UNIVERSITY OF JOENSUU
COMPUTER SCIENCE
DISSERTATIONS 13

ISMO KÄRKKÄINEN

METHODS FOR FAST AND RELIABLE CLUSTERING

ACADEMIC DISSERTATION

To be presented, with the permission of the Faculty of Science of the University of Joensuu, for public criticism in Louhela Auditorium of the Science Park, Länsikatu 15, Joensuu, on June 2nd, 2006, at 12 noon.

UNIVERSITY OF JOENSUU
2006

Supervisor	Professor Pasi Fränti Department of Computer Science University of Joensuu Joensuu, Finland
Reviewers	Professor Olli Nevalainen Department of Computer Science University of Turku Turku, Finland
	Professor Xiaowei Xu Information Science Department University of Arkansas at Little Rock Arkansas, USA
Opponent	Professor Laurence S. Dooley Faculty of Information Technology Monash University Victoria, Australia

ISBN	952-458-815-3	(printed)
ISBN	952-458-816-1	(PDF)
ISSN	1238-6944	(printed)
ISSN	1795-7931	(PDF)

Computing Reviews (1998) Classification: H.3.3, I.5.1, I.5.3, G.1.6

Yliopistopaino
Joensuu 2006

Methods For Fast And Reliable Clustering

Ismo Kärkkäinen
Department of Computer Science
University of Joensuu
P.O.Box 111, FIN-80101 Joensuu FINLAND
Ismo.Karkkainen@cs.joensuu.fi

University of Joensuu, Computer Science, Dissertations 13
Joensuu, 2006, 108 pages
ISBN 952-458-815-3 (printed)
ISBN 952-458-816-1 (PDF)
ISSN 1238-6944 (printed)
ISSN 1795-7931 (PDF)

Abstract

Clustering is used in many areas as a tool to inspect the data or to generate a representation of the data that is better suited to the application.

In this work, different parts related to clustering are studied. Focus is mainly on making the algorithms faster while preserving reliability. Attention is paid to the ability of the algorithms to find the number of clusters. Binary data sets and large data sets present problems for clustering algorithms. Some improvements for handling these cases are proposed.

For a fixed number of clusters, the clustering problem can be solved in a fast and reliable manner, but the problem changes when the number of clusters is unknown. Performing the clustering repeatedly is no longer fast. The proposed solutions to performing clustering rapidly involve reusing the results of previous work and focusing the search to more promising model sizes. Using the previous results as a starting point improves the speed of clustering when solving the clustering for the next model size. Performing the search so that the model size is optimized along with the model produces much greater speed-up. This is due to less work is done on the models of much larger or smaller size than the number of clusters in the data.

Binary data causes problems for certain clustering algorithms. These problems are addressed by changing the distance function. One proposed method is designed for a specific clustering criterion. The distance function is used to decide how to best improve the value of the criterion locally at each step of the algorithm. The second proposed method changes the distance function gradually from L_∞ to L_1.

The proposed algorithm for large data sets converts the data into a model using only one pass over the data. The data need not be stored in memory. In this way, the data can be processed much faster, and without excessive memory consumption.

Keywords: clustering, number of clusters, binary data, distance function, large data sets, centroid model, Gaussian mixture model, unsupervised learning.

Acknowledgments

The work done for this thesis has been carried out at the Department of Computer Science, University of Joensuu, Finland, during the years 2000-2006. During 2000-2002 I was an assistant at the department, and from 2003 on I held position funded by East Finland Graduate School in Computer Science (ECSE).

I owe thanks to my supervisor, Professor Pasi Fränti, for criticism, ideas and support during my studies. I thank the co-authors of some parts of this thesis, Dr Tomi Kinnunen and Dr Mantao Xu for their help.

I express my gratitude to Professor Olli Nevalainen and Professor Xiaowei Xu, the reviewers of this thesis, for their helpful comments and recommendations. Finally, everyone else who has read the thesis in part or in full, deserves thanks for their comments.

Joensuu, 3rd of May, 2006
Ismo Kärkkäinen

List of original publications

P1. I. Kärkkäinen, P. Fränti: “Stepwise Algorithm for Finding Unknown Number of Clusters,” in *Proceedings of ACIVS 2002 (Advanced Concepts for Intelligent Vision Systems)*, Ghent, Belgium, September 9-11, 2002, pp 136–143.

P2. I. Kärkkäinen, P. Fränti: “Dynamic local search for clustering with unknown number of clusters,” in *Proceedings of the 16th International Conference on Pattern Recognition*, Québec City, QC, Canada, August 11-15, 2002, pp. 240–243.

P3. I. Kärkkäinen, P. Fränti: "Minimization of the value of Davies-Bouldin index,” in *Proceedings of the IASTED International Conference on Signal Processing and Communications (SPC'2000)*, Marbella, Spain, 2000, pp. 426–432.

P4. P. Fränti, M. Xu, I. Kärkkäinen: “Classification of binary vectors by using DeltaSC-distance to minimize stochastic complexity,” *Pattern Recognition Letters*, 24 (1-3), 2003, pp. 65–73.

P5. I. Kärkkäinen, P. Fränti: “Variable Metric for Binary Vector Quantization,” in *Proceedings of IEEE International Conference on Image Processing (ICIP'04)*, Singapore, vol. 3, October 2004, pp. 3499–3502.

P6. I. Kärkkäinen, P. Fränti: “Gradual Model Generator for Single-pass Clustering,” in *Proceedings of the Fifth IEEE International Conference on Data Mining*, Houston, Texas, November 2005, pp. 681–684.

P7. T. Kinnunen, I. Kärkkäinen, P. Fränti: "Is Speech Data Clustered? -Statistical Analysis of Cepstral Features,” in *Proceedings of 7th European Conference on Speech Communication and Technology (EUROSPEECH 2001)*, vol. 4, Aalborg, Denmark, Sept. 3-7, 2001, pp. 2627–2630.

Contents

1 Introduction

Clustering is a problem, where the user has a collection of *objects* and she wants to place them into groups, or *clusters*, so that some condition is satisfied. Objects to be clustered can belong to just one cluster, or to several [31]. If all objects belong to the same cluster, we can conclude that the data is not clustered.

The number of clusters is usually unknown. We may have assumptions about what the clusters might be like, for example, spherical or ellipsoidal. Clusters can also be of arbitrary-shape, in which case the objects inside the same cluster share some property with at least one other object in the cluster. Alternatively, we may enforce certain restrictions that arise from the application. Assumption such as spherical clusters may not hold for the data, but it may not be important for the application. Testing the validity of the assumptions can be part of the process.

Clustering is *unsupervised learning* [53], meaning that the user of the algorithm does not affect the outcome of the algorithm during the clustering process. Clustering algorithm should be able to obtain general information from the objects. With regard to the information that is available as input, clustering is close to the *classification* problem, with the significant difference that in classification we have explicit information about the classes the objects belong to, whereas in clustering no a priori information is available. Hence, in classification the object classes have already been defined, and we wish to be able to represent them in an efficient and accurate manner. In clustering the problem is to find the clusters, or observe the lack of any clusters, and usually to find a proper representation for the clusters. The difference is illustrated Figure 1, three classes of a classification problem are shown with different symbols in the left panel of the figure, and the corresponding clustering problem with no class information in the right panel.

Quite often clustering algorithms are used to generate a model for the data, even though the data is not clustered. Clustering algorithms have a tendency to find structure from the data, despite the lack of any structure. For example when producing a model for speaker recognition by using *vector quantization* (VQ), the qualitative differences between the results of different algorithms are small [61]. This is explained by the lack of clustering structure in the data computed from speech signal [P7].

The fields where clustering algorithms are used are quite diverse. For example, in gene research expressed sequence tags are clustered [91, 111]. In image processing, clustering is used widely in image segmentation [96], and detection of line segments from an image [117]. An environment map can be clustered in order to find topological features [75]. The features are connected sub-graphs found in the graph that describes the entire terrain. The features are then intended to be used in path-planning of autonomous mobile systems on the clustered map. In astronomy, clustering is used to distinguish between stars and galaxies, and between galaxies of different type, such as spiral and elliptical [18]. Remote-sensing data is clustered to produce a climatic classification [101].

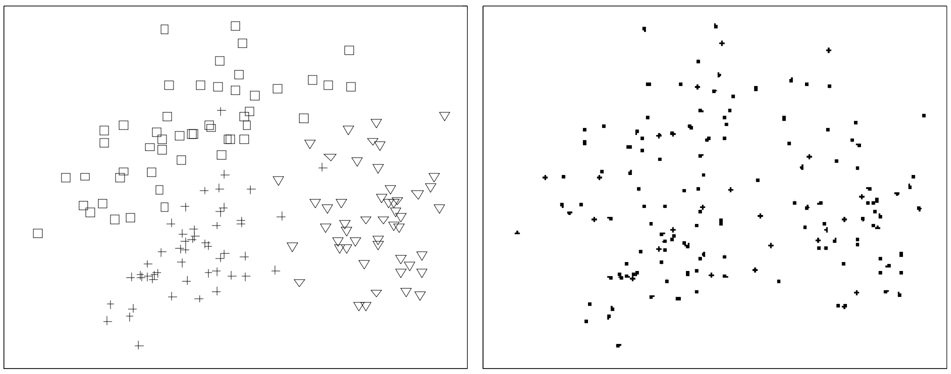
Figure 1: Classification problem (left) and a clustering problem (right).

2 Clustering problem

In this chapter, the clustering problem setup is first described followed by descriptions of clustering and clustering criteria. Once the user decides what kind of representation will be used for the clustering, the problem becomes better defined. The potential drawback is that the decision has been incorrect, and the results of clustering can then be misleading.

2.1 Problem setup

The input for clustering is a data set X of N objects $\{x_1, x_2, \ldots, x_N\}$. In all of the following formulas, x_i stands for a row of a matrix, or row vector, whenever the distinction between row and column vectors is significant. The output of clustering is usually either a division of the data set into K subsets, or a representation for each cluster. K is generally referred to as the number of clusters regardless of whether we want to find the clusters, or to split the data set into subsets that satisfy the requirements of the application.

A data object is an ordered set of *attributes*, or *features*, or variables. The attributes need not be of same type. Attributes of *nominal* type have a set of unordered values. Values of *ordinal* type have an order. *Interval* data consists of values that have an order but it is not possible to say that some value is twice as much as some other value when the attribute represents date, for example. *Ratio* data, such as distances or weights [100], allows such statements. For objects consisting of the latter two types, the objects can be described using a D-dimensional real-valued vector.

Clustering algorithms commonly require a measure of difference, in some cases a measure of similarity or closeness. Three types of measures of difference or similarity can occur. Between objects, between parts of the representation of clustering, and between objects and representation of clustering. *Euclidean distance* fulfills all three roles for *centroid model*, and will be described in Chapter 2.2.1. For the *Gaussian mixture model*, described in Chapter 2.2.2, the three measures are different. There exists a vide variety of distance and similarity functions for various purposes [100, 58, 31]. Selection of proper distance or similarity is one thing to which the user should pay attention.

We will refer to whatever measure of dissimilarity is in use as *distance function* or *distance*. Measure of similarity is likewise referred to as *similarity function* or *similarity*. Measure of closeness is appropriate when an object and a cluster representation are compared. Cluster representation needs not be identical to the object representation.

Therefore, similarity is not appropriate. In this case, the measure quite often determines the membership of an object in a cluster.

2.2 Cluster representation

A cluster is usually thought to have the property that objects belonging to it are more similar to each other than to the other objects. This simple definition sounds reasonable but it leaves a lot of room for interpretation.

Membership of an object in a cluster is usually expressed in relative terms. When the membership is 1, the object belongs to exactly one cluster and has membership 0 in all other clusters. We can express the memberships of N objects in K clusters with an $N \times K$-matrix M. Usually there is a restriction that each object's memberships in clusters sum to 1:

$$\sum_k m_{nk} = 1 \tag{1}$$

No membership values are negative: $m_{nk} \in [0, 1]$ for all $n = 1, 2, \ldots N$, $k = 1, 2, \ldots, K$.

Representing the membership using a matrix is quite space-consuming for large N and relatively large K. It is, however, quite general, since the relationships between objects do not affect how the objects can be divided between clusters. If we make some assumptions about the clusters themselves, we can use more compact representations. A common assumption is that each object belongs to only one cluster. Exactly one m_{nk} is 1 for an index n and the rest of the row elements of M are 0. In that case we only need to store for each object the index k of the matrix entry with value 1. Allowing membership values of only 0 or 1 is called *hard* or *crisp* clustering.

In contrast, terms *soft* or *fuzzy* clustering are used if objects can belong to several clusters. The term fuzzy clustering is used when algorithms utilizing fuzzy sets [118] are used. In other cases the membership values are usually related to the probability that an object belongs to the specific cluster. Numerically they may seem the same but there is a theoretical difference in their definition.

Assumptions about the relationships between objects may allow us to use considerably less space-consuming representations than listing membership values for each object. If, for every data object, the nearby objects also belong to the same clusters with very similar membership then we can represent a cluster with a collection of objects that are surrounded by objects that have the similar membership values. This requires that we can measure the distance between objects. If the objects reside in a vector space it is possible to perform computations with them and compute parameters of distributions. Then we can represent one cluster with a mixture of distributions. Commonly one cluster is represented with just one component.

When each cluster can be represented by a model consisting of a collection of objects or a mixture of distributions, then it is possible to state for any object what would be the cluster it belongs to, by measuring the distances to the models. Hence the model provides independence of the data that was used to generate the model, allowing for a more general statements about the clusters than just stating which object belongs to which cluster. The model can also be used to classify objects that the user may have in the future.

The membership matrix M is rarely used except as an intermediate representation that allows for computation of a model. However, for the special case of each object belonging to exactly one cluster, and a small number of objects, presenting the cluster labels can be useful. Labels and information about the possible clusters can be represented in the form of a dendrogram, a tree that shows the order in which objects could be joined together into clusters. The height at which two branches representing clusters or individual objects are joined together indicates the cost of joining the clusters or objects, providing a visualization of the clustering.

In chapters 2.2.1 and 2.2.2 the centroid model and Gaussian mixture model are described in more detail. Models can also consist of lines, planes or shells [34], for example.

2.2.1 Centroid model

The centroid model is a simple model that can be used for data in vector space. The assumption made about the data is that all clusters are spherical with the same variance. The model is an ordered set $C = \{ c_1, c_2, \ldots, c_K \}$of K centroid vectors. If needed, the information about how many objects are mapped into each centroid, n_k, can be stored. Then we can consider the model as a histogram, in which the Voronoi cell around each centroid vector represents one histogram bin. Each data object is mapped to the nearest centroid. Let P be a mapping from data objects to centroids, or *partitioning*. Let p_j indicate the centroid vector to which object j is mapped. The centroid, or mean vector of all objects with the same p_j is:

$$c_k = \frac{1}{n_k} \sum_{p_j = k} x_j \ . \tag{2}$$

Centroid model is widely used in vector quantization, where it is called a *codebook*. Euclidean distance is widely used measure of difference:

$$d_E(x, y) = \sqrt{(x - y)(x - y)^T} \ .$$

Euclidean distance is usable when the objects lie in vector space. It can be used as a distance between objects, between centroids and between an object and a centroid, since the objects and centroids have same type.

The model consisting of representative objects is similar to the centroid model. There is no requirement that the objects can be summed or scaled, and the objects of the model are a subset of the data set.

Figure 2 shows an example of a centroid model of size 15. Gray dots are data vectors.

2.2.2 Gaussian mixture model

Gaussian mixture model (GMM) is related to centroid model, but it is used to represent probability density distribution. It is also restricted to attributes with numeric values like the centroid model. The main difference is that data objects belong to all components of the model with varying degree of membership. As a result of this, each component has a weight, rather than the number of objects mapped to it. These weights are normalized to sum to 1. An example of a GMM of size 15 is shown in Figure 3. One component of a GMM consists of mean vector, covariance matrix and component weight. Given memberships, or *a posteriori* probabilities, of *j*th vector to *k*th component, m_{jk}, the mixing weight, or *a priori* probability, of *k*th component, is:

$$w_k = \frac{\sum_j m_{jk}}{\sum_k \sum_j m_{jk}} \tag{3}$$

When there are no noise clusters or similar things involved, the double sum in the divisor is N, due to the restriction of Equation (1). When all memberships are 0 or 1, equation (3) then equals n_k / N.

The mean vector of the component is:

$$c_k = \frac{\sum_j m_{jk} x_j}{\sum_j m_{jk}} \tag{4}$$

The above equation equals to (2) in case all memberships are either 0 or 1. A necessary parameter of the GMM is also the covariance matrix, which can be computed as a weighted sum of the outer products of differences of data and mean vectors:

$$v_k = \frac{\sum_j m_{jk} \left(x_j - c_k\right)^T \left(x_j - c_k\right)}{\sum_j m_{jk}} \tag{5}$$

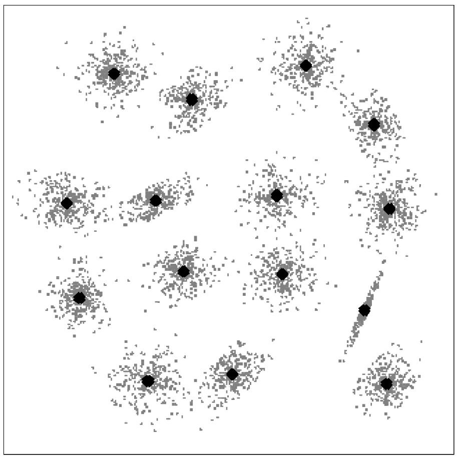
Figure 2: Example of a codebook.

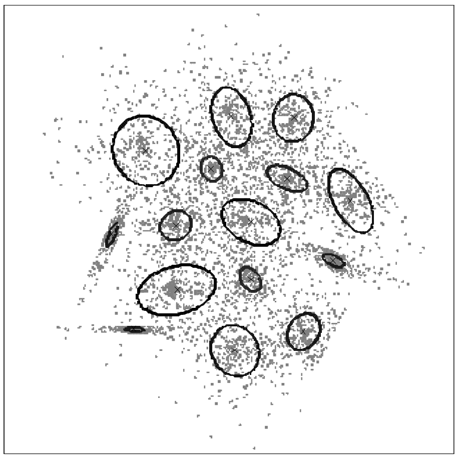
Figure 3: Example of a GMM.

We only need to compute the diagonal values of the outer product when we know that the variables are independent, or we do not care about the covariances. If variances of different variables are expected to be equal in practice, then we can compute the squared mean of the standard deviations. The user may know that the variances can be left out if she has prior experience with the data. Leaving out variance information leaves essentially the centroid model.

The covariance matrix contains shape information about the distribution of objects from which it was computed. Mahalanobis-distance is a distance function that takes the shape information into account and here it is used to compute distance from a object to the mean of the component of a GMM. Inverse of the covariance matrix is used to negate the effect of shape:

$$d_M(x,y) = \sqrt{(x-y)v_k^{-1}(x-y)^T}$$

The above equation reduces to the Euclidean distance if the covariance matrix equals identity matrix. With the Mahalanobis-distance it is possible to compute the probability density of the distribution at location *x* with respect to *k*th component of the GMM.

$$P_k(x, c_k, v_k, w_k) = \frac{w_k e^{-\frac{1}{2}d_M(x,c_k)^2}}{\sqrt{(2\pi)^d |v_k|}} \tag{6}$$

The probability density at the location of an object with respect to the entire model is the sum of the densities of the individual components.

2.3 Clustering criteria

There are numerous functions that measure the quality of clustering. When the centroid model is used, a valid criterion for the error is *mean squared error* (MSE), which is calculated as:

$$MSE(X,C) = \frac{1}{N}\sum_j d(x_j, c_{p_j})^2$$

A related criterion is the *mean absolute error* (MAE), in which the distance is not squared.

While MSE indicates how well the centroid model represents the data, it is monotonically decreasing as the model size increases, see Figures 4 and 5. The data set used in Figure 4 is shown in Figure 2. Figure 3 shows the data set used in Figure 5. There is more or less a visible bend or “knee” in the graph of Figure 4 at model size 15, which would be the most suitable model size for this data set. In this data set the clusters are fairly well separated but in the data set used in Figure 5, the clusters overlap and there is no visible bend in the graph. Hence, finding such a bend cannot always be used as a criterion for finding the most suitable model size.

Research has been done in identifying the location of the knee in the graph. In [21] the ratio of subsequent distances between joined clusters is used alongside the values of the clustering criterion to determine the best model size. In [93] only the values of the clustering criterion are used. The values of the clustering criterion are needed for all model sizes in the desired range.

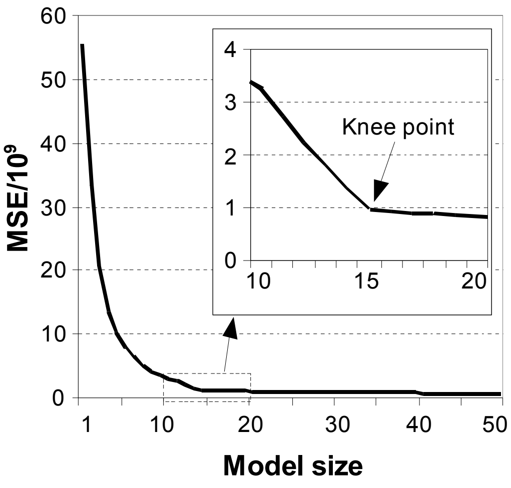

Figure 4: Plot of MSE with bend.

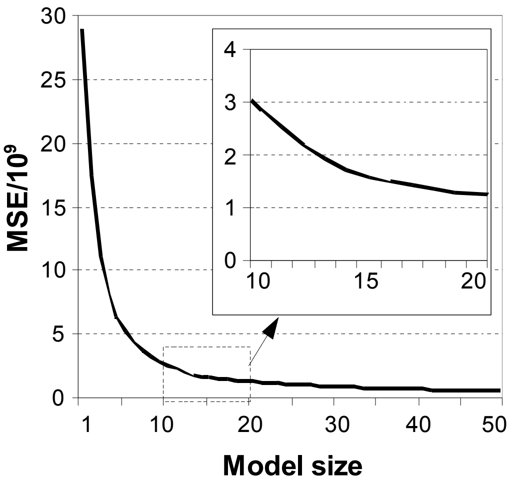

Figure 5: Plot of MSE without bend.

Likelihood is a measure similar to MSE for probability density distributions. Product of the probability densities at individual objects is computed instead of distances to components. For reasons of numerical precision, it is easier to compute logarithm of the likelihood. It can be divided by N, which corresponds to the logarithm of the Nth root of the likelihood and makes the *log-likelihoods* of different size data sets comparable. The equation of log-likelihood for GMM is:

$$LL(X,C,V,W) = \frac{1}{N}\sum_j \log \sum_k P_k\left(x_j, c_k, v_k, w_k\right) \tag{7}$$

Likelihood increases monotonically as the model size increases. Therefore the model size with maximum value does not indicate the proper model size.

Several criteria have been developed for the explicit purpose of finding the best model size, or the number of clusters. A common goal is maximizing the difference between groups and minimizing the difference within groups. Several criteria are listed in [80], but see also [58, 31, 89, 13, 12, 17, 42]. For GMMs, [79] contains a comparison of several criteria. Here we describe two criteria in detail since they have been utilized in publications [P1, P2, P3, P6, P7].

Davies-Bouldin index (DBI) [26] is designed to use a *dispersion measure*, such as MSE or MAE, and distance between clusters, which can simply be the distance between centroids or representative objects. Denoting the dispersion measure for cluster i with S_i and distance between clusters i and j with M_{ij}, the *cluster separation measure* R_{ij} is

$$R_{ij} = \frac{S_i + S_j}{M_{ij}}$$

The index to be minimized is the average of the maximum R_{ij} values for each i:

$$DBI = \frac{1}{K}\sum_i \max R_{ij}$$

Hence, the basic idea is to measure the separation of each cluster from its nearest neighbor so that dispersion inside clusters in minimized and distance to nearest centroid is maximized. Davies and Bouldin present a set of requirements that the dispersion measure and the cluster separation measure should have in order to be usable in DBI. Figure 6 shows graphs of DBI values for the data sets in Figures 2 (data set A) and 3 (data set B). For data set A, the index can clearly find the optimal number of clusters, but for data set B the index indicates that the proper model size is 14, not 15. Hence the number of clusters has been under-estimated.

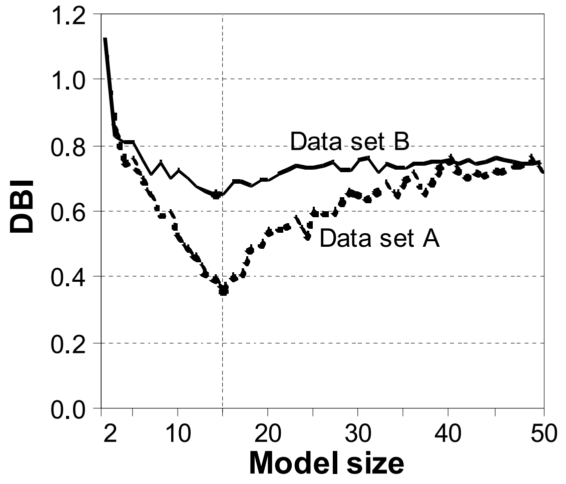

Figure 6: Example graphs of DBI.

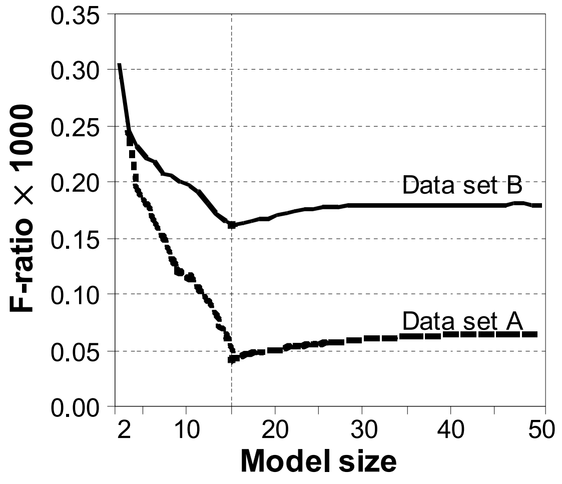

Figure 7: Example graphs of F-ratio.

There are several tests for clustering based on the *scatter matrices* computed from the data [31]. These originate from statistical analysis of variance. The *within scatter matrix* represents the scatter of objects inside cluster. *Between scatter matrix* represents the scatter of clusters. The within scatter matrix is the sum of outer products of differences of data objects and centroids:

$$W = \sum_n \left(x_n - c_{p_n}\right)^T \left(x_n - c_{p_n}\right) \tag{8}$$

In case of GMM, we can compute a scatter matrix using the covariance matrices of the components. Therefore, the original data is not necessarily needed. The matrices will not be identical since covariance matrices have been computed using membership probabilities instead of partitioning. The formula is:

$$W = N\sum_k w_k v_k$$

The between scatter matrix is computed using component means and the mean vector of the data. Using $w_k = n_k/N$ for centroid model gives:

$$B = \sum_k N w_k \left(c_k - \bar{x}\right)^T \left(c_k - \bar{x}\right). \tag{9}$$

Using the Equations (8) and (9) it is possible to compute *f-ratio* [52] with traces (sums of diagonal elements) of the matrices:

$$f = \frac{(K-1)tr(W)}{(N-K)tr(B)}. \tag{10}$$

By rearranging the terms of trace, one can see that the trace of W is the same as the sum of squared Euclidean distances from data vectors to nearest centroids. Likewise, the trace of B is the sum of squared Euclidean distances from cluster centroids to the mean vector of data. Thus, we can rewrite the formula as:

$$F = \frac{N(K-1)MSE(X,C)}{K(N-K)MSE(C,\bar{X})}.$$

Figure 7 shows a graph of F-ratio values for the data sets in Figures 2 and 3. The criterion indicates that there are 15 clusters for both data sets.

2.4 Preprocessing the data

The data can be processed before clustering. The need for this varies. Some features may have ranges that are orders of magnitude larger than those of other features, hence the contribution of the other features may be negligible without *normalization*. This can be usually done by subtracting the mean of the feature, and then dividing the result by standard deviation of that feature.

Dimensionality reduction can be done to the data by using *principal component analysis* (PCA) [55, 109] or *independent component analysis* (ICA) [51] for data that is not Gaussian. Dimensionality reduction using PCA might decrease the number of features significantly but it can also make interpretation of the results harder since the new features have no clear relation to the original ones. It has been observed [116] that the effect of PCA varies depending on which principal components the user chooses and which distance function is used. There is no single preprocessing technique that can be applied in every case, or a technique that is unquestionably suitable. Therefore, the decision to preprocess has to be made by the user. In clustering algorithms it is not presumed that any specific preprocessing is done, so preprocessing can be omitted.

An example of PCA and normalization is shown in Figure 8. The image on the left is a grayscale presentation of the original color image. In middle and left are the RGB triples from the image. Right of it is the same data reduced to two dimensions using PCA. On the right is the data after the standard deviations of both variables have been normalized to 1.

Objects that are far away from all other objects are called *outliers*, and they can decrease the quality of the clustering. Hence, the removal of outliers, or rather, detection if there are any outliers, may need to be done. However, this topic is quite wide and what is considered to be an outlier varies [76]. Outliers will also have an effect on normalization and dimensionality reduction. Therefore, the normalization may need to be re-done if normalization is a required step in the outlier detection process. Noise and outliers have been considered in clustering algorithms, for example in so-called *noise clustering* [23]. In noise clustering, the idea is that all objects are equally far from the noise cluster. Therefore, due to the restriction of Equation (1) in fuzzy or in probabilistic clustering, the objects that are far away from the model have smaller effect on the model since the objects are closest to the noise cluster. Another approach is to consider the farthest objects to be noise [83].

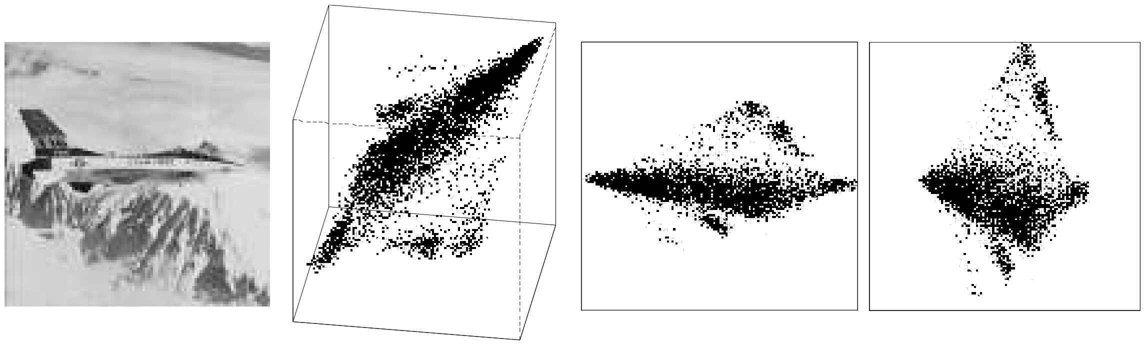
Figure 8: Example of data normalization. Original image (left), RGB color values (middle left), two principal components (middle right), and the normalized data (right).

3 Algorithms for a fixed number of clusters

After the decision about representation of clustering has been made there remains the question about the number of clusters. Fixing the number beforehand is a valid choice in cases where the clustering algorithm is used to obtain a representation that consumes less space but still represents the data well enough.

3.1 Algorithms for the centroid model

The most widely known algorithm for the centroid model is *k-means* [74]. The iterative algorithm alternates between making a partitioning from a centroid model and vice versa. Partitioning is made by mapping each object to the nearest centroid. The algorithm converges to a (locally) optimal solution. The algorithm is also known as *Generalized Lloyd algorithm* (GLA), *hard c-means* or *Linde, Buzo, Grey* (LBG) algorithm [72]. The simplest way to generate the initial model is to select objects randomly. More organized approach is splitting the data recursively and taking the means of the resulting partitions as the initial centroids [72].

The tendency of k-means to get stuck on local optima has been dealt with in various ways. Repeating the algorithm using different initial solution is a common method. Fritzke [35] proposed an LBG-U algorithm with deterministic swap operation. The swap operation moves the least useful centroid to a location near the centroid that contributes most to the total squared error. To find the least useful centroid, each centroid is removed in turn, MSE is computed, and the model with lowest MSE indicates the least useful centroid. Then k-means is iterated until convergence. In Figure 9 are seen the initial model, the models with one centroid removed in turn on the bottom row, the model after swap on middle of top row and the final model on top right corner. The whole procedure is iterated until the error after the swap and k-means has not decreased any more. The LBG-U algorithm is less sensitive to the initial model, but the deterministic swap can still get stuck on a local optimum.

A proposed solution to this is to use random swap to move a centroid to a location of an object in a local search algorithm [37], denoted RLS. Major improvement happens during the first few iterations of k-means. If there is no improvement, compared to the best model found so far, the new model is discarded and a new swap is performed starting from the currently best model. Pseudocode for the algorithm is presented in Figure 10.

Experimentally only two iterations of k-means was found to be sufficient. Result of random swap and two k-means iterations are shown in Figure 11. On the left is the initial model, in the middle the model after swap and on the right the model after the k-means iterations.

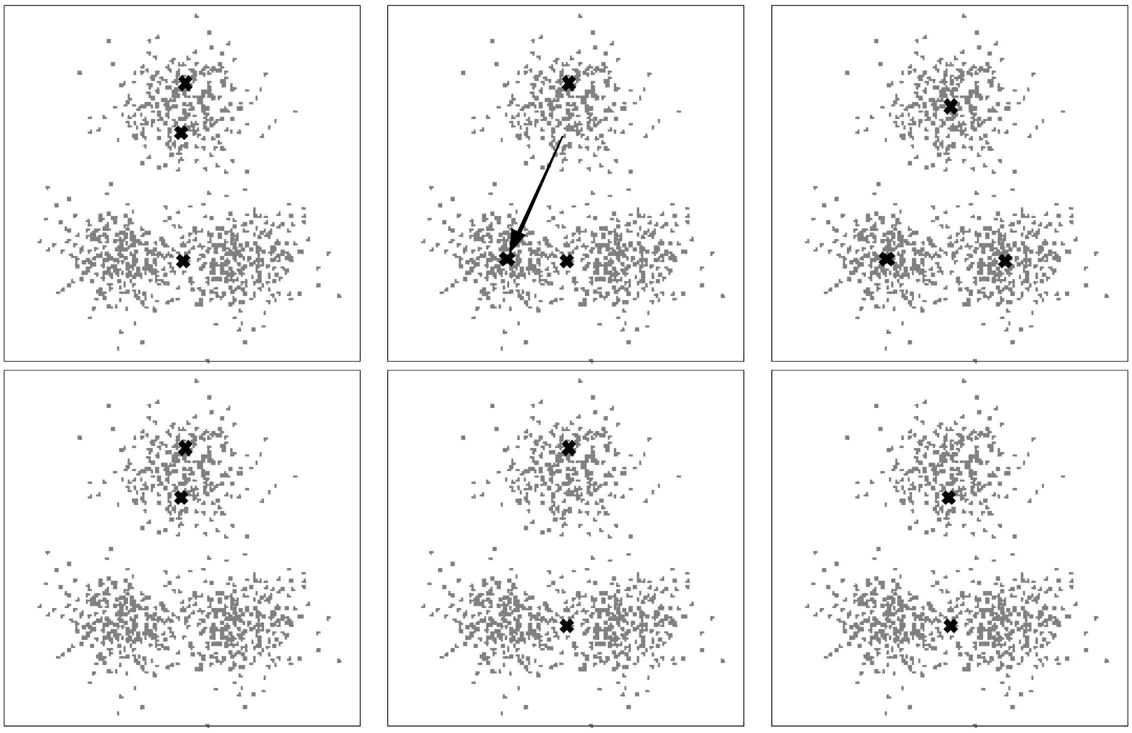
Figure 9: Example of LBG-U iteration. Initial model (top left), models used in determining utility (bottom row), after swap (top middle) and after k-means (top right).

```
C, P := RLS(X, C, P)
Repeat
    $C_{new}$, $P_{new}$ := C, P
    $C_{new}$ := Swap(X, $C_{new}$)
    $C_{new}$, $P_{new}$ := k-means(X, $C_{new}$, $P_{new}$)
    If Error($C_{new}$, $P_{new}$) < Error(C, P) Then
        C, P := $C_{new}$, $P_{new}$
Until stopping criterion is true
Return C, P
```

Figure 10: Pseudocode of the RLS algorithm.

There is no way to detect if the algorithm has converged to an optimal solution. The user must specify either a fixed number of iterations or otherwise determine when the algorithm has advanced close enough to the optimal solution so that further iterations would give no significant improvement.

An alternative to moving centroids is to add and remove them. Iterative split-and-merge algorithm [59] performs a number of split and merge operations on the clusters so that after splits and merges, the model size is the same as before the operations. For the leftmost data and model in Figure 11, the algorithm can split the lower cluster and then merge the two upper centroids into one. The final model is obtained after iterating k-means.

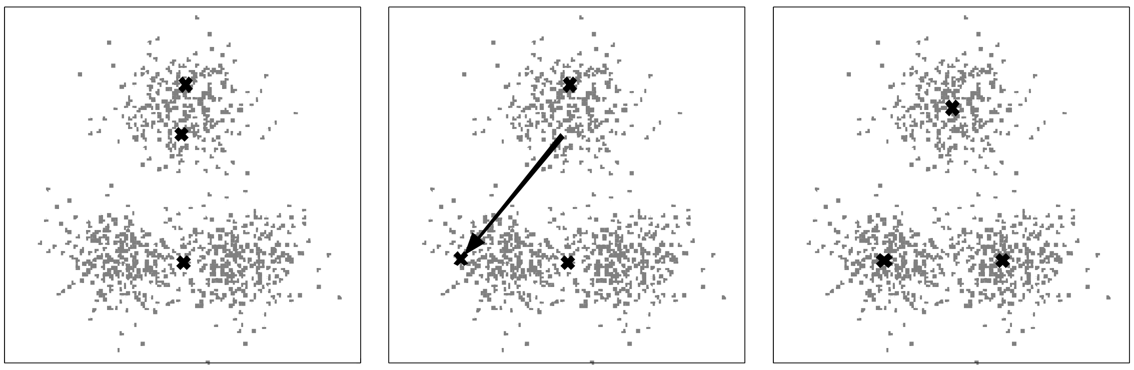
Figure 11: One iteration of RLS: initial model (left), after swap (middle) and after k-means (right).

Changes to the partitioning are proposed in [48]. Model is changed by first moving a number of objects to randomly selected partitions and then updating the model. After the update, an algorithm similar to RLS, but without k-means iterations, is used to fine-tune the model.

Approaches that change the entire model include stochastic relaxation [119], where noise is added to the centroid vectors. The amount of noise decreases as the algorithm is iterated. The noise is meant to allow the model to move from one optimum to another so that the global optimum will be reached. Deterministic annealing [92] starts with objects belonging to all groups with nearly equal membership values. The groups are gradually allowed to become non-overlapping by restricting the memberships to 0 and 1 in the end.

Changes to k-means to make it less sensitive to initial model tend to make the algorithm more complex, and increase the processing time. However, in cases where finding a good clustering is difficult, the additions reduce the need to repeat the algorithm with different initializations and thus decrease the required processing time. If it is easy to find a good clustering for particular kind of data then using a more advanced algorithm may not be necessary [61].

Clustering algorithms based on genetic algorithm (GA) [43] have also been proposed. Some utilize binary representation of the model [47]. Representation specific to the application is used in [77]. Combination of GA and k-means [87, 97] uses GA to search for global optimum and k-means to speed up the search by converging each member of the population to nearest local optimum.

There are cases where there is no possibility to add the vectors together and scale them. In place of centroids, we select the object that has the smallest sum of distances to other objects in the partition. This requires going through all pairs of objects inside each partition. This is done in an algorithm analogous to k-means called partitioning around medoids (PAM) [58], or k-medoids. It should be simple to convert an algorithm for centroid model to use representative objects if the algorithm relies on properties of vector space only in the centroid computations.

It is possible to operate on the matrix of distances between objects, if the objects themselves are used only in distance computations and selection of cluster representatives. A variant of using representative objects is to use only the values for each feature that the set of objects contains. For each variable, the allowed values are the values that occur within the set, and the representative object is constructed by selecting the most appropriate value for each feature. The value can be the most common value inside the partition, for example.

An variant of k-means algorithm for the case where the number of objects in each cluster must not exceed a predefined limit is given in [84]. Partitioning step is replaced by solving the transportation problem [3, 60]. As a result, only specified number of objects is mapped to each centroid. Prior knowledge about which objects belong to the same cluster and which do not is the constraint discussed in [108].

If DBI is used as the clustering criterion, the mean vector is not an optimal cluster representative, as noted in [P3]. Moving the centroid slightly away from the mean will improve the value of DBI, at the cost of increasing MSE. Furthermore, after the centroid has been moved, it is possible that the partitioning could change as well. Therefore no changes in partitioning were made after the model alterations.

3.2 Algorithms for the Gaussian mixture model

When GMM is used, the basic algorithm for obtaining a model from the data is the *Expectation Maximization* (EM) algorithm [24, 78]. The algorithm maximizes likelihood of the data with respect to the model, see equation (7). As in k-means, EM algorithm works by alternatively updating the model considering which object belongs to each component and with what probability, and mapping the objects to components. This is repeated until the model converges to a (local) optimum. A variant that updated the model several times per one pass over the data is presented in [86].

Membership of an object to a component is computed using Equation (6). Sum of densities over all components is normalized to 1, and the normalized values are the membership values m_{jk} used in Equations (3, 4, 5) when the component weights, means and covariances are computed.

GMM is more complex than the centroid model and initializing GMM is therefore more demanding. A common practice in EM is to first iterate k-means and then use the partitions to initialize component weights and covariance matrices. This has the advantage that the user does not need to make educated guesses about the initial covariance matrices or use a scaled version of the initial covariance matrix of the entire data set. A potential drawback is that k-means is affected by the scaling of the variables in the data, which may affect the final solution of the EM algorithm. Technically one can use the Mahalanobis-distance with the covariance matrix of all of the data which would make the data appear normalized to k-means.

The GMM obtained directly from partitioning of the data may be usable, as noted in [74]. Such model has been used for speaker recognition [64].

A deterministic annealing variant of EM algorithm has been proposed in [62]. The component weights and covariance matrices are constrained to be close to each other during the annealing process until the constraints are removed at the end. A similar algorithm is given in [106], where only the component weights are constrained. Figueiredo and Jain point out in [32] that EM algorithm exhibits annealing behavior when component weights are initialized to $1/K$, where K is the model size.

There is a variant of the EM algorithm that merges nearby components and splits large components. The SMEM algorithm [105] uses the sum of products of memberships between two components to determine suitability of the pair for merge:

$$J_{merge}(j,k) = \sum_n m_{nj} m_{nk}$$

Selecting a component for splitting relies on finding a component which describes the data poorly. The authors define local Kullback divergence as:

$$J_{split}(c_k, v_k, w_k) = \int f(x,k) \log\left(\frac{f(x,k)}{P(x, c_k, v_k, w_k)} \right) dx. \tag{11}$$

Function f in Equation (11) measures local data density around the component and is defined as:

$$f(x,k) = \frac{\sum_n \delta(x - x_n) m_{nk}}{\sum_n m_{nk}}$$

Function δ is an estimate of the local data density around location x. The larger the value of Equation (11) is, the worse the component represents the data locally. Such component is a good candidate for split.

In the SMEM algorithm one split and one merge operation is performed for each iteration of the algorithm, keeping the model size constant. After the operations, the changed components are first optimized locally, so that only their parameters are allowed to change. After that, the EM algorithm is run until the model converges. The entire process is iterated for an user-specified number of times.

3.3 Fuzzy and possibilistic clustering algorithms

Object memberships can be assigned in *fuzzy* [118] or *possibilistic* manner. A fuzzy variant of k-means is called *fuzzy c-means* (FCM) algorithm [25]. The algorithm uses fuzzy membership values in the same way as the EM algorithm uses a posteriori probabilities in computing the update weight of each object. The process is iterated until convergence. Since fuzzy sets leave room for the designer to decide how the fuzzy memberships are computed [69], fuzzy algorithms can be, to some extent, tuned for the application in question.

Possibilistic approach [68] relaxes the restriction of Equation (1) that the sum of memberships of any object should be 1. It is replaced by the requirement that the maximum membership of an object in some cluster should be greater than 0. This variant of k-means is called *possibilistic c-means* (PCM). The clusters become uncoupled and the algorithm becomes more immune to noise than FCM. The noise immunity is achieved because the memberships of outliers remain small and the outliers have only small effect on the centroid computations. A consequence of the uncoupled clusters is that PCM can produce clusters that contain the same objects with nearly same memberships [8].

In the FCM algorithm the model is centroid model. The centroids can be closer to each other than when a hard partitioning is used. This is because the membership values are commonly computed using squared inverse distances, so objects attract the centroids of neighboring clusters. Techniques such as deterministic annealing can be used with fuzzy logic as well [92].

There are several fuzzy, or soft, clustering algorithms, many of which have their counterparts in hard clustering algorithms. This work is restricted mainly to hard clustering and soft clustering with GMM.

3.4 Binary variables

When the data set consists of binary vectors there are some special conditions to be considered [40]. Due to binary-valued attributes, the mean vector is not usually a binary vector and hence, cannot be used in the centroid model. When mean vectors are used in intermediate steps of an algorithm they have to be rounded in the end, see Figure 12. Using binary vectors as a model, on the other hand, prevents gradual changes as the value can only change from one 0 to 1, and vice versa.

Stochastic relaxation and annealing algorithms are usable, as they do not rely on gradual changes alone. Random changes made to the model cause changes in the partitions, and consequently, the model vectors are able to move around.

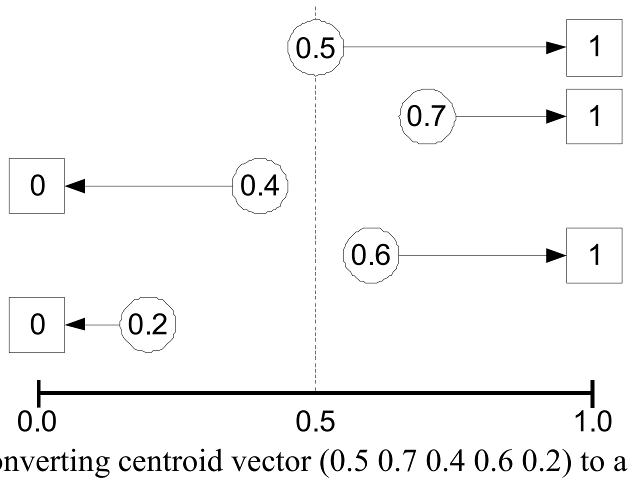

Figure 12: Converting centroid vector (0.5 0.7 0.4 0.6 0.2) to a binary vector.

4 Varying the number of clusters

When the number of clusters is unknown, the basic approach is to vary the model size in a given range and then keep the model that is the most suitable for the needs of the user. There are several approaches to changing the model size depending on the range of model sizes, and whether we need to store all models for later inspection. If we want to have all models from the specific range, then we must generate them all and the main problem is how to generate them efficiently.

4.1 Hierarchic algorithms

A basic feature of hierarchic algorithms [100, 58, 31] is that they initially consider all objects either to belong to the same cluster, or each object to belong to its own cluster. *Divisive* algorithms proceed by splitting the clusters. *Agglomerative* algorithms join the clusters. The process continues until the desired model size or number of partitions has been reached. Therefore, hierarchic algorithms can produce models of all sizes. Clustering criteria can be used to select the most suitable model among all generated models of different sizes. The algorithms perform irreversible decisions. If two objects are separated at some level, then they remain separated.

Certain hierarchic algorithms do not use the data objects. Instead, they operate on a matrix of distances between the objects [31]. Similarities can be used as well. Objects are needed only to compute the matrix. Therefore, the algorithms are applicable to any kind of data. In *single linkage* algorithm, for example, joining is done between the two clusters with the smallest distance between objects between clusters, see Figure 13. *Average linkage* uses average distance between all pairs of objects in two different clusters as the cost of joining the clusters. In *complete linkage*, joining is done based on the maximum distance between all pairs of objects between the clusters, see Figure 14.

Ward's method [110] is an agglomerative algorithm that joins the pair that results in the best value of the user-chosen criterion function among all candidates. In vector quantization, the algorithm is generally known as *pairwise nearest neighbor* (PNN) algorithm [28]. Equitz gives an equation for the increase of error as the result of joining two clusters:

$$I(j,k) = \frac{n_j n_k}{n_j + n_k} d_E(c_j, c_k)^2 .$$

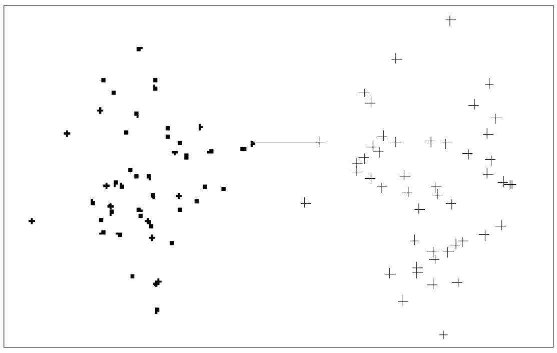
Figure 13: Joining in single linkage.

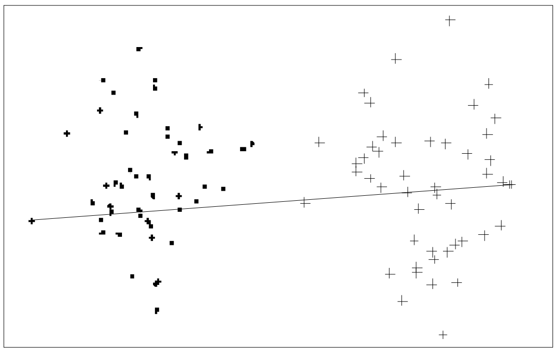
Figure 14: Joining in complete linkage.

Performing a division of a cluster into two new clusters is not as simple as joining two clusters. A straightforward way is to select two objects farthest from each other inside a partition, and then divide the remaining objects according to these objects [100]. For data in vector space, the largest principal component of the objects in the partition can be used in finding a hyper-plane that is used to split the cluster [114].

If hierarchic algorithm is run to the end, the result is a *dendrogram* that can be used in trying to decide how many clusters there are, and which objects belong to which cluster. In Figure 15 there is a data set with two clusters. Dendrogram produced by average linkage is shown in Figure 16. Each subtree represents a subset of the data. Vertical distance between horizontal lines joining the subsets is directly proportional to the average distance between all pairs of objects between the two clusters.

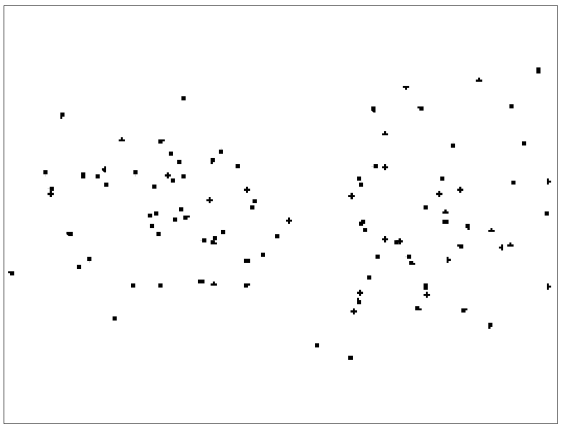
Figure 15: Data set with two clusters.

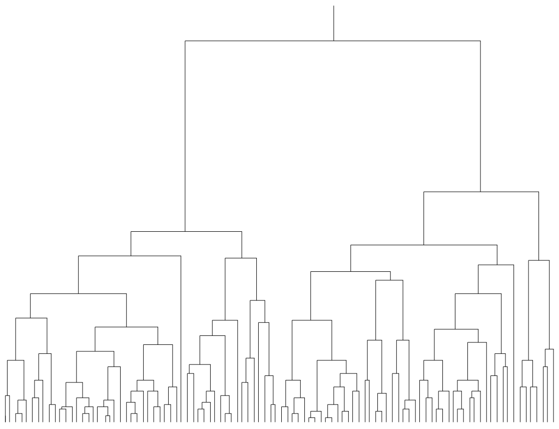
Figure 16: Dendrogram produced using average linkage agglomerative clustering.

As a by-product, models of all sizes from 1 to N are obtained. The extreme case of one object per cluster is not likely to be useful, and neither are the other large models. One can assume that the interesting models are the smaller ones, hence a divisive approach is expected to give the result faster than an agglomerative approach.

Modification of the distance matrix is a basic operation when speeding up the algorithm or changing the clustering produced by hierarchic algorithms. For example in

[112], all distances except the ones where one object is in the *k*th nearest neighborhood of the other are set to ∞. Similar approach has been used to speed up the pairwise nearest neighbor algorithm [38] to achieve $O(N\log N)$ time complexity. Likewise for the single linkage algorithm [27], the time complexity decreases to $O(kN^2)$. In Chameleon [56], a *k*th nearest neighbor (kNN) graph is used as a starting point. The graph is partitioned to get the initial clusters. Clusters are joined by using interconnectedness and closeness information from the graph. Only the distances in the kNN-graph are used by the algorithm. The authors demonstrate that the algorithm is capable of finding arbitrary-shaped clusters.

One need not use distance function to compute the values in the matrix. Similarity matrix can be constructed by running k-means repeatedly and then assigning similarity between object pairs depending on how many times the objects ended up in the same partition [33]. This relies on the tendency of k-means algorithm to move the centroids to positions where the density of objects is high when the model size is high. In such case, centroids do not end up in areas of low density between clusters.

There are variants of hierarchic algorithms that perform local optimization using, for example, k-means after each split or merge to obtain better intermediate codebooks [22, 90, 107]. These variants are not strictly speaking hierarchic as they do not produce dendrograms, but they still provide the user with multiple codebooks. The variants usually require that the objects lie in a vector space. Abandoning the strictly hierarchic nesting of clusters inside larger clusters produces better results in terms of lower error, because the partitions can be adjusted to the current model. Figure 17 illustrates the difference.

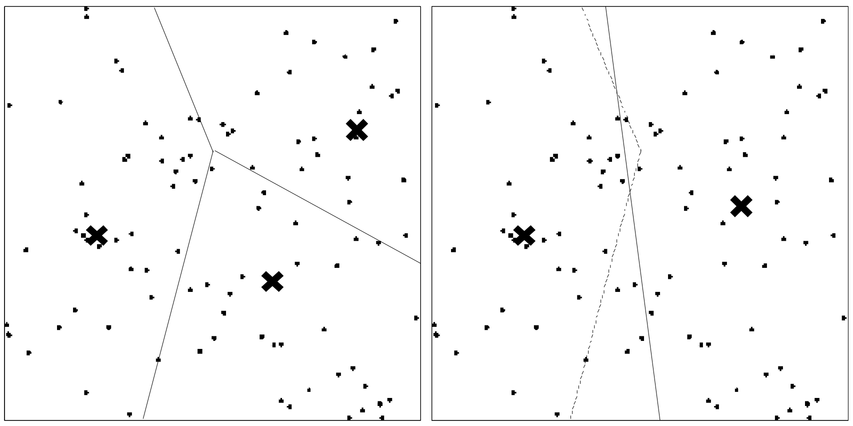

Figure 17: Original partitions (left) and alternative partitions (right) with joined partitions (dashed line) or partitioning to closest centroid (solid line).

Instead of joining the nearest neighbors in agglomerative clustering, an alternative inspired by gravitation has been proposed in [113, 70]. Every object is moved to the direction determined by gravity caused by the other objects. The aim is that nearby objects clump together first. The formed groups gradually move closer and form larger groups that attract their neighbors more strongly. These two algorithms are iterative and at each iteration, objects that are close enough to each other are joined together. These algorithms provide a dendrogram, but since the objects and clusters move towards the center of gravity of the data set during the process, the joined objects cannot be used directly as centroids.

Model size can be changed by more than one. Competitive Agglomeration [34] is a fuzzy algorithm, which discards all clusters that have too few objects. The limit is given by the user. The algorithm does not produce models of all sizes and it stops automatically when the model size does not change and the algorithm has converged.

Another approach that resembles hierarchic methods is *X-means* [90]. The idea is to split each partition if two centroids describe the partition better than one centroid. Let X_j be the set of objects that belong to cluster j and let p be the number of parameters needed to represent the component. The method uses *Schwarz criterion* or *Bayesian information criterion* [57]:

$$BIC\left(X_j, c_j, v_j, w_j\right) = LL\left(X_j, c_j, v_j, w_j\right) - \tfrac{p}{2}\log n_j \tag{12}$$

The criterion is computed inside each partition separately. *LL* is log-likelihood given by Equation (7), when computed using one component only. Each split is performed independently of each other, so several splits can occur at once. Value of Equation (12) is to be maximized.

GAs have been used to find hierarchic clustering [73]. The algorithm is based on the existence of a mapping from dendrogram to distance matrix. Individuals of the population represent the order of objects and the tree structure. The squared error between the distance matrix and initial distance matrix is minimized. The optimal order and tree structure should yield minimal difference between the two distance matrices.

It is not possible to start with one component formed of just one object, when the selected model is a GMM. This is because the covariance matrix is singular until the number of objects that contribute to a component is at least one greater than the dimensionality of the data. Using agglomerative clustering until the groups are large enough to be handled with GMM is proposed in [39]. Different criteria to be optimized in agglomerative clustering algorithm for different types of covariance matrices have also been presented there. The starting point can be an existing GMM, which is reduced to model size 1 in a hierarchic manner [44]. The algorithm proceeds by clustering the components and joining them together. Joining several components during one step is also possible. In [32], those components are removed, for which the sum of object memberships multiplied by two is lower than the number of parameters required to represent the component.

4.2 Generating models within a range

A simple method to obtain all models from a given minimum size to a given maximum size is to generate them one by one. When the models are generated independently of each other, we can use any algorithm described in Chapter 3. Another approach is to utilize the best solution for the preceding model size, and change it by adding or removing a component. In this way, the changed model may be close to optimal and the algorithm may converge faster or find a better solution within a given time constraint.

Removal of a component is simpler of the two alternatives, since the number of removal positions is much smaller than the places where one can add a new component. Adding new components offers a natural starting point if we start with model of size 1. Algorithms that change the previous model size by one and generate several candidates generally can be considered as hierarchic algorithms when the number of candidate models depends on the previous model size [90]. That is, when each component is removed in turn or each cluster is split in turn.

Using k-means as a starting point, *global k-means* [71] starts with a model of size 1 and keeps adding new centroids until the given model size has been reached. The new centroids are selected from internal nodes of a *kd-tree* [11] to find good positions and to cover the entire area where there is data with the candidate vectors. Each candidate solution is fine-tuned with k-means and the best one is used as the initial solution for the next round. Using kd-tree provides a sufficient number of candidates so that the quality of the solutions is identical to that of using all data vectors as the set of candidates. Using the smaller candidate set requires much less time.

Generating a model for all model sizes in the search range may not be necessary. Skipping some model sizes is a viable approach if the clustering criterion has reasonably smooth graph, see Figure 19. The fewer local optima the graph has, the better for the algorithm. The values surrounding the optimum value should also differ enough from each other so that the optimum value is clearly distinguishable. Figure 18 shows a range in the graph of DBI where it is potentially possible to find the optimum value by changing the model size by simple descending approach. If the search can be focused to this range, the optimal model size is expected to be found quickly compared to searching the entire range. Finding the narrow range, however, is harder than finding a wider range, as in Figure 19. There small changes to model size are, in theory, sufficient to find the optimal model size.

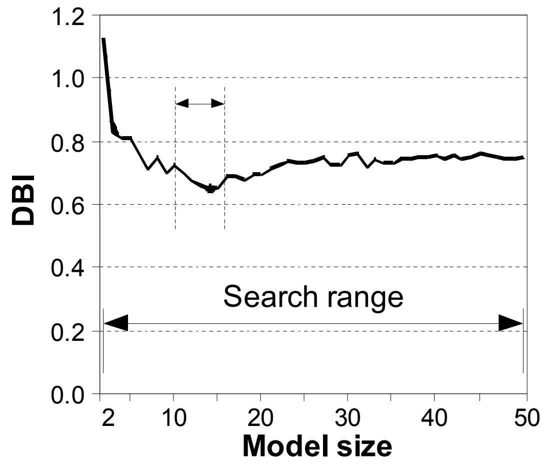

Figure 18: Narrow range around optimum value of DBI.

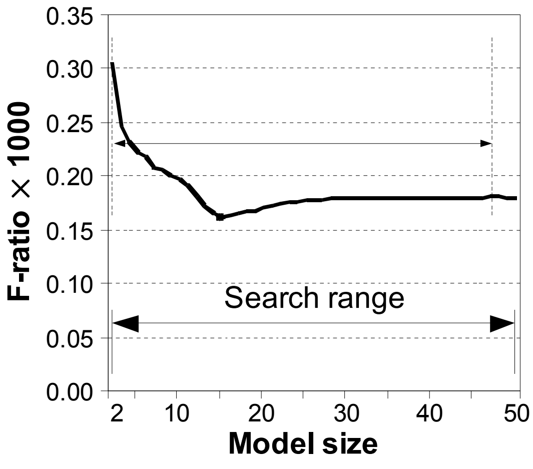

Figure 19: Wide range around optimum value of F-ratio.

Two GA-based approaches [102, 103], related to single linkage, use a two-stage process. The first stage divides the data into subsets which are then clustered together in the second stage. Average distance to nearest neighbor is used to find the subsets. The average distance is computed over the entire data set. The subsets are the connected components formed by linking objects to their neighbors whenever the distance is less than the average distance to nearest neighbor. Clusters are formed in the second stage by joining the subsets together. Each individual in the population of the GA is a single cluster, so the algorithms need to find a set of most fit individuals. Distances between subsets in same cluster should be minimized and distances between different clusters maximized [102]. In [103], after the subsets have been formed, their mean vectors are computed. Distances between mean vectors are used instead of distances between objects in the subsets. The authors present a heuristic strategy for determining the number of clusters, based on minimum distances between centroids and the spread of the clusters.

Davies-Bouldin index is used in the fitness function of a GA in [7]. The algorithm allows for the model size to vary within a pre-specified range. This is accomplished with a value that indicates that the centroid is not used. Hence, different number of values that indicate missing cluster in the individual produce models of different size. In [41], F-ratio is used inside the cross-over operator to select the model returned by the operator.

4.3 Small changes to the model size

When the proper model size is known approximately, a suitable way to alter the model size is to change it in small steps. An early example is ISODATA [5] where the user gives a lower and upper limit to the partition diameter. Two neighboring components are joined if the diameter stays within limits. Any partition that is too large is split. Consequently the model size changes, and if the initial clusters were much larger or smaller than the given constraints, the change in the model size can be quite large.

A competitive EM algorithm [120] uses the same split and merge criteria as the SMEM algorithm [105]. The algorithm chooses either a split or a merge operation whenever the model has converged during the EM iterations. Components with small mixing weight are annihilated as in [32].

Adding new components to areas where distribution of data does not match the distribution indicated by a GMM is done in [99]. The area covered by each component is divided into equally probable areas. Each area should have an equal number of objects, provided that the model represents the distribution properly. For each area, objects are counted. There is a parameter that indicates the allowed deviation from the expected value. If the deviation is exceeded, the area is considered to have too many or too few objects. Another parameter controls how many deviating areas the component is allowed to have. New components are added to areas with too many objects if there are too many areas that deviate from the expectation.

A similar idea is used in an iterative algorithm in [94]. The maximum difference between the density estimate computed from data objects and the density computed from the model is used to determine where the new component should be added. Components with small weight are candidates for removal. Adding new components and removing components with small weight is random to some extent. Regions with higher difference of the density estimates have higher probability of being selected as locations for new components. Likewise, inverse component weight is used to compute the probability of removal.

Only one component is added or removed per iteration of the algorithm in [94]. EM algorithm is iterated a few times during each iteration of the main algorithm. Probability of component removal or addition is initially 1, and it linearly decreases to 0. Likewise, number of EM iterations is initially 1 and it linearly increases to 10. When the model size changes, the component removal is chosen to be performed if the removal improved the model during the previous iteration, or if addition did not improve the model. Same logic is used for adding a new component. If the score of the model did not improve after the EM iterations, the new model is discarded and the previous one is used as the starting point.

4.4 Partitioning objects or components together to form clusters

When only a partitioning of the data is used, there are some alternatives besides the basic hierarchic algorithms, such as single and complete linkage. Even if a model is used, each component of the model can be thought of as an object, and linked together with other components to form arbitrary-shaped clusters. Hence, joining components together is not a new idea. The main feature of the algorithms presented in this Chapter is the capability to produce arbitrary-shaped clusters. The result of the clustering is either a partitioning of data objects or a partitioning of the components of the model. In cases where a model produced from the data is used as the starting point, the model should be sufficiently detailed so that the actual shape of the clusters can be still found by linking the components.

An algorithm based on forming trees [67] assigns a parent to each object whenever possible. Parents are chosen from the neighborhood of each object. Objects that are close and have much higher estimated density are likely to be chosen as the parent. Each object has only one parent, and the algorithm restricts the links so that only trees are formed. The number of trees corresponds to the number of clusters. An object that has no objects of higher density close to it will become a root of a tree. The results indicate that the algorithm tends to split clusters into several trees but the largest trees tend to hold most of the objects.

Refining an existing partitioning can be done using majority-vote inside a given neighborhood of an object [65, 66]. For each object, the algorithm finds the partition with most objects in the neighborhood. An example is shown in Figure 20. The object at the center will move to partition consisting of square objects. The process is iterated until no object changes partition. The algorithm resembles k-means, except that there is no centroid model.

Boosting-based clustering [36] relies on sampling the data set repeatedly. A model is generated for each random sample, for example using k-means or fuzzy c-means. A new partitioning for the data set is computed using the model. The new partitioning is combined with an aggregate partitioning using a voting scheme. When the actual clusters are divided in a partitioning, the borders of partitions end up in arbitrary places inside clusters, but remain mostly in the same places between the clusters. As a result, objects at different sides of cluster borders are voted into different clusters even if they end up in the same partition occasionally.

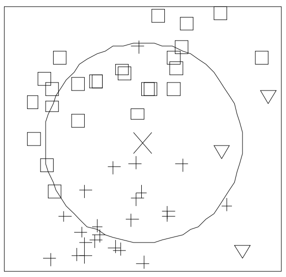
Figure 20: Voting to partition objects.

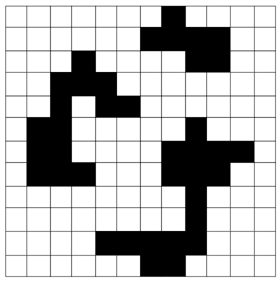
Figure 21: Clustering by joining grid cells.

CLIQUE [2], ENCLUS [20] and MAFIA [82] all attempt to find clusters in subspaces. Dense areas are joined together. CLIQUE uses a fixed-size grid. Two grid cells that have sufficiently high density are joined together to form clusters provided the grid cells share a side. ENCLUS uses entropy of the grid cell instead of density. Low entropy indicates clusters. MAFIA uses an adaptive grid and density information. Figure 21 shows an example of the clusters produced on a fixed-size grid.

An algorithm that links centroids is presented in [104]. After initial clustering by k-means using a large model size, a distance matrix between the centroids is constructed. The distance between two centroids is inversely proportional to number of objects in a hyper-cylinder that has its end-points at the centroids. The number of objects corresponds to density of objects between the centroids. Two centroids with lots of data objects between them are more likely to lie inside the same cluster than two with just few objects between them. Distances are computed only between neighboring centroids. The linking is done with an agglomerative clustering algorithm. After linking and deciding a cut-off point for the dendrogram, the centroids have been divided into clusters. It is possible to construct a piecewise linear model consisting of the line segments between centroids in the same cluster. In this model, the distance of an object to a cluster is the distance to the nearest line segment.

The same idea can be used for GMM. The advantage is that the probability density can be computed anywhere directly from the model. Therefore, if the density estimate computed using the objects correlates with the probability density computed using the model, it is possible to compute the values of the distance matrix without using the original data. Since the objects are not counted, the hyper-cylinder can be replaced with straight line between the component means. The minimum probability density along the line is taken as the density between the components, since it corresponds to the separation between the components.

Technically, one should use minimum probability density along a path between the two components, with the path chosen so that all other possible paths have lower minimum probability density. For nearby components, the straight line between component means is likely to be a reasonable approximation. Since there can be a path which goes from one mean to the other mean entirely along area of higher probability density, the straight line provides a lower limit to the true value.

A difference between relative and absolute densities can be made when constructing the density matrix. If there are clusters that have quite low density compared to some other cluster, but which still have higher density than the surrounding area, then using absolute densities will result in no components being joined inside these clusters. If the density is relative to, for example, maximum density of the end-point components at their means, then the components in clusters with lower overall density can be joined together earlier. Using relative densities favors linking in areas where the density between two components remains roughly the same as near the component means. The difference is shown in Figure 22. Density distribution of the data is shown on the left. In the middle is the result of linking using absolute densities, on the right the result of linking using relative densities. Dark lines between components are stronger links than thin, gray lines.

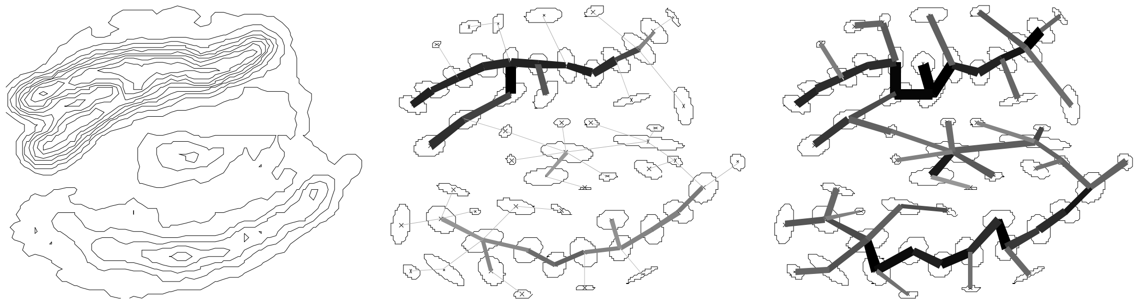

Figure 22: Density contours of data distribution (left), result of linking using absolute densities (middle), and result of linking using relative densities (right).

5 Large data sets

The algorithms presented in the previous chapters make the implicit assumption that all data can be kept in memory at once, and the explicit assumption that multiple passes are acceptable. This is not always the case. If all data does not fit into memory, the data needs to be read from mass storage once per every pass. For example, every partitioning step in k-means requires one pass. This may be prohibitively slow. In this chapter, algorithms for large data sets are considered.

Need for processing large data sets has been around for a long time, but the notion of what is large has changed during the years. Even though computers get faster and have more memory than before, an efficient algorithm that finds good solutions for large data sets can most likely be used for smaller data sets as well.

There are different requirements for algorithms that deal with large data sets. *Data stream* algorithms [45] are allowed to store some data objects, but each object is read only once. Data stream algorithms are not required to take action after an object arrives, so they can store a set of objects that are not modeled yet. These algorithms should be able to accept the data in arbitrary order as there is no guarantee that the data is stored anywhere. *On-line* algorithms should be capable of updating the model after new data has arrived. Some algorithms allow removal of data objects as well [1]. On-line algorithms model the previously seen data so that the model is available at all times.

Some algorithms concentrate only on using less memory. They may read all data, or just take random samples. In cases where the algorithm uses only a small part of the data, the whole set can be partitioned afterwards, once the model has been created. Some algorithms rely on suitable data structures so that the required objects and their nearest neighbors are found efficiently. Without such structure, these algorithms may have $\Omega(N^2)$ time complexity due to the neighborhood search [4]. Only the objects in the neighborhood need to be stored in memory at once.

Several of the algorithms for large data sets take outliers into consideration. Even if the probability that an object is an outlier is low, in a large data set there will be some outliers. In the algorithms described in this chapter, objects that remain alone after some clustering steps or objects that have low estimated density are usually considered to be outliers.

Even when the entire data set fits into the memory, $\Omega(N^2)$ time complexity may be too high in practice. This may exclude agglomerative algorithms, which typically have at least quadratic time complexity. Fast agglomerative algorithms [38] rely on fast kNN search so that the overall time complexity is less than $\Omega(N^2)$. Performing a thorough search over some range of model sizes may also become too expensive. A clustering algorithm for fixed model size has usually at least linear time complexity, because it has to go through all objects at least once. Performing the search for different model sizes, possibly several times for each model size may have too high time complexity for practical purposes. If the lower bound for the time complexity of a clustering algorithm for fixed model size is $\Omega(N)$ and the number of model sizes that are used is proportional to N, the total time complexity will be $\Omega(N^2)$. The clustering algorithms presented in this chapter either determine the model size or the number of partitions automatically, or they keep the model size fixed.

5.1 Data reduction algorithms

A way to cope with large amounts of data is to reduce it before generating the model. Data reduction algorithms try to represent denser areas with fewer objects while maintaining some accuracy in areas where the objects are sparsely distributed. The simplest approach is to take a random sample and use it instead of the entire data set. The sample must be large enough to capture the distribution of the data with sufficient detail. This becomes harder when the dimensionality of the data increases, as the tails of the distributions contain larger part of the probability density [98].

BIRCH [121] is an algorithm designed to reduce a large data set into *clustering features*, which contain position and distribution information. The user can set the maximum amount of memory the algorithm may use. As data is read, the algorithm combines objects into larger and larger clustering features to stay below the memory limit. These clustering features consist of the number of objects stored into the feature, their sum and sum of squares. Thus, it is possible to compute the mean and variance.

Outliers are dealt with as an optional special case. During the joining of clustering features, if the algorithm encounters a clustering feature that has far fewer objects than the other clustering features, that clustering feature can be considered to be an outlier. They are stored externally and read in afterwards at which point the final decision is made about whether the clustering features represent outliers or not.

After the data set has been processed, the clustering features can be clustered using any algorithm that has been modified to use the clustering features instead of the data objects. Basically the clustering features contain the same information as a GMM, so in some sense the modified algorithm operates on a GMM. However, the major difference is that due to limited radius of the clustering features, the algorithms only need to consider the distances between the features. The spread of objects within the clustering features can be ignored since the features do not overlap and are mostly of the same shape. Data reduction using BIRCH before the actual clustering has also been used in [21, 54].

In a way, BIRCH produces a very detailed centroid model with extra information. Therefore, it might be possible to produce a model directly with BIRCH by limiting the available memory. Then BIRCH would be suitable for incremental updates of the model as well. However, the data compression stage in BIRCH does not keep the number of clustering features constant. Also, it is argued in [19] that BIRCH depends on the order of the data so that the clustering features may end up containing objects from neighboring clusters.

Whether a random sample or the clustering features are used, it is suggested in [16] that both the objects in the sample and clustering features should be weighted according to how many objects they represent. For representative objects, this requires finding the nearest neighbor among the representative objects for all data objects. The authors introduce the *data bubble*, which contains the location, weight, radius information, and the average kNN-distance of all objects represented by the data bubble. The radius and average kNN-distance indicate the wideness of the area over which the original objects were spread. This information along with a specific distance functions for data bubbles helps to decrease the loss of information due to random sampling, or compression as used in BIRCH.

Figure 23 shows a data set of 10000 objects before and after data reduction. The number of objects has been reduced to 1081. The size of the dot is proportional to how many objects are represented.

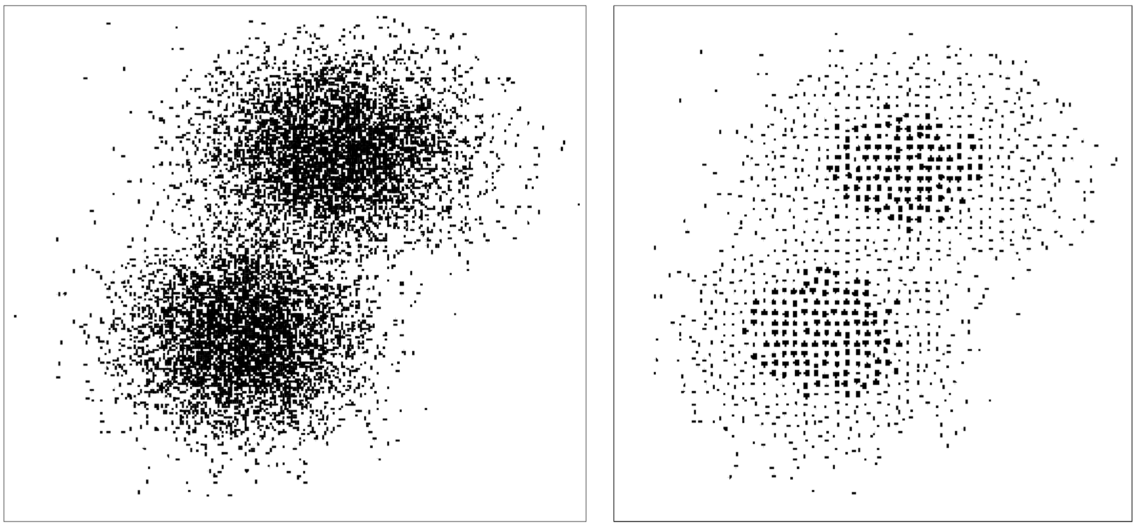

Figure 23: Data set of 10000 objects (left), and after reduction to 1081 objects (right).

Equations (8, 9) can be computed from the clustering features as well as from the original data. One must only take into account the weighting of the features. The result of the Equation (9) is accurate. The Equation (8) gives an estimate unless one stores the entire covariance matrix in each clustering feature. However, Equation (10) can still be computed accurately from the clustering features. In practice, clustering features force a slightly different partitioning than the original data would allow.

5.2 Partition and representative object -based algorithms

Having the information about clusters stored in a partition has the drawback that there are as many partition indices as there are data objects. Therefore, partitioning may be impractical for huge data sets. In some cases, producing a partitioning is an optional final step, as in [121].

Leader [49] is an algorithm that constructs a model of the input data by picking representative objects from the input data. Any object further than the user-specified limit is added to the model. Objects are mapped to the first representative in the model that is close enough. The algorithm requires only one pass, but the limit must be set to a suitable value. A large limit will produce a small model that does not describe the model in any detail, and too small limit can result in a model of excessive size.

CLARA [58] uses random sampling and PAM [58] to find the representative objects. PAM is run until convergence, and the random sample is then replaced with another one. The previous model is used as the initial model for PAM with new random sample. CLARANS [85] is an improved version of CLARA [58]. In CLARANS the random sample is drawn at each step of PAM. Hence, there is a greater possibility that good representative objects would be selected, but at the cost of increased need to access the data. Once the final representative objects have been found, the entire data set can be partitioned.

The main idea of CURE [46] is to cluster a random sample using a hierarchical algorithm, remove outliers, and then use several objects from each cluster as representatives of the cluster. Outlier removal is done during the agglomerative clustering. At some user-specified point, clusters consisting of one or two objects are removed. Then, at the time when the number of clusters is slightly above the desired number, smallest clusters are removed as outlier clusters. After that the whole data set can be partitioned using the representatives. Objects are mapped to the cluster with the closest representative. The initial random sample must be large enough so that small clusters are not missed during the clustering phase.

The approach used in [45] reads the data set in blocks. For each block, a model is produced. The models are combined, and they are then treated as data and clustered. To keep the amount of memory limited, there are intermediate models in several levels. Each intermediate model has a fixed maximum size and if it is exceeded, a smaller, higher level model is generated. At the end, the final model is generated.

5.3 Density-based algorithms

Algorithms that rely on using density estimates usually require a fast method of obtaining density estimates for data objects and possibly the nearest neighbors, or neighbors inside a specified range, usually by using a suitable indexing structure, such as R*-tree [10]. Therefore, the user will have to create the structure unless it already exists. These algorithms usually produce a partitioning since the clusters can be of arbitrary shape.

DBSCAN [30] is based on the idea that the object density around each object in a cluster should exceed some user-given limit. In DBSCAN, a fixed radius is used. The number of objects inside the radius, or neighborhood, is the density estimate. Objects for which the user-given density limit is exceeded are called core objects. Any object that can be reached from a core object by moving through any chain of core object neighborhoods belongs to the same cluster. Hence clusters can be arbitrary-shaped. All objects which are not reachable from any core object have lower density than the user-given limit and are regarded as noise.

An incremental version of DBSCAN is presented in [29]. The main observation behind the algorithm is that insertions and deletions of objects affect only small neighborhood around them. Since core objects define the clusters, it is only necessary to consider the core objects close to an object that has changed core status due to insertion or removal. An object becomes a core object if the density estimate exceeds the user-given limit, and likewise, a core object becomes a common object when the density estimate falls below the limit. Objects can become noise if enough objects are removed from their vicinity, and vice versa. Consequently for relatively small number of updates the algorithm is much faster than re-running DBSCAN.

OPTICS [4] sorts the objects into a decreasing order according to the distances that DBSCAN uses when performing clustering. Using the ordering and stored distance information, the data set can be clustered in the same way as DBSCAN would cluster the data using any user-given radius that is smaller than the radius used in generating the ordering. As in DBSCAN, an efficient region-query is required. An online version is introduced in [1].

DBCLASD [115] relies on identifying clusters based on the distribution of distances to nearest neighbors inside cluster. Candidates are selected from the neighborhoods of the objects that already belong to the cluster. A candidate object is added to the cluster if the distribution of distances to nearest neighbors inside the cluster still matches the expected distribution according to χ^2 test with given confidence level. The neighborhood radius depends on the area the cluster covers and inversely on the number of objects already in the cluster.

DENCLUE [50] is a gradient-ascent algorithm. Basically, a Parzen window [88] is used as a density estimator and the gradient search is done to find a mode of the distribution. Algorithm uses Gaussian window function with tail of the distribution cut off. Modes of the distribution are used to identify clusters. Several modes can form one cluster if the probability density between modes does not decrease below a limit specified by the user. Any mode with probability density below the limit is discarded as noise. To speed up the search, DENCLUE divides the data using a grid. All grid cells with too few objects, determined using an optional limit given by the user, are not used to start searches. Cells are stored in B^+-tree [9] for faster access during the gradient search. For further speed-up, all objects that end up inside the density estimation window during the search are mapped to the same cluster as the object where the search started.

5.4 GMM-based algorithms

There are at least three online-EM algorithms [6, 95, 122]. The algorithm in [6] is a competitive learning [63] algorithm designed to use GMM. The one by Sato and Ishii [95] can create new components if the new object is far away from the model. The covariance matrix of the new component is a scaled identity matrix. The scaling factor is a product of an user-specified scaling factor and minimum squared distance from the object to component means, divided by data dimensionality. A component is removed if the component weight becomes too small. In [95], the user must give the probability density limit for determining when the object is too far, and the weight limit for removing a component. In [122], component is removed if the weight becomes zero or less.

All three algorithms forget old data gradually. Forgetting old data might not seem appropriate if the intention is to model the entire data set. However, forgetting old data might allow for the components to move more freely so that they are not tied to the original positions, should the initialization match the actual distribution of the data poorly. When the data is ordered in some manner, forgetting old data may move the entire model away from a region where the model was initially [95].

Scalable EM [14] takes the approach of converting the data into compressed object sets. Some data is read in, then model is updated. After update the data is compressed to make room for more data. For compressed object sets the same statistics are stored as in BIRCH. The algorithm contains EM algorithm that has been modified to use the sufficient statistics as well as individual objects, and the compression of the data is done using agglomerative clustering algorithm. In it, compressed object sets are joined together as long as the set radius does not exceed a limit. This limit will increase as the algorithm proceeds so that later on, more objects can be compressed together. The algorithm also contains another method of compression. All objects that are close enough to a component are compressed together. In practice this compresses together all objects in an area centered around a component mean, with the same shape as the component. This might compress together objects from different clusters if a component covers two or more clusters. The authors have also presented similar variant of k-means [15]. The main feature is the detection of small, compact subsets that can be turned into compressed objects using k-means, and the agglomeration, when necessary, of the compressed object sets. Technically, the agglomerative algorithm could handle both compressed object sets and the uncompressed data.

An EM algorithm that utilizes a multi-resolution kd-tree is presented in [81]. Each node of the tree contains information that summarizes the sub-tree. Namely, the number of objects, their mean, covariance and bounding hyper-rectangle. In order to avoid storing all data, a set of objects in a sub-tree is left together once their bounding hyper-rectangle becomes smaller than a user-given limit. The error that arises during model update of the components of the GMM is expected to be small. Also, when estimated minimum and maximum memberships of an internal node are very close to each other for all components, the node is treated as a leaf node. A component with very small estimated membership value for a sub-tree can be omitted from further computations in that sub-tree.

6 Summary of Publications

In Publication P1 we present the use of previous models as the initial model when performing a search for the optimal model size systematically inside a range of model sizes. Since the algorithm for models of fixed size does not improve the model during each iteration, some stopping criteria to indicate a sufficient number of iterations are also proposed. A few dozen initial iterations are required before we can start to use the stopping criteria. The results indicate that the stopping criteria that assume the best clustering criterion value should follow a smooth curve, are reasonably good. A simpler criterion based on whether the model has improved or not, worked poorly. Using the previous solution decreases the required number of iterations to 33–43 %.

In Publication P2 we propose a generalization of the RLS algorithm that optimizes also the size of the model during the process in order to find the number of clusters faster. Instead of performing just random swaps, centroids can also be removed or added. Comparison shows that the proposed approach spends only 3 % of the number of iterations used by a simple brute force approach of going through the given range of model sizes systematically. For data sets with overlapping clusters, much better chance of finding the proper number of clusters is achieved by using only 5.5 % of iterations required by the brute force approach.

In Publication P3 we propose a method for improving the values of Davies-Bouldin index [26] by altering the centroid model. The main goal of the research was to improve the ability to find the intended number of clusters. DBI is calculated using MSE of each cluster and distances between all pairs of centroids. The greater the distances between the centroids are, the better the clustering is. For DBI, the optimal position for centroids is not the mean vector. Moving the centroids slightly away from each other so that the increase in errors within clusters is offset by the increase in distances between the centroids results in lower values of DBI. The decrease of the values of DBI is too small to have a practical effect on the clustering. It was not possible to determine the number of clusters any better than without the changes to the model.

In Publication P4 we cluster binary data. Some distance functions used with binary data do not work properly when a cluster has only one object. We propose a distance function to be used with stochastic complexity. The distance function indicates how much the criterion value changes when an object is moved from one partition to another. The results of the clustering are used to perform classification. The classification error for the proposed distance function is 1.13 % while for the other distance functions it is 3–7 %.

In Publication P5 we cluster binary data using gradient-descent type algorithms. These algorithms rely on small, gradual changes of the model. With binary data only abrupt changes are possible. We use a variable metric, and therefore variable criterion function and non-binary model, to improve performance. The exponent of a Minkowski metric is changed from ∞ to 1, and in the process, the model gradually becomes a set of binary vectors. Initially the differences between optimal values in different partitions for single variable are very close regardless of how many ones and zeroes there are in a partition, thus allowing for small changes to occur in the model. The final solution becomes a set of pure binary vectors in a natural manner. Test results indicate that gradient-descent algorithms perform clearly better with variable metric. The average decrease in error is 11 %.

In Publication P6 we propose a one pass algorithm for generating a mixture model from large data set. It is useful in cases where only a small part of the data fits into the main memory at once. The algorithm identifies suitable subsets of the data, generates and adds new components into the model, and updates the existing model with data objects as they are read in. Updating the model is always the primary choice. The resulting model can be processed afterwards without the original data. During the execution of the algorithm, only a small user-specified amount of the data is kept in memory at once. The results indicate that the model conforms reasonably well to the distribution of the data. The order in which the data objects are given to the algorithm does not have major effect on the results. The whole process requires 0.5–10 % of the time taken by the EM algorithm to generate the same amount of models.

In Publication P7 we attempt to discover clusters from speech data by systematically searching through a wide range of model sizes. Three different features are computed from the speech signal. F-ratio is used as the clustering criterion. The monotonic behavior of the criterion values indicates that there are no clusters. Visual inspection of the data does not indicate the presence of clusters either. Using different window functions or normalizing the data has no effect. As a consequence, to create a model from speech data, the clustering algorithm has to model the distribution of the data. Algorithms that automatically attempt to find the number of clusters are not usable.

The contributions of the author of the thesis to the publications can be summarized as follows. In publications P1, P2, P3, P5 and P6 the author is responsible for implementing the algorithms, performing the tests, and most of the writing. Ideas have been developed jointly with the co-author. In publication P4, the author implemented the algorithms that were used and helped with the data sets. In publication P7, the author implemented the algorithms, performed the data conversions and experiments.

7 Conclusions

In this work, algorithms for speeding up the search of both a good clustering and the number of clusters are proposed. The approach is guided by a clustering criterion, which is capable of indicating the number of clusters even when the clusters overlap. The proposed algorithm is 5 times faster than an alternative approach of utilizing previous results, and 18 times faster than a simple approach of generating each model from scratch.

Algorithms to modify the centroids in order to obtain better values of Davies-Bouldin index are also proposed. The proposed algorithms improves the optimization of the criterion values but the ability of the criterion to indicate the number of clusters for data sets with overlapping cluster is not improved.

Algorithms that rely on gradual changes can be applied for binary data better by changing the representation of the clustering slowly from non-binary to pure binary vectors. In most cases, the results are considerably better than when the model consists of binary vectors, or when the model consists of mean vectors that are rounded as the last step.

A model can be generated from a large data set by first generating it from the data set in one pass and then reducing the model size in a separate post-processing step, without the need of the original data. The model size reduction is fast compared to processing the entire data set. Therefore, several models can be generated with low cost compared to the initial generation of the model from data.

A future extension to this work would be to develop a better method of selecting the objects from the buffer for the algorithm presented in [P6]. Ideally, the algorithm should quickly identify a set of objects that is distributed according to a multivariate Gaussian distribution. This would allow the algorithm to identify potential clusters.

It may also be possible to generalize the approach of [P5] to attributes with few ordinal values, such as 3-5 different values. Whether gradient descent algorithms have problems with such attributes should be verified first.

References

[1] E. Achtert, C. Bohm, H-P. Kriegel, P. Kröger: "Online Hierarchical Clustering in a Data Warehouse Environment Data Mining," in *Proceedings of the Fifth IEEE International Conference on Data Mining*, 2005, pp.10–17.

[2] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan: "Automatic subspace clustering of high dimensional data for data mining applications," in *Proceedings of 1998 ACM-SIGMOD International Conference on Management of Data*, 1998, pp. 94–105.

[3] R. Ahuja, T. Magnanti, J. Orlin: "Network Flows: Theory, Algorithms, and Applications," Englewood Cliffs, Prentice-Hall, NJ, 1993.

[4] M. Ankerst, M. Breunig, H-P. Kriegel, J. Sander: "OPTICS: Ordering Points to Identify the Clustering Structure," in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, 1999, pp. 49–60.

[5] G. Ball, D. Hall: "A clustering technique for summarizing multivariate data," *Behavioral Science*, 12, 1967, pp. 153–155.

[6] S. De Backer, P. Scheunders: "A competitive elliptical clustering algorithm," *Pattern Recognition Letters*, 20, 1999, pp. 1141–1147.

[7] S. Bandyopadhyay, U. Maulik: "Genetic clustering for automatic evolution of clusters and application to image classification," *Pattern Recognition*, 35, 2002, pp. 1197–1208.

[8] M. Barni, V. Cappellini, A. Mecocci: "Comments on 'A Possibilistic Approach to Clustering'," *IEEE Transactions on Fuzzy Systems*, 4(3), 1996, pp. 393–396.

[9] R. Bayer, E.M. McCreight: "Organization and Maintenance of Large Ordered Indices," *Acta Informatica*, 1, 1972, pp. 173–189.

[10] N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger: "The R*-tree: an efficient and robust access method for points and rectangles," in *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, 1990, pp. 322–331.

[11] J.L. Bentley: "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, 18(9), 1975, pp. 509–517.

[12] J.C. Bezdek, N.R. Pal: "Some New Indexes of Cluster Validity," *IEEE Transactions On Systems, Man, and Cybernetics - Part B: Cybernetics*, 28(3), 1998, pp. 301–315.

[13] H. Bischof, A. Leonardis, A. Selb: "MDL Principle for Robust Vector Quantization," *Pattern Analysis and Applications*, 1999, pp. 59–72.

[14] P. Bradley, U. Fayyad, C. Reina: "Clustering Very Large Databases Using EM Mixture Models," in *Proceedings of the 15th International Conference on Pattern Recognition*, vol. 2, 2000, pp. 76–80.

[15] P. Bradley, U. Fayyad, C. Reina: “Scaling Clustering Algorithms to Large Databases,” in *Proceedings. of the 4th International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 9–15.

[16] M.M. Breunig, H.-P. Kriegel, P. Kröger, J. Sander: “Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering,” in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2001, pp. 79–90.

[17] J. Buhmann, H. Kuhnel: “Vector quantization with complexity costs,” *IEEE Transactions on Information Theory*, 39(4), 1993, pp. 1133–1145.

[18] R. R. de Carvalho, S. G. Djorgovski, N. Weir, U. Fayyad, K. Cherkauer, J. Roden, A. Gray: “Clustering Analysis Algorithms and Their Applications to Digital POSS-II Catalogs,” in *Astronomical Data Analysis Software and Systems IV*, ASP Conference Series, vol. 77, 1995.

[19] C.-Y. Chen, S.-C. Hwang, Y.-J. Oyang: “An Incremental Hierarchical Data Clustering Algorithm Based on Gravity Theory,” in *Proceeding of Advances in Knowledge Discovery and Data Mining, 6th Pacific-Asia Conference*, PAKDD 2002, LNCS 2336, 2002, pp. 237–250.

[20] C. Cheng, A.W. Fu, Y. Zhang: “Entropy-based Subspace Clustering for Mining Numerical Data,” in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999, pp. 84–93.

[21] T. Chiu, D. Fang, J. Chen, Y. Wang, C. Jeris: “A Robust and Scalable Clustering Algorithm for Mixed Type Attributes in Large Database Environment,” in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 263–268.

[22] D. Cutting, D. Karger, J. Pedersen, J. Tukey: "Scatter-gather: A cluster-based approach to browsing large document collections,” in *Proceedings of SIGIR'92*, 1992, pp. 318-329.

[23] R.N. Dave: “Characterization and Detection of Noise In Clustering,” *Pattern Recognition Letters*, 12(11), 1991, pp. 657–664.

[24] A. Dempster, N. Laird, D. Rubin: “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the Royal Statistical Society B*, 39, 1977, pp. 1–38.

[25] J.C. Dunn: “A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters,” *Journal of Cybernetics*, 3(3), 1974, pp. 32–57.

[26] D.L. Davies, D.W. Bouldin: “A cluster separation measure,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2), 1979, pp. 224–227.

[27] Y. El-Sombaty, M.A. Ismail: “On-line Hierarchical Clustering,” *Pattern Recognition Letters*, 19, 1998, pp. 1285–1291.

[28] W.H. Equitz: "A New vector Quantization Clustering Algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(19), 1989, pp. 1568–1575.

[29] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X. Xu: "Incremental Clustering for Mining in a Data Warehousing Environment," in *Proceedings of the 24rd International Conference on Very Large Data Bases*, 1998, pp. 323–333

[30] M. Ester, H-P. Kriegel, J. Sander, X. Xu: "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[31] B.S. Everitt: "Cluster Analysis," 3rd Edition. Edward Arnold / Halsted Press, London, 1992.

[32] M.A.F. Figueiredo, A.K. Jain: "Unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3), 2002, pp. 381–396.

[33] A. Fred, A.K. Jain: "Robust data clustering," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 128–133.

[34] H. Frigui, R. Krishnapuram: "Clustering by competitive agglomeration," *Pattern Recognition*, 30(7), 1997, pp. 1109–1119.

[35] B. Fritzke: "The LBG-U method for vector quantization - An improvement over LBG inspired from neural networks," *Neural Processing Letters*, 5(1), 1997, pp. 35–45.

[36] D. Frossyniotis, A. Likas, A. Stafylopatis: "A clustering method based on boosting," *Pattern Recognition Letters*, 25, 2004, pp. 641–654.

[37] P. Fränti, J. Kivijärvi: "Randomized local search algorithm for the clustering problem," *Pattern Analysis and Applications*, 3(4), 2000, pp. 358–369.

[38] P. Fränti, O. Virmajoki, V. Hautamäki: "Fast PNN-based clustering using k-nearest neighbor graph," in *Proceedings of IEEE International Conference on Data Mining (ICDM 2003)*, 2003, pp. 525–528.

[39] C. Fraley: "Algorithms for Model Based Gaussian Hierarchical Clustering," *SIAM Journal of Scientific Computing*, 20(1), 1998, pp. 270–281.

[40] P. Fränti, T. Kaukoranta: "Binary vector quantizer design using soft centroids," *Signal Processing: Image Communication*, 14(9), 1999, pp. 677–681.

[41] P. Fränti, O. Virmajoki: "Iterative shrinking method for clustering problems," *Pattern Recognition*, 39(5), 2006, pp. 761-765.

[42] A.B. Geva, Y. Steinberg, S. Bruckmair, G. Nahum: "A comparison of cluster validity criteria for a mixture of normal distributed data," *Pattern Recognition Letters*, 21, 2000, pp. 511–529.

[43] D.E. Goldberg: "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley, Reading, MA., 1989.

[44] J. Goldberger, S. Roweis: "Hierarchical Clustering of a Mixture Model," *Neural Information Processing Systems 17* (NIPS'04), 2004, pp. 505–512.

[45] S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O'Callaghan: "Clustering data streams: Theory and practice," *IEEE Transactions on Knowledge and Data Engineering*, 15(3), 2003, pp. 515–528.

[46] S. Guha, R. Rastogi, K. Shim: "CURE: An Efficient Clustering Algorithm for Large Databases," in *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, 1998, pp. 73–84.

[47] L.O. Hall, I.B. Özyurt, J.C. Bezdek: "Clustering with a Genetically Optimized Approach," *IEEE Transactions On Evolutionary Computation*, 3(2), 1999, pp. 103–112.

[48] P. Hansen, N. Mladenovic: "J-MEANS: a new local search heuristic for minimum sum of squares clustering," *Pattern Recognition*, 34(2), 2001, pp. 187–529.

[49] J.A. Hartigan: "Clustering Algorithms". Wiley Series in probability and mathematical statistics. John Wiley and Sons, Inc., 1975.

[50] A. Hinneburg, D.A. Keim: "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," in *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, (KDD)*, 1998, pp. 58–65.

[51] A. Hyvärinen, J. Karhunen, E. Oja: "Independent Component Analysis," John Wiley & Sons, 2001.

[52] P.K. Ito: "Robustness of ANOVA and MANOVA Test Procedures," in P.R. Krishnaiah (ed). *Handbook of Statistics 1: Analysis of Variance*. North-Holland Publishing Company, 1980, pp. 199–236.

[53] A.K. Jain, M.N. Murty, P.J. Flynn: "Data Clustering: A Review," *ACM Computing Surveys*, 31(3), 1999, pp. 264–323.

[54] H. Jin, K.-S. Leung, M.-L. Wong, Z.-B. Xu: "Scalable model-based cluster analysis using clustering features," *Pattern Recognition*, 38(5), 2005, pp. 637–649.

[55] I.T. Joliffe: "Principal Component Analysis," Springer, New York, 1986.

[56] G. Karypis, E.-H. Han, V. Kumar: "Chameleon: hierarchical clustering using dynamic modeling," *Computer*, 32(8), 1999, pp. 68–75.

[57] R. Kass, L. Wasserman: "A reference Bayesian test for nested hypotheses and its relationship to the Schwarz criterion," *Journal of the American Statistical Association*, 90, 1994, pp. 773–795.

[58] L. Kaufman, P.J. Rousseeuw: "Finding Groups in Data: An Introduction to Cluster Analysis". John Wiley Sons, New York, 1990.

[59] T. Kaukoranta, P. Fränti, O. Nevalainen: "Iterative split-and-merge algorithm for VQ codebook generation," *Optical Engineering*, 37(10), 1998, pp. 2726–2732.

[60] J. Kennington, R. Helgason: “Algorithms for Network Programming,” Wiley, New York, 1980.

[61] T. Kinnunen, T. Kilpeläinen, P. Fränti: "Comparison of clustering algorithms in speaker identification,” in *Proceedings of IASTED International Conference on Signal Processing and Communications (SPC'2000)*, 2000, pp. 222–227.

[62] M. Kloppenburg, P. Tavan: “Deterministic Annealing for Density Estimation by Multivariate Normal Mixtures,” *Physical Review Letters E*, 55(3), 1997, pp. R2089–R2092.

[63] T. Kohonen: “Self-organizing Maps,” Springer, New York, 1995.

[64] G. Kolano, P. Regel-Brietzmann: “Combination of Vector Quantization and Gaussian Mixture Models for Speaker Verification,” in *Proceedings of EURO-SPEECH-1999*, 1999, pp. 1203–1206.

[65] W.L.G. Koontz, K. Fukunaga: “A Nonparametric Valley-Seeking Technique for Cluster Analysis,” *IEEE Transactions on Computers*, C-21(2), 1972, pp. 171–178.

[66] W.L.G. Koontz, K. Fukunaga: “Asymptotic Analysis of a Nonparametric Clustering Technique,” *IEEE Transactions on Computers*, C-21(9), 1972, pp. 967–974.

[67] W.L.G. Koontz, P.M. Narendra, K. Fukunaga: “A Graph-Theoretic Approach to Nonparametric Cluster Analysis,” *IEEE Transactions on Computers*, C-25 (9), 1976, pp. 936–944.

[68] R. Krishnapuram, J.M. Keller: “A Possibilistic Approach to Clustering,” *IEEE Transactions on Fuzzy Systems*, 1(2), 1993, pp. 98–110.

[69] L.I. Kuncheva: “Fuzzy Classifier Design,” Physica-Verlag, Heidelberg, 2000.

[70] S. Kundu: “Gravitational Clustering: a new approach based on the spatial distribution of the points,” *Pattern Recognition*, 32, 1999, pp. 1149–1160.

[71] A. Likas, N. Vlassis, J. J. Verbeek: "The global k-means clustering algorithm,” *Pattern Recognition*, 36(2), 2003, pp. 451–461.

[72] Y. Linde, A. Buzo, R.M. Gray: “An algorithm for vector quantizer design,” *IEEE Transactions on Communications*, 28(1), 1980, pp. 84–95.

[73] J.A. Lozano, P. Larrañaga: “Applying genetic algorithms to search for the best hierarchical clustering of a dataset,” *Pattern Recognition Letters*, 20, 1999, pp. 911–918.

[74] J. MacQueen: “Some Methods for Classification and Analysis of Multivariate Observations,” in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. I, 1967, pp. 281–297.

[75] D. Maio, D. Maltoni, S. Rizzi: “Dynamic clustering of maps in autonomous agents,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11), 1996, pp. 1080–1091.

[76] M. Markou, S. Singh: “Novelty detection: a review—part 1: statistical approaches,” *Signal Processing*, 83(12), 2003, pp. 2481–2497.

[77] U. Maulik, S. Bandyopadhyay: “Genetic algorithm-based clustering technique,” *Pattern Recognition*, 33, 2000, pp. 1455–1465.

[78] G. McLachlan, T. Krishnan: “The EM Algorithm and Extensions,” John Wiley & Sons, New York, 1997.

[79] G. McLachlan, D. Peel: “Finite Mixture Models,” John Wiley & Sons, New York, 2001.

[80] G.W. Milligan: “A Monte Carlo Study of Thirty Internal Criterion Measures for Cluster Analysis,” *Psychometrika*, 46(2), 1981, pp. 187–199

[81] A. Moore: “Very Fast EM-based Mixture Model Clustering Using Multiresolution kd-trees,” in M. Kearns and D. Cohn (eds.) *Advances in Neural Information Processing Systems*, Morgan Kaufman, 1999, pp. 543–549.

[82] H. Nagesh, S. Goil, A. Choudhary: “Adaptive Grids for Clustering Massive Data Sets,” in *Proceedings of the SIAM Conference on Data Mining*, 2001, pp. 506–517.

[83] O. Nasraoui, R. Krishnapuram: “A Robust Estimator Based on Density And Scale Optimization and it’s Application to Clustering,” in *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, vol. 2, 1996, pp.1031–1035.

[84] M.K. Ng: “A note on constrained k-means algorithms,” *Pattern Recognition*, 33, 2000, pp. 515–519.

[85] R. Ng, J. Han: “CLARANS: A Method for Clustering Objects for Spatial Data Mining,” *IEEE Transactions on Knowledge and Data Engineering*, 14(5), 2002, pp. 1003–1016.

[86] C. Ordonez, E. Omiecinski: “FREM: Fast and Robust EM Clustering for Large Data Sets,” in *Proceedings of the 11th International conference on Information and Knowledge Management*, 2002, pp. 590–599.

[87] J.S. Pan, F.R. McInnes, M.A. Jack: “VQ codebook design using genetic algorithms,” *Electronics Letters*, 31 (17), 1995, pp. 1418–1419.

[88] E. Parzen: ” On estimation of a probability density function and mode,” *Annals of Mathematical Statistics*, 33, 1962, pp. 1065–1076.

[89] R. Peck, L. Fisher, J. van Ness: “Approximate Confidence Intervals for the Number of Clusters,” *Journal of the American Statistical Association*, 84(405), 1989, pp. 184–191.

[90] D. Pelleg, A. Moore: “X-means: Extending K-means with Efficient Estimation of the Number of Clusters,” in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 727–734.

[91] A. Ptitsyn, W. Hide: “CLU: A new algorithm for EST clustering,” *BMC Bioinformatics*, 6(Suppl 2):S3, 2005.

[92] K. Rose: "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems,” *Proceedings of IEEE*, 86(11), 1998, pp. 2210–2239.

[93] S. Salvador, P. Chan: “Determining the Number of Clusters/Segments in Hierarchical Clustering/Segmentation Algorithms,” in *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, 2004, pp. 576–584.

[94] P. Sand, A.W. Moore: “Repairing Faulty Mixture Models using Density Estimation,” in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 457–464.

[95] M. Sato, S. Ishii: “On-line EM Algorithm for the Normalized Gaussian Network,” *Neural Computation*, 12(2), 2000, pp. 407–432.

[96] B.J. Schachter, L.S. Davis, A. Rosenfeld: “Some experiments in image segmentation by clustering of local feature values,” *Pattern Recognition*, 11, 1979, pp. 19–28.

[97] P. Scheunders: “A genetic c-means clustering algorithm applied to color image quantization,” *Pattern Recognition*, 30(6), 1997, pp. 859–866.

[98] D.W. Scott: “Multivariate density estimation: theory, practice, and visualization,” John Wiley & Sons, New York, 1992.

[99] J. Shanmugasundaram, U. Fayyad, P. S. Bradley: “Compressed data cubes for OLAP aggregate query approximation on continuous dimensions,” in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999, pp. 223–232.

[100] H. Späth: “Cluster Analysis Algorithms for Data Reduction and Classification of Objects,” Ellis Horwood Limited, West Sussex, UK, 1980.

[101] L.M. Talbot, B.G. Talbot, R.E. Peterson, H.D. Tolley, H.D. Mecham: “Application of Fuzzy Grade-of-Membership Clustering to Analysis of Remote Sensing Data,” *Journal of Climate*, 12(1), 1999, pp. 200–219.

[102] L.Y. Tseng, S.B. Yang: “A genetic clustering algorithm for data with non-spherical-shape clusters,” *Pattern Recognition*, 33, 2000, pp. 1251–1259.

[103] L.Y. Tseng, S.B. Yang: “A genetic approach to the automatic clustering problem,” *Pattern Recognition*, 34, 2001, pp. 415–424.

[104] E.W. Tyree, J.A. Long: “The use of linked line segments for cluster representation and data reduction,” *Pattern Recognition Letters*, 20, 1999, pp. 21–29.

[105] N. Ueda, R. Nakano, Z. Ghahramani, G.E. Hinton: “SMEM Algorithm for Mixture Models,” *Neural Computation*, 12(9), 2000, pp. 2109-2128.

[106] N. Ueda, R. Nakano: “Deterministic annealing EM algorithm,” *Neural Networks*, 11(2), 1998, pp. 189–376.

[107] O. Virmajoki, P. Fränti, T. Kaukoranta: "Iterative shrinking method for generating clustering," in *Proceedings of IEEE International Conference on Image Processing (ICIP'02)*, vol. 2, 2002, pp. 685–688.

[108] K. Wagstaff, C. Cardie, S. Rogers, S. Schroedl: “Constrained K-means Clustering with Background Knowledge,” in *Proceedings of the International Conference on Machine Learning (ICML)*, 2001, pp. 577–584.

[109] M.E. Wall, A. Rechtsteiner, L.M. Rocha: "Singular value decomposition and principal component analysis". in D.P. Berrar, W. Dubitzky, M. Granzow, (eds.) *A Practical Approach to Microarray Data Analysis*, Kluwer, Norwell, MA, 2003, pp. 91–109.

[110] J.H. Ward: "Hierarchical grouping to optimize an objective function," *Journal of American Statistical Association*, 58(301), 1963, pp. 236–244.

[111] N. Wicker, D. Dembele, W. Raffelsberger, O. Poch: "Density of points clustering, application to transcriptomic data analysis," *Nucleic Acids Research*, 30(18), 2002, pp. 3992–4000.

[112] M.A. Wong, T. Lane: "A kth Nearest Neighbour Clustering Procedure," *Journal of the Royal Statistical Society B*, 45(3), 1983, pp. 362–368.

[113] W.E. Wright: "Gravitational Clustering," *Pattern Recognition*, 9, 1977, pp. 151–166.

[114] X. Wu, K. Zhang: "A Better Tree-Structured Vector Quantizer," in *Proceedings of the Data Compression Conference*, 1991, pp. 392–401.

[115] X. Xu, M. Ester, H.-P. Kriegel, J. Sander: "A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases," in *Proceedings of the 14th International Conference on Data Engineering (ICDE'98)*, 1998, pp. 324–331.

[116] K.Y. Yeung, W.L. Ruzzo: "Principal component analysis for clustering gene expression data," *Bioinformatics*, 17, 2001, pp. 763–74.

[117] W.P. Yu, G.W. Chu, M.J. Chung: "A robust line extraction method by unsupervised line clustering," *Pattern Recognition*, 32, 1999, pp. 529–546.

[118] L.A. Zadeh: "Fuzzy sets," *Information and Control*, 8, 1965, pp. 338–353.

[119] K. Zeger, A. Gersho: "Stochastic Relaxation Algorithm for Improved Vector Quantizer Design," *Electronics Letters*, 25(14), 1989, pp. 896–898.

[120] B. Zhang, C. Zhang, X. Yi: "Competitive EM algorithm for finite mixture models," *Pattern Recognition*, 37, 2004, pp. 131–144.

[121] T. Zhang, R. Ramakrishnan, M. Livny: "BIRCH: A New Data Clustering Algorithm and Its Applications," *Data Mining and Knowledge Discovery*, 1(2), 1997, pp. 141–182.

[122] Z. Zivkovic, F. van der Heijden: "Recursive unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5), 2004, pp. 651–656.