UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
ISSN 0103-2569

Improved Visual Clustering of Large Multi-Dimensional Data Sets

Eduardo Tejada
Rosane Minghim

Nº 215

RELATÓRIOS TÉCNICOS

São Carlos – SP
Set./2003

Contents

Improved visual clustering of large multi-dimensional data sets

Eduardo Tejada and Rosane Minghim
{eduardo,rminghim}@icmc.usp.br
High Performance Computing Laboratory
Departamento de de Ciências de Computação e Estatística
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
P.O.Box 668
CEP 13560-970, São Carlos, Brazil.

Abstract

Lowering computational cost of data analysis techniques is an essential step towards including the user in the process and achieving scalability of algorithms for large scale visualization.

In this paper we present an improved algorithm for visual clustering of large multi-dimensional data sets. This algorithm is a version with lower computational cost of the IPCLUS algorithm. The original algorithm is an approach that deals efficiently with multi-dimensionality using various projections of the data in order to perform multi-space clustering, pruning outliers through direct user interaction.

The algorithm presented here, named HC-Enhanced, adds a scalability level to the approach without reducing clustering quality. Additionally, an algorithm to improve clusters is added to the approach. A number of test cases is presented with good results.

Keywords: Visual mining, clustering, visual clustering, scalability, multi-dimensionality, large data sets, IPCLUS, HC-Cooperative, HC-Enhanced.

Resumo

Reduzir o custo computacional das técnicas de análise de dados é um passo essencial em direção à inclusão do usuário e a atingir a escalabilidade de algoritmos para visualização de grandes conjuntos de dados.

Neste trabalho apresentamos um algoritmo melhorado para agrupamento visual de grandes conjuntos de dados multidimensionais. Esse algoritmo é uma versão com menor custo computacional do algoritmo IPCLUS. O algoritmo original é uma abordagem que lida eficientemente com multidimensionalidade usando várias projeções dos dados para realizar um agrupamento em múltiplos espaços, eliminando *outliers* através da interferência do usuário.

O algoritmo apresentado aqui, denominado *HC-Enhanced*, adiciona um nível de escalabilidade à abordagem sem reduzir a qualidade do agrupamento. Adicionalmente, um algoritmo para melhorar os grupos é incluído na abordagem. Vários teste são apresentados com bons resultados.

Palavras chave: Mineração visual, agrupamento, agrupamento visual, escalabilidade, multidimensionalidade, grandes conjuntos de dados, *IPCLUS*, *HC-Cooperative*, *HC-Enhanced*.

1 Introduction

Clustering large multi-dimensional data sets presents two major problems besides generating a good clustering: scalability and capacity for dealing with multi-dimensionality. Scalability usually means linear complexity ($O(n)$) or better ($O(n \log n)$, $O(\log n)$, etc.). For highly interactive systems, such as visual clustering techniques [1, 14, 21], scalability is critical. Algorithms with linear complexity but high constant ($O(kn)$, k constant) are not suitable for such systems: interactiveness is a key issue that must be included in visual mining techniques [25], thus the response time must be as low as possible.

Treating multi-dimensionality is also a very difficult task. The research done for tackling this problem are based in three approaches: subspace clustering, co-clustering [5], and feature selection [24]. These approaches are focused on solving the problem known as the "dimensionality curse", which is the incapacity for generating significative structures (patterns or models) from high dimensional data. For clustering algorithms in most cases this means more than 15 dimensions [5, 6].

Subspace clustering refers to approaches that apply dimensionality reduction before clustering the data. Different approaches for dimensionality reduction have been developed, such as PCA [15], Fastmap [9], Singular Value Decomposition (SVD) [22], and Fractal-based techniques [16, 18]. Recently we developed a novel technique named *Nearest-Neighbor Projection* (NNP) for multi-dimensional data exploration on bidimensional spaces [23]. Although some of these techniques present a means to measure the amount of information lost, there is no warranty of dimensionality reduction without loosing a considerable amount of information. Besides that, different clusters could be found on different projections of the same data as shown in the literature [1, 2, 3]. Thus, clustering in projected subspaces could lead to a poor result of the clustering process.

The expected result of every clustering algorithm is a good clustering. However, evaluating what is a good clustering is not a trivial matter, and depends heavily on the knowledge the evaluator has about the data, the domain of the data, and the application for which the clustering was generated. That concept cannot be generally stated for all applications and data sets. This is why generating a good clustering, for an specific application, cannot be achieved without direct user interference. Visual clustering techniques exploit this fact in order to make use of the capacity of humans beings to make decisions combined with the processing power of computers. This human-computer approach also tends to reduce the complexity of the algorithm replacing usually costly heuristics with user decisions.

In order to present a suitable interface to the user, the algorithm must support an intuitive idea of cluster. That is, the concept of cluster used by the algorithm must match the notion of group the user has (dense regions with arbitrary shapes).

Based on these facts, we can state what a clustering algorithm must accomplish in order to be appropriate for real data sets:

1. find clusters in multiple projections,
2. employ users' knowledge of the domain,
3. generate clusters with arbitrary shape, and
4. be scalable.

It is also very desirable to provide mechanisms for handling outliers [1] and to define a metric for determining whether a cluster is consistent with the user responses.

Table 1 summarizes the features for the most representative approaches of the various families of clustering algorithms [2]. Aggarwal's IPCLUS algorithm [1], from now on called HC-Cooperative (from Human-Computer Cooperative), accomplishes almost all the requirements we stated as necessary for an algorithm be appropriate for real data sets.

HC-Cooperative includes user interference during cluster creation. This interaction provides the user with means to understand the data visualizing the clusters formed during the process, as well as the outliers the data presents.

It also presents a metric to measure how consistent a cluster is with the user responses, an efficient way to deal with outliers, an intuitive idea of cluster based on density and the use of multiple projections for treating high dimensional data reducing the information lost.

[1] Instances that cannot be included in any cluster.

[2] See the work by Berkhin for details on most clustering algorithms found in the literature [5].

	Scalability	Multi-dimensionality	Arbitrary Shape	Outliers	User Interaction	Metric
Birch [26]	√					
Chameleon [17]			√			
Cure [13]	√		√	√		
Bubble [11]	√					
Orclus [3]		√		√		
DBScan [8]	√		√	√		
HC-Cooperative. [1]		√	√	√	√	√
HD-Eye [14]	√		√	√	√	

Table 1: Features of representative clustering algorithms.

However, HC-Cooperative cannot be applied to very large data sets due to the costly processes used for projecting the data and estimating the density. We developed mechanisms for reducing the time spent in those processes. Those mechanisms were introduced in different steps of the algorithm. However, the most important gain was achieved in the density estimation step, where we used the fact that not every instance is relevant for the estimation of the density in every point of the grid that holds the densities.

We also introduce an approach for improving projections where the clusters cannot be identified properly, based in the technique proposed by Tejada *et al.* [23]. Such approach can be used interactively by the user in each iteration (projections of the data set) during the clustering process. It was embedded in one of the steps of the clustering algorithm constituting an user option for improving the density map of each projection used during the clustering, before the final density estimation step. For making such technique suitable for large data sets, a mesh is built using the Delaunay triangulation [7] in order to index the nearest neighbors of the current instance and use its neighborhood for calculating the displacement instead of using all instances. Results demonstrate a processing time reduction of 50% to 92%, as well as clustering improvement in some cases and same clustering quality in all the others. Software, code, and documentation are open and available.

In next section we detail the HC-Cooperative algorithm. In section 3 we discuss HC-Enhanced, with highlight on the improvements performed on HC-Cooperative for reducing its processing time and improving the quality of the clustering. In section 4 we present the results on processing time and quality of the clustering generated. Finally, section 5 is devoted to present our conclusions and future work.

2 The HC-Cooperative algorithm

HC-Cooperative is a visual density-based algorithm that employs multiple projections to cluster the data in order to reduce the loss of information and admits user interaction for handling outliers and determining when the process must stop (i.e. all significant clusters were discovered).

The algorithm can be summarized as follows:

---

1. Sample l randomic data points a_i and obtain their r-neighborhood $\mathcal{M}_i$.
2. Normalize the groups $\mathcal{M}_i$ to obtain the groups $\mathcal{O}_i$.
3. Determine a bidimensional subspace $\mathbb{S}$ where the groups $\mathcal{M}_i$ form well defined clusters.
4. Project the data in subspace $\mathbb{S}$.
5. Estimate density in subspace $\mathbb{S}$.
6. Specify a noise level λ where the outliers are not included in clusters creation.
7. Create and store the clusters for current projection.
8. Repeat steps 1 through 7 until all clusters were identified.
9. Create final clusters.

Each of the l $\mathcal{M}_i$ groups, where l is the number of anchors a_i defined by the user, is formed by the instances in the r-neighborhood of a_i (i.e., the result of a range query with center a_i and radius r, where r is user-defined). These groups are normalized such that their centroids z_i become $\overline{0}$. This is achieved subtracting z_i from all instances in $\mathcal{M}_i$ generating a normalized group $\mathcal{O}_i$.

The bidimensional subspace $\mathbb{S}$ where $\mathcal{M}_i$ form well defined groups is that where the moment of all $\mathcal{O}_i$ is minimal. That is, the subspace where $\sum_{i=1}^{l} \mu(\mathcal{O}_i)$ is minimal, where $\mu(\mathcal{O}_i)$ is the moment of $\mathcal{O}_i$ defined by the expression $\mu(\mathcal{O}_i) = \sum_{j=1}^{k} d(z_i', m_j)$. $d(z_i', m_j)$ is the distance between the centroid z_i' of $\mathcal{O}_i$ and the instance (m_j). This is equal to minimizing $\mu(\cup_{i=1}^{l} \mathcal{O}_i)$.

It was proven by Jolliffe [15] that the d-dimensional subspace where a group of data points $\mathcal{G}$ has the minimum moment is given by the d eigenvectors associated to the d smaller eigenvalues of the covariance matrix of $\mathcal{G}$. That is, the d smaller components resulting from applying PCA (Principal Components Analysis) [15] to the set $\cup_{i=1}^{l} \mathcal{O}_i$[3].

HC-Cooperative uses this result to find the subspace $\mathbb{S}$, incrementally decreasing by a half the dimensionality of the data set.

Once the subspace $\mathbb{S}$ has been determined, the data set is projected onto it, and the density $\hat{f}_D$ is estimated using a Gaussian kernel estimator:

$$\hat{f}_D = \frac{1}{nh^d} \sum_{i}^{n} \frac{1}{\sqrt{2\pi}h} e^{-\frac{(x-x_i)^2}{2h^2}} \tag{1}$$

where x_i is the i^{th} data point, h is the smoothing factor, n the number of instances in the data set, and x represents a point in the bidimensional Cartesian grid where the density is stored.

The density estimation is mapped into a heightmap presented to the user. The user then specifies a value for the noise level λ such that the outliers will not be included in the clusters creation for the current iteration (projection). This is done using interaction techniques such as cutting planes (figure 1). With λ established, the clusters are created searching for the grid cells that have at least three of its vertices over the noise level. These grid cells form connected components, each one representing the interior of a cluster. Each cluster is assigned with an identifier that will be added to the *IDString* of each instance belonging to that cluster. The *IDString* of an instance stores the identifiers of the clusters to which the instance belongs in the different iterations (projections). This can be seen in table 2.

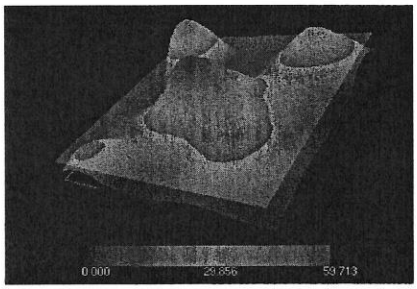


Figure 1: Cutting plane over a heightmap representing the density estimation for specifying noise level λ.

	Projection				
Instance	I	II	III	IV	V
1	1	3	2	1	2
2	2	1	1	3	3
3	3	1	3	2	1
4	1	3	2	1	2
5	1	3	2	2	2

Table 2: Example of *IDStrings*. The element in each cell of the table identifies the cluster to which the instance belongs in that projection. It is highly likely that instances 1, 4, and 5 belong to the same cluster in the original multi-dimensional space.

Once the user decides (based on the visualization) that all the clusters could have been identified, the final clusters are created searching for substrings. Each substring has in its i^{th} position the identifier

[3] In our case $\mathcal{G} = \cup_{i=1}^{l} \mathcal{O}_i$.

of a cluster created in the i^{th} projection, or "$*$", where "$*$" means that this position is not considered when searching for the instances that meet the substring. Thus, each substring represents a final cluster. For example, for the case of table 2, the substring "132*2" defines the cluster formed by instances 1,4, and 5, whilst the substring "13213" defines the cluster formed by instances 1 and 4. This provides a means for measuring the degree of consistency of a cluster with the responses of the user. This metric is called the *interest ratio* which is given by the expression:

$$IR(S) = \theta/(\beta_1 \cdot \beta_2 \cdot \beta_3 ... \beta_l) \tag{2}$$

where β_i is the fractionary support of the cluster m_i in the substring $S = m_1...m_l$ (i.e., the fraction of $IDStrings$ for which the i^{th} position is equal to m_i) and θ is the support of S (i.e., the number of $IDStrings$ that meet S, where an $IDString$ meets a substring $S = m_1...m_l$ if they are the same except in the positions where $m_i = *$).

An interest ratio above 1 indicates that the cluster reveals significant affinity between its instances, based on user's behavior.

The next section presents the changes performed to this algorithm, in order to seek improvement of processing time and of clustering quality.

3 The HC-Enhanced algorithm

The improvements performed on the HC-Cooperative algorithm lead to the HC-Enhanced[4] version which outperforms the original algorithm in terms of processing time without compromising the quality of the clustering obtained. The improvements were directed to three main processes: the range queries performed for forming the groups $\mathcal{M}_i$ (step 1), the searching for subspace $\mathbb{S}$ (step 3), and the density estimation (step 5) which is the key point of the improvement in scalability obtained. Within step 3 we also introduce a mechanism for improving projections in cases where the groups cannot be identified clearly.

3.1 Step 1 - Range queries with Omni-sequential

The Omni-sequential is part of a family of data structures proposed by Santos *et al.* [10] that exploits the fact that the intrinsic dimensionality[5] of real data sets is usually considerably low.

The Omni-concept is the base of such family, which basically is a coordinate transformation to the so called Omni-foci base. This base is a set $\mathcal{F} = \{f_1, f_2, ..., f_k\}$ of instances belonging to the metric data set to be indexed called foci or focal points. Using this base, the Omni-coordinates are calculated for each instance. The Omni-coordinates of an instance s are the distances between s and each f_i. The data set is then indexed using such coordinates, maintaining the original data in a different file. The number of foci can be estimated as the fractal dimensionality, which is an approximation to the intrinsic dimensionality of the data set. A linear algorithm for calculating the fractal dimension of a data set can be found in the work by Traina *et al.* [16].

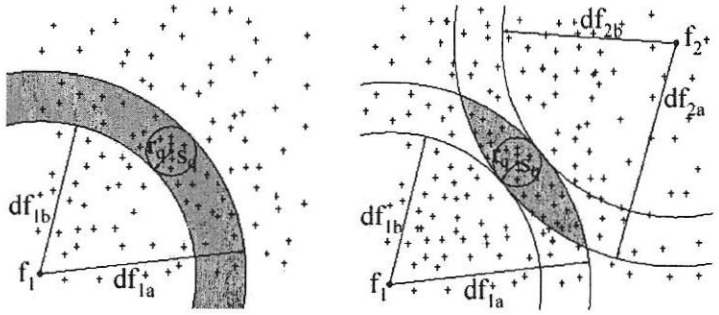
The result of using the Omni-concept is twofold. First, the indexing benefits from this "dimensionality reduction" allowing the data structure to index high dimensional data sets without suffering from the "dimensionality curse". Second, when a range query is performed, a considerable amount of distance calculations is pruned as shown in figure 2.

When a range query with center in s_q and radius r_q is performed, the distance df_i from each focus to the query point is calculated. Then, a range is set for the focus, defining a region formed by the intersection of two balls with center in f_i and radii $df_ia = df_i + r_q$ and $df_ib = df_i - r_q$ respectively, as shown in figure 2a. The intersection of the region formed by all foci defines the *minimum bounding Omni region* (*mbOr*). The instances belonging to the *mbOr* belong to the partial result of the range query. Thus, only the distances between these instances and s_q must be calculated.

It is possible to use this concept together with any data structure. However, the gains of using a data structure like the R*-Tree [4] over a sequential approach are negligible when few queries are performed [10], as in our case. Thus, we used the sequential version (Omni-sequential) in the first step of the algorithm in order to accelerate the range queries performed to generate the groups $\mathcal{M}_i$.

[4]Available at http://www.lcad.icmc.usp.br/~powervis/visual/.

[5]The dimensionality of the space actually occupied by the data (a line in $\mathbb{R}^n$ has intrinsic dimensionality equal to 1).



(a) (b)

Figure 2: A range query with center s_q and radius r_q. The shadow illustrates the instances returned from the pruning. (a) Using one focus f_1. (b) Using two foci f_1 and f_2.

3.2 Step 3 - Finding and improving the polarization subspace

In the step 3 of the HC-Cooperative algorithm, a subspace $\mathbb{S}$ (the *polarization subspace*) where $\mathcal{M}_i$ form well defined groups must be found. In order to do that, as we discussed in section 2, the dimensionality of the data set is incrementally decreased by half, projecting the data set in the d_i-dimensional subspace defined by the d_i eigenvectors associated to the d_i smaller eigenvalues of the covariance matrix of the data set, where d_i is half the current dimensionality. This is the same of applying PCA [15] to the set $\cup_{i=1}^{l}\mathcal{O}_i$[6] and choosing the last d_i components as the base of the d_i-dimensional subspace.

Mathematically speaking, eigendecomposition or PCA, reduces to solving the equation $RW = W\Lambda$ where R is any real square matrix [12], as the covariance matrix of $\cup_{i=1}^{l}\mathcal{O}_i$. Since PCA is a costly process, we experiment using online-PCA [20] and Simple-PCA [19] for finding subspace $\mathbb{S}$. Both of these methods are based in the fact that an eigenvector w is related to its rotated version by R as stated in the equation:

$$w = \frac{Rw}{w^T Rw} \tag{3}$$

Then, it is possible to establish an iterative process as:

$$w(n) = \frac{R(n)w(n-1)}{w^T(n-1)R(n)w(n-1)} \tag{4}$$

where $R(n)$ is the estimate of the covariance matrix at the n^{th} time step.

The minor components are calculated using the deflation technique, subtracting the previous eigencomponents from the original signal x (data) as in the expression:

$$x_i' = x_i - (w^T x_i)w \tag{5}$$

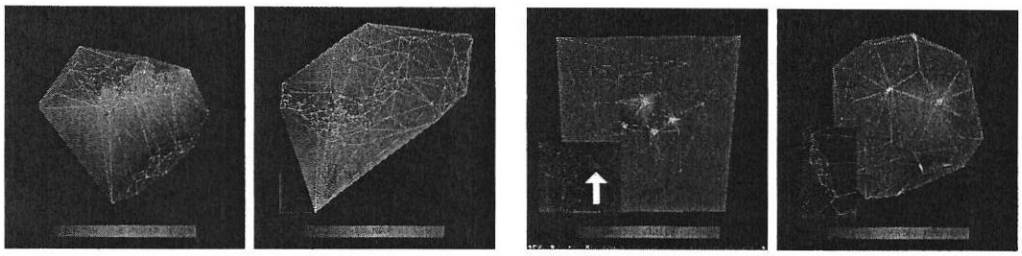
where x_i is the i^{th} sample of signal x (the i^{th} instance).

From the results we could notice that due to the fact that these algorithms need previous eigenvectors to calculate the next ones, and since we always need the minor components, the time spent by them is higher than the time consumed by the original PCA. Moreover, since the PCA is performed over a sample of the data set ($\cup_{i=1}^{l}\mathcal{O}_i$), the time spent by PCA is negligible compared with the time spent by other process. Because of that, we concentrate our efforts in devising a way to decrease the processing time of the density estimation step (see next subsection), and improving the projections where the clusters are not well defined, as described in the following text.

In some polarization subspaces the groups could not be identified clearly since it is not ensured that for every set of anchors a_i and every radius r defining the initial groups $\mathcal{M}_i$, the polarization subspace obtained from them is one where the clusters in the original multi-dimensional space will form well defined groups. Thus, we proposed an approach for improving such projections [23]. The approach uses

[6] As mentioned in section 2, $\mathcal{O}_i$ is the normalized version of $\mathcal{M}_i$ around $\overline{0}$

the fact that the relation between the distance in the original space and the bidimensional subspace must be constant for every pair of data points (x_i, x_j). In figure 3 we show that this will generate a better projection when applied to the results of techniques such as Fastmap [9] or PCA [15]. The technique, called the Force scheme, is presented below.



(a) (b)

Figure 3: Improvement of the Fastmap projection of the (a) Iris data set and (b) a 10-dimensional synthetic data set with 1000 instances, using the approach proposed. In both cases the color maps the cluster to which each data point belongs. In (b) we show a close-up of one of the clusters. An instance classified as noise (pointed by the arrow) is part of this cluster originally. In the improved projection the instance is removed from the cluster.

From an instance x_i, we calculate the vectors $\overrightarrow{v_{ij}}$ to each instance x_j in its neighborhood. Then, we apply a perturbation to each point in the neighborhood in the direction of the vector calculated for that data point. In order to make this process suitable for very large data sets, we modified the original approach using the Delaunay triangulation [7] for retrieving the neighborhood of a projected data point as being the star of the corresponding point inserted in the Delaunay triangulation.

The modified Force scheme, for each iteration, can be summarized as follows:

---

1. Calculate the Delaunay triangulation for the current projection.
2. For each projected data point p.
   2.1. Get the star $\mathcal{S}$ of p.
   2.2. For each point s in $\mathcal{S}$.
      2.2.1. Calculate $\overrightarrow{v}$ being the vector from p to s.
      2.2.2. Move s in the direction of $\overrightarrow{v}$ a fraction of Δ.
3. Normalize the projection to the range [0,1].

Here the error Δ is given by:

$$\Delta = \frac{d(p^n, s^n) - dmin}{dmax - dmin} - d_2(p, s) \tag{6}$$

where $d_2(p, s)$ is the distance between p and s in the bidimensional projected subspace, $d(p^n, s^n)$ is the distance between the respective points p^n and s^n in the n-dimensional original space, and $dmax$ and $dmin$ are the maximum and minimum distances respectively in the n-dimensional space.

This process tries to converge to an optimal projection. Since the normalization of the distances is a costly process, it cannot be performed in each iteration for the projected space. This is why the distances are normalized once for the original space, and we perform a normalization of the projected instances instead of the projected distances in each iteration. Thus, the process actually generates an approximation of the optimal case.

In order to illustrate the results of this process, figure 4 shows some iterations for the projection of a synthetic data set of 15 5-dimensional instances. We can notice that the last iteration presents the data better grouped.

We can approximate the relative error $Q(P)$ of the projection P obtained in each iteration as in the

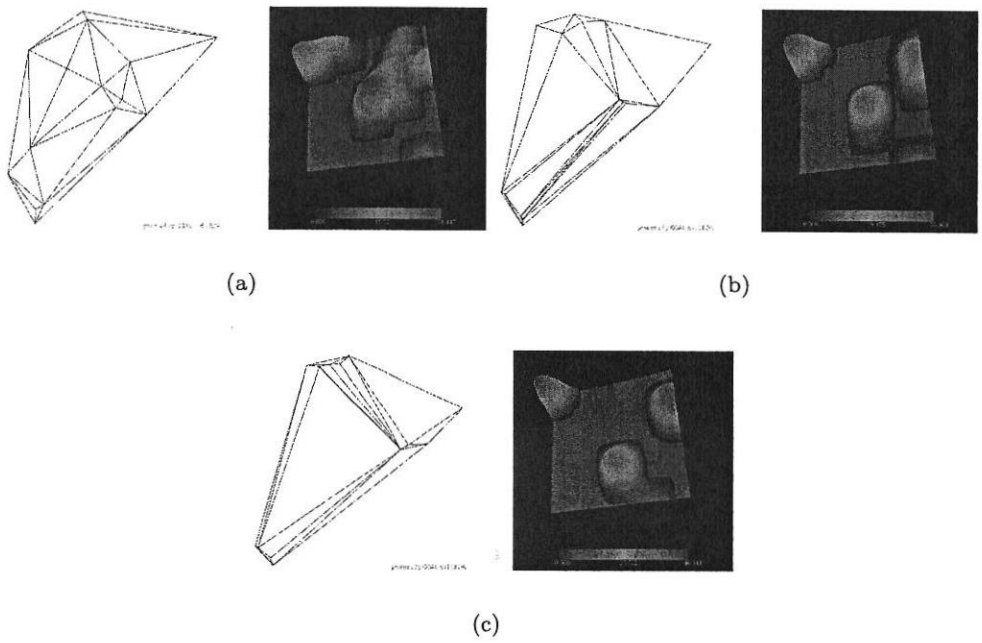


Figure 4: Improvement of a projection of a synthetic dataset of 15 5-dimensional instances. (a) Iteration 1. (b) Iteration 13. (c) Iteration 25.

expression:

$$Q(P) = \frac{1}{2M} \sum_{k=1}^{M} |\Delta_k| \tag{7}$$

Notice that the error given by $Q(P)$ is a relative error useful for comparing different projections of the same data set. However, since equation 6 defines an approximate displacement, $Q(P)$ over-estimates the projection error. This is, the optimal projection P' of a data set has an error $Q(P') \geq 0$.

This approach can be used interactively by the user over the projection generated in each iteration (within step 3 of the algorithm), generating a new projection and, thus, a new density map. The density map is generated with an improved process in relation to the original approach of Aggarwal, as described in next subsection.

3.3 Step 5 - Density estimation

This step is the critic point of the algorithm. Every previous steps of the algorithm has complexity $O(kn)$, where k is a constant and n the number of instances. However, contrary to all other steps, in the density estimation k is a high constant (the number of grid points that will hold the density).

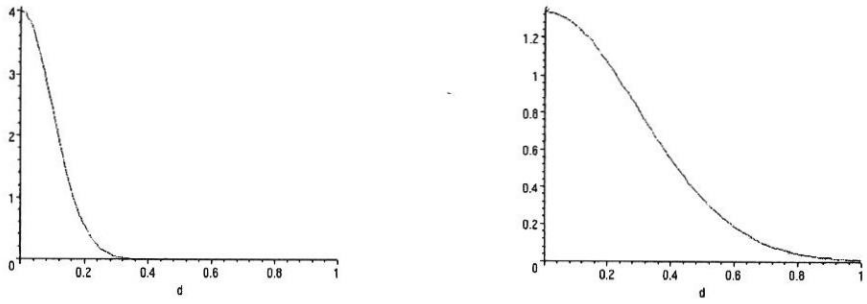
We propose an approach for reducing the constant k based on the individual contribution of each instance to the density accumulated in the grid points.

In figure 5 we present the contribution $\hat{f}_D^i$ of an instance x_i in the calculus of the density estimation in a grid point x as a function of the distance between x and x_i for different values of the smoothing factor h. Such contribution is given by the expression:

$$\hat{f}_D^i = \frac{1}{\sqrt{2\pi}h} e^{-\frac{(x-x_i)^2}{2h^2}} \tag{8}$$

We can see that after some distance, and depending of the smoothing factor, the contribution of instance x_i to the calculus of the density accumulated in the grid point could be neglected.

Due to the fact that the data set is normalized to the range $[0, 1]$ in all dimensions during the projection, the smoothing factor h is between 0 and 1, being that for the data sets tested the optimal h factor from the formula presented by Silverman [?] is about 0.1. We can see that, for the case of the optimal value of h (0.1), the contribution of an instance to the calculus of the density accumulated in a grid point decreases rapidly as the distance increases.



(a) (b)

Figure 5: Contribution of an instance x_i in the calculus of the density accumulated in a grid point x as function of the distance d between x and x_i for (a) $h = 0,1$ and (b) $h = 0,3$.

Thus, we define a neighborhood v of the grid cells for which an instance will be considered in the calculus of the density accumulated in them. In figure 6 we show an example for neighborhood $v = 2$ of an instance of the grid 9×9.

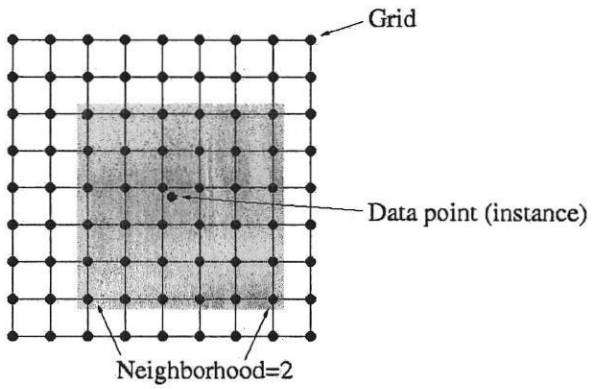


Figure 6: Neighborhood of an instance.

The approach proposed for calculating the density estimation is as follows.

For each instance x_i with bidimensional coordinates x_{ix} and x_{iy}.

1. $index_x = \lfloor x_{ix}.m \rfloor$.
2. $index_y = \lfloor x_{iy}.m \rfloor$.
3. Check for boundary conditions ($index_x + v + 1 > m$, $index_x - v < 0$, $index_y + v + 1 > m$, or $index_y - v < 0$).
4. Accumulate the contribution of x_i for all grid points with index in the range $[index_x - v, index_x + v + 1; index_y - v, index_y + v + 1]$

Here m indicates the number of samples of the grid (i.e., for a grid 30×30, $m = 30$) and v is the neighborhood.

The original approach, presented here for comparison, is as follows.

For eachgrid point x
    For each instance x_i
        Accumulate the contribution of x_i for x.

The optimal neighborhood depends on the number of samples m of the grid, as well as in the precision p required and the smoothing factor h. The precision indicates the minimum contribution

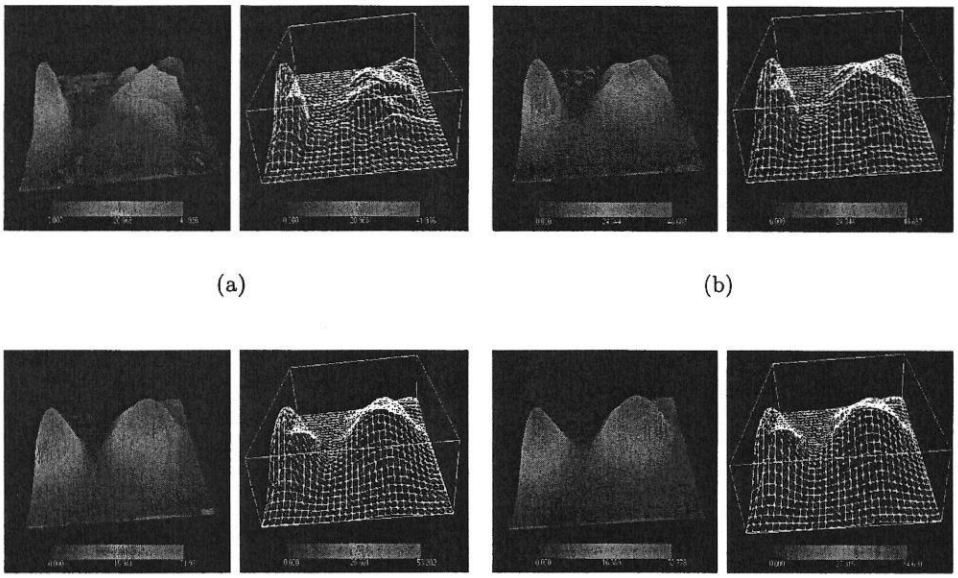
that an instance must provide to the calculus of the density accumulated in a grid point for it to be considered relevant for such grid point.

Such relation is given by the expression:

$$v = m\sqrt{-2h^2 \ln(p\sqrt{2\pi}h)} \tag{9}$$

directly derived from equation 8.

Figure 7 shows the results of the calculus of the density estimation for different neighborhoods v for the Iris data set over a grid 30×30. We can see that with a neighborhood of $v = 5$ (the optimal neighborhood is $v = 5$ for a precision of 0.008) the density estimation approximates the density estimated using all grid points as the neighborhood (original approach).



(a) (b)

(c) (d)

Figure 7: Density estimation for the Iris data set over a grid 30×30 with neighborhood (a) $v = 2$, (b) $v = 3$, (c) $v = 5$, and (d) $v = 30$.

In the next section we present the results od applying both the HC-Cooperative and the HC-Enhanced algorithms to different data sets.

4 Results

In figure 8 we present a comparison between the results obtained with the original HC-Cooperative algorithm and those obtained with the modified version HC-Enhanced in terms of processing time for different sizes of the data set. The tests were performed on a PC computer with a AMD Athlon 1.0 GHz processor and 1.0 GB of memory.

We can see a considerable gain in processing time. These results where obtained without loss in the quality of the clustering. Thus HC-Enhanced is a more interactive and scalable version of the HC-Cooperative algorithm, which was the main goal of this work.

The algorithms were also applied to the Iris and IDH data sets, as well as two synthetic data sets of 1000 10-dimensional and 10000 50-dimensional instances respectively, and to a quadrapeds data set which are described below [7].

The Iris data set is formed by 150 4-dimensional instances of three classes of plants classified according to the sepal length and width, and the petal length and width. This data set is usually separated by the clustering algorithms in two clusters. One of them formed by the instances of the first class of plants

[7] All data sets, with the exception of Synt10d and Synt50d, were taken from the Machine Learning Repository of the University of California, Irvine (http://www.ics.uci.edu/~mlearn/MLRepository).

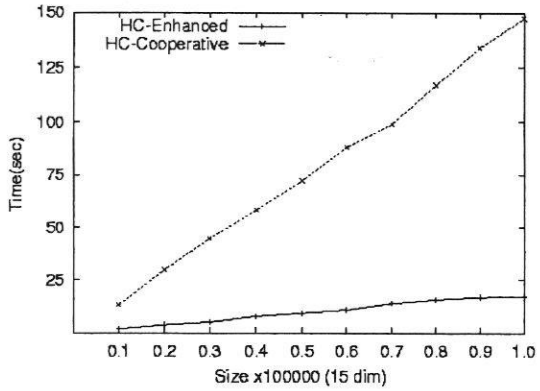


Figure 8: Performance comparison between HC-Cooperative and HC-Enhanced with data sets of different sizes and 15 dimensions.

and the other by the two remaining classes. Using the HC-Cooperative and HC-Enhanced algorithms the first cluster is formed by the 92% of the instances of the first class, whilst the second cluster is formed by the 85% of the instances of the second and third classes. It is important to notice that there were no false positives in the clusters. This result is usually obtained with most clustering algorithms. Besides these two clusters, the HC algorithms return a cluster formed by 86% of the instances of the second class of plants. Thus, it is possible to infer that there exist three classes of plants in the Iris data set and that two of them are very similar. This result cannot be obtained with another clustering algorithm. The HC algorithms are capable of presenting such result due to the nature of the creation of the final clusters process based on multiple projections.

The processing times of the HC-Cooperative and HC-Enhanced algorithms for the Iris data set were $0,97$s and $0,14$s respectively, for a 40×40 grid using a neighborhood $v = 6$. The maximum densities were $32,648$ for HC-Cooperative, and $31,502$ for HC-Enhanced.

The IDH data set stores data of 173 countries relative to their quality of life. This data set has 9 dimensions plus the ranking of each country, which was not included during the tests. The HC-Cooperative and HC-Enhanced algorithms separated this data set in two clusters. The first one was formed of the first 25 instances in the ranking. The second cluster was formed by the instances 48 to 173. The remaining instances $(26-47)$ were classified as noise due to the fact that they rely between the two clusters. From the Delaunay triangulation of the Fastmap projection [9] shown in figure 9 we can notice that this result is in accordance with the distribution of the data. There was one false positive in the first cluster and 4 and 9 false negatives in the first and second clusters respectively.

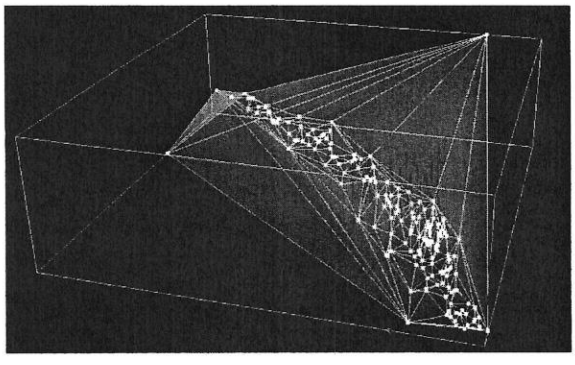


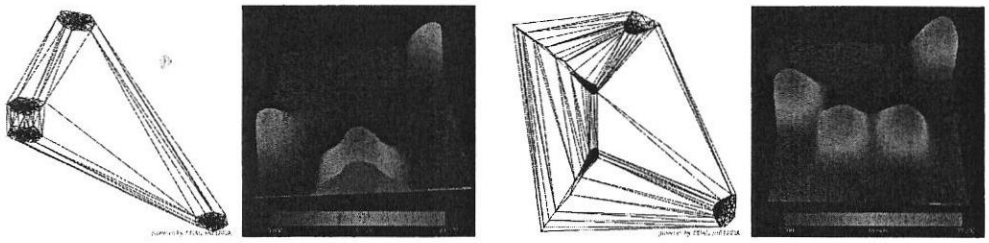
Figure 9: Delaunay triangulation of the Fastmap projection of the IDH data set. The height and color map the ranking of the country, being that the best ranked is mapped as blue-low and the worst ranked as red-high.

The time spent by the algorithms for the IDH data set was $1,1$s (HC-Cooperative) and $0,14$s (HC-Enhanced), for a 40×40 grid using a neighborhood $v = 6$. The maximum densities were $54,054$ and $51,615$ respectively.

The first synthetic data set, Synt10d, was created with 1000 10-dimensional instances forming 4

groups, two of them considerably close to each other, and 20% of noise. Applying HC-Cooperative and HC-Enhanced to this data set led to the following results: four clusters were obtained, the first of them with 88% of the original instances of the first synthetic group, the second with 100% of the instances of the second synthetic group, the third with 88% of the instances of the third synthetic group, and the fourth with 100% of the instances of the last synthetic group, none presenting false positives. The second and third groups presented false negative because they were too close and the instances in the boundaries were assumed to be noise. Normally, only one cluster would be generated from these two groups.

This result was improved using the Force scheme presented in subsection 3.2. The resulting clustering presented 98% (96%, 100%, 96%, 100%, for groups 1 to 4 respectively) of accuracy on average after improving the projection using such scheme. We can see in figure 10a the four groups with two of them too close to each other. In figure 10b we show the improved projection, where those groups were separated.



(a) (b)

Figure 10: Delaunay triangulation and density estimation calculated over the (a) original and (b) improved projections of a synthetic data set of 1000 10-dimensional instances used for testing the algorithms.

The processing time for this data set was 6,42s (HC-Cooperative) and 0,93s (HC-Enhanced), for a 40×40 grid using a neighborhood $v = 6$. The maximum densities were 45,601 and 43,660 respectively.

A second synthetic data set, Synt50d, of 10000 instances and 50 dimensions, was created with three groups and tested with both algorithms. The output obtained was three clusters formed by the instances of the three original groups, presenting the following errors: the first cluster had 1 false negative and 4 false positives. The second presented 9 false negatives and 0 false positives. The third cluster had 0 false negatives and 0 false positives. This results in a precision of 99.9985%, 99.9973% and 100% for the three clusters respectively.

The time spent by the algorithms HC-Cooperative and HC-Enhanced was 72,59s and 20,26s respectively, for a 40×40 grid using a neighborhood $v = 6$. The maximum densities were 46,1914 and 46,190 respectively.

Finally, a data set formed by 96487 instances grouped in four classes of quadrapeds (dogs, cats, giraffes, and horses) was also used to test both algorithms. Both algorithms generated four clusters. The first of them was formed by 90.80% of the instances of the first class (dogs). From the 25521 instances included in this cluster, 3480 were false positives corresponding to the second class (cats). The second cluster was formed by 70.24% of the instances from to the second class (cats). The third cluster was formed by 67.92% of the instances corresponding to the third class (horses). Finally, the fourth cluster was formed by 33.86% of the instances from to the third class (giraffes). Neither of the latter three clusters presented false positives. 30.875% of the instances of the original data set were considered noise. This data set is typically difficult to cluster considering that attributes are not always capable of distinguishing animals from different groups. For the same reason, the Force scheme has no effect here, because it has been designed to recover distance information lost in the projection process. For the quadrapeds, the distance itself does not codify animals well.

The time used by the HC-Cooperative algorithm for this last data set was 391,21s, whilst the time used by the HC-Enhanced algorithm was 201,5s, for a 40×40 grid using a neighborhood $v = 6$. The maximum densities were 56,02 and 55,98 respectively.

Tables 3 and 4 summarize these results, presenting a comparison between the algorithms in relation

both to the processing time and the quality of the clustering.

	HC-Cooperative			
Data set	**Average precision**	**Time**	**False positives**	**False negatives**
Iris	87,667	0,97s	0	19
IDH	91,907	1,10s	1	13
Synt10d	94,0	6,42s	0	60
Synt50d	99,9986	72,59s	4	10
Quadrapeds	65,518	391,21s	3480	29670

Table 3: Results of the HC-Cooperative algorithm on different data sets in terms of precision and processing time.

	HC-Enhanced			
Data set	**Average precision**	**Time**	**False positives**	**False negatives**
Iris	87,667	0,14s	0	19
IDH	91,907	0,14s	1	13
Synt10d	98,0	0,93s	0	20
Synt50d	99,9986	20,26s	4	10
Quadrapeds	65,518	201,5s	3480	29670

Table 4: Results of the HC-Enhanced algorithm on different data sets in terms of precision and processing time.

We can notice that the only data set to which it was necessary to apply the projection improvement process was Synt10d. The decision of using such improvement was because of the visual results presented to the user during the process, in which it was possible to see that there were clusters too near to each other that should be separated. The rest of the data sets presented well defined clusters in the original projections or, in the case of the Quadrapeds data set, the metric used was not suitable for separating the two last clusters due to the fact that they were "overlapped" in the Euclidean space. It also affected the performance of the algorithms in terms of processing time, considering that it was the only case where the improvement in the performance was below 70%.

5 Conclusion

We proposed an enhancement of the Aggarwal's algorithm HC-Cooperative, for reducing its processing time, improving the quality of the clustering generated, and allow a better interaction with the user. The advantages of HC-Cooperative are meaningful for the understanding of the user about the data. During the process the user gains a much deeper knowledge about the data and, thus, about the domain from which the data was originated.

HC-Cooperative deals efficiently with outliers and multi-dimensional data, lowering the information lost during dimensionality reduction by performing multi-subspace clustering. The information about the clusters generated in each subspace is combined to create the final clusters, allowing the definition of a metric for the consistency of the clusters with the responses supplied by the user. This metric is totally independent from the application which is an important advantage of this algorithm. However, due to the high computational cost of the processes involved, the algorithm cannot be applied to very large data sets without increasing the response time to prohibitive levels. Since interactiveness is a key issue for visual mining techniques, such as Aggarwal's approach, the response time must be sufficiently low for allowing a continuous feedback between user and computer.

The improvements to this approach presented here for lowering its processing time, led to a version of the HC-Cooperative algorithm with a better degree of scalability called HC-Enhanced. The gain in processing time obtained from these improvements is considerably high. This reduction exceeded 87% in most of cases. There were only two data sets that presented improvements of 50% and 70%. The quality of the clustering generated by the original algorithm was maintained and the interactiveness of the approach was improved.

We also developed an approach for improving the results of quality of clustering when two or more groups are very close to each other in relation to the total domain of the projected subspace. This is

performed relocating the projected data points following an optimization heuristic based in the relation between the distances in the original and the projected spaces.

As future work we can mention the need for devising mechanism for improving the interaction between user and machine in order to increase the interference of the user during the clustering process, which will decrease the response time, allow a better understanding of the data, generate better clusterings, and turn the approach applicable to a wider range of situations. It will be also desirable to allow multiple visualizations and interaction techniques in order to increase the accuracy of the user-driven cluster specification.

The inclusion of fast mechanisms for projecting the data is also necessary for obtaining a larger reduction in the processing time. The application of such mechanisms must be studied considering the original idea of Aggarwal of finding the bidimensional subspaces where the instances concentrate, forming well defined clusters.

It is also important to mention that one of the main difficulties found in the density-based clustering is the treatment of categorical data and non-continuous data. The fact that the bidimensional grid that stores the density needs spatial locations for calculating the density estimation originates an important problem that must be studied. The extension of the HC-Enhanced algorithm for clustering categorical data is a work that should be approached in future work.

6 Acknowledgements

The authors want to acknowledge the finance of CNPq and FAPESP Brazilian financial agencies.

References

[1] C. Aggarwal. A human-computer cooperative system for effective high dimensional clustering. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 221–226, CA, USA, 2001.

[2] C. Aggarwal. Towards meaningful high-dimensional nearest neighbor search by human-computer interaction. In *ICDE'02, International Conference on Data Engineering*, pages 593–604, San Jose, California, Fevereiro 2002. IEEE Press.

[3] C. Aggarwal and P. Yu. Finding generalized projected clusters in high dimensional spaces. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *ACM SIGMOD International Conference on Management of Data*, pages 70–81, Dallas, Texas, USA, 2000. ACM.

[4] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An efficient and robust access method for points and rectangles. In *ACM SIGMOD International. Conference. on Management of Data*, volume 1, pages 322–331, 1990.

[5] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[6] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? *Lecture Notes in Computer Science*, 1540:217–235, 1999.

[7] Herbert Edelsbrunner. *Geometry and Topology for Mesh Generation.* Cambridge Monographs on Applied and Computational Mathematics. Cambridge, 2001.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han, and U. Fayyad, editors, *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–331, Portland, Oregon, 1996. AAAI Press.

[9] C. Faloutsos and K. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia databases. In Michael J. Carey and Donovan A. Schneider, editors, *ACM SIGMOD'95 International Conference on Management of Data*, pages 163–174, San Jose, California, 1995. ACM.

[10] R. F. Santos Filho, A.J.M. Traina, C. Traina Jr, and C. Faloutsos. Similarity search without tears: The omni-family of all-purpose access methods. In *17th IEEE International Conference on Data Engineering (ICDE)*, pages 623–630, Heidelberg, Germany, 2001.

[11] V. Ganti, M Ramakrishnan, J. Gehrke, A. Powell, and J. French. Clustering large datasets in arbitrary metric spaces. Technical report, Department of Computer Science, University of Wisconsin at Madison, Madison, Wisconsin, USA, 1998.

[12] G. H. Golub and C.F. Van Loan. *Matrix Computation.* Then John Hopkins University Press, 1991.

[13] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: An efficient clustering algorithm for large databases. In Laura M. Haas and Ashutosh Tiwary, editors, *ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pages 73–84, Seattle, Washington, USA, 1998. ACM Press.

[14] A. Hinneburg, D. Keim, and M. Wawryniuk. HD-eye: Visual mining of high himensional hata. *IEEE Computer Graphics and Applications*, 19(5):22–31, Setembro/Outubro 1999.

[15] I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, 1986.

[16] C. Traina Jr., A.J.M. Traina, Leejay Wu, and C. Faloutsos. Fast feature selection using fractal dimension. In *Simpósio Brasileiro de Banco de Dados - SBBD'00*, João Pessoa-PA, Brazil, 2000.

[17] G. Karypis, E. Han, and V. Kumar. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. *COMPUTER*, 32:68–75, 1999.

[18] B.-U Pagel, F. Korn, and C. Faloutsos. Deflating the dimensionality curse using multiple fractal dimensions. In *International Conference on Data Engineering - ICDE'00*, pages 589–598, San Diego-CA, USA, 2000.

[19] M. Partridge and R.A. Calvo. Fast dimensionality reduction and simple pca. *Intelligent Data Analysis*, 2(3), 1998.

[20] Y. N. Rao and J. C. Principe. A fast, on-line algorithm for pca and its convergence characteristics. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 299–308. IEEE Press, 2000.

[21] W. Ribarsky, J. Katz, F. Jiang, and A. Holland. Discovery visualization using fast clustering. *IEEE Computer Graphics and Applications*, 19(5):32–39, Setembro/Outubro 1999.

[22] G. Strang. *Linear Algebra and its Applications.* Academic Press, 1980.

[23] E. Tejada, R. Minghim, and L. G. Nonato. On improved projection techniques to support visual exploration of multi-dimensional data sets. *Submitted to Information Visualization Journal, Special Issue on Coordinated & Multiple Views in Exploratory Visualization*, 2(4), 2003.

[24] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999.

[25] P. Chung Wong. Visual data mining. *IEEE Computer Graphics and Applications*, 19(5):20–21, Setembro/Outubro 1999.

[26] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD'96 International Conference on Management of Data*, pages 103–114, New York, USA, 1996. ACM.