LEGClust—A Clustering Algorithm Based on Layered Entropic Subgraphs

Jorge M. Santos, Joaquim Marques de Sá, and Luís A. Alexandre

Abstract—Hierarchical clustering is a stepwise clustering method usually based on proximity measures between objects or sets of objects from a given data set. The most common proximity measures are distance measures. The derived proximity matrices can be used to build graphs, which provide the basic structure for some clustering methods. We present here a new proximity matrix based on an entropic measure and also a clustering algorithm (LEGClust) that builds layers of subgraphs based on this matrix and uses them and a hierarchical agglomerative clustering technique to form the clusters. Our approach capitalizes on both a graph structure and a hierarchical construction. Moreover, by using entropy as a proximity measure, we are able, with no assumption about the cluster shapes, to capture the local structure of the data, forcing the clustering method to reflect this structure. We present several experiments on artificial and real data sets that provide evidence on the superior performance of this new algorithm when compared with competing ones.

Index Terms—Clustering, entropy, graphs.

1 Introduction

Clustering deals with the process of finding possible different groups in a given set, based on similarities or differences among their objects. This simple definition does not convey the richness of such a wide area of research. What are the similarities, and what are the differences? How do the groups differ? How can we find them? These are examples of some basic questions, none with a unique answer. There is a wide variety of techniques to do clustering. Results are not unique, and they always depend on the purpose of the clustering. The same data can be clustered with different acceptable solutions. Hierarchical clustering, for example, gives several solutions depending on the tree level chosen for the final solution.

There are algorithms based on similarity or dissimilarity measures between the objects of a set, like sequential and hierarchical algorithms; others are based on the principle of function approximation, like fuzzy clustering or density-based algorithms, yet others are based on graph theory or competitive learning. In this paper, we combine hierarchical and graph approaches and present a new clustering algorithm based on a new proximity matrix that is built with an entropic measure. With this measure, connections between objects are sensitive to the local structure of the data, achieving clusters that reflect that same structure.

- *J.M. Santos is with the Department of Mathematics, ISEP-Polytechnic, School of Engineering, R. Dr. António Bernardino de Almeida, 431, 4200-072 Porto, Portugal, and the INEB-Biomedical Engineering Institute, Porto, Portugal. E-mail: jms@isep.ipp.pt.*
- *J. Marques de Sá is with the Department of Electrical and Computer Engineering at FEUP-Engineering University, Porto, Portugal, and the INEB-Biomedical Engineering Institute, Porto, Portugal. E-mail: jmsa@fe.up.pt.*
- *L.A. Alexandre is with the Department of Informatics, UBI-Beira Interior University, Covilhã, Portugal, and the Networks and Multimedia Group of IT, Covilhã, Portugal. E-mail: lfbaa@di.ubi.pt.*

Manuscript received 6 Nov. 2006; revised 1 Mar. 2007; accepted 12 Mar. 2007; published online 4 Apr. 2007.
Recommended for acceptance by J. Buhmann
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0788-1106.
Digital Object Identifier no. 10.1109/TPAMI.2007.1142

In Section 2, we introduce the concepts and notation that serve as the basis to present our algorithm. In Section 3, we present the clustering algorithm (designated by LEGClust) components: a new dissimilarity matrix and a new clustering process. The experiments are described in Section 4 and the conclusions in the last section.

2 Basic Concepts

2.1 Proximity Measures

Let X be the data set $X = \{\mathbf{x}_i\}, i = 1, 2, \ldots, N$, where N is the number of objects, and $\mathbf{x}_i$ is an l-dimensional vector representing each object. We define S, an s-clustering of X, as a partition of X into s clusters $C_1, C_2, \ldots, C_s$, obeying the following conditions: $C_i \neq \oslash$, $i = 1, \ldots, s$; $\cup_{i=1}^{s} C_i = X$ and $C_i \cap C_j = \oslash$, $i \neq j$, $i, j = 1, \ldots, s$. Each vector (point), given these conditions, belongs to a single cluster. Our proposed algorithm uses this so-called *hard clustering*. (There are algorithms like those based on fuzzy theory in which a point has degrees of membership for each cluster.) Points belonging to the same cluster have a higher degree of similarity with each other than with any other point of the other clusters. This degree of similarity is usually defined using similarity (or dissimilarity) measures.

The most common dissimilarity measure between two real-valued vectors $\mathbf{x}$ and $\mathbf{y}$ is the weighted l_p metric,

$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{l} w_i |x_i - y_i|^p \right)^{\frac{1}{p}}, \quad (1)$$

where x_i and y_i are the ith coordinates of $\mathbf{x}$ and $\mathbf{y}$, $i = 1, \ldots, l$, and $w_i \geq 0$ is the ith weight coefficient. The unweighted $(\mathbf{w} = 1)$ l_p metric is also known as the Minkowski distance of order p $(p \geq 1)$. Examples of this distance are the well-known euclidean distance, obtained by setting $p = 2$, the Manhattan distance, $p = 1$, and the l_∞ or the Chebyshev distance.

2.2 Overview of Clustering Algorithms

Probably the most used clustering algorithms are the hierarchical agglomerative algorithms. They, by definition, create a hierarchy of clusters from the data set. Hierarchical clustering is widely used in biology, medicine, and also computer science and engineering. (For an overview on clustering techniques and applications, see [1], [2], [3], and [4]). Hierarchical agglomerative algorithms start by assigning each point to a single cluster and then, usually based on dissimilarity measures, proceed to merge small clusters into larger ones in a stepwise manner. The process ends when all the points in the data set are members of a single cluster. The resulting hierarchical tree defines the clustering levels. Examples of hierarchical clustering algorithms are CURE [5] and ROCK [6] developed by the same researchers, AGNES [7], BIRCH [8], [9], and Chameleon [10].

The merging phase of the agglomerative algorithms differs in the sense that depending on the measures used to compute the similarity or dissimilarity between clusters, different merge results can be obtained. The most common methods to perform the merging phase are the Single-Link, Complete-Link, Centroid, and Ward's methods. The Single-Link method usually creates elongated clusters, and the Complete-Link usually results in more compact clusters. The Centroid method acts in a midway basis, yielding clusters somewhere between the two previous methods. Ward's method is considered very effective in producing balanced clusters; however, it has several problems in dealing with outliers and elongated clusters. In [11], one can find a probabilistic interpretation of these classical agglomerative methods.

Another type of algorithms is the one based on graphs and graph theory. Clustering algorithms based on graph theory are usually divisive algorithms, meaning that they start with a single highly connected graph (that corresponds to a single cluster) that is then split using consecutive cuts. A cut in a graph corresponds to the removal of a set of edges that disconnects the graph. A minimum cut (min-cut) is the removal of the smallest number of edges that produces a cut. The result of a cut in the graph causes the splitting of one cluster into, at least, two clusters. An example of a min-cut clustering algorithm can be found in [12]. Clustering algorithms based on graph theory have existed since the early 1970s. They use the high connectivity in similarity graphs to perform clustering [13], [14]. More recent works such as [15], [16], and [17] also perform clustering using highly connected graphs and subsequent partition by edge cutting to obtain subgraphs. Chameleon, mentioned earlier as a hierarchical agglomerative algorithm, also uses a graph-theoretic approach. It starts by constructing a graph, based on k-nearest neighbors; then, it performs the partition of the graph into several clusters (using the hMetis [18] algorithm) such that it minimizes the edge cut. After finding the initial clusters, it repeatedly merges these small clusters using relative cluster interconnectivity and closeness measures.

Graph cutting is also used in spectral clustering, commonly applied in image segmentation and, more recently, in Web and document clustering and bioinformatics. The rationale of spectral clustering is to use the special properties of the eigenvectors of a Laplacian matrix as the basis to perform clustering. Fiedler [19] was one of the first to show the application of eigenvectors to graph partitioning. The Laplacian matrix is based on an affinity matrix built with a similarity measure. The most common similarity measure used in spectral clustering is $A_{ij} = \exp(-d_{ij}^2/2\sigma^2)$, where d_{ij} is the euclidean distance between vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, and σ is a scaling parameter. With matrix A, the Laplacian matrix L is computed as $L = D - A$, where D is the diagonal matrix whose elements are the sums of all row elements of A.

There are several spectral clustering algorithms that differ in the way they use the eigenvectors in order to perform clustering. Some researchers use the eigenvectors of the "normalized" Laplacian matrix [20] (or a similar one) in order to perform the cutting usually using the second smallest eigenvector [21], [22], [23]. Others use the highest eigenvectors as input to another clustering algorithm [24], [25]. One of the advantages of this last approach is that by using more than one eigenvector, enough information may be provided to obtain more than two clusters as opposed to cutting strategies where clustering must be performed recursively to obtain more than two clusters. A comparison of several spectral clustering algorithms can be found in [26].

The practical problems encountered with graph-cutting algorithms are basically related to the belief that the subgraphs produced by cutting are always related to real clusters. This assumption is frequently true with well separated compact clusters; however, in data sets with, for example, elongated clusters, this may not occur. Also, if we use weighted graphs, the choice of the threshold to perform graph partition can produce very different clustering solutions.

Other clustering algorithms use the existence of different density regions of the data to perform clustering. One of the density-based clustering algorithms, apart from the well-known DBScan [27], is the Mean Shift algorithm. Mean Shift was introduced by Fukunaga and Hostetler [28], rediscovered in [29], and also studied in more detail by Comaniciu and Meer [30], [31] with applications to image segmentation. The original algorithm, with a flat kernel, works this way: In each iteration, for each point P, the cluster center is obtained by repeatedly centering the kernel (originally centered in P) by shifting it in the direction of the mean of the set of points inside the same kernel. The process is similar if we use a Gaussian kernel. The mean shift vector is aligned with the local gradient estimate and defines a path leading to a stationary point in the estimated density [31]. This algorithm seeks modes in the sample density estimation and so is considered to be a gradient mapping algorithm [29]. Mean Shift has some very good results in image segmentation and computer vision applications, but like other density-based algorithms, it builds clusters with the assumption that each of them is related to a mode of the density estimation. For problems like the one depicted in Fig. 1a, with clusters of different densities very close to each other, this kind of algorithm usually has difficulties in performing the right partition because it finds only one mode in the density function. (If we use a smaller smoothing parameter it will find several local modes in the low-density region). This behavior is also observable in data sets like the double spiral data set depicted in Fig. 10.

Another example of a clustering algorithm is the path-based pairwise clustering algorithm [32], [33]. This clustering method also groups objects according to their connectivity. It uses a pairwise clustering cost function with a dissimilarity measure that emphasizes connectedness in feature space to deal with cluster compactness. This simple approach gives good results with compact clusters. To deal with structured clusters, a new objective function, conserving the same properties of the pairwise cost function, is used. This new

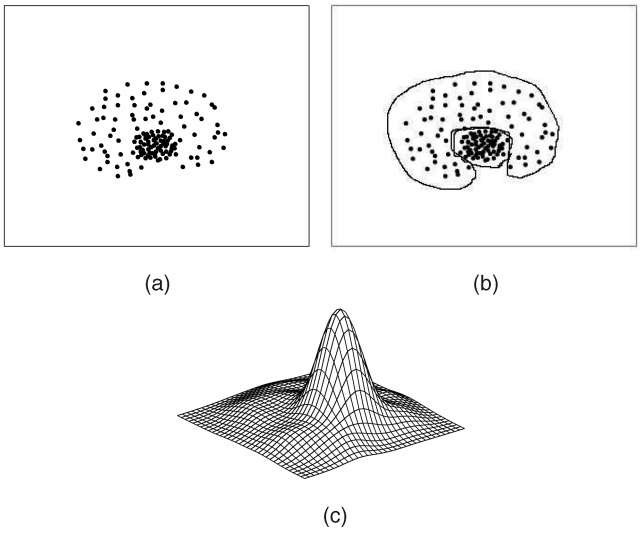

Fig. 1. An example of a data set difficult to cluster using density-based clustering algorithms like Mean Shift. (a) The original data set. (b) The possible clustering solution. (c) Density function.

objective function is based on the *effective dissimilarity* and the length of the minimal connecting path between two objects and is the basis for the path-based clustering. Some of the applications of this clustering algorithm are edge detection and texture image segmentation.

2.3 Renyi's Quadratic Entropy

Since the introduction by Shannon [34] of the concept of entropy, information theory concepts have been applied in learning systems.

Shannon's entropy, $H_S(X) = -\sum_{i=1}^{N} p_i \log p_i$, measures the average amount of information conveyed by the events $X = \mathbf{x}_i$ that occur with probability p_i. Entropy can also be seen as the amount of uncertainty of a random variable. The more uncertain the events of X, the larger the information content, with a maximum for equiprobable events.

The extension of Shannon's entropy to continuous random variables is $H(X) = -\int_C f(\mathbf{x}) \log f(\mathbf{x}) d\mathbf{x}$, where $X \in C$, and $f(\mathbf{x})$ is the probability density function (pdf) of the variable X.

Renyi generalized the concept of entropy [35] and defined the (Renyi's) α-entropy of a discrete distribution as

$$H_{R\alpha}(X) = \frac{1}{1-\alpha} \log\left(\sum_{i=1}^{N} p_i^{\alpha}\right), \tag{2}$$

which becomes Shannon's entropy when $\alpha \to 1$. For continuous distributions and $\alpha = 2$, one obtains the following formula for Renyi's Quadratic Entropy [35]:

$$H_{R2}(X) = -\log\left(\int_C [f(\mathbf{x})]^2 d\mathbf{x}\right). \tag{3}$$

The pdf can be estimated using the Parzen Window method allowing the determination of the entropy in a nonparametric and computationally efficient way. The Parzen window method [36] estimates the pdf $f(\mathbf{x})$ as

$$f(\mathbf{x}) = \frac{1}{Nh^m} \sum_{i=1}^{N} G\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}, \mathbf{I}\right), \tag{4}$$

where N is the number of data vectors, G can be a radially symmetric Gaussian kernel with zero mean and diagonal covariance matrix

$$G(\mathbf{x}; 0, \mathbf{I}) = \frac{1}{(2\pi)^{\frac{m}{2}} |\mathbf{I}|^{\frac{1}{2}}} exp\left(-\frac{1}{2} x^T \mathbf{I}^{-1} x\right),$$

m is the dimension of the vector $\mathbf{x}$ ($\mathbf{x} \in \mathbb{R}^m$), h is the bandwidth parameter (also known as the smoothing parameter or kernel size), and $\mathbf{I}$ is the $m \times m$ identity matrix. Substituting (4) in (3) and applying the integration of Gaussian kernels [37], Renyi's Quadratic Entropy can be estimated as

$$\begin{aligned}\hat{H}_{R2} &= -\log\left[\int_{-\infty}^{+\infty} \left(\frac{1}{Nh^m} \sum_{i=1}^{N} G(\frac{x}{h}; \frac{x_i}{h}, I)\right)^2 d\mathbf{x}\right] \\ &= -\log\left(\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G(x_i - x_j; 0, 2h^2\mathbf{I})\right).\end{aligned}$$

In our algorithm, we use Renyi's Quadratic Entropy because of its simplicity; however, one could use other entropic measures as well. Some examples of the application of entropy and concepts of information theory in clustering are the minimum entropic clustering [38], entropic spanning graphs clustering [39], and entropic subspace clustering [40]. In some works, the entropic concepts are usually related to measures similar to the Kullback-Leibler divergence. In some recent works, several authors used entropy as a measure of proximity or interrelation between clusters. Examples of these algorithms are those proposed by Jenssen et al.[41] and Gokcay and Príncipe [42], which use a so-called Between-Cluster Entropy, and the one proposed by Lee and Choi [43], [44], which uses the Within-Cluster Association. Despite the good results in several data sets, these algorithms are heavily time consuming, and they start by selecting random seeds for the first clusters that may produce very different results in the final cluster solution. These algorithms usually give good results for compact and well-separated clusters.

3 The Clustering Algorithm Components

One of the main concerns when we started searching for an efficient clustering algorithm was to find an extremely simple idea, based on very simple principles, that did not need complex measures of intracluster or intercluster association. Keeping this in mind, we performed clustering tests involving several types of individuals (including children) in order to grasp the mental process of data clustering. The results of these tests can be found in [45]. The tests used two-dimensional (2D) data sets similar to those presented in Section 4. An example of different clustering solutions to a given data set suggested by different individuals is shown in Fig. 2.

One of the most important conclusions from our tests is that human clustering exhibits some balance between the importance given to local (for example, connectedness) and global (for example, structuring direction) features of the data, a fact that we tried to reflect with our algorithm. The tests also provided the majority choices of clustering solutions against which one can compare the clustering algorithms.

Below, we introduce two new clustering algorithm components: a new proximity matrix and a new clustering

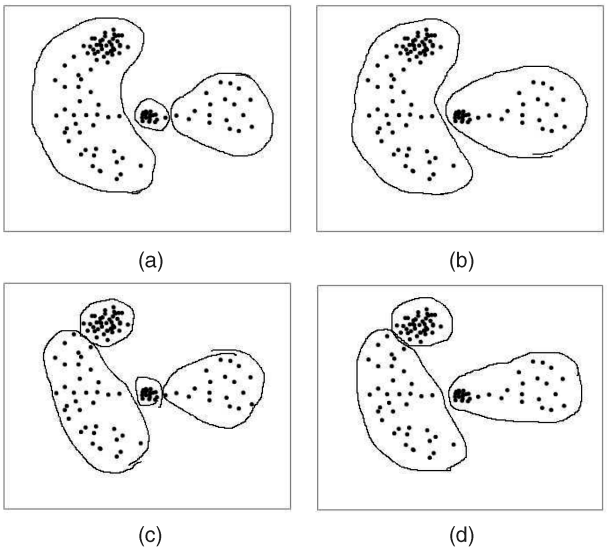

Fig. 2. An example of some clustering solutions for a particular data set. Children usually propose solution (b) and adults solutions (b) and (c). Solution (d) was never proposed.

process. We first present the new entropic dissimilarity measure and, based on that, the computing procedure of a layered entropic proximity matrix (EPM); following that, we present the LEGClust algorithm.

3.1 The Entropic Proximity Matrix

Given a set of vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $\mathbf{x_i} \in \mathbb{R}^m$, corresponding to a set of objects, each element of the dissimilarity matrix A, $A \in \mathbb{R}^{N \times N}$, is computed using a dissimilarity measure $A_{i,j} = d(\mathbf{x}_i, \mathbf{x}_j)$. Using this dissimilarity matrix, one can build a proximity matrix L, where each ith line represents the data set points, each jth column represents the proximity order (1st column $=$ closest point $\ldots$ last column $=$ farthest point), and each element represents the point reference that, according to row point i, is in the jth proximity position. An example of a proximity matrix is shown in Table 5 (to be described in detail later on). The points referenced in the first column (L1) of the proximity matrix are those that have the smallest dissimilarity value to each one of the row elements.

Each column of the proximity matrix corresponds to one layer of connections. We can use this proximity matrix to build subgraphs for each layer, where each edge is the connection between a point and the corresponding point of that layer.

If we use a proximity matrix based on a dissimilarity matrix built with the euclidean distance to connect each point with its corresponding L1 point (first layer), we get a subgraph similar to the one presented in Fig. 3a for the data set in Fig. 10f. We will call the clusters formed with this first layer the elementary clusters. Each of these resulting elementary clusters (not considering directed edges) is a Minimum Spanning Tree.

As we can see in Fig. 3a, these connections have no relation with the structure of the given data set. In Fig. 3b, we present what we think should be the "ideal" connections. These ideal connections should, in our opinion, reflect the local structuring direction of the data. However, using classical distance measures, we are not able to achieve this behavior. As we will see bellow, entropy will allow us to do

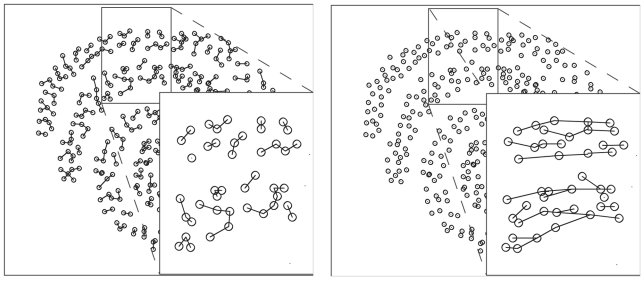

Fig. 3. Connections of the first layer using the euclidean distance and the "ideal" connections for the spiral data set in Fig. 10f. (a) Connections based on the euclidean distance. (b) "Ideal" connections.

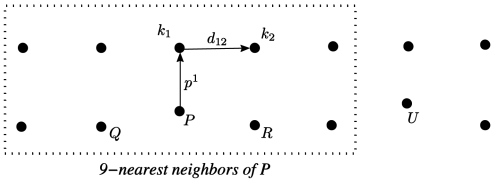

Fig. 4. A simple example with the considered M-nearest neighbors of point P, $M = 9$. The M-neighborhood of P corresponds to the dotted region.

it. The main idea behind the entropic dissimilarity measure is to make the connections follow the local structure of the data set, where the meaning of "local structure" will be clarified later. This concept can be applied to data sets with any number of dimensions.

Let us consider the set of points depicted in Fig. 4. These points are in a square grid except for points P and U. For simplicity, we use a 2D data set, but the analysis is valid for higher dimensions. Let us denote

- $K = \{k_i\}$, $i = 1, 2, \ldots, M$, the set of the M-nearest neighbors of P;
- d_{ij} the difference vector between points k_i and k_j, $i, j = 1, 2, \ldots, M$, $i \neq j$, which we will call the *connecting vector* between those points; and
- p^i the difference vector between point P and each of the M-nearest neighbors k_i.

We wish to find the connection between P and one of its neighbors that best reflects the local structure. Without making any computation and just by "looking" at the points, we can say, despite the fact that the shortest connection is p_1, that the ideal candidates for "best connection" are those connecting P with Q or with R because they are the ones that best reflect the structuring direction of the data points.

Let us represent all d_{ij} connecting vectors translated to a common origin as shown in Fig. 5a. We will call this an *M-neighborhood vector field*. An M-neighborhood vector field can be interpreted as a pdf in correspondence with the 2D histogram shown in Fig. 5b, where in each bin, we plot the number of occurrences of d_{ij} vector ends. This histogram estimates the pdf of d_{ij} connections. It can be interpreted as a Parzen window estimate of the pdf using a rectangular kernel.

The pdf associated with point P reflects, in this case, a horizontal M-neighborhood structure and, therefore, we must choose a connection for P that follows this horizontal direction. Although the direction is an important factor, we should also consider the size of the connections and avoid the

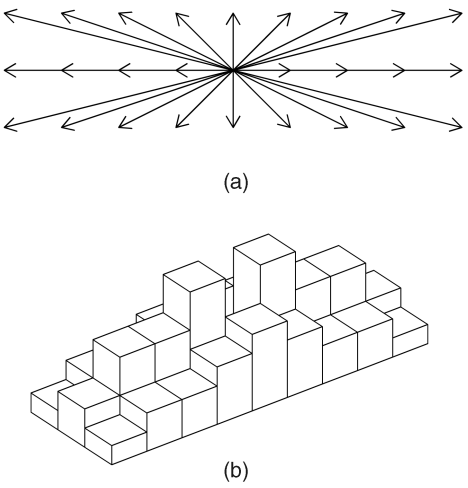

Fig. 5. (a) The M-neighborhood vector field of point P and (b) the histogram representation of the pdf.

TABLE 1
Entropic Dissimilarities Relative to Point P (10)

Points	1	2	3	4	5	6	7	8	9	10	11	12	13	14
10	8.83	8.73	8.72	8.73	8.83			8.66	8.58		8.58	8.66		

TABLE 2
Entropic Proximities Relative to Point P (10)

Points	L1	L2	L3	L4	L5	L6	L7	L8	L9
10	11	9	12	8	3	4	2	5	1

selection of connections between points far apart. Taking this into consideration, we can also see that in terms of the pdf, the small connecting vectors are the most probable ones.

Now, since we want to choose a connection for point P based on ranking all possible connections, we have to compare all the pdf's resulting from adding each connection p^i to the set of connection of the M-neighborhood vector field. To perform this comparison between pdf's, we will use an entropic measure. Basically, what we are going to do is to rank connection p^i according to the variation they introduce in the pdf. The connection that introduces less disorder into the system (that least increases the entropy of the system) will be top ranked as the *stronger* connection, followed by the other $M-1$ connections in decreasing order.

Let $D=\{d_{ij}\}$, $i,j=1,2,\ldots,M$, $i\neq j$. Let $H(D,p^i)$ be the entropy associated with connection p^i, the entropy of the set of all connections d_{ij} plus connection p^i, such that

$$H(D,p^i)=H(\{D\}\cup\{p^i\}), i=1,2,\ldots,M. \quad (5)$$

This entropy is our dissimilarity measure. We compute for each point the M possible entropies and build an entropic dissimilarity matrix and the corresponding EPM (an example is shown in Tables 5 and 6). The elements of the first column of the proximity matrix are those corresponding to the points having the smallest entropic dissimilarity value (strongest entropic connection), followed by those in the subsequent layers in decreasing order.

Regarding our simple example in Fig. 4, we show in Tables 1 and 2 the dissimilarity and proximity values for point P and their neighbors. We use Renyi's quadratic entropy computed as explained in Section 2.3. The points in Fig. 4 are referenced left to right and top to bottom as 1 to 14.

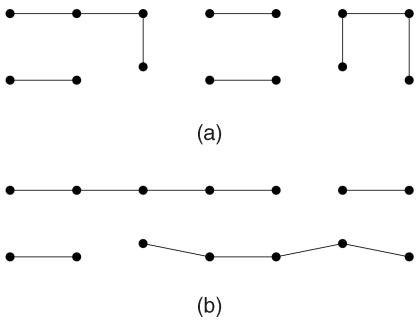

Fig. 6. Difference on elementary clusters using a dissimilarity matrix based (a) on the euclidean distance and (b) on our entropic measure.

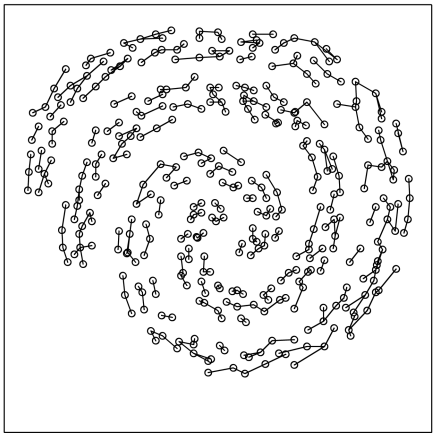

Fig. 7. The first layer connections following the structure of the data set when using an EPM.

TABLE 3
Pseudocode for Computing the EPM with Any Dissimilarity Measure

```
For i = 1 to N (number of objects)
    For j = 1 to M (number of nearest neighbors)
        Compute H(D, p_j) = H({D} ∪ {p_j})
    end j
end i
Build the (N × M) entropic proximity matrix.
```

In Fig. 6, we show the first layer connections, where we can see the difference between using a dissimilarity matrix based on distance measures (Fig. 6a) and a dissimilarity matrix based on our entropic measure (Fig. 6b). The connections derived by the first layer when using the entropic measure clearly follow an horizontal line and despite the fact that point k_1 is the closest one, the stronger connection for point P is the connection between P and R, as expected. This different behavior can also be seen in the spiral data set depicted in Fig. 7. The connections that produce the elementary first-layer clusters are clearly following the structuring direction of the data. We obtain the same behavior for the connections of all the layers favoring the union of those clusters that follow the structure of the data.

The pseudocode to compute the EPM is presented in Table 3.

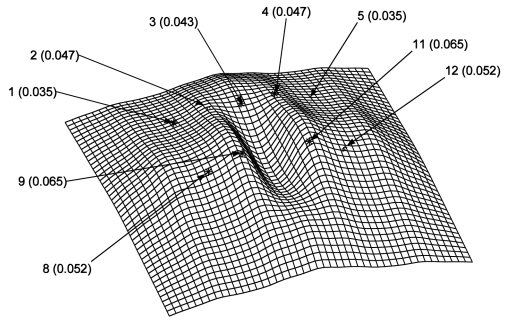

Fig. 8. The pdf of the M-neighborhood vector field and the points corresponding to the p^i connections. The labels indicate the element number and the pdf value.

TABLE 4
Pseudocode for the LEGClust Algorithm

Compute the entropic proximity matrix. (Table III)
Form the elementary clusters using the first layer.
Define k - the minimum number of connections.
While number-of-clusters> 1 do
    Go to next layer (L)
    Join each cluster with the one having the highest number of connections with it ($\geq k$) in L
End While

The process just described is different from the apparently similar process of ranking the connections p^i according to the value of the pdf derived from the M-neighborhood vector field. In Fig. 8, we show the estimated pdf and the points corresponding to the p^i connections. The corresponding point ranking according to decreasing pdf value is 11, 9, 12, 8, 4, 2, 3, 5, 1; we can see that even in this simple example, a difference exists between the pdf ranking and the entropy ranking previously reported in Table 2 (fifth rank). As a matter of fact, one must bear in mind that our entropy ranking is a way of summarizing (and comparing) the degree of randomness of the *several* pdf's corresponding to the p^i connections, whereas *single*-value pdf ranking cannot clearly afford the same information.

3.2 The Clustering Process

Having created the new EPM, we could use it with an existing clustering algorithm to cluster the data. However, the potentialities of the new proximity matrix can be exploited with a new hierarchical agglomerative algorithm that we propose and call LEGClust. The basic foundations for this new clustering algorithm are unweighted subgraphs. More specifically, they are directed, maximally connected, unweighted subgraphs, built with the information provided by the EPM. Each subgraph is built by connecting each point with the corresponding point of each layer (column) of the EPM. An example of such a subgraph was already shown in Fig. 7. The clusters are built hierarchically by joining together the clusters that correspond to the layer subgraphs.

We will start by presenting, in Table 4, the pseudocode of the LEGClust algorithm. This will be followed by a further explanation using a simple example.

To illustrate the procedure applied in the clustering process, we use a simple 2D data set example (Fig. 9a). This data set consists of 16 points apparently constituting two

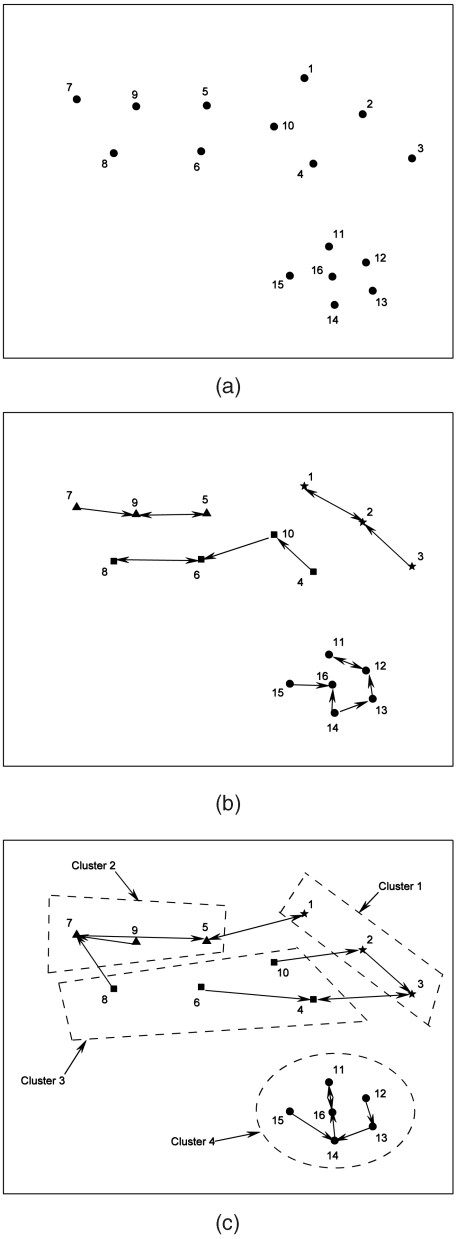

Fig. 9. The clustering process in a simple 2D data set. (a) The data points. (b) First-layer connections and the resulting elementary clusters. (c) The four elementary clusters and the second-layer connections.

clusters with 10 and 6 points each. Since the number of clusters in a data set is highly subjective, the assumption that it has a specific number of clusters is always affected by the knowledge about the problem.

In Table 6, we present the EPM built from the entropic dissimilarity matrix in Table 5.

The EPM defines the connections between each point and those points in each layer: point 1 is connected with point 2 in the first layer, with point 5 in the second layer, with point 10 in the third layer, and so on (see Table 6). We start the process by defining the elementary clusters. These clusters are built by connecting, with an oriented edge, each point with the corresponding point in the first layer (Fig. 9b). There are four elementary clusters in our simple example.

In the second step of the algorithm, we connect, with an oriented edge, each point with the corresponding point in

TABLE 5
The Dissimilarity Matrix for Fig. 9 Data Set

Points	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	-	5.64	-	6.36	5.66	6.32	-	-	-	5.77	-	-	-	-	-	-
2	6.00	-	6.03	6.26	6.40	-	-	-	-	6.12	-	-	-	-	-	-
3	-	6.19	-	6.61	-	-	-	-	-	7.03	6.76	6.48	-	-	-	-
4	6.48	6.50	6.58	-	-	-	-	-	-	6.36	6.47	-	-	-	-	-
5	6.10	-	-	-	-	6.24	-	6.16	6.09	6.18	-	-	-	-	-	-
6	-	-	-	6.05	6.21	-	-	6.03	6.14	6.11	-	-	-	-	-	-
7	-	-	-	-	5.61	6.06	-	5.67	5.28	6.84	-	-	-	-	-	-
8	-	-	-	-	6.09	5.54	5.82	-	5.89	6.27	-	-	-	-	-	-
9	-	-	-	-	5.78	5.98	5.79	6.03	-	5.99	-	-	-	-	-	-
10	6.06	5.98	-	6.05	6.00	5.98	-	-	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	-	-	-	-	3.74	4.41	4.73	3.93	3.86
12	-	-	-	-	-	-	-	-	-	-	3.86	-	3.88	4.36	4.60	3.95
13	-	-	-	-	-	-	-	-	-	-	4.39	3.78	-	3.79	4.75	3.87
14	-	-	-	-	-	-	-	-	-	-	4.71	4.36	3.80	-	3.93	3.86
15	-	-	-	-	-	-	-	-	-	-	3.85	4.81	5.01	3.82	-	3.71
16	-	-	-	-	-	-	-	-	-	-	4.19	4.18	4.18	4.18	4.18	-

TABLE 6
The Proximity Matrix for Fig. 9 Data Set

Points	L1	L2	L3	L4	L5
1	2	5	10	6	4
2	1	3	10	4	5
3	2	4	11	1	10
4	10	3	6	2	11
5	9	1	8	10	6
6	8	4	10	9	5
7	9	5	8	6	10
8	6	7	9	5	10
9	5	7	10	6	8
10	6	2	5	4	1
11	12	16	15	13	14
12	11	13	16	14	15
13	12	14	16	11	15
14	13	16	15	12	11
15	16	14	11	12	13
16	14	11	13	12	15

the second layer (Fig. 9c). In order to build the second-step clusters, we apply a rule based on the number of connections to join each pair of clusters. We can use the simple rules of 1) joining each cluster with the ones having at least k connections with it or 2) joining each cluster with the one having the highest number of connections with it not less than a predefined k. In the performed experiments, this second rule proved to be more reliable, and the resulting clusters were usually "better" than using the first rule. The parameter k must be greater than 1 in order to avoid outliers and noise in the clusters. In our simple example, we chose to join the clusters with the maximum number of connections larger than 2 ($k > 2$). In the second step, we form three clusters by joining clusters 1 and 3 with three edges connecting them (note that the edge connecting points 3 and 4 is a double connection).

The process is repeated, and the algorithm stops when only one cluster is present or when we get the same number of clusters in consecutive steps. The resulting number of clusters for this simple example was 4-3-2-2-2. As we can see, the number of clusters in Steps 3 and 4 is the same (2); therefore, we will consider it to be the acceptable number of clusters.

3.3 Parameters Involved in the Clustering Process

3.3.1 Number of Nearest Neighbors

The first parameter that one must choose in LEGClust is the number of nearest neighbors (M). We do not have a specific rule for this. However, one should not choose a very small value because a minimum number of steps in the algorithm is needed in order to guarantee reaching a solution. Choosing a relatively high value for M is also not a good alternative because one loses information about the local structure, which is the main focus of the algorithm.

Based on the large amount of experiments performed with the LEGClust algorithm on several data sets, we came to a rule of thumb of using M values not higher than 10 percent of the data set size. Note that since the entropy computation for all the data set has complexity $O(N(\binom{M}{2}+1)^2)$, the value of M has a large influence on the computational time. Hence, for large data sets, a smaller M is recommended, down to 2 percent of the data size. For image segmentation, M can be reduced to less than 1 percent due to the nature of image characteristics (elements are much closer to each other than in a usual classification problem).

3.3.2 The Smoothing Parameter

The h parameter is very important when computing the entropy. In other works, [41], [42], using Renyi's Quadratic Entropy to perform clustering, it is assumed that the smoothing parameter is experimentally selected and that it must be fine tuned to achieve acceptable results. One of the formulas for an estimate of the Gaussian kernel smoothing parameter for unidimensional pdf estimation, assuming Gaussian distributed samples, was proposed by Silverman [46]:

$$h_{op} = 1.06\,\sigma\,N^{-0.2}, \tag{6}$$

where σ is the sample standard deviation, and N is the number of points. For multidimensional cases, also assuming normal distributions and using the Gaussian kernel, Bowman and Azzalini [47] proposed the following formula:

$$h_{op} = \sigma\left(\frac{4}{(m+2)N}\right)^{\frac{1}{m+4}}, \tag{7}$$

where m is the dimension of vector x. Formulas (6) and (7) were also compared by Jenssen et al. in [48], where they use (6) to estimate the optimal one-dimensional kernel size for each dimension of the data and use the smallest value as the smoothing parameter.

In a previous paper [49], we have proposed the formula $h_{op} = 25\sqrt{m/N}$ and experimentally showed that higher values of h than those given by (7) produce better results in neural network classification using error entropy minimization as a cost function. Following the same approach, we propose a formula similar to (7) but with the introduction of the mean standard deviation

$$h_{op} = 2\,\sigma^*\left(\frac{4}{(m+2)N}\right)^{\frac{1}{m+4}}, \tag{8}$$

where σ^* is the mean value of the standard deviations for each dimension. All experiments of LEGClust were performed using (8).

Although the value of the smoothing parameter is important, it is not crucial in order to obtain good results. As we increase the h value, the kernel becomes smoother, and the EPM becomes similar to the euclidean distance proximity matrix. Extremely small values of h will produce

TABLE 7
Real Data Sets Used in the Experiments

Data set	*# Objects*	*# Features*	*# Classes*
20NewsGroups	1000	565	20
DHN	2000	3	10
Iris	150	4	3
NCI Microarray	64	6830	12
Olive	572	8	9
Wdbc	569	30	2
Wine	178	13	3
UBIRIS	image data set		

undesirable behaviors because the entropy will have high variability. Using h values in a small interval near the optimal value does not affect the final clustering results (for example, we used in the spiral data set (Fig. 3) values between 0.05 and 0.5 without changing the final result).

3.3.3 Minimum Number of Connections

The third parameter that must be chosen in the LEGClust algorithm is the value of k, the minimum number of connections to join clusters in consecutive steps of the algorithm. As mentioned earlier, we should not use $k = 1$ to avoid outliers and noise, especially if they are located between clusters. In our experiments, we obtained good results using either $k = 2$ or $k = 3$. If the elementary clusters have a small number of points, we do not recommend higher values for k because it can cause the impossibility of joining clusters due to lack of a sufficient number of connections among them.

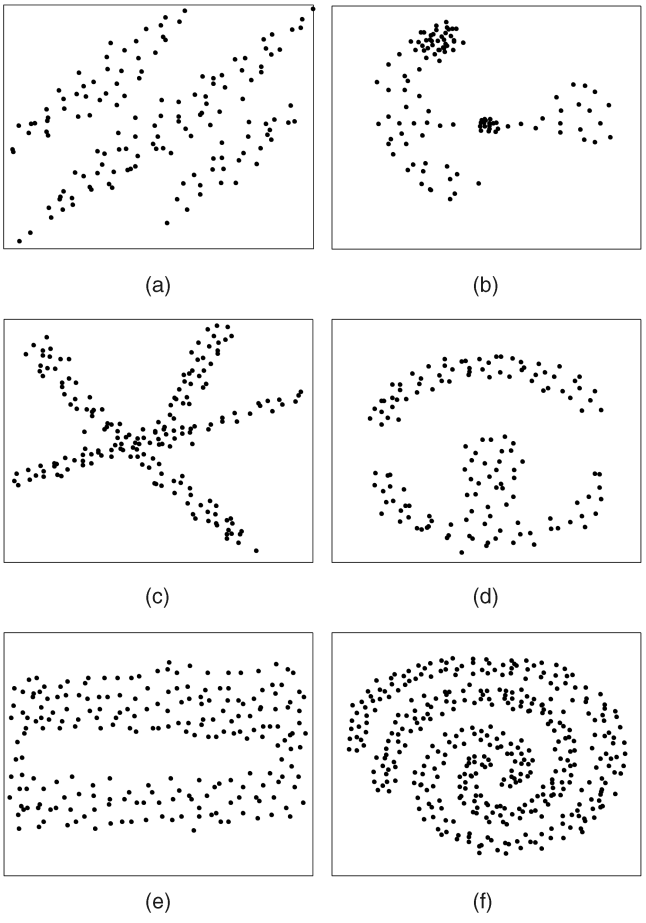

Fig. 10. Some of the artificial data sets used in the experiments (in brackets the number of points). (a) Data set 7 (142). (b) Data set 13 (113). (c) Data set 15 (184). (d) Data set 22 (141). (e) Data set 34 (217). (f) Spiral (387).

4 Experiments

We have experimented the LEGClust algorithm in a large variety of applications. We have performed experiments with real data sets, some of them with a large number of features, and also with several artificial 2D data sets. The real data sets are summarized in Table 7. They are from the public domain. Data set UBIRIS can be found in [50], NCI Microarray in [51], 20NewsGroups, Dutch Handwritten Numerals (DHN), Iris, Wdbc, and Wine in [52], and Olive in [53]. The artificial data sets were created in order to better visualize and control the clustering process, and some examples are depicted in Fig. 10. For the artificial data set problems, the clustering solutions yielded by different algorithms were compared with the majority choice solutions obtained in the human clustering experiment mentioned in Section 3 and described in [45]. For real data sets, the comparison was made with the supervised classification of these data sets with the exception of the UBIRIS data set where the objective of the clustering task was the correct segmentation of the eye's iris. In both cases —majority choice or supervised classes—we will designate these solutions as reference solutions or reference clusters.

We have compared our algorithm with several well-known clustering algorithms: Chameleon algorithm, two Spectral clustering algorithms, DBScan algorithm, and Mean Shift algorithm.

The Chameleon clustering algorithm, included in Cluto [54], is a software package for clustering low and high-dimensional data sets. The parameters used in the experiments among the innumerous used by Chameleon are referred in the results. In fact, the number of parameters needed to tune this algorithm was one of the main problems we encountered when we tried to use it in our experiments. To perform the experiments with Chameleon, we followed the advice in [10] and in the manual for the Cluto software [55].

For the experiments with the spectral clustering approaches, we implemented the algorithms (Spectral-Ng) and (Spectral-Shi) presented in [24] and [21], respectively. One of the difficulties with both Spectral-(Ng/Shi) algorithms is the choice of the scaling parameter. Extremely small changes in the scaling parameter produced very different clustering solutions. In these algorithms, the number of clusters is the number of eigenvectors used to perform clustering. The number of clusters is a parameter that is chosen by the user in both algorithms. We tried to make this choice, in Spectral-Ng, an automatic procedure by implementing the algorithm presented in [56]; this, however, produced poor results. When making the choice of the cluster centroids in the K-Means clustering used in Spectral-Ng, we performed a random initialization and 10 restarts (deemed acceptable by the authors).

We tested the adaptive Mean Shift algorithm [31] in our artificial data sets, and the results were very poor. In most of the cases, the proposed clustering solution has a high number of modes and, consequently, a high number of clusters. For problems having a small number of points, the

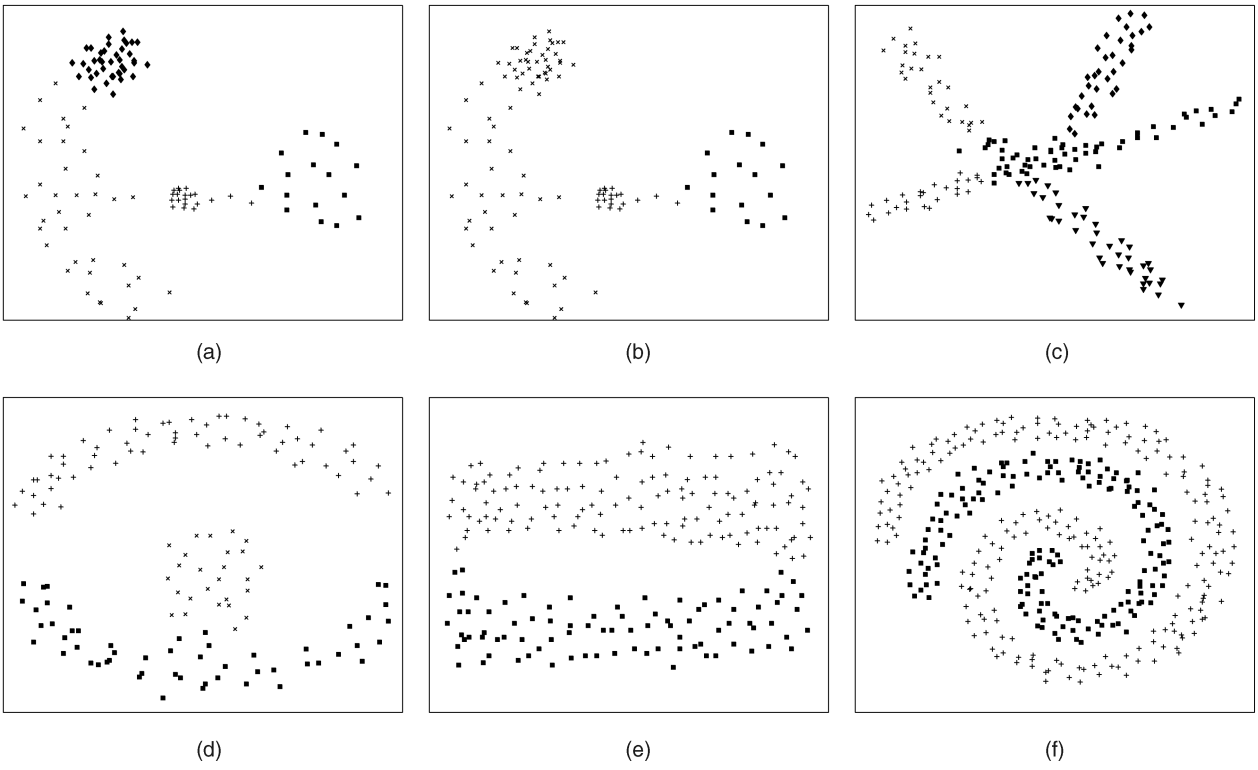

Fig. 11. The clustering solutions for each data set suggested by LEGClust. Each label shows the data set name, the number of neighbors (M), the number of connections to join clusters (k), and the number of clusters found in each step of the algorithm (underlined is the considered step). (a) Data set 13, $\mathrm{M} = 10$, $\mathrm{k} = 3$, 11 7 5 $\underline{4}$ 3 3 3. (b) Data set 13, $\mathrm{M} = 10$, $\mathrm{k} = 3$, 11 7 5 4 $\underline{3}$ 3 3. (c) Data set 15, $\mathrm{M} = 18$, $\mathrm{k} = 2$, 55 37 16 7 $\underline{5}$ 3 2 1. (d) Data set 22, $\mathrm{M} = 14$, $\mathrm{k} = 2$, 45 26 12 8 5 $\underline{3}$ 2 2. (e) Data set 34, $\mathrm{M} = 20$, $\mathrm{k} = 3$, 68 59 36 25 15 8 5 3 $\underline{2}$ 2. (f) Spiral, $\mathrm{M} = 30$, $\mathrm{k} = 2$, 116 72 28 14 5 3 $\underline{2}$ 1.

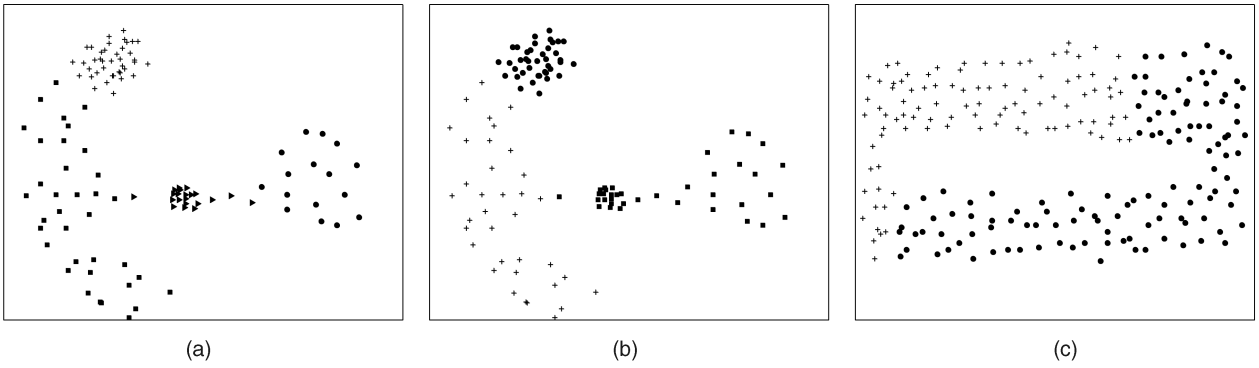

Fig. 12. Some clustering solutions suggested by Chameleon. The considered values nc, a, and n are shown in each label. (a) Data set 13, $\mathrm{nc} = 4$, $\mathrm{a} = 20$, $\mathrm{n} = 20$. (b) Data set 13, $\mathrm{nc} = 3$, $\mathrm{a} = 20$, $\mathrm{n} = 20$. (c) Data set 34, $\mathrm{nc} = 2$, $\mathrm{a} = 50$, $\mathrm{n} = 6$.

estimated density function will present, depending on the window size, either a unique mode if we use a large window size or several modes not corresponding to really existing clusters if we use a small window size. We think that this algorithm probably works better with large data sets. An advantage of this algorithm is the fact that one does not have to specify the number of clusters as these will be driven by the data according to the number of modes.

The DBScan algorithm is a density-based algorithm that claims to find clusters of arbitrary shapes but presents basically the same problems as the Mean Shift algorithm. It is based on several density definitions between a point and its neighbors. This algorithm only requires two input parameters, Eps and MinPts, but small changes in their values, especially in Eps, produce very different clustering solutions. For our experiments, we used an implementation of DBScan available in [57].

In the LEGClust algorithm, the parameters involved are the smoothing parameter (h), related to the Parzen pdf estimation; the number of neighbors to consider (M); and the number of connections to join clusters (k). For the parameter h, we used in all experiments the proposed formula (8). For the other two parameters, we indicate in each experiment the chosen values.

Regarding the experiments with artificial data sets, depicted in Fig. 10, we present in Fig. 11 the results obtained with LEGClust.

In Fig. 12, we present the solutions obtained with the Chameleon algorithm that differ from those suggested by LEGClust.

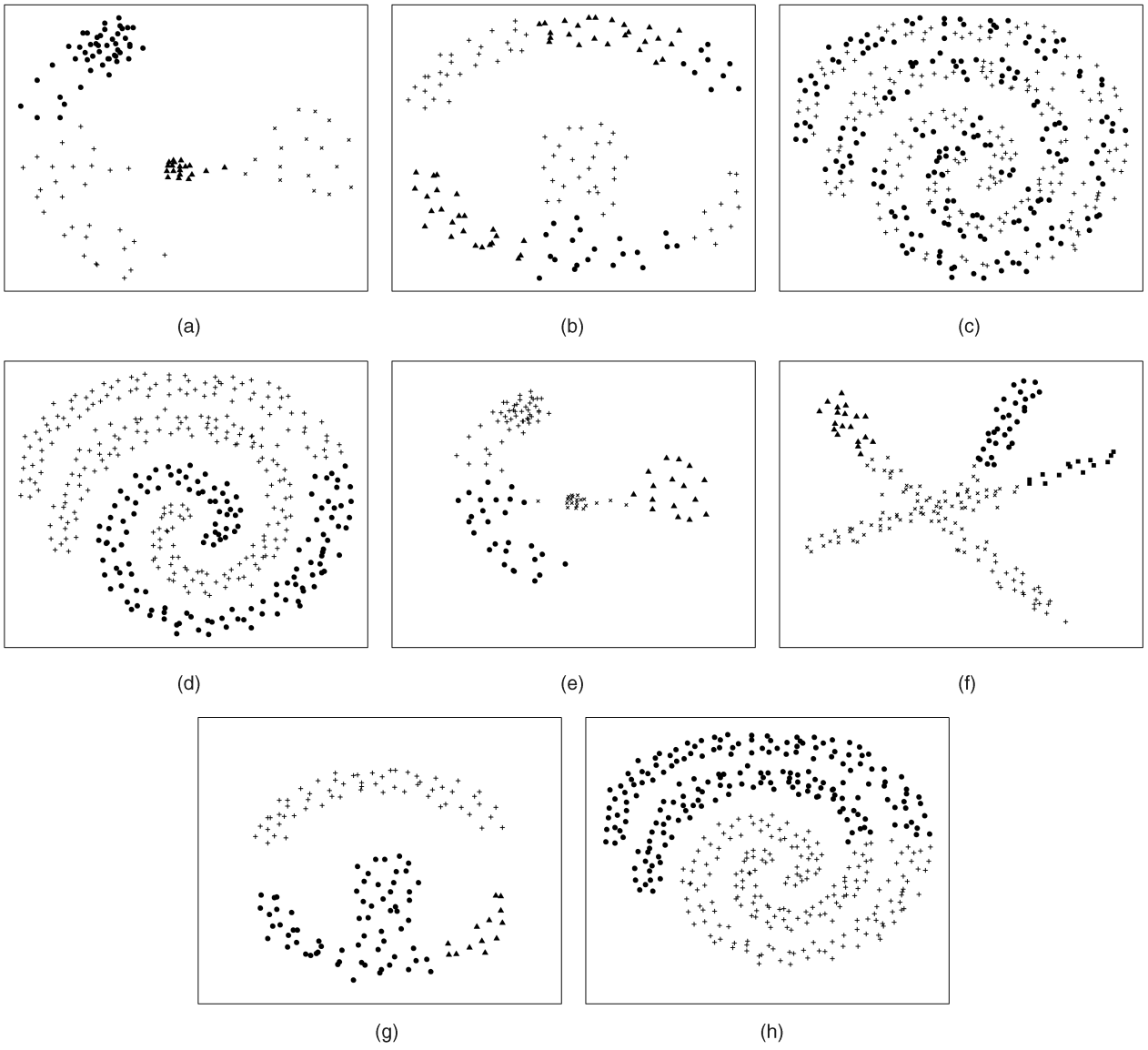

Fig. 13. Some clustering solutions suggested by Spectral-Ng (a), (b), (c), and (d) and by Spectral-Shi (e), (f), (g), and (h). Each label shows the data set name, the preestablished number of clusters, and the σ value. (a) Data set 13, $\mathrm{nc} = 4$, $\sigma = 0.065$. (b) Data set 22, $\mathrm{nc} = 3$, $\sigma = 0.071$. (c) Spiral; $\mathrm{nc} = 2$, $\sigma = 0.0272$. (d) Spiral, $\mathrm{nc} = 2$, $\sigma = 0.0275$. (e) Data set 13, $\mathrm{nc} = 4$, $\sigma = 0.3$. (f) Data set 15, $\mathrm{nc} = 5$, $\sigma = 0.3$. (g) Data set 22, $\mathrm{nc} = 3$, $\sigma = 0.15$. (h) Spiral, $\mathrm{nc} = 2$, $\sigma = 0.13$.

From the performed experiments, an important aspect noticed when using the Chameleon algorithm was the different solutions obtained for slightly different parameter values. The data set in Fig. 12c was the one where we had more difficulties in tuning the parameters involved in the Chameleon algorithm. A particular difference between the Chameleon and LEGClust results corresponds to the curious solution given by Chameleon and is depicted in Fig. 12b. When choosing three clusters as input parameter $(\mathrm{nc} = 3)$, this solution is the only solution that is not suggested by the individuals that performed the tests referred in Section 3. The solutions for this same problem, given by LEGClust, are shown in Figs. 11b and 11c.

The spectral clustering algorithms gave some good results for some data sets, but they were unable to resolve some nonconvex data sets like the double spiral problem (Fig. 13).

The DBScan algorithm clearly fails in finding the reference clusters in all data sets (except the one in Fig. 10a).

Comparing the results given by all the algorithms applied to the artificial data sets, we clearly see, as expected, that the solutions obtained with the density-based algorithms are worse than those obtained with any of the other algorithms. The best results were achieved with LEGClust and Chameleon algorithms.

We now present the performed experiments with LEGClust in real data sets and the comparative results with the different clustering algorithms.

UBIRIS is a data set of eye images used for biometric recognition. In our experiments, we used a sample of 12 images from this data set, some of which are shown in Fig. 14a. The biometric identification process starts by detecting and isolating the iris with a segmentation algorithm. The results for this image segmentation problem with LEGClust and Spectral-Ng are depicted in Figs. 14b and 14c. In all experiments with LEGClust, we used the values $M = 30$ and $k = 3$. For the experiments with Spectral-Ng, we chose 5

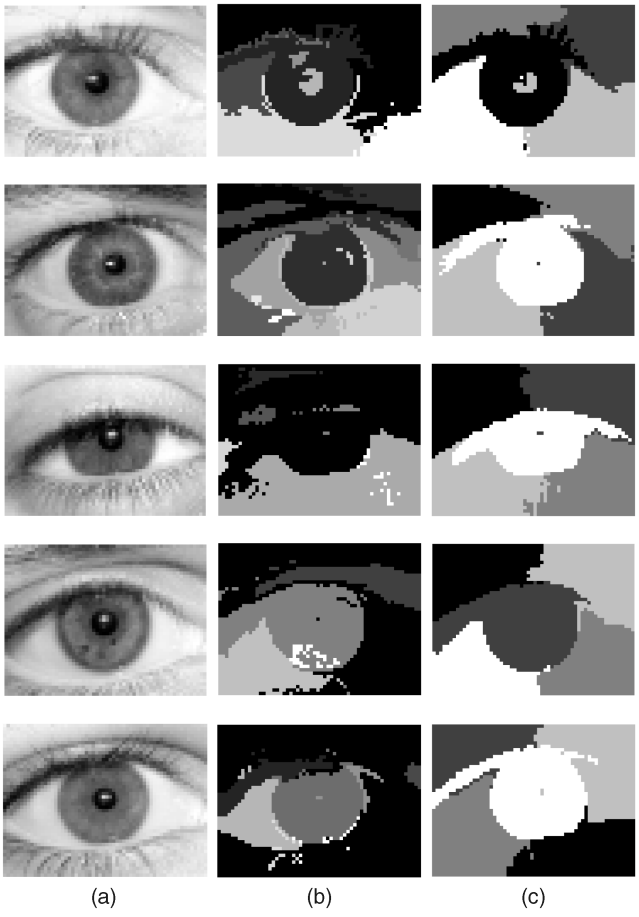

Fig. 14. Sample from the (a) UBIRIS data set and (b) the results of the LEGClust and (c) Spectral clustering algorithms. The number of clusters for LEGClust was 8, 12, 7, 5, and 8 with $M = 30$ and $k = 3$.

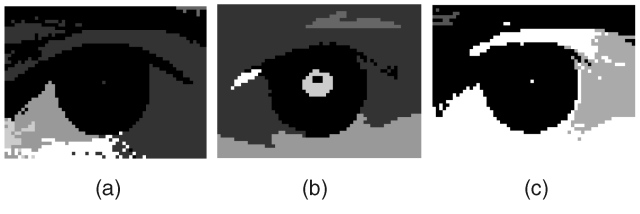

Fig. 15. Segmentation results for the fourth image (line 4) in Fig. 14 using different values of k or M. (a) $k = 2$. (b) $M = 10$. (c) $M = 20$.

as the number of final clusters. We can see by the segmentations produced that both algorithms gave acceptable results. However, one of the striking differences is the way Spectral clustering splits each eyelid in two by its center region (Fig. 14c is a good example of this behavior), which is also observable if we choose a different number of clusters.

To test the sensitivity of our clustering algorithm to different values of the parameters, we have made some experiments with different values of M and k in the UBIRIS data set sample. An example is shown in Fig. 15. We can see that different values of M and k do not affect substantially the final result of the segmentation process; the eye iris in all solutions is distinctly obtained.

The Dutch Handwritten Numerals (DHN) data set consists of 2,000 images of handwritten numerals ("0"-"9") extracted from a collection of Dutch utility maps [58]. A sample of this data set is depicted in Fig. 16. In this data set, the first two

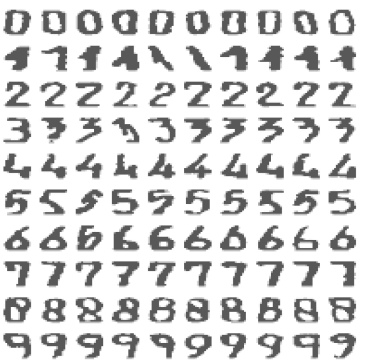

Fig. 16. A sample of the DHN data set.

TABLE 8
The Results and Parameters Used in the Comparison of LEGClust and Spectral Clustering in Experiments with DHN, 20NewsGroups, and NCI Microarray Data Sets

	LEGClustt			Spectral-Ng			Spectral-Shi		
	M	k	ARI	nc	σ	ARI	nc	σ	ARI
DHN	30	10	0.628	10	10	0.287	10	12	0.573
	30	8	0.608						
	30	12	0.574						
20NewsGroups	20	3	0.289	20	12	0.479	20	20	0.006
	20	2	0.287						
NCI Microarray	4	2	0.148	3	80	0.177	3	10	0.138
	6	3	0.148						
	10	3	0.148						

features represent the pixel position, and the third one, the gray level. Experiments with this data set were performed with LEGClust and Spectral clustering, and their results were compared. The results are presented in Table 8. ARI stands for Adjusted Rand Index, a measure for comparing results of different clustering solutions when the labels are known [59]. This index is an improvement of the Rand Index, it lies between 0 and 1, and the higher the ARI index, the better the clustering solution. The parameters for both Spectral clustering and LEGClust were tuned to give the best possible solutions. We can see that in this problem, LEGClust performs much better than Spectral-Shi and with similar (but slightly better) results than Spectral-Ng. We also show in Table 8 some different results for LEGClust for different choices of the minimum number of connections (k) to join clusters. In these results, we can see that different values of k produce results with small differences in the ARI index.

In the experiments with the 20NewsGroups data set, we used a random subsample of 1,000 elements from the original data set. This data set is a 20-class text classification problem obtained from 20 different news groups. We have prepared this data set by stemming words according to the Porter Stemming Algorithm [60]. The size of the corpus (the number of different words presented in all the stemmed data set) defines the number of features. In this subsample, we consider only the words that occur at least 40 times, thus obtaining a corpus of 565 words. The results of the experiments with LEGClust and Spectral clustering are shown in Table 8.

TABLE 9
The Results and Parameters Used in the Comparison of LEGClust and Chameleon in Experiments with Four Real Data Sets

	Chameleon			LEGClust		
	a	n	ARI	M	k	ARI
Iris	9	50	0.658	15	3	0.750
Olive	40	40	0.733	25	3	0.616
Wdbc	40	25	0.410	20	3	0.574
Wine	30	21	0.400	15	3	0.802

The NCI Microarray data set is a human tumor microarray data and an example of a high-dimensional data set. The data are a $64 \times 6,830$ matrix of real numbers, each representing an expression measurement for a gene (column) and a sample (row). There are 12 different tumor types, one with just one representative and three with two representatives. We have performed some experiments with LEGClust and compared the results with Spectral clustering. We chose three clusters, following the example in [61], as the final number of clusters for both algorithms. The results are also shown in Table 8. Again, the results produced by LEGClust were quite insensitive to the choice of parameter values.

The results presented in Table 8 show that the LEGClust algorithm performs better than the Spectral-Shi algorithm in the three data sets, and compared with Spectral-Ng, it gives better results in the DHN data set and similar ones in the NCI Microarray.

In the experiments with the data sets Iris, Olive, Wdbc, and Wine, we compared the clustering solutions given by LEGClust and Chameleon. The parameters used for each experiment and the results obtained with both algorithms are shown in Table 9. Each experiment with the Chameleon algorithm followed the command **vcluster** dataset_name number_of_clusters = **nc** − clmethod = graph − sim = dist − agglofrom = **a** − agglocr = wslink − nnbrs = **n** given in [55]. The final number of clusters is the same as the number of classes. We can see that the results with LEGClust are better than the ones obtained with Chameleon, except for the data set "Olive."

Finally, we also experimented our algorithm in two images from [32], used to test textured image segmentation. We show in Fig. 17 the results obtained and the comparison with those obtained by Fischer et al. [32] with their path-based algorithm. We are aware that our algorithm was not designed having in mind the specific requirements of texture segmentation; as expected, the results were not as good as those obtained in [32], but nevertheless, LEGClust was still capable of detecting some of the structured texture information.

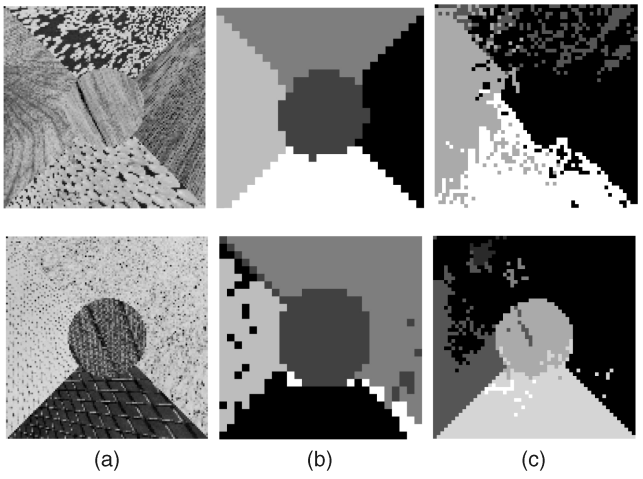

Fig. 17. Segmentation results for (a) textured images with (b) Fischer et al.'s path-based clustering and (c) LEGClust. The parameters used in LEGClust were $M = 30$ and $k = 3$, and the final number of clusters was 4 (top) and 6 (bottom).

5 Conclusion

The present paper presented a new proximity matrix, built with a new entropic dissimilarity measure, as input for clustering algorithms. We also presented a simple clustering process that uses this new proximity matrix and performs clustering by combining a hierarchical approach with a graph technique.

The new proximity matrix and the methodology implemented in the LEGClust algorithm allows taking into account the local structure of the data, represented by the statistical distribution of the connections in a neighborhood of a reference point achieving a good balance between structuring direction and local connectedness. In this way, LEGClust is able, for instance, to correctly follow a structuring direction presented on the data, with the sacrifice of local connectedness (minimum distance), as human clustering often does.

The experiments with the LEGClust algorithm in both artificial and real data sets have shown that

- LEGClust achieves good results compared with other well-known clustering algorithms.
- LEGClust is simple to use since it only needs to adjust three parameters and simple guidelines for these adjustments were presented.
- LEGClust often yields solutions that are majority voted by humans.
- LEGClust's sensitivity to small changes of the parameter values is low.
- LEGClust is a valid proposal for data sets with any number of features.

In our future work, we will include our entropic measure in other existing hierarchical and graph based algorithms and compare them with the LEGClust algorithm in order to try to establish the importance of the entropic measure in the clustering process. We will also implement another clustering process using as input our entropic dissimilarity matrix with a different approach than the one presented here that does not depend on the choice of parameters by the user and that can give us, for example, a fixed number of clusters if so desired.

Acknowledgments

This work was supported by the Portuguese Fundação para a Ciencia e Tecnologia (project POSC/EIA/56918/2004).

References

[1] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data*. Prentice Hall, 1988.

[2] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys,* vol. 31, no. 3, pp. 264-323, 1999.

[3] A. Jain, A. Topchy, M. Law, and J. Buhmann, "Landscape of Clustering Algorithms," *Proc. 17th Int'l Conf. Pattern Recognition,* vol. 1, pp. 260-263, 2004.

[4] P. Berkhin, "Survey of Clustering Data Mining Techniques," technical report, Accrue Software, San Jose, Cailf., 2002.

[5] S. Guha, R. Rastogi, and K. Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," *Proc. ACM Int'l Conf. Management of Data,* pp. 73-84, 1998.

[6] S. Guha, R. Rastogi, and K. Shim, "ROCK: A Robust Clustering Algorithm for Categorical Attributes," *Information Systems,* vol. 25, no. 5, pp. 345-366, 2000.

[7] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, 1990.

[8] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Clustering Method for Very Large Databases," *Proc. ACM SIGMOD Workshop Research Issues on Data Mining and Knowledge Discovery,* pp. 103-114, 1996.

[9] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: A New Data Clustering Algorithm and Its Applications," *Data Mining and Knowledge Discovery,* vol. 1, no. 2, pp. 141-182, 1997.

[10] G. Karypis, E.-H.S. Han, and V. Kumar, "Chameleon: Hierarchical Clustering Using Dynamic Modeling," *Computer,* vol. 32, no. 8, pp. 68-75, 1999.

[11] S.D. Kamvar, D. Klein, and C.D. Manning, "Interpreting and Extending Classical Agglomerative Clustering Algorithms Using a Model-Based Approach," *Proc. 19th Int'l Conf. Machine Learning,* pp. 283-290, 2002.

[12] E.L. Johnson, A. Mehrotra, and G.L. Nemhauser, "Min-Cut Clustering," *Math. Programming,* vol. 62, pp. 133-151, 1993.

[13] D. Matula, "Cluster Analysis via Graph Theoretic Techniques," *Proc. Louisiana Conf. Combinatorics, Graph Theory and Computing,* R.C. Mullin, K.B. Reid, and D.P. Roselle, eds., pp. 199-212, 1970.

[14] D. Matula, "K-Components, Clusters and Slicings in Graphs," *SIAM J. Applied Math.,* vol. 22, no. 3, pp. 459-480, 1972.

[15] E. Hartuv, A. Schmitt, J. Lange, S. Meier-Ewert, H. Lehrachs, and R. Shamir, "An Algorithm for Clustering cDNAs for Gene Expression Analysis," *Proc. Third Ann. Int'l Conf. Research in Computational Molecular Biology,* pp. 188-197, 1999.

[16] E. Hartuv and R. Shamir, "A Clustering Algorithm Based on Graph Connectivity," *Information Processing Letters,* vol. 76, nos. 4-6, pp. 175-181, 2000.

[17] Z. Wu and R. Leahy, "An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Learning,* vol. 15, no. 11, pp. 1101-1113, Nov. 1993.

[18] G. Karypis and V. Kumar, "Multilevel Algorithms for Multi-Constraint Graph Partitioning," Technical Report 98-019, Univ. of Minnesota, Dept. Computer Science/Army HPC Research Center, Minneapolis, May 1998.

[19] M. Fiedler, "A Property of Eigenvectors of Nonnegative Symmetric Matrices and Its Application to Graph Theory," *Czechoslovak Math. J.,* vol. 25, no. 100, pp. 619-633, 1975.

[20] F.R.K. Chung, *Spectral Graph Theory.* Am. Math. Soc., no. 92, 1997.

[21] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.

[22] R. Kannan, S. Vempala, and A. Vetta, "On Clusterings: Good, Bad, and Spectral," *Proc. 41st Ann. Symp. Foundation of Computer Science,* pp. 367-380, 2000.

[23] C. Ding, X. He, H. Zha, M. Gu, and H. Simon, "A Min-Max Cut Algorithm for Graph Partitioning and Data Clustering," *Proc. Int'l Conf. Data Mining,* pp. 107-114, 2001.

[24] A.Y. Ng, M.I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," *Advances in Neural Information Processing Systems,* vol. 14, 2001.

[25] M. Meila and J. Shi, "A Random Walks View of Spectral Segmentation," *Proc. Eighth Int'l Workshop Artificial Intelligence and Statistics,* 2001.

[26] D. Verma and M. Meila, "A Comparison of Spectral Clustering Algorithms," Technical Report UW-CSE-03-05-01, Washington Univ., 2003.

[27] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. Second Int'l Conf. Knowledge Discovery and Data Mining,* pp. 226-231, 1996.

[28] K. Fukunaga and L.D. Hostetler, "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition," *IEEE Trans. Information Theory,* vol. 21, pp. 32-40, 1975.

[29] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 8, pp. 790-799, Aug. 1995.

[30] D. Comaniciu and P. Meer, "Mean Shift Analysis and Applications," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1197-1203, 1999.

[31] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach toward Feature Space Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 5, pp. 603-619, May 2002.

[32] B. Fischer, T. Zöller, and J.M. Buhmann, "Path Based Pairwise Data Clustering with Application to Texture Segmentation," *Proc. Int'l Workshop Energy Minimization Methods in Computer Vision and Pattern Recognition,* pp. 235-250, 2001.

[33] B. Fischer and J.M. Buhmann, "Path-Based Clustering for Grouping of Smooth Curves and Texture Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 4, pp. 513-518, Apr. 2003.

[34] C. Shannon, "A Mathematical Theory of Communication," *Bell System Technical J.,* vol. 27, pp. 379-423 and 623-656, 1948.

[35] A. Renyi, "Some Fundamental Questions of Information Theory," *Selected Papers of Alfred Renyi,* vol. 2, pp. 526-552, 1976.

[36] E. Parzen, "On the Estimation of a Probability Density Function and Mode," *Annals of Math. Statistics,* vol. 33, pp. 1065-1076, 1962.

[37] D. Xu and J. Príncipe, "Training MLPS Layer-by-Layer with the Information Potential," *Proc. Int'l Joint Conf. Neural Networks,* pp. 1716-1720, 1999.

[38] H. Li, K. Zhang, and T. Jiang, "Minimum Entropy Clustering and Applications to Gene Expression Analysis," *Proc. IEEE Computational Systems Bioinformatics Conf.,* pp. 142-151, 2004.

[39] A.O. Hero, B. Ma, O.J. Michel, and J. Gorman, "Applications of Entropic Spanning Graphs," *IEEE Signal Processing Magazine,* vol. 19, no. 5, pp. 85-95, 2002.

[40] C.H. Cheng, A.W. Fu, and Y. Zhang, "Entropy-Based Subspace Clustering for Mining Numerical Data," *Proc. Int'l Conf. Knowledge Discovery and Data Mining,* 1999.

[41] R. Jenssen, K.E. Hild, D. Erdogmus, J. Príncipe, and T. Eltoft, "Clustering Using Renyi's Entropy," *Proc. Int'l Joint Conf. Neural Networks,* pp. 523-528, 2003.

[42] E. Gokcay and J.C. Príncipe, "Information Theoretic Clustering," *IEEE Trans. Pattern Analysis and Machine Learning,* vol. 24, no. 2, pp. 158-171, Feb. 2002.

[43] Y. Lee and S. Choi, "Minimum Entropy, K-Means, Spectral Clustering," *Proc. IEEE Int'l Joint Conf. Neural Networks,* vol. 1, pp. 117-122, 2004.

[44] Y. Lee and S. Choi, "Maximum Within-Cluster Association," *Pattern Recognition Letters,* vol. 26, no. 10, pp. 1412-1422, July 2005.

[45] J.M. Santos and J. Marques de Sá, "Human Clustering on Bi-Dimensional Data: An Assessment," Technical Report 1, INEB —Instituto de Engenharia Biomédica, Porto, Portugal,http://www.fe.up.pt/~nnig/papers/JMS_TechReport2005_1.pdf, Oct. 2005.

[46] B.W. Silverman, *Density Estimation for Statistics and Data Analysis,* vol. 26, Chapman & Hall, 1986.

[47] A.W. Bowman and A. Azzalini, *Applied Smoothing Techniques for Data Analysis.* Oxford Univ. Press. 1997.

[48] R. Jenssen, T. Eltoft, and J. Príncipe, "Information Theoretic Spectral Clustering," *Proc. Int'l Joint Conf. Neural Networks,* pp. 111-116, 2004.

[49] J.M. Santos, J. Marques de Sá, and L.A. Alexandre, "Neural Networks Trained with the EEM Algorithm: Tuning the Smoothing Parameter," *Proc. Sixth World Scientific and Eng. Academy and Soc. Int'l Conf. Neural Networks,* 2005.

[50] H. Proença and L.A. Alexandre, "UBIRIS: A Noisy Iris Image Database," *Proc. Int'l Conf. Image Analysis and Processing,* vol. 1, pp. 970-977, 2005.

[51] "Stanford NCI60 Cancer Microarray Project,"http://genome-www.stanford.edu/nci60/, 2000.

[52] C. Blake, E. Keogh, and C. Merz, "UCI Repository of Machine Learning Databases," http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.

[53] M. Forina and C. Armanino, "Eigenvector Projection and Simplified Non-Linear Mapping of Fatty Acid Content of Italian Olive Oils," *Annali di Chimica,* vol. 72, pp. 127-155, 1981.

[54] G. Karypis, "Cluto: Software Package for Clustering High-Dimensional Datasets," version 2.1.1, Nov. 2003.

[55] G. Karypis, *Cluto: A Clustering Toolkit*, Univ. of Minnesota, Dept. Computer Science, Minneapolis, Nov. 2003.
[56] G. Sanguinetti, J. Laidler, and N.D. Lawrence, "Automatic Determination of the Number of Clusters Using Spectral Algorithms," *Proc. Int'l Workshop Machine Learning for Signal Processing*, pp. 55-60, 2005.
[57] X. Xu, "DBScan," http://ifsc.ualr.edu/xwxu/, 1998.
[58] R.P. Duin, "Dutch Handwritten Numerals," http://www.ph.tn.tudelft.nl/~duin, 1998.
[59] L. Hubert and P. Arabie, "Comparing Partitions," *J. Classification*, vol. 2, no. 1, pp. 193-218, 1985.
[60] M.F. Porter, "An Algorithm for Suffix Stripping," *Program*, vol. 14, no. 3, pp. 130-137, 1980.
[61] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2001.

Jorge M. Santos received the degree in industrial informatics from the Engineering Polytechnic School of Porto (ISEP) in 1994, the MSc degree in electrical and computer engineering from the Engineering Faculty of Porto University (FEUP) in 1997, and the PhD degree in engineering sciences from FEUP in 2007. He is presently an assistant professor in the Department of Mathematics at ISEP and a member of the Signal Processing Group of the Biomedical Engineering Institute at Porto.

Joaquim Marques de Sá received the degree in electrical engineering from the Engineering Faculty of Porto University (FEUP) in 1969 and the PhD degree in electrical engineering (Signal Processing) from FEUP in 1984. He is presently a full professor at FEUP and the leader of the Signal Processing Group of the Biomedical Engineering Institute at Porto.

Luís A. Alexandre received the degree in physics and applied mathematics from the Faculty of Sciences of the Porto University in 1994 and both the MSc and PhD degrees in electrical engineering from the Engineering Faculty of Porto University in 1997 and 2002, respectively. He is currently an auxiliar professor in the Department of Informatics at the University of Beira Interior (UBI) and a member of the Networks and Multimedia Group of the Institute of Telecommunications at UBI.