the distance between $\mathbf{x}_j$ and its P^{th} neighbor. However, as it can be seen from table III, it still creates 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 σ to each sample can distort the structure of the data.

FLeCK on the other hand, was able to partition the data better than kNERF and Self tuning spectral. Referring to the FLeCK clustering results shown in table IV, 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 5 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

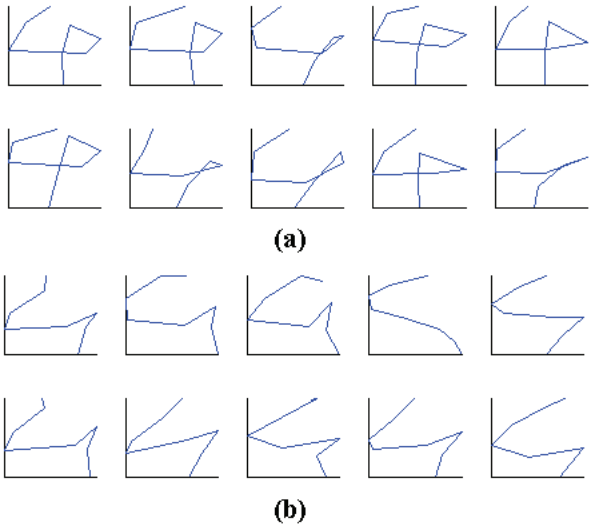

Fig. 5. The top 10 representatives of (a) cluster 11 and (b) cluster 16.

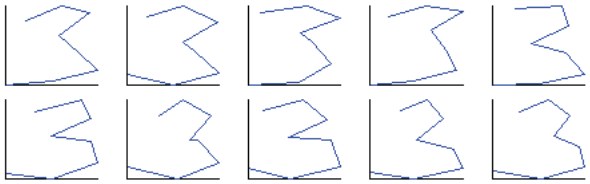

Fig. 6. The top 10 representatives of cluster 9.

with more consistent writing style. Figure 6 displays the top 10 representatives of one such cluster representing digit 3.

FLeCK learned a scaling parameter for each cluster. These parameters are reported in table V. 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 similarity) 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).

V. CONCLUSIONS

We have proposed an approach that performs relational fuzzy clustering and local kernel learning. The learned kernel is based on a Gaussian function with a cluster dependent scaling parameter. The proposed approach minimizes both the inter-cluster dissimilarity and the intra-cluster dissimilarity to learn a scaling parameter with respect to each cluster. This allows FLeCK to adapt the distance to the characteristics of each cluster. FLeCK is formulated to work on pairwise relational

data. This makes it applicable when data is available only in relational form and when similar objects cannot be represented efficiently by a single prototype.

Using synthetic and real data sets, we have shown that our approach outperforms kNERF and self tuning spectral. In particular, we have shown that when data include clusters with various inter and intra cluster distances, learning cluster dependent scaling parameters is crucial in obtaining a good partition.

Currently, FLeCK uses fuzzy memberships that are constrained to sum to one. This type of membership does not discriminate between typical points that are close to multiple clusters and noise points. Consequently, FLeCK is not robust to noise and outliers. One way to overcome this limitation is to replace the fuzzy memberships with possibilistic memberships [21]. We are currently investigating this alternative.

ACKNOWLEDGMENTS

This work was supported in part by U.S. Army Research Office Grants Number W911NF-08-0255 and by a grant from the Kentucky Science and Engineering Foundation as per Grant Agreement No. KSEF-2079-RDE-013 with the Kentucky Science and Technology Corporation.

REFERENCES

[1] F. Klawonn, “Fuzzy Clustering: Insights and a New Approach”, *Mathware & Soft Computing 11*, Vol 11, No 3, 2004.

[2] M. Girolami, “Mercer kernel based clustering in feature space”, *IEEE Transactions on Neural Networks*, Vol. 13, No. 3, 2002.

[3] R. Inokuchi and S. Miyamoto, “LVQ clustering and SOM using a kernel function”, *IEEE Conf. on Fuzzy Systems*, 2004.

[4] A. K. Qinand and P. N. Suganthan, “Kernel neural gas algorithms with application to cluster analysis”, *ICPR*, 2004.

[5] Raju G., Binu Thomas, Sonam Tobgay, Th. Shanta Kumar, “Fuzzy Clustering Methods in Data Mining: A Comparative Case Analysis,” *icacte*, pp.489-493, 2008

[6] J.C Bezdek, “Pattern Recognition with fuzzy objective function algorithm”, *Plenum Press*, New york, 1981.

[7] R. J Hathaway, J.W. davenport, and J,C Bezdek, “Relational duals of the c-means algorithms”, *Pattern Recognition*, vol. 22, p.205 , 1989.

[8] Pang-Ning Tan, Michael Steinbach, Vipin Kumar,” Introduction to data mining”, *Addison Wesley,* 2006.

[9] Luxburg, “U. A tutorial on spectral clustering”, *Statistics and Computing* , 2007.

[10] B. Scholkopf, A.J. Smola, and K. R. Muller, “Nonlinear component analysis as a kernel eigenvalue problem”, *Neural Computation*, vol 10, p. 1299-1319,1998.

[11] T. Graepel and K. Obermayer, “Fuzzy topographic kernel clustering”, *5th GI Workshop Fuzzy Neuro Systems*, 1998.

[12] Z. D. Wu, W. X. Xie, and J. P. Yu, “Fuzzy c-means clustering algorithm based on kernel method”, *CIMA*, 2003.

[13] D.-Q. Zhang and S.-C. Chen, “Fuzzy clustering using kernel method”, *ICCA*, 2002.

[14] D. Macdonald and C. Fyfe, “The kernel self-organising map”, *KBIESAT*, 2000.

[15] R.J.Hathaway, J.M. Huband, and J.C.Bezdek, ”A Kernelized Non-Euclidean Relational Fuzzy c- Means Algorithm”, *FUZZ-IEEE*, 2005.

[16] P. Perona and W. T. Freeman, “A Factorization Approach to Grouping”, *5th European Conference on Computer Vision*, 1998.

[17] L. Zelnik-Manor, P. Perona, “Self-Tuning spectral clustering”, *NIPS*, 2004.

[18] Enrique Frias-Martinez, Sherry Y. Chen, and Xiaohui Liu. “Survey of Data Mining Approaches to User Modeling for Adaptive Hypermedia”, *IEEE Transactions on systems, man and cybernetics,* Vol. 36, No. 6, 2006.

[19] F. Alimoglu, E. Alpaydin, "Methods of Combining Multiple Classifiers Based on Different Representations for Pen-based Handwriting Recognition," *TAINN*, 1996, Istanbul.

[20] C. Borgelt and R. kruse, “Finding the number of fuzzy clusters by resampling”, *IEEE Conf. on Fuzzy Systems*, 2006.

[21] Krishnapuram, R. and Keller, J., "A Possibilistic Approach to Clustering", *IEEE Transactions on Fuzzy Systems*, Vol. 1, No. 3, pp. 222-233, 1993.