CLUSTERING IN MASSIVE DATA SETS

FIONN MURTAGH

School of Computer Science, The Queen's University of Belfast, Belfast BT7 1NN, Northern Ireland.
E-mail: f.murtagh@qub.ac.uk

Received: 25th May 2000 / Published 11th May 2001

ABSTRACT

We review the time and storage costs of search and clustering algorithms. We exemplify these, based on case studies in astronomy, information retrieval, visual user interfaces, chemical databases, and other areas. First we describe nearest neighbor searching, an elemental form of clustering, and a basis for clustering algorithms to follow. Next we review a number of families of clustering algorithms. Finally we discuss visual or image representations of data sets, from which a number of interesting algorithmic developments arise.

'Now', said Rabbit, 'this is a search, and I've organised it - '
'Done what to it?' said Pooh.
'Organised it. Which means - well, it's what you do to a Search,
when you don't all look in the same place at once.'
A.A. Milne, The House at Pooh Corner (1928) –M. S. Zakaria

INTRODUCTION

Nearest neighbor searching is considered first for one main reason: its utility for the clustering algorithms reviewed later. They are the building blocks for the most efficient implementations of hierarchical clustering algorithms, and they can be used to speed up other families of clustering algorithms. We will then deal with facets of visual or image representations of data sets.

The best match or nearest neighbor problem is important in many disciplines. In statistics, *k*-nearest neighbors, where *k* can be 1, 2, *etc.*, is a method of non-parametric discriminant analysis. In pattern recognition, this is a widely used method for unsupervised classification (see [1]).

Nearest neighbor algorithms are the building block of clustering algorithms based on nearest neighbor chains; or of effective heuristic solutions for combinatorial optimization algorithms such as the traveling salesman problem, which is a paradigmatic problem in many areas. In the database and more particularly data mining fields, NN searching is called similarity query, or similarity join. [2]

In the next section, we begin with data structures where the objective is to break the O(n) barrier for determining the nearest neighbor (NN) of a point. A database record or tuple may be taken as a point in a space of dimensionality m, the latter being the associated number of fields or attributes. These approaches have been very successful, but they are restricted to low dimensional NN-searching. For higher dimensional data, a wide range of bounding approaches have been proposed, which remain O(n) algorithms but with a low constant of proportionality.

We assume familiarity with basic notions of similarity and distance, the triangular inequality, ultrametric spaces, Jaccard and other coefficients, normalization and standardization. For an implicit treatment of data theory and data coding, see [3]. Useful background reading can be found in [4]. In

particular output representational models include discrete structures, *e.g.* rooted labeled trees or dendrograms, and spatial structures, [5] with many hybrids.

BINNING OR BUCKETING

In this approach to NN-searching, a preprocessing stage precedes the searching stage. All points are mapped onto indexed cellular regions of space, so that NNs are found in the same or in closely adjacent cells. Taking the plane as as example, and considering points (x_i, y_i), the maximum and minimum values on all coordinates are obtained (*e.g.* (x_j^{min}, y_j^{min})). Consider the mapping (Fig. 1)

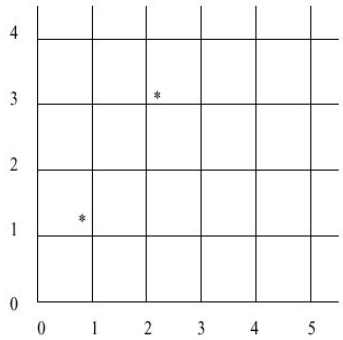
$$x_i \rightarrow \left\{ \left\lfloor \left(x_{ij} - x_j^{min} \right) \Big/ r \right\rfloor \right\}$$

where constant r is chosen in terms of the number of equally spaced categories into which the interval $[x_j^{min}, x_j^{max}]$ is to be divided. This gives to x_i an integer value between 0 and $\lfloor (x_{ij}^{max} - x_{ij}^{min})/r \rfloor$ for each attribute j. $O(nm)$ time is required to obtain the transformation of all n points, and the result may be stored as a linked list with a pointer from each cell identifier to the set of points mapped onto that cell.

NN-searching begins by finding the closest point among those that have been mapped onto the same grid cell as the target point. This gives a current NN point. A closer point may be mapped onto some other grid cell if the distance between target point and current NN point is greater than the distance between the target point and any of the boundaries of the cell containing it. Some further implementation details can be found in [6].

A powerful theoretical result regarding this approach is as follows. For uniformly distributed points, the NN of a point is found in $O(1)$, or constant, expected time (see [7] or [8] for proof). Therefore this approach will work well if approximate uniformity can be assumed or if the data can be broken down into regions of approximately uniformly distributed points.

$$x^{min}, y^{min} = 0,0,\ x^{max}, y^{max} = 50,40,\ r = 10$$



Point (22,32) is mapped onto cell (2,3); point (8,13) is mapped onto cell (0,1).

Figure 1: Example of simple binning in the plane.

Simple Fortran code for this approach is listed, and discussed, in [9]. The search through adjacent cells requires time that increases exponentially with dimensionality (if it is assumed that the number of points assigned to each cell is approximately equal). As a result, this approach is suitable for low dimensions only. Rohlf [10] reports on work in dimensions 2, 3, and 4; and Murtagh [11] in the plane. Rohlf also mentions the use of the first 3 principal components to approximate a set of points in 15-dimensional space.

From the constant expected time NN search result, particular hierarchical agglomerative clustering methods can be shown to be of linear expected time, $O(n)$. [11] The expected time complexity for Ward's minimum variance method is given as $O(n \log n)$. Results on the hierarchical clustering of up to 12,000 points are discussed.

The limitation on these very appealing computational complexity results is that they are only really feasible for data in the plane. Bellman's curse of dimensionality manifests itself here as always. For dimensions greater than 2 or 3 we proceed to the situation where a binary search tree can provide us with a good preprocessing of our data.

Multidimensional Binary Search or kD Tree

A binary search tree preprocesses the data to be searched through by two-way subdivision, and subdivisions continue until some prespecified number of data points is arrived at. See example in Fig. 2. We associate with each node of the decision tree the definition of a subdivision of the data only, and we associate with each terminal node a pointer to the stored coordinates of the points. Using the approximate median of projections keeps the tree balanced, and consequently $O(\log n)$ levels, at each of which $O(n)$ processing is required. Hence the construction of the tree takes $O(n \log n)$ time.

The search for a NN then proceeds by a top-down traversal of the tree. The target point is transmitted through successive levels of the tree using the defined separation of the two child nodes at each node. On arrival at a terminal node, all associated points are examined and a current NN selected. The tree is then backtracked: if the points associated with any node could furnish a closer point, then subnodes must be checked out.

The approximately constant number of points associated with terminal nodes (hyper-rectangular cells in the space of points) should be greater than 1 in order that some NNs may be obtained without requiring a search of adjacent cells (other terminal nodes). Friedman *et al*. [12] suggest a value of the number of points per bin between 4 and 32 based on empirical study.

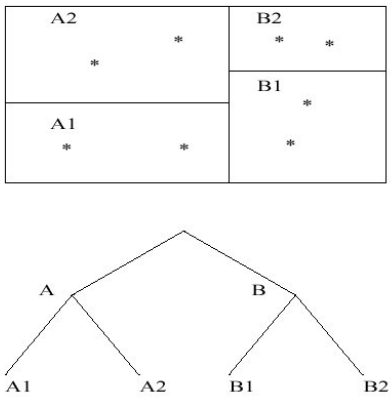


Figure 2: A MDBST using planar data

The MDBST approach only works well with small dimensions. To see this, consider each coordinate being used once and once only for the subdivision of points, *i.e.* each attribute is considered equally useful. Let there be p levels in the tree, *i.e.* 2^p terminal nodes. Each terminal node contains approximately c points by construction and so $c2^p = n$. Therefore $p = \log_2 n/c$. As sample values, if $n = 32768$; $c = 32$; then $p = 10$. That is in 10-dimensional space, using a large number of points associated with terminal nodes, more than 30000 points will need to be considered. For high dimensional spaces, two alternative MDBST specifications are as follows.

All attributes need not be considered for splitting the data if it is known that some are of greater interest than others. Linearity present in the data may manifest itself *via* the variance of projections of points on the coordinates; choosing the coordinate with greatest variance as the discriminator coordinate at each node may therefore allow repeated use of certain attributes. This has the added effect that the hyper-rectangular cells into which the terminal nodes divide the space will be approximately cubic in shape. In this case, Friedman *et al*. [12] show that search time is $O(\log n)$ on average for the finding of a NN. Results

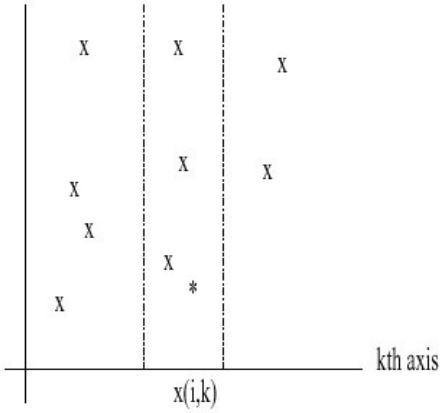


Figure 3: Two-dimensional example of projection-based bound. Points with projections within distance c of given point's (*) projection, alone, are searched. Distance c is defined with reference to a candidate or current nearest neighbor.

obtained for dimensionalities of between 2 and 8 are reported in [12], and in the application of this approach to minimal spanning tree construction in Bentley and Friedman. [13] LISP code for the MDBST is discussed in [14].

The MDBST has also been proposed for very high dimensionality spaces, *i.e.* where the dimensionality may be greater than the number of points, as could be the case in a keyword-based system. Keywords (coordinates) are batched, and the following decision rule is used: if some one of a given batch of node-defining discriminating attributes is present, then take the left subtree, else take the right subtree. Large n, well in excess of 1400, was stated as necessary for good results. [15, 16] General guidelines for the attributes that define the direction of search at each level are that they be related, and the number chosen should keep the tree balanced. On intuitive grounds, our opinion is that this approach will work well if the clusters of attributes, defining the tree nodes, are mutually well separated.

An MDBST approach is used by Moore [17] in the case of Gaussian mixture clustering. Over and above the search for nearest neighbors based on Euclidean distance, Moore allows for the Mahalanobis metric, *i.e.* distance to cluster centers that are "corrected" for the (Gaussian) spread or morphology of clusters. The information stored at each node of the tree includes covariances.

Moore [17] reports results on numbers of objects of around 160,000, dimensionalities of between 2 and 6, and speedups of 8-fold to 1000-fold. Pelleg and Moore [18] discuss results on some 430,000 two-dimensional objects from the Sloan Digital Sky Survey (see the section "k-Means and Family" below).

PROJECTIONS AND OTHER BOUNDS

Bounding using Projection or Properties of Metrics

Making use of bounds is a versatile approach, which may be less restricted by dimensionality. Some lower bound on the dissimilarity is efficiently calculated in order to dispense with the full calculation in many instances.

Using projections on a coordinate axis allows the exclusion of points in the search for the NN of point x_i. Points x_k, only, are considered such that $(x_{ij} - x_{kj})^2 \leq c^2$ where x_{ij} is the j^{th} coordinate of x_i, and where c is some prespecified distance (see Fig. 3).

Alternatively, more than one coordinate may be used. The prior sorting of coordinate values on the chosen axis or axes expedites the finding of points whose full distance calculation is necessitated. The preprocessing required with this approach involves the sorting of up to m sets of coordinates, *i.e.* $O(mn \log n)$ time.

Using one axis, it is evident that many points may be excluded if the dimensionality is very small, but that the approach will worsen as the latter grows. Friedman *et al.* [19] give the expected NN search time, under the assumption that the points are uniformly distributed, as $O(mn^{1-1/n})$. This approaches the brute force $O(nm)$ as n gets large. Reported empirical results are for dimensions 2 to 8.

Marimont and Shapiro [20] extend this approach by the use of projections in subspaces of dimension greater than 1 (usually about m=2 is suggested). This can be further improved if the subspace of the principal components is used. Dimensions up to 40 are examined. The Euclidean distance is very widely used. Two other members of a family of Minkowski metric measures require less computation time to calculate, and they can be used to provide bounds on the Euclidean distance. We have:

$$d_1(x,x') \geq d_2(x,x') \geq d_\infty(x,x')$$

where d_1 is the Hamming distance defined as $\Sigma_j |x_j - x'_j|$, the Euclidean distance is given by the square root of $\Sigma_j (x_j - x'_j)^2$; and the Chebyshev distance is defined as $\max_j |x_j - x'_j|$.

Kittler [21] makes use of the following bounding strategy: reject all points y such that $d_1(x,y) \geq \sqrt{(m)}\delta$ where δ is the current NN d_2-distance. The more efficiently calculated d_1-distance may thus allow the rejection of many points (90% in 10-dimensional space is reported by Kittler). Kittler's rule is obtained by noting that the greatest d_1-distance between x and x´ is attained when

$$|x_j - x'_j|^2 = d_2^2(x,x')/m$$

for all coordinates, j. Hence $d_1(x,x') = d_2(x,x') \geq \sqrt{m}$ is the greatest d_1-distance between x and x'. In the case of the rejection of point y, we then have:

$$d_1(x,y) \leq d_2(x,y)/\sqrt{m}$$

and since, by virtue of the rejection

$$d_1(x,y) \geq \sqrt{m}\delta$$

it follows that $\delta \leq d_2(x,y)$.

Yunck [22] presents a theoretical analysis for the similar use of the Chebyshev metric. Richetin *et al.* [23] propose the use of both bounds. Using uniformly distributed points in dimensions 2 to 5, the latter reference reports the best outcome when the rule: reject all y such that $d_\infty(x,y) \geq \delta$ precedes the rule based on the d_1-distance. Up to 80% reduction in CPU time is reported.

Bounding using the Triangular Inequality

The triangular inequality is satisfied by distances: $d(x,y) \leq d(x,z) + d(z,y)$, where x, y and z are any three points. The use of a reference point, z, allows a full distance calculation between point x, whose NN is sought, and y to be avoided if

$$|d(y,z) - d(x,z)| \geq \delta$$

where δ is the current NN distance. The set of all distances to the reference point are calculated and stored in a preprocessing step requiring $O(n)$ time and $O(n)$ space. The above cut-off rule is obtained by noting that if

$$d(x,y) \geq |d(x,y) - d(y,z)|$$

then, necessarily, $d(x,y) \geq \delta$. The former inequality above reduces to the triangular inequality irrespective of which of $d(y,z)$ or $d(x,z)$ is the greater.

The set of distances to the reference point, $\{d(x,z)|x\}$, may be sorted in the preprocessing stage. Since $d(x,z)$ is fixed during the search for the NN of x, it follows that the cut-off rule will not then need to be applied in all cases.

Shapiro [25] generalized the single reference point approach, due to Burkhard and Keller, [24] to multiple reference points. The sorted list of distances to the first reference point, $\{d(x,z_1)|x\}$, is used as described above as a preliminary bound. Then the subsequent bounds are similarly employed to further reduce the points requiring a full distance calculation. The number and the choice of reference

points to be used is dependent on the distributional characteristics of the data. Shapiro [25] finds that reference points ought to be located away from groups of points. In 10-dimensional simulations, it was found that at best only 20% of full distance calculations were required (although this was very dependent on the choice of reference points).

Hodgson [26] proposes the following bound, related to the training set of points, y, among which the NN of point x is sought. Determine in advance the NNs and their distances, d(y,NN(y)) for all points in the training set. For point y, then consider δy = ½ d(y,NN(y)). In seeking NN(x), and having at some time in the processing a candidate NN, y', we can exclude all y from consideration if we find that d(x,y') ≤ $\delta y'$. In this case, we know that we are sufficiently close to y' that we cannot improve on it.

We return now to the choice of reference points: Vidal Ruiz [27] proposes the storing of inter-point distances between the members of the training set. Given x, whose NN we require, some member of the training set is used as a reference point. Using the bounding approach based on the triangular inequality, described above, allows other training set members to be excluded from any possibility of being NN(x). Micó *et al.* [28] and Ramasubramanian and Paliwal [29] discuss further enhancements to this approach, focused especially on the storage requirements. Fukunaga and Narendra [30] make use of both a hierarchical decomposition of the data set (they employ repeatedly the k-means partitioning technique), and bounds based on the triangular inequality. For each node in the decomposition tree, the center and maximum distance to the center of associated points (the "radius") are determined. For 1000 points, 3 levels were used, with a division into 3 classes at each node. All points associated with a non-terminal node can be rejected in the search for the NN of point x if the following rule (Rule 1) is not verified:

$$d(x,g) - r_g < \delta$$

where δ is the current NN distance, g is the center of the cluster of points associated with the node, and r_g is the radius of this cluster. For a terminal node, which cannot be rejected on the basis of this rule, each associated point, y, can be tested for rejection using the following rule (Rule 2):

$$|d(x,g) - d(y,g)| \geq \delta.$$

These two rules are direct consequences of the triangular inequality.

A branch and bound algorithm can be implemented using these two rules. This involves determining some current NN (the bound) and subsequently branching out of a traversal path whenever the current NN cannot be bettered. Not being inherently limited by dimensionality, this approach appears particularly attractive for general purpose applications.

Other rejection rules are considered by Kamgar-Parsi and Kanal. [31] A simpler form of clustering is used in the variant of this algorithm proposed by Niemann and Goppert. [32] A shallow MDBST is used, followed by a variant on the branching and bounding described above.

Bennett *et al.* [2] use the nearest neighbor problem as a means towards solving the Gaussian distribution mixture problem. They consider a preprocessing approach similar to Fukunaga and Narendra [30] but with an important difference: to take better account of cluster structure in the data, the clusters are multivariate normal but not necessarily of diagonal covariance structure. Therefore very elliptical clusters are allowed. This in turn implies that a cluster radius is not of great benefit for establishing a bound on whether or not distances need to be calculated Bennett *et al.* [2] address this problem by seeking a stochastic

guarantee on whether or not calculations can be excluded. Technically, however, such stochastic bounds are not easy to determine in a high dimensional space.

An interesting issue raised in Beyer *et al.* [33] is also discussed by Bennett *et al.* [2] if the ratio of the nearest and furthest neighbor distances converges in probability to 1 as the dimensionality increases, then is it meaningful to search for nearest neighbors? This issue is not all that different from saying that neighbors in an increasingly high dimensional space tend towards being equidistant. In section 5, we will look at approaches for handling particular classes of data of this type.

Fast Approximate Nearest Neighbor Finding

Kushilevitz *et al.*, [34] working in Euclidean and L_1 spaces, propose fast approximate nearest neighbor searching, on the grounds that in systems for content-based image retrieval, approximate results are adequate. Projections are used to bound the search. Probability of successfully finding the nearest neighbor is traded off against time and space requirements.

The Special Case of Sparse Binary Data

"High-dimensional", "sparse" and "binary" are the characteristics of keyword-based bibliographic data, with values possibly in excess of 10000 for both n and m. Such data is usually stored as list data structures, representing the mapping of documents onto index terms, or *vice versa*. Commercial document collections are usually searched using a Boolean search environment. Documents associated with particular terms are retrieved, and the intersection (AND), union (OR) or other operations on such sets of documents are obtained. For efficiency, an *inverted file,* which maps terms onto documents, must be available for Boolean retrieval. The efficient NN algorithms, to be discussed, make use of both the document-term and the term-document files.

The usual algorithm for NN-searching considers each document in turn, calculates the distance with the given document, and updates the NN if appropriate. This algorithm is shown schematically in Fig. 4 (top). The inner loop is simply an expression of the fact that the distance or similarity will, in general, require O(m) calculation: examples of commonly used coefficients are the Jaccard similarity, and the Hamming (L_1 Minkowski) distance.

If $\overline{m}$ and $\overline{n}$ are, respectively, the average numbers of terms associated with a document, and the average number of documents associated with a term, then an average complexity measure, over n searches, of this usual algorithm is O($n\overline{m}$). It is assumed that advantage is taken of some packed form of storage in the inner loop (*e.g.* using linked lists).

Croft's algorithm (see [35] and Fig. 4) is of worst case complexity O(nm^2). However, the number of terms associated with the document whose NN is required will often be quite small. The National Physical Laboratory test collection, for example, which was used by Murtagh [36] has the following characteristics: n = 11429, m = 7491, $\overline{m}$ = 19.9, and $\overline{n}$ = 30.4. The outermost and innermost loops in Croft's algorithm use the document-term file. The center loop uses the term-document inverted file. An average complexity measure (more strictly, the time taken for best match search based on an average document with associated average terms) is seen to be O($\overline{nm}^2$).

```
Usual algorithm:
      Initialize current NN
      For all documents in turn do:
      ... For all terms associated with the document do:
      ... ... Determine (dis)similarity
      ... Endfor
      ... Test against current NN
      Endfor

Croft's algorithm:
      Initialize current NN
      For all terms associated with the given document do:
      ... For all documents associated with each term do:
      ... ... For all terms associated with a document do:
      ... ... ... Determine (dis)similarity
      ... ... Endfor
      ... ... Test against current NN
      ... Endfor
      Endfor

Perry-Willett algorithm:
      Initialize current NN
      For all terms associated with the given document, i, do:
      ... For all documents, i', associated with each term, do:
      ... ... Increment location i' of counter vector
      ... Endfor
      Endfor
```

Figure 4: Algorithms for NN-searching using high-dimensional sparse binary data.

In the outermost loop of Croft's algorithm there will eventually come about a situation where – if a document has not been thus far examined - the number of terms remaining for the given document do not permit the current NN document to be bettered. In this case we can cut short the iterations of the outermost loop. The calculation of a bound using the greatest possible number of terms that could be shared with a so-far unexamined document has been exploited by Smeaton and van Rijsbergen [37] and by Murtagh [36] in successive improvements on Croft's algorithm.

The complexity of all the above algorithms has been measured in terms of operations to be performed. In practice, however, the actual accessing of term or document information may be of far greater cost. The document-term and term-document files are ordinarily stored on direct access file storage because of their large sizes. The strategy used in Croft's algorithm, and in improvements on it, does not allow any viable approaches to batching together the records which are to be read successively, in order to improve accessing-related performance.

The Perry-Willett algorithm (see Perry and Willett, [38]) presents a simple but effective solution to the problem of costly I/O. It focuses on the calculation of the number of terms common to the given document x and each other document, y, in the document collection. This set of values is built up in a computationally efficient fashion. O(n) operations are subsequently required to determine the (dis)similarity, using another vector comprising the total numbers of terms associated with each document. Computation time (the same "average" measure as that used above) is O($\overline{nm}+n$). We now turn our attention to numbers of direct-access reads required.

In Croft's algorithm, all terms associated with the document whose NN is desired may be read in one

read operation. Subsequently, we require $\overline{nm}$ reads, giving in all $1+\overline{nm}$. In the Perry-Willett algorithm, the outer loop again pertains to the one (given) document, and so all terms associated with this document can be read and stored. Subsequently, $\overline{m}$ reads, *i.e.* the average number of terms, each of which demands a read of a set of documents, are required. This gives, in all, $1 + \overline{m}$. Since these reads are very much the costliest operation in practice, the Perry-Willett algorithm can be recommended for large values of n and m. Its general characteristics are that it requires, (i) as do all the algorithms discussed in this section, the availability of the inverted term-document file; and (ii) in-memory storage of two vectors containing *n* integer values.

HIERARCHICAL AGGLOMERATIVE CLUSTERING

The algorithms discussed in this section can be characterized as greedy. [39] A sequence of irreversible algorithm steps is used to construct the desired data structure.

We will not review hierarchical agglomerative clustering here. For essential background, the reader is referred to Murtagh and Heck, [3] Gordon, [40] or Jain and Dubes. [41] This section borrows on Murtagh. [42]

One could practically say that Sibson [43] and Defays [44] are part of the prehistory of clustering. Their $O(n^2)$ implementations of the single link method and of a (non-unique) complete link method, respectively, have been widely cited.

In the early 1980s a range of significant improvements were made to the Lance-Williams, or related, dissimilarity update schema, [45, 46] which had been in wide use since the mid-1960s. Murtagh [47] presents a survey of these algorithmic improvements. We will briefly describe them here. The new algorithms, which have the potential for exactly replicating results found in the classical but more computationally expensive approach, are based on the construction of nearest neighbor chains and reciprocal or mutual NNs (NN-chains and RNNs).

A NN-chain consists of an arbitrary point (a in Fig. 5); followed by its NN (b in Fig. 5); followed by the NN from among the remaining points (c, d, and e in Fig. 5) of this second point; and so on until we necessarily have some pair of points which can be termed reciprocal or mutual NNs. (Such a pair of RNNs may be the first two points in the chain; we have assumed that no two dissimilarities are equal.)

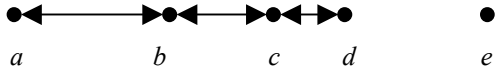


Figure 5: Five points, showing NNs and RNNs.

In constructing a NN-chain, irrespective of the starting point, we may agglomerate a pair of RNNs as soon as they are found. What guarantees that we can arrive at the same hierarchy as we would if we used traditional "stored dissimilarities" or "stored data" algorithms? Essentially this is the same condition as that under which no inversions or reversals are produced by the clustering method. Fig. 6 gives an example of this, where s is agglomerated at a lower criterion value (*i.e.* dissimilarity) than was the case at the previous agglomeration between *q* and *r*.

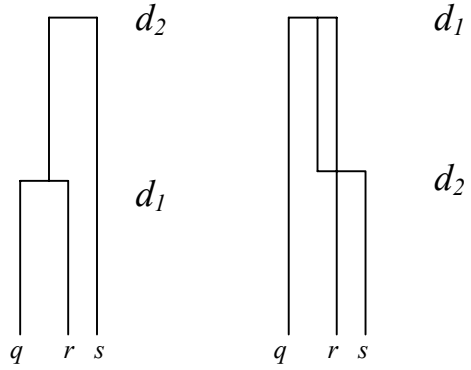


Figure 6: Alternative representations of a hierarchy with an inversion. Assuming dissimilarities, as we go vertically up, criterion values (d_1, d_2) decrease. But here, undesirably, $d_2 > d_1$.

Our ambient space has thus contracted because of the agglomeration. This is due to the algorithm used - in particular the agglomeration criterion - and it is something we would normally wish to avoid.

This is formulated as:

Inversion impossible if

$$d(i,j) < d(i,k) \text{ or } d(j,k) \Rightarrow d(i,j) < d(i \cup j, k)$$

This is essentially Bruynooghe's reducibility property [48] (see also [49]). Using the Lance-Williams dissimilarity update formula, it can be shown that the minimum variance method does not give rise to inversions; neither do the linkage methods; but the median and centroid methods cannot be guaranteed not to have inversions.

To return to Fig. 5, if we are dealing with a clustering criterion that precludes inversions, then c and d can justifiably be agglomerated, since no other point (for example, b or e) could have been agglomerated to either of these.

The processing required, following an agglomeration, is to update the NNs of points such as b in Fig. 5 (and on account of such points, this algorithm was dubbed *algorithme des célibataires* in [45]). The following is a summary of the algorithm:

NN-chain algorithm

> **Step 1** Select a point arbitrarily.
>
> **Step 2** Grow the NN-chain from this point until a pair of RNNs is obtained.
>
> **Step 3** Agglomerate these points (replacing with a cluster point, or updating the dissimilarity matrix).
>
> **Step 4** From the point which preceded the RNNs (or from any other arbitrary point if the first two points chosen in Steps 1 and 2 constituted a pair of RNNs), return to Step 2 until only one point remains.

In Murtagh [11, 47, 49] and Day and Edelsbrunner, [50] one finds discussions of O(n^2) time and O(n) space implementations of Ward's minimum variance (or error sum of squares) method and of the centroid and median methods. The latter two methods are termed the UPGMC and WPGMC criteria by Sneath and Sokal. [51] Now, a problem with the cluster criteria used by these latter two methods is that the reducibility property is not satisfied by them. This means that the hierarchy constructed may not be unique as a result of inversions or reversals (non-monotonic variation) in the clustering criterion value determined in the sequence of agglomerations. Murtagh [49] describes O(n^2) time and space implementations for the single link method, the complete link method and for the weighted and unweighted group average methods (WPGMA and UPGMA). This approach is quite general *vis á vis* the dissimilarity used and can also be used for hierarchical clustering methods other than those mentioned.

Day and Edelsbrunner [50] prove the exact O(n^2) time complexity of the centroid and median methods using an argument related to the combinatorial problem of optimally packing hyperspheres into an m-dimensional volume. They also address the question of metrics: results are valid in a wide class of distances including those associated with the Minkowski metrics.

The construction and maintenance of the nearest neighbor chain as well as the carrying out of agglomerations whenever reciprocal nearest neighbors meet, both offer possibilities for parallelization. Willet described implementations on an SIMD machine. [52]

Evidently both coordinate data and graph (*e.g.*, dissimilarity) data can be input to these agglomerative methods. Gillet et. al. [53] in the context of clustering chemical structure databases refer to the common use of the Ward method, based on the reciprocal nearest neighbors algorithm, on

data sets of a few hundred thousand molecules.

Applications of hierarchical clustering to bibliographic information retrieval are assessed in Griffiths *et al.* [54] Ward's minimum variance criterion is favored.

From details in White and McCain, [55] the Institute of Scientific Information (ISI) clusters citations (science, and social science) by first clustering highly cited documents based on a single linkage criterion, and then four more passes are made through the data to create a subset of a single linkage hierarchical clustering.

GRAPH CLUSTERING

Hierarchical clustering methods are closely related to graph-based clustering. Firstly, a dendrogram is a rooted labeled tree. Secondly, and more importantly, some methods like the single and complete link methods can be displayed as graphs, and are very closely related to mainstream graph data structures.

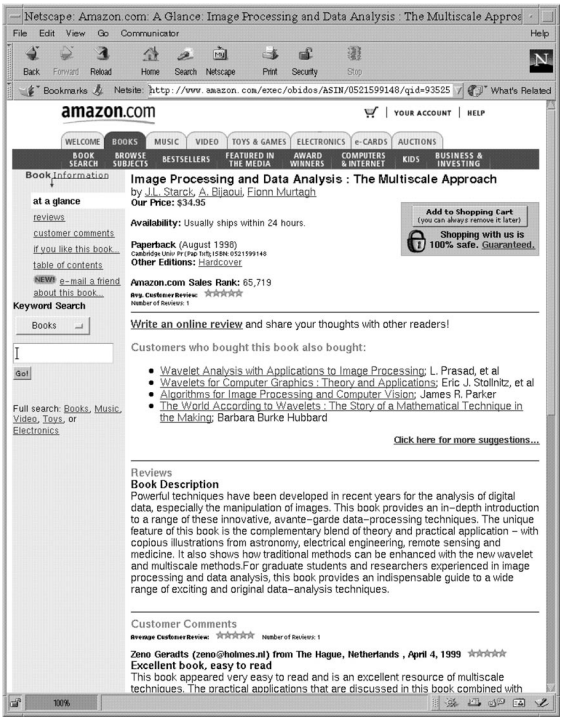


Figure 7: Example of a graph clustering in a data mining perspective at Amazon.com: "Customers who bought this book also bought …"

An example of the increasing prevalence of graph clustering in the context of data mining on the web is presented in Fig. 7: Amazon.com provides information on what other books were purchased by like-minded individuals.

The single link method was referred to in the previous section, as a widely used agglomerative, hence hierarchical, clustering method. Rohlf [56] reviews algorithms for the single link method with complexities ranging from O(n log n) to O(n^5). The criterion used by the single link method for cluster formation is weak, meaning that noisy data in particular give rise to results that are not robust.

The minimal spanning tree (MST) and the single link agglomerative clustering method are closely related: the MST can be transformed irreversibly into the single link hierarchy. [57] The MST is defined as of minimal total weight, it spans all nodes (vertices) and is an unrooted tree. The MST has been a method of choice for at least four decades now either in its own right for data analysis, [58] as a data structure to be approximated (*e.g.* using shortest spanning paths, see Murtagh, [47], p. 96), or as a basis for clustering. We will look at some fast algorithms for the MST in the remainder of this section.

Perhaps the most basic MST algorithm, due to Prim and Dijkstra, grows a single fragment through n-1 steps. We find the closest vertex to an arbitrary vertex, calling these a fragment of the MST. We determine the closest vertex, not in the fragment, to any vertex in the fragment, and add this new vertex into the fragment. While there are fewer than n vertices in the fragment, we continue to grow it. This algorithm leads to a unique solution. A default O(n^3) implementation is clear, and O(n^2) computational cost is possible ([47], p. 98).

Sollin's algorithm constructs the fragments in parallel. For each fragment in turn, at any stage of the construction of the MST, determine its closest

fragment. Merge these fragments, and update the list of fragments. A tree can be guaranteed in this algorithm (although care must be taken in cases of equal similarity) and our other requirements (all vertices included, minimal total edge weight) are very straightforward. Given the potential for roughly halving the data remaining to be processed at each step, not surprisingly the computational cost reduces from $O(n^3)$ to $O(n^2 \log n)$.

The real interest of Sollin's algorithm arises when we are clustering on a graph and do not have all $n(n\text{-}1)/2$ edges present. Sollin's algorithm can be shown to have computational cost m log n, where m is the number of edges. When $m \ll n(n\text{-}1)/2$ then we have the potential for appreciable gains.

The MST in feature spaces can of course make use of the fast nearest neighbor finding methods studied earlier in this article. See [47], (section 4.4) for various examples.

Other graph data structures that have been proposed for data analysis are related to the MST. We know, for example, that the following subset relationship holds:

$$MST \subseteq RNG \subseteq GG \subseteq GT$$

where RNG is the relative neighborhood graph, GG is the Gabriel graph, and DT is the Delaunay triangulation. The latter, in the form of its dual, the Voronoi diagram, has been used for analyzing the clustering of galaxy locations. References to these and related methods can be found in Murtagh. [59]

NEAREST NEIGHBOR FINDING ON GRAPHS

Clustering on graphs may be required because we are working with (perhaps complex non-Euclidean) dissimilarities. In such cases where we must take into account an edge between each and every pair of vertices, we will generally have an $O(m)$ computational cost where m is the number of edges.

In a metric space we have seen that we can look for various possible ways to expedite the nearest neighbor search. An approach based on visualization - turning our data into an image - will be looked at below. However, there is another aspect of our similarity (or other) graph that we may be able to turn to our advantage. Efficient algorithms for sparse graphs are available. Sparsity can be arranged - we can threshold our edges if the sparsity does not suggest itself more naturally.

A special type of sparse graph is a planar graph, *i.e.* a graph capable of being represented in the plane without any crossovers of edges. For sparse graphs, algorithms with $O(m \log \log n)$ computational cost were described by Yao [60] and Cheriton and Tarjan. [61] A short algorithmic description can be found in Murtagh [47] (pp. 107-108) and we refer in particular to the latter. The basic idea is to preprocess the graph, in order to expedite the sorting of edge weights (why sorting? - simply because we must repeatedly find smallest links, and maintaining a sorted list of edges is a good basis for doing this). If we were to sort all edges, the computational requirement would be $O(m \log m)$. Instead of doing that, we take the edge set associated with each and every vertex. We divide each such edge set into groups of size k. (The fact that the last such group will usually be of size $< k$ is taken into account when programming.)

Let n_v be the number of incident edges at vertex v, such that $\Sigma_v n_v = 2m$. The sorting operation for each vertex now takes $O(k \log k)$ operations for each group, and we have n_v/k groups. For all vertices the sorting requires a number of operations which is of the order of $\Sigma_v n_v \log k = 2m \log k$. This looks like a questionable - or small - improvement over $O(m \log m)$. Determining the lightest edge incident on a vertex requires $O(n_v/k)$ comparisons since we have to check all groups. Therefore the lightest edges incident on all vertices are found with $O(m/k)$

operations.

When two vertices, and later fragments, are merged, their associated groups of edges are simply collected together, therefore keeping the total number of groups of edges that we started out with.

We will bypass the issue of edges which, over time, are to be avoided because they connect vertices in the same fragment: given the fact that we are building an MST, the total number of such edges-to-be-avoided cannot surpass 2m. To find what to merge next, again O(m/k) processing is required. Using Sollin's algorithm, the total processing required in finding what to merge next is O(m/k log n). The total processing required for grouping the edges, and sorting within the edge-groups, is O(m log k), *i.e.* it is one-off and accomplished at the start of the MST-building process.

The total time is O(m/k log n)+O(m log k). If we fix k = log n, the second term dominates and gives overall computational complexity as O(m log log n). This result has been further improved to near linearity in m by Gabow *et al.*, [62] who develop an algorithm with complexity O(m log log log … n) where the number of iterated log terms is bounded by m/n.

Motwani and Raghavan [63] (chapter 10) base a stochastic O(m) algorithm for the MST on random sampling to identify and eliminate edges that are guaranteed not to belong to the MST.

Let us turn our attention now to the case of a planar graph. For a planar graph we know that $m \leq 3n$-6 for m >1. (For proof, see for example Tucker, [64] or any book on graph theory).

Referring to Sollin's algorithm, described above, O(n) operations are needed to establish a least cost edge from each vertex, since there are only O(n) edges present. On the next round, following fragment-creation, there will be at most ceil(n/2) new vertices, implying of the order of n/2 processing to find the least cost edge. The total computational cost is seen to be proportional to: n + n/2 + n/4 +…=O(n).

So determining the MST of a planar graph is linear in numbers of either vertices or edges. Before ending this review of very efficient clustering algorithms for graphs, we note that algorithms discussed so far have assumed that the similarity graph was undirected. For modeling transport flows, or economic transfers, the graph could well be directed. Components can be defined, generalizing the clusters of the single link method, or the complete link method. [65] provides an algorithm for the latter agglomerative criterion which is of computational cost O(m log n).

K-MEANS AND FAMILY

The non-technical person more often than not understands clustering as a partition. K-means looked at in this section, or the distribution mixture approach looked at in the section on fast model-based clustering, provide solutions. A mathematical definition of a partition implies no multiple assignments of observations to clusters, *i.e.* no overlapping clusters. Overlapping clusters may be faster to determine in practice, and a case in point is the one-pass algorithm described in Salton and McGill. [66] The general principle followed is: make one pass through the data, assigning each object to the first cluster which is close enough, and making a new cluster for objects that are not close enough to any existing cluster.

Broder *et al.* [67] use this algorithm for clustering the web. A feature vector is determined for each HTML document considered, based on sequences of words. Similarity between documents is based on an inverted list, using an approach like those described for the special case of binary data above. The similarity graph is thresholded, and components sought.

Broder [68] solves the same clustering objective

using a thresholding and overlapping clustering method similar to the Salton and McGill one. The application described is that of clustering the Altavista repository in April 1996, consisting of 30 million HTML and text documents, comprising 150 GBytes of data. The number of serviceable clusters found was 1.5 million, containing 7 million documents. Processing time was about 10.5 days. An analysis of the clustering algorithm used by Broder can be found in Borodin *et al.*, [69] who also consider the use of approximate minimal spanning trees.

The threshold-based pass of the data, in its basic state, is susceptible to lack of robustness. A bad choice of threshold leads to too many clusters or two few. To remedy this, we can work on a well-defined data structure such as the minimal spanning tree. Or, alternatively, we can iteratively refine the clustering. Partitioning methods, such as k-means, use iterative improvement of an initial estimation of a targeted clustering.

A very widely used family of methods for inducing a partition on a data set is called k-means, c-means (in the fuzzy case), Isodata, competitive learning, vector quantization and other more general names (non-overlapping non-hierarchical clustering) or more specific names (minimal distance or exchange algorithms).

The usual criterion to be optimized is:

$$\frac{1}{|I|}\sum_{q\in Q}\sum_{i\in q}\left\|\vec{i}-\vec{q}\right\|^2$$

where I is the object set, $|.|$ denotes cardinality, q is some cluster, Q is the partition, and q denotes a set in the summation, whereas $\vec{q}$ denotes some associated vector in the error term, or metric norm.

This criterion ensures that clusters found are compact, and therefore assumed homogeneous. The optimization criterion, by a small abuse of terminology, is more often referred to as a minimum variance one. A necessary condition that this criterion be optimized is that vector $\vec{q}$ be a cluster mean, which for the Euclidean metric case is:

$$\vec{q} = \frac{1}{|q|}\sum_{i\in q}\vec{i}$$

A batch update algorithm, due to Lloyd, [70] Forgy, [71] and others, makes assignments to a set of initially randomly chosen vectors, $\vec{q}$, as step 1. Step 2 updates the cluster vectors, $\vec{q}$. This is iterated. The distortion error, equation 1, is non-increasing, and a local minimum is achieved in a finite number of iterations.

An online update algorithm is due to MacQueen. [72] After each presentation of an observation vector, $\vec{i}$, the closest cluster vector, $\vec{q}$, is updated to take account of it. Such an approach is well-suited for a continuous input data stream (implying "online" learning of cluster vectors).

Both algorithms are gradient descent ones. In the online case, much attention has been devoted to best learning rate schedules in the neural network (competitive learning) literature: Darken and Moody [73, 74], Darken *et al.*, [75] Fritzke. [76]

A difficulty, less controllable in the case of the batch algorithm, is that clusters may become (and stay) empty. This may be acceptable, but also may be in breach of our original problem formulation. An alternative to the batch update algorithm is Späth's exchange algorithm. [77] Each observation is considered for possible assignment into any of the other clusters. Späth gives updating and "downdating" formulae. This exchange algorithm is stated to be faster to converge and to produce better (smaller) values of the objective function. Over decades of use, we have also verified that it is a superior algorithm to the minimal distance one.

K-means is very closely related to Voronoi

(Dirichlet) tesselations, to Kohonen self-organizing feature-maps, and various other methods. The batch-learning algorithm above may be viewed as

1. An assignment step, which we will term the E (estimation) step: estimate the posteriors,

   P(observations | cluster centers)

2. A cluster update step, the M (maximization) step, which maximizes a cluster center likelihood.

Neal and Hinton [78] cast the k-means optimization problem in such away that the both E- and M-steps monotonically increase the maximand's values. The EM algorithm may, too, be enhanced to allow for online as well as batch learning. [79]

In Thiesson *et al*., [80] k-means is implemented (i) by traversing blocks of data, cyclically, and incrementally updating the sufficient statistics and parameters, and (ii) instead of cyclic traversal, sampling from subsets of the data is used. Such an approach is admirably suited for very large data sets, where in-memory storage is not feasible. Examples used by Thiesson *et al*. [80] include the clustering of a half million 300-dimensional records.

FAST MODEL BASED CLUSTERING

It is traditional to note that models and (computational) speed do not mix. We review recent progress in this section.

Modeling of Signal and Noise

A simple and applicable model is a distribution mixture, with the signal modeled by Gaussians, in the presence of Poisson background noise.

Consider data which are generated by a mixture of (G-1) bivariate Gaussian densities, $f_k(x;\theta)\sim N(\mu_k;\Sigma_k)$, for clusters $k = 2; \ldots ; G$, and with Poisson background noise corresponding to $k = 1$. The overall population thus has the mixture density

$$f(x;\theta) = \sum_{k=1}^{G} \pi_k f_k(x;\theta)$$

where the mixing or prior probabilities, π_k, sum to 1, and $f_1(x;\theta) = A^{-1}$, where A is the area of the data region. This is the basis for model-based clustering. [81-84]

The parameters, θ and π, can be estimated efficiently by maximizing the mixture likelihood

$$L(\theta,\pi) = \prod_{i=1}^{n} f(x_i;\theta),$$

with respect to θ and π, where x_i is the i^{th} observation.

Now let us assume the presence of two clusters, one of which is Poisson noise, the other Gaussian. This yields the mixture likelihood

$$L(\theta,\pi) = \prod_{i=1}^{n}\left[\pi_1 A^{-1} + \pi_2 \frac{1}{2\pi\sqrt{|\Sigma|}}\exp\left\{-\frac{1}{2}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)\right\}\right],$$

where $\pi_1 + \pi_2 = 1$.

An iterative solution is provided by the expectation-maximization (EM) algorithm of Dempster *et al*. [85] We have already noted this algorithm in informal terms in the last section, dealing with *k*-means. Let the "complete" (or "clean" or "output") data be $y_i = (x_i, z_i)$ with indicator set $z_i = (z_{i1}, z_{i2})$ given by (1,0) or (0,1). Vector z_i has a multinomial distribution with parameters $(1;\pi_1,\pi_2)$. This leads to the complete data log-likelihood:

$$l(y,z;\theta,\pi) = \sum_{i=1}^{n}\sum_{k=1}^{2} z_{ik}\left[\log\pi_k + \log f_k(x_k;\theta\right]$$

The E-step then computes $\hat{z}_{ik} = E(z_{ik}|x_1,\ldots,x_n,\theta)$, *i.e.* the posterior probability that x_i is in cluster k. The M-step involves maximization of the expected complete data log-likelihood:

$$l^*(y;\theta,\pi) = \sum_{i=1}^{n}\sum_{k=1}^{2} \hat{z}_{ik}\left[\log\pi_k + \log f_k(x_i;\theta)\right].$$

The E- and M-steps are iterated until convergence.

For the 2-class case (Poisson noise and a Gaussian cluster), the complete-data likelihood is

$$L(y,z;\theta,\pi) = \prod_{i=1}^{n}\left[\frac{\pi_1}{A}\right]^{zi1}\left[\frac{\pi_2}{2\pi\sqrt{|\Sigma|}}\exp\left\{-\frac{1}{2}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)\right\}\right]^{zi2}$$

The corresponding expected log-likelihood is then used in the EM algorithm. This formulation of the problem generalizes to the case of G clusters, of arbitrary distributions and dimensions.

Fraley [86] discusses implementation of model-based clustering, including publicly available software.

In order to assess the evidence for the presence of a signal-cluster, we use the Bayes factor for the mixture model, M_2 that includes a Gaussian density as well as background noise, against the "null" model, M_1, that contains only background noise. The Bayes factor is the posterior odds for the mixture model against the pure noise model, when neither is favored *a priori*. It is defined as $B = p(x|M_2)/p(x|M_1)$, where $p(x|M_2)$ is the integrated likelihood of the mixture model M_2, obtained by integrating over the parameter space. For a general review of Bayes factors, their use in applied statistics, and how to approximate and compute them, see Kass and Raftery. [87]

We approximate the Bayes factor using the Bayesian Information Criterion (BIC). [88] For a Gaussian cluster and Poisson noise, this takes the form:

$$2\log B \approx BIC = 2\log L(\hat{\theta},\hat{\pi}) + 2n\log A - 6\log n,$$

where $\hat{\theta}$ and $\hat{\pi}$ are the maximum likelihood estimators of θ and π, and $L(\hat{\theta},\hat{\pi})$ is the maximized mixture likelihood.

A review of the use of the BIC criterion for model selection - and more specifically for choosing the number of clusters in a data set - can be found in Fraley and Raftery. [89]

An application of mixture modeling and the BIC criterion to gamma-ray burst data can be found in Mukherjee *et al.* [90] So far around 800 observations have been assessed, but as greater numbers become available we will find the inherent number of clusters in a similar way, in order to try to understand more about the complex phenomenon of gamma-ray bursts.

Application to Thresholding

Consider an image or a planar or 3-dimensional set of object positions. For simplicity we consider the case of setting a single threshold in the image intensities, or the point set's spatial density.

We deal with a combined mixture density of two univariate Gaussian distributions $f_k(x,\theta) \sim N(\mu_k,\sigma_k)$. The overall population thus has the mixture density

$$f(x;\theta) = \sum_{k=1}^{2}\pi_k f_k(x;\theta)$$

where the mixing or prior probabilities, π_k, sum to 1.

When the mixing proportions are assumed equal, the log-likelihood takes the form

$$l(\theta) = \sum_{i=1}^{n}\ln\left[\sum_{k=1}^{2}\frac{1}{2\pi\sqrt{|\sigma_k|}}\exp\left\{-\frac{1}{2\sigma_k}(x_i-\mu_k)^2\right\}\right]$$

The EM algorithm is then used to iteratively solve this (see Celeux and Govaert, [91]). This method is used for appraisals of textile (jeans and other fabrics) fault detection in Campbell *et al.* [92]. Industrial vision inspection systems potentially produce large data streams, and fault detection can be a good application for fast clustering methods. We are currently using a mixture model of this sort on SEM (scanning electron microscope) images of cross-sections of concrete to allow for subsequent characterization of physical properties.

Image segmentation, *per se*, is a relatively straightforward application, but there are novel and interesting aspects to the two studies mentioned. In the textile case, the faults are very often perceptual and relative, rather than "absolute" or capable of

being analyzed in isolation. In the SEM imaging case, a first phase of processing is applied to de-speckle the images, using multiple resolution noise filtering.

Turning from concrete to cosmology, the Sloan Digital Sky Survey [92] is producing a sky map of more than 100 million objects, together with 3-dimensional information (redshifts) for a million galaxies. Pelleg and Moore [18] describe mixture modeling, using a k-D tree preprocessing to expedite the finding of the class (mixture) parameters, *e.g.* means, covariances.

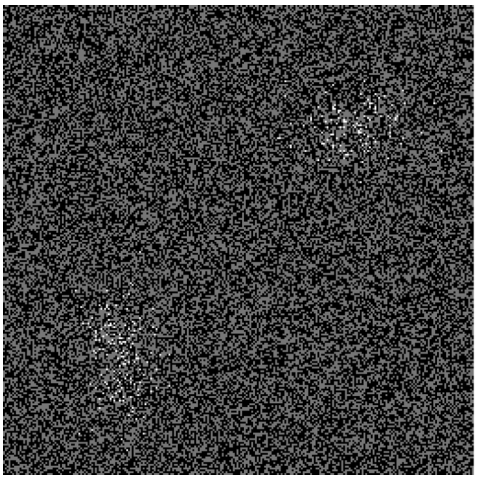


Figure 8: Data in the plane. The 256 × 256 image shows 550 "signal" points – two Gaussian-shaped clusters in the lower left and in the upper right – with in addition 40,000 Poisson noise points added. Details of recovery of the clusters are discussed in [95].

NOISE MODELING

In Starck *et al*. [93] and in a wide range of papers, we have pursued an approach for the noise modeling of observed data. A multiple resolution scale vision model or data generation process is used, to allow for the phenomenon being observed on different scales. In addition, a wide range of options are permitted for the data generation transfer path, including additive and multiplicative, stationary and non-stationary, Gaussian ("read out" noise), Poisson (random shot noise), and so on.

Given point pattern clustering in two- or three-dimensional spaces, we will limit our overview here to the Poisson noise case.

Poisson Noise with Few Events Using the à trous Transform

If a wavelet coefficient $w_j(x,y)$ is due to noise, it can be considered as a realization of the sum $\sum_{k\in K} n_k$ of independent random variables with the same distribution as that of the wavelet function (n_k being the number of events used for the calculation of $w_j(x,y)$). This allows comparison of the wavelet coefficients of the data with the values that can be taken by the sum of n independent variables.

The distribution of one event in wavelet space is then directly given by the histogram H_1 of the wavelet ψ. As we consider independent events, the distribution of a coefficient w_n (note the changed subscripting for w, for convenience) related to n events is given by n autoconvolutions of H_1 :

$$H_n = H_1 \otimes H_1 \otimes ... \otimes H_1$$

For a large number of events, H_n converges to a Gaussian. Fig. 8 shows an example of where point pattern clusters - density bumps in this case – are sought, with a great amount of background clutter. Murtagh and Starck [94] refer to the fact that there is no computational dependence on the number of points (signal or noise) in such a problem, when using a wavelet transform with noise modeling.

Some other alternative approaches will be briefly noted. The Haar transform presents the advantage of its simplicity for modeling Poisson noise. Analytic formulae for wavelet coefficient distributions have been derived by Kolaczyk, [96] and Jammal and Bijaoui. [97] Using a new wavelet transform, the Haar *à trous* transform, Zheng *et al.* [98] appraise a denoising approach for financial data streams, - an important preliminary step for

subsequent clustering, forecasting, or other processing.

Poisson Noise with Nearest Neighbor Clutter Removal

The wavelet approach is certainly appropriate when the wavelet function reflects the type of object sought (*e.g.* isotropic), and when superimposed point patterns are to be analyzed. However, non-superimposed point patterns of complex shape are very well treated by the approach described in Byers and Raftery. [99] Using a homogeneous Poisson noise model, they derive the distribution of the distance of a point to its k^{th} nearest neighbor.

Next, Byers and Raftery [99] consider the case of a Poisson process which is signal, superimposed on a Poisson process which is clutter. The k^{th} nearest neighbor distances are modeled as a mixture distribution: a histogram of these, for given k, will yield a bimodal distribution if our assumption is correct. This mixture distribution problem is solved using the EM algorithm. Generalization to higher dimensions, *e.g.* 10, is also discussed.

Similar data were analyzed by noise modeling and a Voronoi tesselation preprocessing of the data in Allard and Fraley. [100] It is pointed out there how this can be a very useful approach with the Voronoi tiles have meaning in relation to the morphology of the point patterns. However, it does not scale well to higher dimensions, and the statistical noise modeling is approximate.

Ebeling and Wiedenmann, [101] reproduced in Dobrzycki *et al.*, [102] propose the use of a Voronoi tesselation for astronomical X-ray object detection and characterization.

CLUSTER-BASED USER INTERFACES

Doyle first described Information retrieval by means of "semantic road maps" in detail. [103] The spatial metaphor is a powerful one in human information processing. The spatial metaphor also lends itself well to modern distributed computing environments such as the web. The Kohonen self-organizing feature map (SOM) method is an effective means towards this end of a visual information retrieval user interface. We will also provide an illustration of web-based semantic maps based on hyperlink clustering.

The Kohonen map is, at heart, k-means clustering with the additional constraint that cluster centers be located on a regular grid (or some other topographic structure) and furthermore their location on the grid be monotonically related to pairwise proximity. [104] The nice thing about a regular grid output representation space is that it lends itself well as a visual user interface.

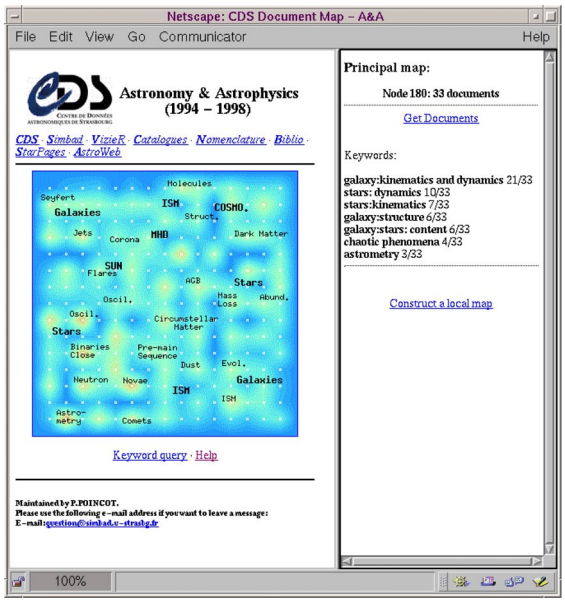


Figure 9: Visual interactive user interface to the journal *Astronomy and Astrophysics*.

Fig. 9 shows a visual and interactive user interface map, using a Kohonen self-organizing feature map (SOM). Color is related to density of document clusters located at regularly-spaced nodes of the map, and some of these nodes/clusters are annotated. The map is installed as a clickable

image-map, with CGI programs accessing lists of documents and - through further links - in many cases, the full documents. In the example shown, the user has queried a node and results are seen in the right-hand panel. Such maps are maintained for (currently) 12000 articles from the Astrophysical Journal, 7000 from Astronomy and Astrophysics, over 2000 astronomical catalogs, and other data holdings. More information on the design of this visual interface and user assessment can be found in Poinçot *et al*. [105, 106]

Guillaume [107] developed a Java-based visualization tool for hyperlink-based data, consisting of astronomers' names, astronomical object names, article titles, and with the possibility of other objects (images, tables, *etc.*). Through weighting, the various types of links could be prioritized. An iterative refinement algorithm was developed to map the nodes (objects) to a regular grid of cells, which as for the Kohonen SOM map, are clickable and provide access to the data represented by the cluster. Fig. 10 shows an example for an astronomer (Prof. Jean Heyvaerts, Strasbourg Astronomical Observatory).

These new cluster-based visual user interfaces are not computationally demanding. They are not however, scalable in their current implementation. Document management (see *e.g.* Cartia, [108]) is not so much the motivation, but rather the interactive user interface.

IMAGES FROM DATA

It is quite impressive how 2D (or 3D) image signals can handle with ease the scalability limitations of clustering and many other data processing operations. The contiguity imposed on adjacent pixels bypasses the need for nearest neighbor finding. It is very interesting therefore to consider the feasibility of taking problems of clustering massive data sets into the 2D image domain. We will look at a few recent examples of work in this direction.

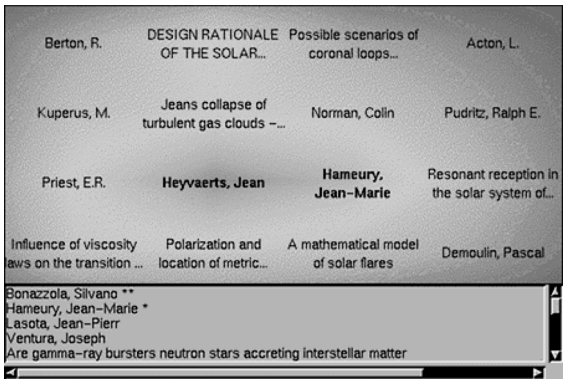


Figure 10: Visual interactive user interfaces, based on graph edges. Map for astronomer Jean Heyvaerts. Original in color.

Church and Helfman [109] address the problem of visualizing possibly millions of lines of computer program code, or text. They consider an approach borrowed from DNA sequence analysis. The data sequence is tokenized by splitting it into its atoms (line, word, character, *etc.*) and then placing a dot at position i,j if the i^{th} input token is the same as the j^{th}. The resulting dotplot, it is argued, is not limited by the available display screen space, and can lead to discovery of large-scale structure in the data.

When data do not have a sequence we have an invariance problem that can be resolved by finding some row and column permutation which pulls large array values together, and perhaps furthermore into proximity to an array diagonal. Berry *et al*. [110] have studied the case of large sparse arrays. Gathering larger (or nonzero) array elements to the diagonal can be viewed in terms of minimizing the envelope of nonzero values relative to the diagonal. This can be formulated and solved in purely symbolic terms by reordering vertices in a suitable graph representation of the matrix. A widely used method for symmetric sparse matrices is the Reverse Cuthill-McKee (RCM) method.

The complexity of the RCM method for ordering rows or columns is proportional to the product of the maximum degree of any vertex in the graph

representing the array values and the total number of edges (nonzeroes in the matrix). For hypertext matrices with small maximum degree, the method would be extremely fast. The strength of the method is its low time complexity but it does suffer from certain drawbacks. The heuristic for finding the starting vertex is influenced by the initial numbering of vertices and so the quality of the reordering can vary slightly for the same problem for different initial numberings. Next, the overall method does not accommodate dense rows (*e.g.*, a common link used in every document), and if a row has a significantly large number of nonzeroes it might be best to process it separately; *i.e.*, extract the dense rows, reorder the remaining matrix and augment fit by the dense rows (or common links) numbered last. Elapsed CPU times for a range of arrays and permuting methods are given in Berry *et al*., [110] and as an indication show performances between 0.025 to 3.18 seconds for permuting a 4000 x 400 array.

A review of public domain software for carrying out SVD and other linear algebra operations on large sparse data sets can be found in Berry *et al.* ([111], section 8.3).

Once we have a sequence-respecting array, we can immediately apply efficient visualization techniques from image analysis. Murtagh *et al*. [112] investigate the use of noise filtering (*i.e.* to remove less useful array entries) using a multiscale wavelet transform approach.

An example follows. From the Concise Columbia Encyclopedia (1989 2nd ed., online version) a set of data relating to 12025 encyclopedia entries and to 9778 cross-references or links was used.

Fig. 11 shows a 500 x 450 subarray, based on a correspondence analysis (*i.e.* ordering of projections on the first factor).

This part of the encyclopedia data was filtered using the wavelet and noise-modeling methodology described in Murtagh *et al*. [112] and the outcome is shown in Fig. 12. Overall the recovery of the more apparent alignments, and hence visually stronger clusters, is excellent. The first relatively long "horizontal bar" was selected - it corresponds to column index (link) 1733 = geological era.

The corresponding row indices (articles) are, in sequence:

SILURIAN PERIOD
PLEISTOCENE EPOCH
HOLOCENE EPOCH
PRECAMBRIAN TIME
CARBONIFEROUS PERIOD
OLIGOCENE EPOCH

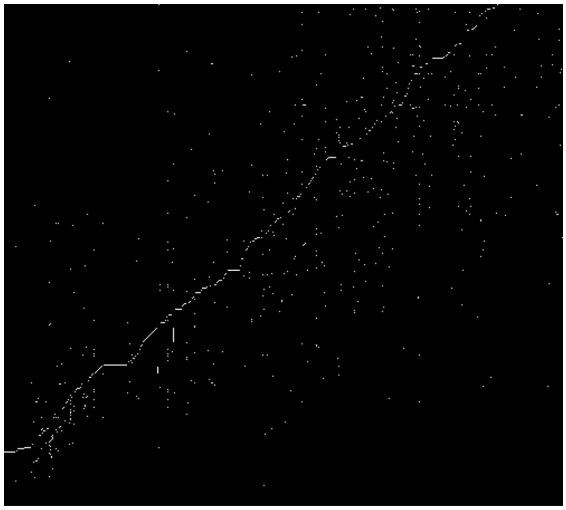


Figure 11: Part (500 × 450) of original encyclopaedia incidence data array.

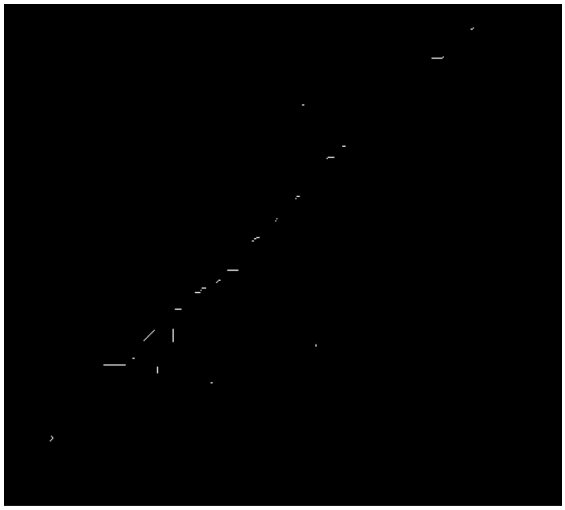


Figure 12: End-product of the filtering of the array shown in Figure 11.

ORDOVICIAN PERIOD
TRIASSIC PERIOD
CENOZOIC ERA
PALEOCENE EPOCH
MIOCENE EPOCH
DEVONIAN PERIOD
PALEOZOIC ERA
JURASSIC PERIOD
MESOZOIC ERA
CAMBRIAN PERIOD
PLIOCENE EPOCH
CRETACEOUS PERIOD

The work described here is based on a number of technologies: (i) data visualization techniques; (ii) the wavelet transform for data analysis; and (iii) data matrix permuting techniques. The wavelet transform has linear computational cost in terms of image row and column dimensions, and is independent of the pixel values.

CONCLUSIONS

Viewed from a commercial or managerial perspective, one could justifiably ask where we are now in our understanding of problems in this area relative to where we were back in the 1960s? Depending on our answer to this, we may well proceed to a second question: Why have all important problems not been solved by now in this area - are there major outstanding problems to be solved?

As described in this chapter, a solid body of experimental and theoretical results has been built up over the last few decades. Clustering remains a requirement that is a central infrastructural element of very many application fields.

There is continual renewal of the essential questions and problems of clustering, relating to new data, new information, and new environments. There is no logjam in clustering research and development simply because the rivers of problems continue to broaden and deepen. Clustering and classification remain quintessential issues in our computing and information technology environments. [113]

ACKNOWLEDGMENTS

Some of this work, in particular in the sections on Nearest Neighbor Finding on Graphs, K-Means and Family and Fast Model-Based Clustering, represents various collaborations with the following: J.L. Starck, CEA; A. Raftery, University of Washington; C. Fraley, University of Washington and MathSoft Inc.; D. Washington, MathSoft, Inc.; Ph. Poinçot and S. Lesteven, Strasbourg Observatory; D. Guillaume, Strasbourg Observatory, University of Illinois and NCSA.

LITERATURE AND NOTES

[1] Dasarathy, B.V., *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, IEEE Computer Society Press, New York, **1991**.

[2] Bennett, K.P., Fayyad, U. and Geiger, D., *Density-based indexing for approximate nearest neighbor queries*, Microsoft Research Technical Report MSR-TR-98-58, **1999**.

[3] Murtagh, F.; and Heck, A. *Multivariate Data Analysis*, Kluwer Academic, Dordrecht, **1987**.

[4] Arabie, P.; Hubert, L. J.; De Soete, G.; Eds., *Clustering and Classification*, World Scientific, Singapore, **1996**.

[5] Arabie, P.; Hubert, L. J. *An Overview of Combinatorial Data Analysis*, in Arabie, P.; Hubert, L. J.; De Soete, G. Eds, *Clustering and Classification*, World Scientific, Singapore, **1996**, 5.

[6] Murtagh, F. *Search Algorithms for Numeric and Quantitative Data* in Heck, A.; Murtagh, F. Eds, *Intelligent Information Retrieval: The Case of Astronomy and Related Space Sciences*, Kluwer Academic, Dordrecht, **1993**, 49.

[7] Delannoy, C. *RAIRO Informatique/Computer Science*, **1980**, 14, 275.

[8] Bentley, J. L.; Weide, B. W.; Yao, A. C. *ACM Transactions on Mathematical Software*, **1980**, 6, 563.

[9] Schreiber, T. *Efficient Search for Nearest Neighbors*, in Weigend, A. S.; Gershenfeld, N. A. Eds, *Predicting the Future and Understanding the Past: A Comparison of*

Approaches, Addison-Wesley, New York, **1993**.
[10] Rohlf, F. J. *Information Processing Letters*, **1978**, 7, 44.
[11] Murtagh, F. *Information Processing Letters*, **1983**, 16, 237.
[12] Friedman, J. H.; Bentley, J. L.; Finkel, R. A. *ACM Transactions on Mathematical Software*, **1977**, 3, 209.
[13] Bentley, J. L.; Friedman, J. H.*IEEE Transactions on Computers*, **1978**, C-27, 97.
[14] Broder, A. J. *Pattern Recognition*, **1990**, 23, 171.
[15] Weiss, S. F. *A Probabilistic Algorithm for Nearest Neighbor Searching*, in R.N. Oddy, R. N. *et al.*, Eds, *Information Retrieval Research,* Butterworths, London, **1981**, 325.
[16] Eastman, C. M.; Weiss, S. F. *Information Systems*, **1982**, 7, 115.
[17] Moore, A. *Advances in Neural Information Processing Systems*, 11, **1999**.
[18] Pelleg, D.; Moore, A. *Accelerating exact k-means algorithms with geometric reasoning*, Proceedings KDD-99, Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August, San Diego, **1999**.
[19] Friedman, J. H.; Baskett, F.; Shustek, L. J. *IEEE Transactions on Computers*, **1975**, C-24, 1000.
[20] Marimont, R. B.; Shapiro, M. B. *Journal of the Institute of Mathematics and its Applications*, **1979**, 24, 59.
[21] Kittler, J. *Kybernetes,* **1978**, 7, 313.
[22] Yunck, T. P. *IEEE Transactions on Systems, Man, and Cybernetics*, **1976**, SMC-6, 678.
[23] Richetin, M.; Rives, G.; Naranjo, M. *RAIRO Informatique/Computer Science*, **1980**, 14, 369.
[24] Burkhard, W. A.; Keller, R. M. *Communications of the ACM*, **1973**, 16, 230.
[25] Shapiro, M. *Communications of the ACM*, **1977**, 20, 339.
[26] Hodgson, M.E., *Remote Sensing of Environment*, **1988**, 25, 117.
[27] Vidal Ruiz, E. *Pattern Recognition Letters*, **1986**, 4, 145.
[28] Micó, L.; Oncina, J.; Vidal, E. An algorithm for finding nearest neighbors in constant average time with a linear space complexity, in *11th International Conference on Pattern Recognition, Volume II*, IEEE Computer Science Press, New York, **1992**, 557.
[29] Ramasubramanian, V.; Paliwal, K. K., *Pattern Recognition Letters*, **1992**, 13, 471.
[30] Fukunaga, K.; Narendra, P. M. *IEEE Transactions on Computers*, **1975**, C-24, 750.
[31] Kamgar-Parsi, B.; Kanal, L. N. *Pattern Recognition Letters*, **1985**, 3, 7.
[32] Niemann, H.; Goppert, R., *Pattern Recognition Letters*, **1988**, 7, 67.
[33] Beyer, K.; Goldstein, J.; Ramakrishnan, R.; Shaft, U. When is nearest neighbor meaningful?, in *Proceedings of the 7th International Conference on Database Theory (ICDT)*, Jerusalem, Israel, **1999**.
[34] Kushilevitz, E.; Ostrovsky, R.; Rabani, Y. Efficient search for approximate nearest neighbors in high-dimensional spaces", *Proc. of 30th ACM Symposium on Theory of Computing (STOC-30)*, **1998**.
[35] Croft, W. B. *Journal of the American Society for Information Science*, **1977**, 28, 341.
[36] Murtagh, F. *Information Processing Letters*, **1983**, 16, 237.
[37] Smeaton, A. F.; van Rijsbergen, C. J. *ACM SIGIR Forum*, **1981**, 16, 83.
[38] Perry, S. A.; Willett, P. *Journal of Information Science*, **1983**, 6, 59.
[39] Horowitz, E.; Sahni, S. *Fundamentals of Computer Algorithms*, Chapter 4 *The Greedy Method*, Pitman, London, **1979**.
[40] Gordon, A. D. *Classification*, 2nd ed., Champman and Hall, **1999**.
[41] Jain, A. K.; Dubes, R. C. *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, **1988**.
[42] Murtagh, F. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **1992**, 14, 1056.
[43] Sibson, R. *Computer Journal*, **1973**, 16, 30.
[44] Defays, D. *Computer Journal*, **1977**, 20, 364.
[45] de Rham, C. *Les Cahiers de l'Analyse des Données*, **1980**, V, 135.
[46] Juan, J. *Les Cahiers de l'Analyse des Données*, 1982, VII, 219.
[47] Murtagh, F. *Multidimensional Clustering Algorithms*, Physica-Verlag, Würzburg, **1985**.
[48] Bruynooghe, M. Statistique et Analyse des Données, **1977**, no. 3, 24.
[49] Murtagh, F. *Computational Statistics Quarterly*, **1984**, 1, 101.
[50] Day, W.H.E.; Edelsbrunner, H. *Journal of Classification*, **1984**, 1, 7..
[51] Sneath, P.H.A.; Sokal, R.R., *Numerical Taxonomy*, W.H. Freeman, San Francisco, **1973**.
[52] Willett, P. *Journal of Documentation*, **1989**, 45, 1.
[53] Gillet, V. J.; Wild, D. J.; Willett, P.; Bradshaw, J. *The Computer Journal*, **1998**, 41, 547.
[54] Griffiths, A.; Robinson, L. A.; Willett, P. *Journal of Documentation*, **1984**, 40, 175.
[55] White, H. D.; McCain, K. W.; in M.E. Williams, Ed., *Annual Review of Information*

Science and Technology (ARIST), **1997**, Vol. 32, 99.
[56] Rohlf, F. J.; *Information Processing Letters*, **1978**, 7, 44.
[57] Rohlf, F. J.; *The Computer Journal*, **1973**, 16, 93.
[58] Zahn, C. T. *IEEE Transactions on Computers*, **1971**, C-20, 68.
[59] Murtagh, F. in Sandqvist, Aa.; Ray, T. P. Eds., *Central Activity in Galaxies: From Observational Data to Astrophysical Diagnostics*, Springer-Verlag, Berlin, **1993**, pp. 209-235.
[60] Yao, A. C. *Information Processing Letters*, **1975**, 4, 21.
[61] Cheriton, D.; Tarjan, D. E. *SIAM Journal on Computing*, **1976**, 5, 724.
[62] Gabow, H. N.; Galil, Z.; Spencer, T.; Tarjan, R. E. *Combinatorica*, **1986**, 6, 109.
[63] Motwani, R.; Raghavan, P. *Randomized Algorithms*, Cambridge University Press, **1995**.
[64] Tucker, A. *Applied Combinatorics*, Wiley, New York**, 1980**.
[65] Tarjan, R. E.*Information Processing Letters*, **1983**, 17, 37.
[66] Salton, G.; McGill, M. J. *Introduction to Modern Information Retrieva*l, McGraw-Hill, New York, **1983**.
[67] Broder, A. Z.; Glassman, S. C.; Manasse, M. S.; Zweig, G.;*Proc. Sixth International World Wide Web Conference*, **1997**, 391.
[68] Broder, A. Z. In Compression and Complexity of Sequences (SEQUENCES'97), pp. 21-29, IEEE Computer Society, **1998**.
[69] Borodin, A.; Ostrovsky, R.; Rabani, Y. "Subquadratic approximation algorithms for clustering problems in high dimensional spaces", *Proc. 31st ACM Symposium on Theory of Computing (STOC-99)*, **1999**.
[70] Lloyd, P. "Least squares quantization in PCM." Technical note, Bell Laboratories, **1957**. Published in *IEEE Transactions on Information Theory*, **1982**.
[71] Forgy, E. *Biometrics*, **1965**, 21, 768.
[72] MacQueen, J., Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, pp. 281-297, Berkeley, University of California Press, **1976**.
[73] Darken, C.; Moody, J. "Note on learning rate schedules for stochastic optimization", *Advances in Neural Information Processing Systems 3*, Morgan Kaufmann, Palo Alto, **1991**.
[74] Darken, C.; Moody, J. "Towards faster stochastic gradient search", *Advances in Neural Information Processing Systems 4*, Morgan Kaufman, San Mateo, **1992**.
[75] Darken, C.; Chang, J.; Moody, J. "Learning rate schedules for faster stochastic gradient search", *Neural Networks for Signal Processing 2, Proceedings of the 1992 IEEE Worksho*p, IEEE Press, Piscataway, **1992**.
[76] Fritzke, B., "Some competitive learning methods", http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/JavaPaper
[77] Späth, H. *Cluster Dissection and Analysis: Theory, Fortran Programs, Examples*, Ellis Horwood, Chichester**, 1985**.
[78] Neal, R; Hinton, G. in M. Jordan, Ed., *Learning in Graphical Models*, Kluwer, Dordrecht, **1998**, pp. 355-371.
[79] Sato, M.; Ishii, S. in *Advances in Neural Information Processing Systems 11*, Kearns, M. S.; Solla, S. A.; Cohn, D. A. Eds., pp. 1052-1058, MIT Press, Cambridge, **1999**.
[80] Thiesson, B.; Meek, C.; Heckerman, D. "Accelerating EM for large databases", *Microsoft Research Technical Report MST-TR-99-31*, **1999**.
[81] Banfield, J. D.; Raftery, A. E. *Biometrics*, **1993**, 49, 803.
[82] Dasgupta, A.; Raftery, A.E. *Journal of the American Statistical Association*, **1998**, 93, 294.
[83] Murtagh, F.; Raftery, A. E. *Pattern Recognition*, **1984**, 17, 479.
[84] Banerjee, S.; Rosenfeld, A. *Pattern Recognition*, **1993**, 26, 963.
[85] Dempster, A. P.; Laird, N. M.; Rubin, D. B. *Journal of the Royal Statistical Society, Series, B* **1977**, 39, 1.
[86] Fraley, C. *SIAM Journal of Scientific Computing*, **1999**, 20, 270.
[87] Kass, R.E.; Raftery, A. E. *Journal of the American Statistical Association*, **1995**, 90, 773.
[88] Schwarz, G. *The Annals of Statistics*, **1978**, 6, 461.
[89] Fraley, C.; Raftery, A. E. *The Computer Journal*, **1998**, 41, 578.
[90] Mukherjee, S.; Feigelson, E. D.; Babu, G. J.; Murtagh, F.; Fraley, C.; Raftery, A. *The Astrophysical Journal*, **1998**, 508, 314.
[91] Celeux, G.; Govaert, G.; *Pattern Recognition*, **1995**, 28, 781.
[92] Sibson, R. *The Computer Journal*, **1973**, 16, 30.
[93] SDSS, Sloan Digital Sky Survey, http://www.sdss.org/
[94] Starck, J. L.; Murtagh, F.; Bijaoui, A. *Image and Data Analysis: The Multiscale Approach*, Cambridge University Press, New York, **1998**.

[95] Murtagh, F.; Starck, J. L.; *Pattern Recognition*, **1998**, 31, 847.
[96] Kolaczyk, E. D.; *Astrophysical Journal*, **1997**, 483, 340.
[97] Jammal, G.; Bijaoui, A. "Multiscale image restoration for photon imaging systems", *SPIE Conference on Signal and Image Processing: Wavelet Applications in Signal and Image Processing VII*, July **1999**.
[98] Zheng, G.; Starck, J. L.; Campbell , J. G.; Murtagh, F.; *Journal of Computational Intelligence in Finance*, 7, **1999**.
[99] Byers, S. D.; Raftery, A. E.; *Journal of the American Statistical Association*, **1998**, 93, 577.
[100] Allard, D.; Fraley, C.; *Journal of the American Statistical Association*, **1997**, 92, 1485.
[101] Ebeling, H.; Wiedenmann, G. *Physical Review E*, **1993**, 47, 704.
[102] Dobrzycki, A.; Ebeling, H.; Glotfelty, K.; Freeman, P.; Damiani, F; Elvis, M; Calderwood, T. *Chandra Detect 1.0 User Guide*, Chandra X-Ray Center, Smithsonian Astrophysical Observatory, Version 0.9, **1999**.
[103] Doyle, L. B. *Journal of the ACM*, **1961**, 8, 553.
[104] Murtagh, F. ; Hernández-Pajares, M., *Journal of Classification*, **1995**, 12, 165.
[105] Poinçot, Ph.; Lesteven, S.; Murtagh, F., *Astronomy and Astrophysics Supplement*, **1998**, 130, 183.
[106] Poinçot, Ph.,;Lesteven, S.; Murtagh, F. *Journal of the American Society for Information Science*, **2000**, 51,1081.
[107] Guillaume, D.; Murtagh, F. *Computer Physics Communications*, **2000**, 127, 215..
[108] Cartia, Inc.*, Mapping the Information Landscape, client-server software system*, http://www.cartia.com/, **1999**.
[109] Church, K. W.; Helfman, J. I. *Journal of Computational and Graphical Statistics*, **1993**, 2, 153.
[110] Berry, M.W.; Hendrickson, B.; Raghavan, P. in Renegar, J., Shub, M. and Smale, S., Eds., *Lectures in Applied Mathematics (LAM) Vol. 32: The Mathematics of Numerical Analysis*, American Mathematical Society, **1996**, 99.
[111] Berry, M. W.; Drmač, Z.; Jessup, E. R. *SIAM Review*, **1999**, 41, 335.
[112] Murtagh, F.; Starck, J. L.; Berry M. *The Computer Journal*, **1999**, submitted.
[113] Murtagh, F. *The Computer Journal*, **1998**, 41, 517.