Clustering Algorithms: Applications to Software Engineering, Computer Security, Biology and Medicine

Bill Andreopoulos

Technical Report CS-2005-09

March 2005

Department of Computer Science and Engineering

4700 Keele Street North York, Ontario M3J 1P3 Canada

Clustering Algorithms: Applications to Software Engineering, Computer Security, Biology and Medicine

Bill Andreopoulos, billa@cs.yorku.ca
Department of Computer Science, York University, Toronto, Ontario, Canada, M3J 1P3

March 2005

1 Introduction

Data sets to be clustered typically contain a set of *N* objects that have a set of *M* attributes associated with them. Usually N>>M, unless the data set is of very small size. The data objects may have attributes of different types associated with them. Objects may contain attributes of either or both of the following types [Han01, Goebel99, Grambeier02, Fasulo99, Han01, Hartigan75]:

Categorical (or nominal) attributes: These are attributes for which there is no logical way to impose a strict ordering. For example, if attributes included 'weather' and 'city' it is not possible to claim that weather should precede city in an ordering of the attributes, or vice versa. Categorical attributes can be viewed as representing prior knowledge on the objects in the data set.

Numerical (or ordinal) attributes: These are real or integer values that are normalized to span the range 0.0 to 1.0 in the data set. Typically, it is possible to define a strict ordering on the numerical attributes associated with data objects. For example, the numerical attributes may be ordered based on the time at which they were observed or recorded.

There exist two types of classification algorithms [Goebel99, Grambeier02, Fasulo99, Han01, Hartigan75]:

1) Unsupervised clustering algorithms: No prior knowledge exists on the correct classification of any of the objects in the data set to be clustered.
2) Supervised classification algorithms: Classification is based on the knowledge of certain classified objects, whose correct classes were known prior to the knowledge discovery process. The objects for which the classifications were known are used to train the algorithm, which then classifies the other objects based on the results of the training.

Some desirable characteristics of good clustering methods include [Han01]:

1) Ability to discover clusters of arbitrary shapes and sizes. CURE is an example.
2) Dealing well with many dimensions (number of attributes). CLIQUE is an example.
3) The algorithm should scale well to large data sets. Performance, regarding both clustering quality and speed, should not deteriorate when the algorithm is applied to data sets with many objects. Examples are CLARA and CLARANS.
4) Successful handling of noise and outliers. K-Medoids is an example.
5) Insensitivity to the order of the input records. WaveCluster is an example.
6) No requirement for the user to specify input parameters such as the number of clusters. Some of the clustering algorithms proposed recently, such as AutoClass and ROCK, do not require the user to specify the number of clusters.
7) Ability to cluster data sets containing both categorical and numerical attributes. A few algorithms have been proposed in recent years for clustering mixed data. K-Prototypes is an example.

This report is organized as follows. In Section 2 we discuss some important applications of clustering, including applications in software clustering, computer security, biology and medicine. In Section 3 we do a literature survey, describing 40 previous clustering methods organized by type – such as methods that are partitioning, hierarchical, grid-based, density-based, model-based, unsupervised and supervised [Han01]. We explain how some of these methods have been applied to various fields. For example, we discuss applications of ACDC to software clustering and we discuss applications of hierarchical clustering to DNA microarray data.

2 Some Applications of Clustering Algorithms

This section gives an overview of applications of clustering to software clustering, intrusion detection analysis and biology.

2.1 Applications to Software Clustering

Clustering is applied to a software system to help new developers create a mental model of the software system structure. It helps developers understand the structure of legacy software and enables them to compare the documented structure with the automatically created structure. It is especially useful in the absence of experts or accurate design documentation [Tzerpos00, Mancoridis99].

Clustering is applied to large software systems in order to group the files such that files containing source code with similar functionality are placed in the same cluster, while files in different clusters contain source code that performs dissimilar functions. Usually both categorical and numerical data sets can contain information pertinent to a software system that may be clustered:

- A categorical data set on a software system represents for each file, which other files it may invoke. Such data sets are referred to as *static* and exist in a market-basket format. Each row of the data set corresponds to a file *x* of the software system. In each row after the file name *x* there is a list of the other filenames that *x* may invoke during execution.
- A numerical data set on a software system contains the results of a profiling of the execution of the system, representing how many times each file invoked other files during the run time. Such data sets are referred to as *dynamic*. Each row of the data set corresponds to a file *x* of the software system. In each row after the file name *x* there is a list of the other filenames that *x* invoked as well as how many times *x* invoked them, during the profiled run time.

Description of applications to software clustering are given for the LIMBO and ACDC clustering methods [Andritsos04, Tzerpos00].

2.2 Applications to Computer Security

Clustering is applied to files with logged behavior of system users over time, to separate instances of normal activity from instances of abusive or attack activity. Clustering is primarily used in Intrusion Detection Systems, especially when the log files of system user behavior are too large for a human expert to analyze. Both unsupervised and supervised clustering methods can be used.

Unsupervised clustering: When unsupervised clustering is applied to Intrusion Detection Systems' data on which no prior knowledge exists, the data is first clustered and then different clusters are labeled as either 'normal' or 'intrusions' based on the size of the cluster. Then, a new unclassified object is classified in the cluster to which it is the closest. Given a new object d, classification proceeds by finding a cluster C which is closest to d and classifying d according to the label of C as either normal or anomalous [Li04, Portnoy01].

IDSs based on unsupervised clustering focus on activity data that may contain features of an individual TCP connection, such as its duration, protocol type, number of bytes transferred and a flag indicating the connection's normal or error status. Other features may include the number of file creation operations, number of failed login attempts, whether root shell was obtained, the number of connections to the same host as the current connection within the past two seconds, the number of connections to the same service as the current connection within the past two seconds and others.

Clustering of the data sets is done using a typical single-linkage clustering, as described in Section 3.1. This is not the most effective type of clustering but it is very fast with a complexity of O(n).

Then, clusters are labeled based on the size of the cluster, as clusters containing normal instances or clusters containing attacks. Since normal instances constitute a majority of the data set, clusters with more objects are more likely to contain normal instances than attacks. Therefore, the clusters containing the largest number of objects are labeled "normal" while the rest of the clusters are labeled "anomalous". A problem with this approach is that many sub-types of normal objects may exist in the data set – such as using different protocols like ftp, telnet, www, etc – in which case a large number of clusters might be produced containing a small number of normal objects, one for each type of normal use of the network. Then, these normal clusters may be incorrectly labeled as "anomalous".

Supervised classification: Supervised classification is applied to Intrusion Detection Systems for Signature Recognition purposes. This means that signatures representing patterns of previous intrusion activities are learned by the system. Then, new patterns of activity are compared to the previously learned signatures to determine if the activities are normal or if they may be intrusions [Ye01].

The automatic learning of intrusion signatures from historic data containing labeled examples of normal and intrusive activity can be done using a data mining technique such as Support Vector Machines. The learned intrusion signatures should be updated whenever more data of normal and intrusive activity becomes available.

IDS based on signature recognition focus on two main types of activity data: network traffic data and computer audit data. Some categorical activity attributes that can be obtained from this data include the user id, event type, process id, command, remote IP address. Some numerical activity attributes that can be obtained from this data include the time stamp, CPU time, etc.

Using SVMs, new activity patterns are matched to the learned signatures to determine if they are more likely to fall in the class of normal or intrusive patterns. An example of the CCA-S clustering algorithm used for intrusion detection systems is described in Section 3.6.

2.3 Applications to Biology and Medicine

Computational biology is an interdisciplinary field lying at the intersection between computational mathematics and molecular biology [Altman98]. The field's goal is to provide computational techniques for solving computationally intensive problems in molecular biology. Such problems usually involve analyzing and understanding the massive genomic data that is being produced by genome-related projects [Butler01]. *Bioinformatics* is a related term that refers to the development and use of biological databases [Scherf00, Kumar00, Altman98]. This literature review starts off by describing the biological motivation for our research and our

computational aims. Then we discuss mathematical clustering of data sets and some of its applications in computational biology. As illustrative examples we describe previous attempts to cluster the yeast genome and work done in clustering of cancer.

We conducted a thorough survey of all mathematical techniques and software implementations that could be used for clustering genomic data sets [Hartigan75, Goebel99]. This section attempts to explain clustering to a biological audience, as well as clustering applications in biology and medicine. Section 3 describes the mathematical techniques used for clustering, assuming a decent level of mathematical understanding.

Clustering of genomic data sets

One of the main challenges for bioinformatics is to develop computational tools for organizing large genomic data sets, such that useful information can be extracted [Khan99, Eisen99, Eisen98, Brown99]. *Clustering* is one of the main tools used for this purpose. Clustering means to partition a large set of data items into groups, such that items with similar characteristics are grouped together; as a result groups of similar items are formed [Hartigan75, Goebel99, Michaels98, Hastie00, Lazzeroni00, BenDor99, Brown00, Mjolsness99, Mjolsness992]. When it comes to clustering a genomic data set, the resulting groups usually contain genes that have a similarity, with respect to a distance measure.

In order to compare two data items for similarity, clustering techniques need to adopt a mathematical description of similarity. If a gene's behavior over *n* time points is represented as an *n*-dimensional vector, then mathematical techniques can be used to measure similarity in two vectors [Hartigan75, Eisen98, Eisen99, Michaels98]. *Figure 1* shows how a mathematical technique, the *Euclidean distance* of two *n*-dimensional vectors, can be used to compare two genes' behavior. Section 3 provides more details about the mathematical techniques used for clustering.

When it comes to clustering a genomic data set, the set can be of either type: a *gene expression data set*, or a *synthetic mutant lethality (SML)* data set. Although in our work we are concerned with the latter type, previous work has focused on gene expression data very similar to our data. Thus, in the next two subsections we describe both types of data sets in more detail.

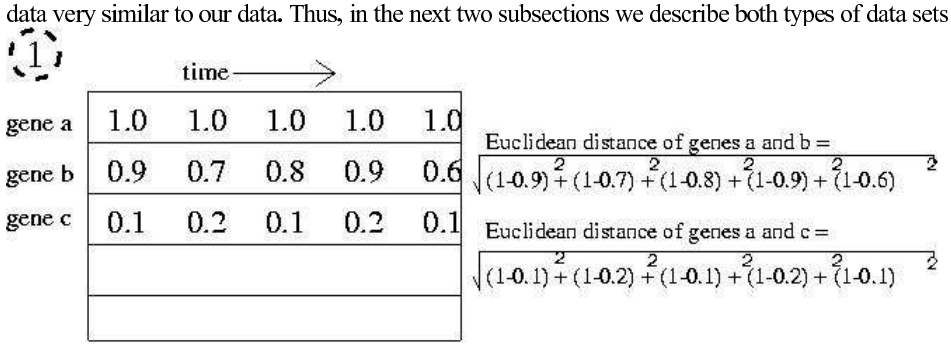


Figure 1 - Every row is a 5-dimensional vector representing a gene's expression pattern over 5 time points. The Euclidean distance between two vectors indicates the similarity between two gene expression patterns.

Clustering of gene expression data sets

For the purpose of understanding gene expression data sets, the method used almost exclusively until now is clustering. A gene expression data set contains measurements of the *expression levels* of many of the genes of an organism [Eisen99, Brown99]. A number of such gene expression measurements are usually taken, either across different points of *time* or across various *tissue* samples of an organism. Given a gene expression data set, clustering can be used either to cluster genes, or to cluster tissues. In the former case clustering forms groups of similar genes, by looking at each gene's pattern of expression across time. In the latter case clustering forms groups of similar tissues, by looking at each tissue and the expression of all genes in it [Eisen98, Gasch00, Spellman98, Michaels98]. Both of these cases are illustrated in Figures 2a and 2b.

Figure 2a shows an example of a gene expression data set. The data set is represented by a table of gene expression measurements, where each *row* corresponds to a gene and each *column* corresponds to a point of time. Each data *point* contains a number, to reflect the level of expression of the gene at that point of time. Figure 2a shows how clustering may separate genes that exhibit different expression patterns across time. The degree of similarity between genes can be assessed by a similarity metric such as that shown in Figure 1. The results of clustering can be displayed graphically, by appending a *tree* to the left of the table to reflect the relationships among genes. This tree orders the genes in the table, such that genes with similar expression patterns are adjacent.

Figure 2b is different, in that each column corresponds to a separate tissue sample and each row corresponds to a gene. Clustering is used to separate molecularly distinct tissues, on the basis of differences in patterns of gene expression across the tissues.

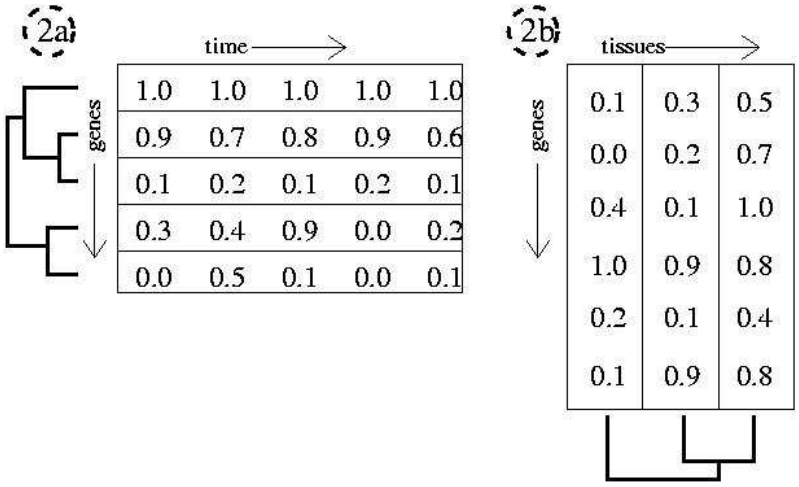


Figure 2a - A gene expression data set of gene expression levels across points of time. Figure 2b - A gene expression data set of gene expression levels across various tissues.

Clustering of synthetic mutant lethality (SML) data sets

A SML data set contains information on which combinations of two genes have been observed to interact in SML studies [Sherman97]. SML searches for interactions between two genes, by examining if simultaneous mutations in both genes cause *synthetic lethality*. Synthetic lethality between two genes is observed when mutations in both genes lead to death (or some other growth defect in the cell) when combined; however, each mutation on its own is not lethal and has no effect on the cell. This can usually be explained by genetic redundancy, that both genes perform the same essential function but each gene carries out the function in a different way [Jarvik75, Novick89, Sandrock99].

Figure 3 illustrates an example of an SML data set, represented as a square table. Columns and rows correspond to genes. Each data point in the table contains a *boolean* value (1 or 0) to reflect whether a synthetic lethality has been observed for the corresponding genes or not. Clustering of SML data forms groups of genes with similar patterns of synthetic lethalities. A *tree* can be appended to the left of the table to reflect the relationships among genes.

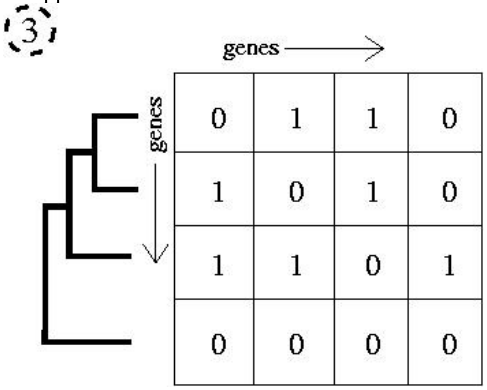


Figure 3 - An SML data set.

Clustering the yeast genes using gene expression data

The community of yeast researchers has accomplished a considerable amount of work in the past [sgd, mips, ypd, kegg]. Previous work has mostly focused on applying large-scale experimental approaches to provide a functional analysis of the yeast genome [Petra00]. The next sections give an overview of the experiments performed on clustering gene expression data from yeast at Stanford University School of Medicine [yeaststress, yeastcellcycle]. Knowledge of protein-protein interactions accumulated from widely used *yeast two-hybrid studies* [Fields89] may be combined with gene expression data for better results.

Clustering the yeast genes in response to environmental changes

Gasch et al. [yeaststress, Gasch00] attempted to cluster the yeast genes based on their expression patterns as they responded to diverse environmental changes (e.g. temperature shocks, amino acid starvation). A large cluster was formed, in which all genes showed a similar response to all environmental changes. Many smaller clusters were also formed, containing genes whose responses were specialized for specific environmental conditions. Clustering of the data was performed using Eisen's hierarchical clustering algorithm (see Section 3.2).

Clustering the cell cycle-regulated yeast genes

Spellman et al. [yeastcellcycle, Spellman98] attempted to cluster the genes that get expressed during the yeast cell cycle. The genes were clustered into groups, according to the phase of the cell cycle during which their expression levels vary. Thus, genes in a cluster share similar expression levels during the same phase of the yeast cell cycle. Clustering of the data was performed using Eisen's hierarchical clustering algorithm (see Section 3.2).

Functional analysis of the yeast genome

The examples above specifically demonstrate that clustering tends to organize genes into functional categories [Petra00, Kumar00]. Genes that are grouped together in the same cluster usually share common functions. In the first clustering example, genes were clustered into functional categories such as carbohydrate metabolism, protein folding, DNA damage repair, and intracellular signaling [Gasch00]. In the second example, it was found that many genes are involved in functions needed for the yeast cell cycle such as DNA replication, DNA repair, budding, cytokinesis and mitosis [Spellman98].

Finding gene functions

Finding gene functions helps identify the genes that have an impact on a biological process [Straus99]. Then, one could find ways to manipulate the biological process, leading to new areas of disease treatment. For example, one could identify the genes involved in cancer tumor predisposition.

Genetic interactions usually occur between genes with common functions, or genes whose products are members of the same functional pathway [Petra00]. Thus, clustering genes on the basis of similarity in interaction patterns, usually results into groups (or "families") of genes with similar functions [Eisen98]. Functions can then be assigned to a gene based on the known functions of its interacting genes. Clustering of genes of known function with poorly characterized genes may provide clues about the functions of many genes for which information is not currently available [Schwik00, Uetz00, Ito00].

Functional prediction

The function of a gene G can be predicted as follows [Schwik00, Uetz00, Ito00]. The annotated functions of all genes in the same cluster with G are ordered in a list, from the most frequent annotation to the least frequent. Functions that occur the same number of times should be ordered arbitrarily. Everything after the third entry in the list can be discarded, and the remaining three or fewer functions can be declared as predictions for the function of G.

In order to predict gene functions this way, it is necessary to assume that functional information is already available on some yeast genes. The research group in Howard Hughes Medical Institute at Stanford University School of Medicine has done many experiments on functional annotation of the yeast genome (an overview of their work was given previously). Currently approximately 2,467 genes have a functional annotation in the *Saccharomyces* Genome Database (available at http://genome-www.stanford.edu/Saccharomyces/). Some functional families to which genes have been assigned include the mitotic cell division cycle [Spellman98] , sporulation [Chu98], the diauxic shift [DeRisi97] , cholesterol biosynthesis, the immediate-early response, signaling and angiogenesis, and tissue remodeling and wound healing [Gasch00].

Cancer Tumor Classification using gene expression data

Cancer is a popular application of clustering. This section gives an overview of some clustering applications to cancer [Scherf00, Wein97, Ross00, Aliz00, Slonim00, Golub99, DeRisi96, Khan992, Kononen98, Sgroi99, Afshari99, Wang99, Hess98]. The clustering technique most widely applied to cancer tumors is Eisen's hierarchical clustering algorithm (see Section 3.2).

In recent years the amount of gene expression data derived from cancer tumors has increased exponentially [DeRisi96, Bittner00, Alon99, Furey00, Zhang97, Perou99, BenDor00]. Researchers have generated large data sets, containing gene expression measurements of when and in which tumors genes are turned on or off. The driving force behind this gene expression data collection effort is the hope that it may provide the information needed to lead to a more reliable classification of cancers [Khan992, Kononen98, Sgroi99, Afshari99, Wang99, Hess98]. Classifying a tumor means to be able to place a tumor into a specific class, all members of which share the same behavioral properties and molecular characteristics [Bittner00, Alon99, Furey00, Zhang97, Perou99, BenDor00]. Many benefits can be derived from classification of tumors.

Reliable classification of tumors is essential for cancer to be treated successfully [Khan992, Kononen98, Sgroi99, Afshari99, Wang99, Hess98]. A big challenge for cancer treatment has been to target specific therapies to molecularly and behaviorally distinct tumor types [Ross00, Aliz00, Slonim00, Golub99, DeRisi96]. This can only be achieved if researchers classify molecularly distinct tumors into subgroups, to reflect the responses of distinct types of tumors to therapy and the different clinical courses they follow [Perou00]. Tumors of identical appearance may progress at very different speeds, and the best course of treatment might be very different for each tumor; for example, prostate cancers of identical grade can follow very different clinical courses after a specific treatment [Cole99].

Researchers have faced two main problems in their search for improved methods of cancer tumor classification: *class discovery* and *class prediction*. Next we describe each of these problems. Three examples of cancer tumor classification are given below.

Class Discovery: *Class discovery* refers to the process of dividing tumor samples into groups with similar behavioral properties and molecular characteristics. Previously unknown tumor subtypes may be identified this way. Class discovery entails two issues:

1. Developing techniques to *classify* tumors by gene expression data. Section 3 provides a description of mathematical clustering techniques that have been successfully applied to gene expression data [Slonim00, Golub99].
2. Determining whether classes produced by such clustering techniques are meaningful - that is, whether they reflect true structure in the data rather than simply random aggregation. In some cases it is difficult to evaluate whether the results of a clustering attempt are meaningful or not. In such cases, the results can be tested by *class prediction* (described in the next subsection).

Class Prediction: *Class prediction* refers to the process of determining the correct class for a new tumor sample, given a set of known classes. Since classes indicate how a patient will respond after treatment with a certain drug, correct class prediction may suggest whether a patient will benefit from treatment. Class prediction entails two issues:

1. Developing tools for this purpose. The main tools usually employed are discriminant analysis and supervised learning [5, 6].
2. Determining the genes that are most relevant to the class distinction to be predicted. *Marker gene identification* refers to the identification of "marker" genes that best characterize the different tumor classes [Slonim00, Golub99].

Detailed examples of cancer tumor classification are described below.

Example of Cancer Tumor Classification: Classification of acute leukemias

Golub et al. applied cancer tumor classification (i.e. class discovery and class prediction) to human *acute leukemias*. Acute leukemias can be divided into two classes, *acute myeloid leukemia* (AML) and *acute lymphoblastic leukemia* (ALL), that originate from cells of myeloid or lymphoid origin respectively [Slonim00, Golub99]. The primary goal of this classification attempt was to find molecularly distinct leukemia classes, on the basis of patterns of gene expression in the tumors.

They succeeded in classifying 38 leukemia tumor samples into two molecularly distinct classes, based on the expression patterns of their genes. The first class contained mostly ALL and the second class contained mostly AML. It was the first time that class discovery was effective at classifying acute leukemias accurately. This taxonomy is of great clinical significance, because the two molecularly distinct leukemia subgroups show big differences in clinical behavior. Class prediction was also applied to leukemias with confirmed pathological diagnoses, verifying that ALL tumors were correctly distinguished from AML tumors [Slonim00, Golub99].

To classify tumors, Golub et al. used a clustering technique called self-organizing maps (SOMs). This clustering technique is described in detail in Section 3.5. In this approach, the user specifies the number of classes to be identified. Then the data set is partitioned into the desired number of classes. The SOM technique automatically discovered the distinction between AML tumors and ALL tumors with no previous knowledge of these classes [Slonim00, Golub99].

Example of Cancer Tumor Classification: Classification of diffuse large B-cell lymphoma tumors

Alizadeh et al. applied classification of cancer tumors to *diffuse large B-cell lymphoma (DLBCL)* tumors, the most common form of non-Hodgkin's lymphoma. The primary goal of this classification attempt was to find molecularly distinct DLBCL classes, on the basis of patterns of gene expression in the DLBCL tumors [Aliz00].

They succeeded in classifying tumors into two molecularly distinct DLBCL classes, according to the expression patterns of their genes. The first class contained germinal center (GC) B-like DLBCL tumors, and the second class activated B-like DLBCL tumors. This taxonomy is of great clinical significance, because the two molecularly distinct DLBCL subgroups show big differences in clinical behavior. Patients in the activated B-like DLBCL class show worse overall survival than in the GC B-like DLBCL class [Aliz00]. To classify tumors, Alizadeh et al. used Eisen's hierarchical clustering algorithm (see Section 3.2).

Example of Cancer Tumor Classification: Classification of 60 cancer cell lines derived from a variety of tumors

The National Cancer Institute (NCI) applied classification to 60 cancer cell lines derived from tumors from various tissues. The cell lines were derived from leukaemias, melanomas, and cancers of renal, ovarian, breast, prostate, lung, colon and central nervous system (CNS) origin. The primary goal of this classification attempt was to classify the cell lines, based on the patterns of gene expression in the cell lines [Scherf00, Wein97, Ross00].

The NCI succeeded in classifying the 60 cell lines into subgroups, according to similarity in patterns of gene expression across the cell lines. Classifying the cell lines based on the patterns of gene expression revealed a correspondence to the origins of the tumors from which the cell lines were derived. For example, all of the colon cancer lines, the CNS lines and the leukaemias clustered into corresponding groups. Thus, there is a consistent relationship between the gene expression patterns and the tissue of origin of the cell lines. This also implies that specific features of the gene expression patterns may be related to properties of the cell lines, such as their doubling time in culture, drug metabolism or the interferon response [Scherf00, Wein97, Ross00]. To classify tumors, the NCI used Eisen's hierarchical clustering algorithm (see Section 3.2).

3 Clustering Algorithms

This section gives an overview of clustering algorithms that have been proposed in the past, along with some technical details.

3.1 Partitioning methods

Partitioning methods are based on the philosophy that objects are partitioned from the beginning into a fixed number of clusters and during the clustering process they change clusters based on their similarity to the closest cluster. Partitioning methods typically require the user to specify beforehand as a parameter, the number of clusters to be produced.

a) Numerical

Several partitioning clustering methods have been developed for data sets with numerical attribute types.

Farthest First Traversal k-center Algorithm

The farthest-first traversal *k*-center algorithm (FFT) is a fast, greedy algorithm that minimizes the maximum cluster radius [Hochbaum85]. In FFT, *k* points are first selected as cluster centers. The remaining points are added to the cluster whose center is the closest. The first center is chosen randomly. The second center is greedily chosen as the point furthest from the first. Each remaining center is determined by greedily choosing the point farthest from the set of already chosen centers, where the furthest point, *x*, from a set, *D*, is defined as, $\max_x\{\min\{d(x,j), j(-D\}\}$.

The farthest-first traversal algorithm is shown in Figure 4.

```
farthest_first_traversal(D: data set, k: integer) {
   randomly select first center;
   //select centers
   for (I= 2,…,k) {
      for (each remaining point) { calculate distance to the current center set; }
      select the point with maximum distance as new center;    }
   //assign remaining points
   for (each remaining point) {
      calculate the distance to each cluster center;
      put it to the cluster with minimum distance;    }
}
```

Figure 4 - Algorithm of farthest-first traversal. Adopted from [Hochbaum85].

The time complexity of FFT is $\Theta(sk)$, where *s* is the number of data objects and *k* is the number of clusters.

Single-linkage clustering

The single-linkage clustering method is a one-pass clustering algorithm. It is not the most effective and accurate clustering algorithm that exists, but it is efficient as it has a complexity of O(n) where n is the number of data objects [Portnoy01].

Single-linkage clustering consists of the following steps:

1) Initialize the set of clusters, S, to the empty set.
2) Obtain an object d from the data set. If S is empty, then create a cluster with d and add it to S. Otherwise, find the cluster in S that is closest to this object. In other words, find the closest cluster C to d in S.
3) If the distance between d and C is less than or equal to a user specified threshold W then associate d with the cluster C. Else, create a new cluster for d in S.
4) Repeat steps 2 and 3 until no objects are left in the data set.

k-Means

k-Means is a clustering algorithm that deals with numerical annotations (NAs) primarily, although it can also be applied to categorical data sets whose annotations have binary values, by viewing the binary values as numerical annotations. The k-Means clustering algorithm for numerical data sets requires the user to specify the number of clusters to be produced and the algorithm builds and refines the specified number of clusters [Alsabti98].

During the k- Means clustering algorithm for numerical data sets, the following generic loop is performed:

```
Insert the first K objects into K new clusters.
Calculate the initial K means for K clusters.
Repeat {
      For (each object O) {
        Calculate the dissimilarity between object O and the means of all clusters.
        Insert object O into the cluster C whose mean is closest to object O.
        }
```

```
        Recalculate the cluster means so that the cluster dissimilarity between mean and
objects is minimized.
} until (no or few objects change clusters)  .
```

Each cluster has a mean associated with it. Means are used to choose the closest cluster to an object by computing the dissimilarity between the cluster's mean and the object. In the loop above, the object is then allocated to the closest cluster and the mean gets updated.

A dissimilarity metric is needed to choose the closest cluster to an object by computing the dissimilarity between the cluster's mean and the object. Assume that each object is described by m numerical attributes. Let $X=\{x_1,x_2,\ldots,x_m\}$ be an object, where x_i is the value for the ith attribute, and $Q=\{q_1,q_2,\ldots,q_m\}$ be the mean of a cluster. The dissimilarity between X and Q is defined as:

$$\text{dissimilarity}(X,Q)= \sum_{j=1}^{m} x_j - q_j / m$$

A mean Q for a cluster C with n objects is found by computing for each attribute position i $\sum_{j=1}^{n} X_{ij} / n$.

The main strength of k-means is that it is efficient with a computational complexity of $O(tkn)$, where n is the number of objects, k is the number of clusters and t is the number of iterations. Normally, $k,t << n$. A weakness is that it often terminates at a local optimum rather than a global optimum. Furthermore, the number of clusters k needs to be specified in advance by the user. K-means is unable to handle noisy data and outliers and it is not suitable to discover clusters with non-convex shapes. Finally, the results depend on the order of the objects in the input data set as different orderings will produce very different results.

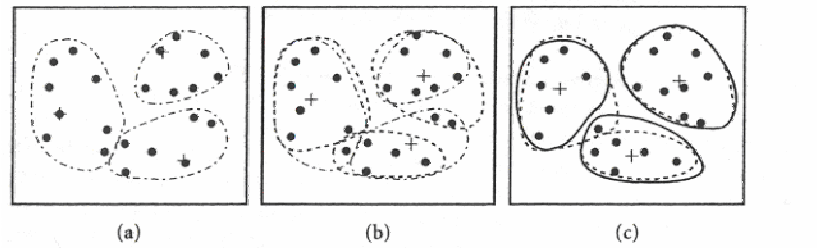


Figure 5 – Clustering of a set of objects using the k-means method. The mean of each cluster is a "+". Adopted from [Han01].

k-Medoids or PAM: Partitioning Around Medoids

k-Medoids deals with the problem of outliers posed by k-Means. For example, if an object has a very different value from the other objects in a cluster then that object will distort the result with k-Means, because the mean of a cluster will have to be altered to deal with the object. k-Medoids is similar to k-Means except that the mean of each cluster is the object that is nearest to the "center" of the cluster [Kaufmann87].

The idea of partitioning with k-Medoids is to reduce the distance between all objects in a cluster and the most centrally located object in the cluster.

The strategy is very similar to the k-Means strategy: first k of the n objects are inserted in k clusters arbitrarily and the medoid for each cluster is set equal to the object inserted in it. Each remaining object is inserted into the cluster whose medoid is most similar to it. Then each medoid i in a cluster may be swapped with one of the non-medoids h, as long as the total swapping cost TC_{ih} is negative. The total cost for swapping medoid i with non-medoid h is determined by using the function: $TC_{ih}=\sum_j C_{jih}$.

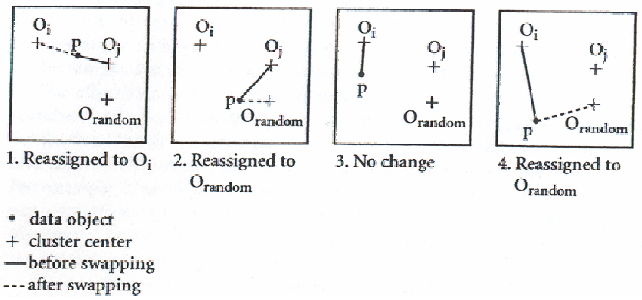


Figure 6 – k-Medoids clustering. The medoid is swapped after each loop, if the total cost $TC_{ih}=\sum_j C_{jih}$ for swapping a medoid with a non-medoid is negative. Adopted from [Kaufmann87].

CLARA

k-Medoids only works effectively for small data sets but does not scale well to large data sets. CLARA (Clustering Large Applications) attempts to overcome this problem. CLARA (Clustering Large Applications) is an extension of k-Medoids whose main focus is to scale well for large data sets. The idea behind CLARA is to take a sample of the whole data set into consideration, as a representative of the entire data set. CLARA selects a sample of the whole data set as a representative of the data set. Medoids are then chosen from this sampling using a method similar to k-Medoids. If the sampling has been done properly, the Medoids chosen from the sample are usually similar to the ones that would have been chosen from the whole data set. The effectiveness of CLARA depends on the size of the sample selected, since it searches for medoids among the selected sample. The computational complexity of CLARA is O(ks2+k(n-k)), where s is the sample size, k is the number of clusters, and n is the total number of objects [Kaufmann90].

CLARANS

CLARANS (Clustering Large Applications Based Upon Randomized Search) improves upon CLARA by allowing for the selected sample to be changed at each iteration. Thus, unlike CLARA, CLARANS does not limit itself to a fixed sample at all stages throughout the execution. CLARANS draws a sample with some randomness at each stage of the search. CLARANS is more effective than k-Medoids and CLARA. CLARANS allows the detection of outliers. However, its computational complexity is O(n2) where n is the number of objects [Ng94].

Fuzzy k-Means Clustering

The clusters produced by the k-means procedure are sometimes called "hard" or "crisp" clusters, since any feature vector **x** either is or is not a member of a particular cluster. This is in contrast to "soft" or "fuzzy" clusters, in which a feature vector **x** can have a degree of membership in each cluster. The **fuzzy-k-means procedure** of Bezdek [Bezdek81, Dembele03, Gasch02] allows each feature vector **x** to have a degree of membership in Cluster i:

- Choose the number of classes k, with 1<*k*<n.
- Choose a value for the fuzziness exponent f, with f>1.
- Choose a definition of distance in the variable-space.
- Choose a value for the stopping criterion e (e= 0.001 gives reasonable convergence).
- Make initial guesses for the means $\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_k$
- Until there are no changes in any mean:
  - Use the estimated means to find the degree of membership u(j,i) of $\mathbf{x}_j$ in Cluster i. For example, if a(j,i) = exp(- $\| \mathbf{x}_j - \mathbf{m}_i \|^2$), one might use u(j,i) = a(j,i) / sum_j a(j,i).
  - For i from 1 to k
    - Replace $\mathbf{m}_i$ with the fuzzy mean of all of the examples for Cluster i -- $$\mathbf{m}_i = \frac{\sum_j u(j,i)^2 \mathbf{x}_j}{\sum_j u(j,i)^2}$$
  - end_for
- end_until

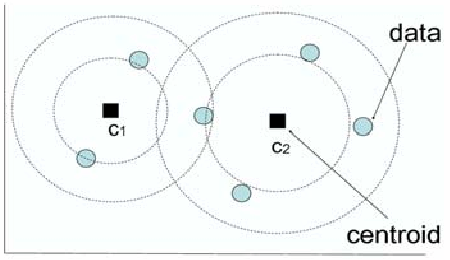


Figure 7 - Fuzzy k-means clustering. Adopted from [Bezdek81].

b) Categorical

Several partitioning clustering methods have been developed for data sets with categorical attribute types.

k-Modes

K-Modes is a clustering algorithm that deals with categorical attributes (CAs) only. The k-Modes clustering algorithm for categorical data sets requires the user to specify from the beginning the number of clusters to be produced and the algorithm builds and refines the specified number of clusters [Huang98].

During the k-Modes clustering algorithm for categorical data sets, the following generic loop is performed:

```
Insert the first K objects into K new clusters.
Calculate the initial K modes for K clusters.
Repeat {
      For (each object O) {
        Calculate the similarity between object O and the modes of all clusters.
        Insert object O into the cluster C whose mode is the most similar to object O.
        }
        Recalculate the cluster modes so that the cluster similarity between mode and objects
is maximized.
} until (no or few objects change clusters)  .
```

Each cluster has a mode associated with it. Modes are used to choose the closest cluster to an object by computing the similarity between the cluster's mode and the object. In the loop above, the object is then allocated to the closest cluster and the mode gets updated.

A similarity metric is needed to choose the closest cluster to an object by computing the similarity between the cluster's mode and the object. Assume that each object is described by m attributes. Let $X=\{x_1,x_2,\ldots,x_m\}$ be an object, where x_i is the value for the ith attribute, and $Q=\{q_1,q_2,\ldots,q_m\}$ be the mode of a cluster. The similarity between X and Q is defined as:

$$\text{similarity}(X,Q)=\sum_{j=1}^{m}\delta(x_j,q_j)$$

where $\delta(x_j,q_j)=1$ *if* $x_j=q_j$, *0 otherwise.*

A mode Q for a cluster C is found by maximizing $\sum_{i=1}^{n} similarity(X_i,Q)$, which is maximized if and only if frequency($X_j=q_j|C$) >= frequency($X_j=c_j|C$) for $q_j \neq c_j$ for all j=1 to m. Frequency($X_j=q_j|C$) is the number of objects in the cluster C that have the value q_j in the jth attribute X_j. Thus, frequency($X_j=q_j|C$) must be maximal for all j=1 to m.

The main strength of k-modes is that it is relatively efficient with a computational complexity of $O(tkn)$, where n is the number of objects, k is the number of clusters and t is the number of iterations. Normally, $k,t << n$. A weakness is that it often terminates at a local optimum rather than a global optimum. Furthermore, the number of clusters k needs to be specified in advance by the user. K-modes is unable to handle noisy data and outliers and it is not suitable to discover clusters with non-convex shapes. Finally, the results depend on the order of the objects in the input data set as different orderings will produce very different results.

Fuzzy k-Modes Clustering

A fuzzy k-Modes algorithm was proposed in [Huang99]. In real applications there is often no sharp boundary between clusters so that fuzzy clustering is often better suited for the data. Membership degrees between zero and one are used in fuzzy clustering instead of crisp assigments of the data to clusters.

The fuzzy k-modes algorithm contains extensions to the fuzzy k-means algorithm for clustering categorical data. By using a simple matching dissimilarity measure for categorical objects and modes instead of means for clusters, a new approach is developed, which allows the use of the k-means paradigm to efficiently cluster large categorical data sets.

Squeezer

Squeezer is introduced in [He2002] as a one-pass algorithm. Squeezer repeatedly reads tuples from the data set one by one. When the first tuple arrives, it forms a cluster alone. The consequent tuples are either put into an existing cluster or rejected by all existing clusters to form a new cluster by the given similarity function.

Squeezer clustering consists of the following steps:

1) Initialize the set of clusters, S, to the empty set.
2) Obtain an object d from the data set. If S is empty, then create a cluster with d and add it to S. Otherwise, find the cluster in S that is closest to this object. In other words, find the closest cluster C to d in S.
3) If the distance between d and C is less than or equal to a user specified threshold W then associate d with the cluster C. Else, create a new cluster for d in S.
4) Repeat steps 2 and 3 until no objects are left in the data set.

c) Mixed Categorical and Numerical

A few partitioning clustering methods have been developed for data sets with mixed numerical and categorical attribute types.

k-Prototypes

An extension of k-Modes called k-Prototypes was proposed in [Huang97] for dealing with mixed numerical and categorical data sets. k-Prototypes is a partition-based algorithm that uses a heterogeneous distance function to compute distance for mixed data. The heterogeneous distance function requires weighting the contribution of the numerical attributes versus that of the categorical attributes. K-Prototypes uses an iterative approach to clustering that continues until objects stop changing clusters.

Bunch

Bunch is a clustering tool intended to aid the software developer and maintainer in understanding, verifying and maintaining a source code base [Mancoridis99]. The input to Bunch is a Module Dependency Graph (MDG). Figure 8 shows an MDG graph.

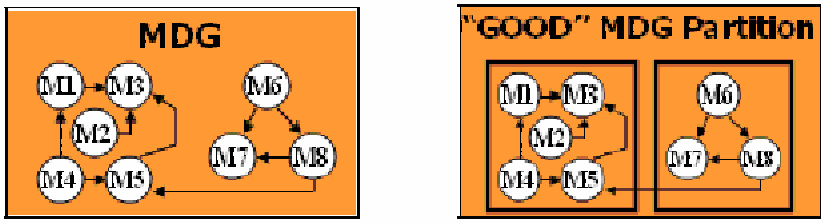


Figure 8 – An MDG graph. Adopted from [Mancoridis99].

Bunch views the clustering problem as trying to find a good partition of an MDG graph. Bunch views a "good partition" as a partition where highly interdependent modules are grouped in the same clusters (representing subsystems) and independent modules are assigned to separate clusters. Figure 8b shows a "good" partitioning of figure 8a. Finding a good graph partition involves systematically navigating through a very large search space of all possible partitions for that graph. Bunch treats graph partitioning (clustering) as an optimization problem. The goal of the optimization is to maximize the value of an objective function, called Modularization Quality (MQ) [Mancoridis99], which is defined as:

Modularization Quality(MQ) *Intra-Connectivity(A)* *Inter-Connectivity(E)*

$$MQ = \begin{cases} \frac{\sum_{i=1}^{k} A_i}{k} - \frac{\sum_{i,j=1 \atop i<j}^{k} E_{i,j}}{\frac{k(k-1)}{2}} & k > 1 \\ A_1 & k = 1 \end{cases} \qquad A_i = \frac{\mu_i}{N_i^2} \qquad E_{i,j} = \begin{cases} 0 & i = j \\ \frac{\varepsilon_{i,j}}{2N_iN_j} & i \neq j \end{cases}$$

The MQ of an MDG that is partitioned into K clusters is the difference between the average interconnectivity and intraconnectivity of the K clusters [Mancoridis99]. The intraconnectivity Ai of cluster i - desired to be high - consisting of Ni nodes and μi intra-edges is the fraction of μi over the maximum number of intra-edges of i (i.e. Ni^2). The inter-connectivity Ei;j between two distinct clusters i and j - desired to be low - consisting of Ni and Nj nodes, respectively, and with "εi;j inter-edges is the fraction of "εi;j over the maximum number of inter-edges between i and j (i.e., 2NiNj). For more information see [Mancoridis99].

A naive algorithm for finding the "best" (optimal) partition of an MDG is to enumerate all of its partitions and select the partition with the largest MQ value. However, this algorithm is not practical for MDGs with a large (over 15) number of modules [Mancoridis99], because the number of partitions of a graph grows exponentially with respect to its number of nodes. Thus, Bunch uses more efficient search algorithms to discover acceptable sub-optimal results. These algorithms are based on hill-climbing and genetic algorithms.

Hill-Climbing Algorithms try to find some better solution, given the current solution and some transition function. They can get caught on one of "the smaller hills" or local maximum. *Steepest Ascent Hill-Climbing* computes *all* the states it can transition to, and takes the best one. *Nearest Ascent Hill-Climbing* does not compute the set of all possible states to transition to. It computes states until a better one is found. It is much faster in practice than Steepest Ascent.

Genetic Algorithms tend to converge to a solution quickly when the solution space is small relative to the search space. Genetic Algorithms do not get stuck at local maxima. They work by representing each state as a string, and then doing 'genetic manipulation' on a 'generation' in ways analogous to nature (1- Crossing Over. 2-Mutations. 3- Selection).

3.2 Hierarchical-based methods

Hierarchical-based clustering methods may be based on 2 different philosophies, agglomerative and divisive, shown in Figure 9:

Agglomerative: Initially many small clusters are formed with high coherence. Then these small clusters are recursively merged based on their similarity. In the end there is one cluster resulting from merging all smaller clusters that contains all objects in the input data set.

Divisive: Initially the entire space of input data objects forms one large cluster that is then decomposed at lower levels into smaller clusters of increasing coherence. At the lowest level each data object forms just one cluster on its own.

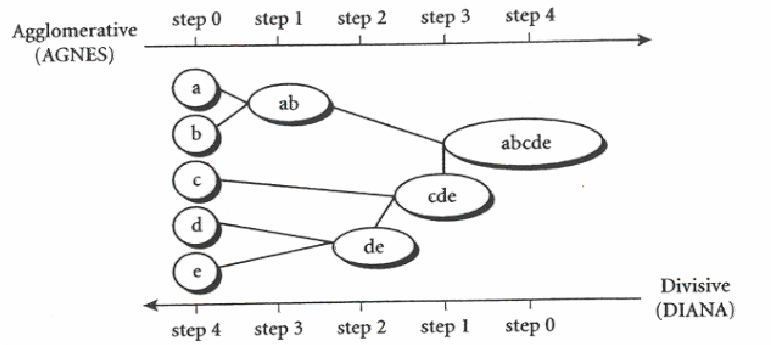


Figure 9 – Agglomerative and Divisive hierarchical clustering on data objects a,b,c,d,e. Adopted from [Han01].

The data objects are decomposed into several levels of nested partitioning (tree of clusters), called a dendrogram. A clustering of the data objects is obtained by cutting the dendrogram at the desired level. Then, each connected component forms a cluster.

a) Numerical

There exist several hierarchical clustering methods that are designed for *numerical* data sets primarily.

Birch

BIRCH is a hierarchical clustering algorithm especially suitable for very large databases. BIRCH's operation is based on clustering features (CF) and clustering feature trees. These are used to summarize the information in a cluster. A CF is a triplet summarizing information about subclusters of objects. A CF typically holds the following information about a subcluster: the number of objects N in the subcluster, a vector holding the linear sum of the N objects in the subcluster, and a vector holding the square sum of the N objects in the subcluster. Thus, a CF is a summary of statistics for the given subcluster [Zhang96].

Figure 10 shows an illustration of a CF tree used in BIRCH clustering. A nonleaf node in this tree contains summaries of the CFs of its children. A CF tree can be considered a multilevel summary of the data that preserves the inherent structure of the data.

The BIRCH algorithm consists of the following 2 phases:

1) Scan the database to build an initial in-memory CF tree. The initial CF tree is built incrementally as new objects are inserted. An object is inserted to the closest leaf (representing a subcluster). If the diameter of the subcluster for the leaf node becomes larger than a threshold value, then the lead node is split. After inserting a new object, information about it affects the rest of the tree up to the root.
2) Apply a selected clustering algorithm to cluster the leaf nodes of the CF tree. For example, a typical partitioning algorithm as described in the previous section can be used to cluster the CF tree.

The main advantage of BIRCH is that it is very efficient with a computational complexity of O(n). Disadvantages include that it has difficulty identifying clusters that are not spherical in shape because it uses the notion of radius for each cluster.

In the paper [Zhang96], one of the experiments they did was studying the application of this algorithm on real datasets. They considered two similar pictures of trees with partly cloudy sky in the background taken in two different wavelengths (NIR – near-infrared band – and VIS – visible wavelength band). Each of the image contained 512x1024 pixels (or 524288 pixels). Therefore, for each of those pixels they have a pair of brightness value which corresponds to NIR and VIS. Soil scientists receive hundreds of such image pairs and try to first filter the trees from the background and then filter the trees into sunlit leaves, shadows and branches for statistical analysis. The filtering process was then accomplished using this BIRCH algorithm on brightness values. It was successful in dealing with this very large database and giving the expected results.

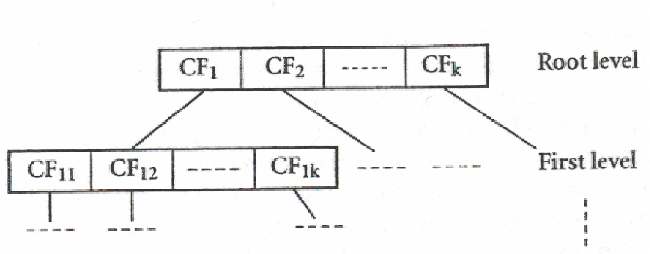


Figure 10 – A CF tree structure. Adopted from [Zhang96, Han01].

CURE

CURE goes a step beyond BIRCH by not favoring clusters with spherical shape – thus being able to discover clusters with arbitrary shape. CURE is also more robust with respect to outliers [Guha98].

CURE uses several representative objects in space to represent a cluster, instead of using a single centroid or object – which are selected to be well-scattered objects for the cluster. The representative objects for each cluster are 'shrinked' or moved toward the cluster center by a user-specified shrinking factor. At each step of the algorithm, the two clusters with the closest pair of representative objects are merged, where each object in the pair is from a different cluster. Figure 11 illustrates CURE clustering, with more than one representative objects per cluster.

CURE performs the following steps:

1) Draw a random sample S of the original objects.
2) Partition the sample S into a set of partitions. Each partition is a partial cluster.
3) Eliminate outliers by random sampling. If a cluster grows too slowly, remove it.
4) Cluster the partial clusters.
5) The representative points falling in each new cluster are "shrinked" or moved toward the cluster center by a user-specified shrinking factor.
6) These objects then represent the shape of the newly formed cluster.

Given a data set with n objects, the complexity of cure is O(n), because CURE requires one scan of the database.

The main advantage of CURE is that it produces good quality clusters even in the presence of outliers – the shrinking of the clusters helps reduce the effect of outliers. Furthermore, it adjusts well to the geometry of non-spherical clusters and allows clusters of complex shapes and different sizes to be discovered – having more than one representative object per cluster permits this to happen. Furthermore, CURE scales well to large data sets without sacrificing the quality of the clustering results. A disadvantage is that the quality of the clustering results is sensitive to the values that the user specifies for the parameters, such as the number of clusters and the shrinking factor.

Eisen's hierarchical clustering algorithm

The hierarchical clustering algorithm by Eisen et al. [Eisen98, Eisen99] is the technique most commonly used for cancer clustering, as well as clustering of genomic data sets in general. In the past it has been applied primarily to gene expression data sets.

Eisen et al. used the pairwise average linking method, for clustering genes according to similarity in pattern of gene expression [Eisen98]. For a set of n genes, an upper-diagonal similarity matrix is computed, containing similarity scores for all pairs of genes. Similarity scores are computed by using a specific metric that we do not describe here. Initially the similarity matrix is scanned to find the highest value, representing the pair of genes with the most similar interaction patterns. The two most similar genes are grouped in a cluster and then the similarity matrix is recomputed, using the average properties of both or all genes in the cluster. More genes are progressively added to the initial pairs to form clusters of genes [Eisen98]. The process is repeated until all genes have been grouped into clusters.

The result is a form of hierarchical clustering, with relationships among genes represented by a tree. Figures 2a, 2b, 3 and 12 display the results of gene expression clustering by appending a tree to the data table to reflect the relationships among genes. The tree orders the genes in the table, such that genes with similar expression patterns are adjacent. Furthermore, the branch lengths reflect the degree of similarity between genes, as assessed by a similarity metric [Eisen98]. The original software implementation of Eisen's algorithm is available at **http://rana.stanford.edu/software/** .

The original software implementation was designed to cluster gene expression data. However, this implementation has disadvantages for the study of gene expression; hierarchical clustering was originally designed for modeling hierarchical descent, such as the evolution of species, as opposed to modeling the various similarities which may exist in gene expression patterns. Eisen's algorithm can cluster any table containing numeric data [Eisen98, Eisen99].

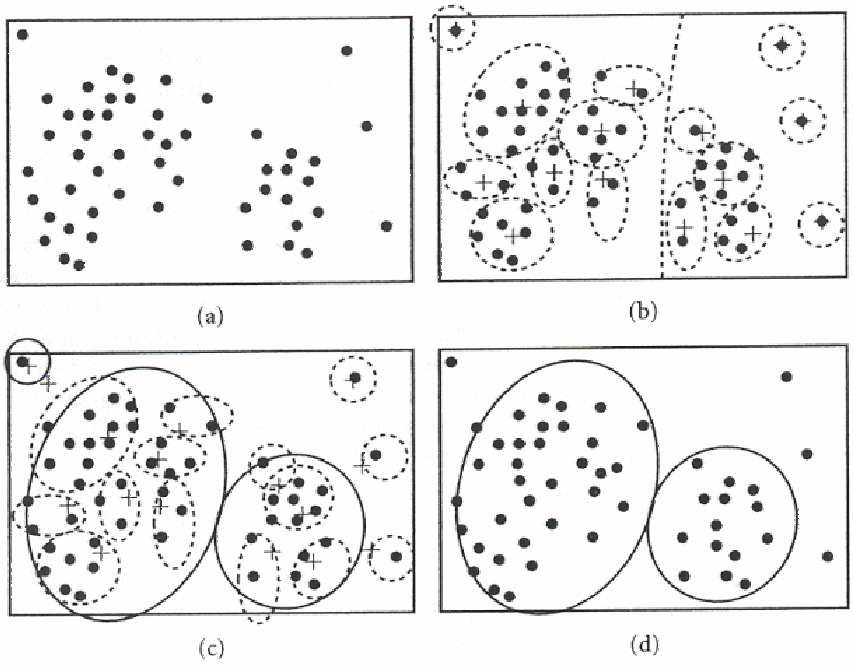


Figure 11 – Clustering a set of objects using CURE. (a) A random sample of objects. (b) Partial clusters. Representative points for each cluster are marked with a "+". (c) The partial clusters are further clustered. The representative points are moved toward the cluster center. (d) The final clusters are nonspherical. Adopted from [Guha98, Han01].

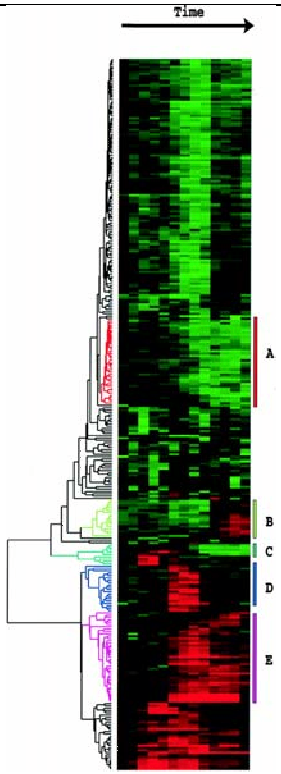


Figure 12 - Clustered display of data from time course of serum stimulation of primary human fibroblasts. Adopted from [Eisen98, Eisen99].

b) Categorical

There exist several hierarchical clustering methods that are designed for *categorical* data sets primarily.

ROCK

ROCK is a hierarchical clustering algorithm that uses a similar idea to CURE, except that it deals with categorical data sets and not numerical data sets. In ROCK clusters are merged hierarchically, by measuring the similarity between two clusters as the number of points from different clusters that have neighbors in common [Guha00].

Initially, each tuple is assigned to a separate cluster and then clusters are merged repeatedly according to the closeness between clusters. The similarity between two clusters is defined by the number of cross links between two clusters. The closeness between clusters is defined as the sum of the number of "links" between all pairs of tuples, where the number of "links" represents the number of common neighbors between two clusters. Link(pi,pj) is the number of common neighbors between two points pi and pj.

Chameleon

Chameleon applies to all types of data, as long as a similarity function is specified. Chameleon was developed to mend the weaknesses of the CURE and ROCK hierarchical clustering algorithms. Specifically, CURE ignores information about the interconnectivity of objects in two different clusters [Karypis99]. ROCK, on the other hand, ignores information about the closeness of two different clusters while emphasizing their objects' interconnectivity.

The basic philosophy behind Chameleon is to merge two clusters if the interconnectivity and closeness between the two clusters is highly related to the internal interconnectivity and closeness of the objects within the clusters. Chameleon uses similarity formulas for the relative interconnectivity and the relative closeness between two clusters, as well as the internal interconnectivity and closeness of the clusters themselves. It merges the two clusters according to the similarity formula results.

Chameleon represents the data objects using the *k-nearest neighbor* graph approach, an example of which is shown in Figure 13. Chameleon first uses a graph partitioning algorithm to partition a graph into a large number of small subclusters with high interconnectivity. It then uses a hierarchical clustering algorithm to repetitively combine or merge the subclusters. To merge clusters, it takes into account the internal characteristics of the clusters to be merged, as well as their interconnectivity and closeness.

The main disadvantage of Chameleon is that it has complexity $O(n^2)$ and for very large data sets it may be especially expensive to run it. The main advantage of Chameleon is that it takes into account the internal characteristics of the clusters to be merged.

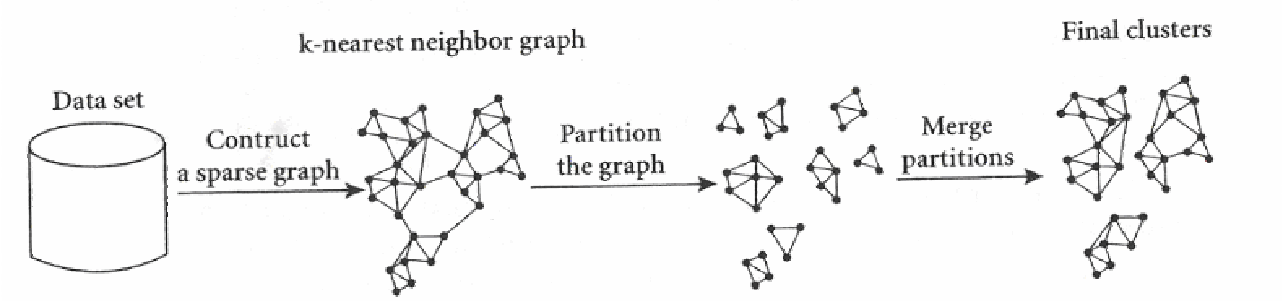


Figure 13 – Chameleon clustering based on k-nearest neighbors and dynamic modeling. Adopted from [Karypis99].

LIMBO

LIMBO is introduced in [Andritsos04] is a scalable hierarchical categorical clustering algorithm that builds on the *Information Bottleneck (IB)* framework for quantifying the relevant information preserved when clustering. LIMBO uses the IB framework to define a distance measure for categorical tuples. LIMBO handles large data sets by producing a memory bounded summary model for the data.

3.3 Density-based methods

Density-based clustering methods view the values of attributes in the entire input data set as a large space with subspaces of higher or lower density. Then, the clustering is based on the idea that the density in each cluster should be maximized while the density between clusters should be minimized. A cluster continues growing as long as the density in the neighborhood is above a threshold. Density-based clustering methods are particularly useful for discovering clusters with arbitrary shape.

a) Numerical

Several density-based clustering methods have been proposed for numerical data sets.

DBSCAN

DBSCAN regards clusters as dense regions of objects in the data space that are separated by regions of low density. A cluster is defined by this algorithm as a maximal set of *density-connected* objects. DBSCAN grows regions with sufficiently high density into clusters. Every object not contained in any cluster is considered to be noise [Ester96, Ester98].

The DBSCAN algorithm finds clusters as follows:

a. Check the e-neighborhood of each object in the database.
b. If the e-neighborhood of an object *o* contains more than MinPts, a new cluster with *o* as a core object is created.
c. Iteratively collect directly density-reachable objects from these core objects, which may involve the merge of a few density-reachable clusters.
d. Terminate the process when no new object can be added to any cluster.

The computational complexity of DBSCAN is O(nlogn) if a spatial index is used. Otherwise, it is $O(n^2)$, where n is the number of database objects.

The main advantage of DBSCAN is that it is capable of discovering clusters of arbitrary shape. The main disadvantage of DBSCAN is that it leaves the user with the responsibility of selecting parameter values for e and MinPts that will lead to high quality clusters. Furthermore, the quality of the resulting clusters is sensitive to the user-defined parameters.

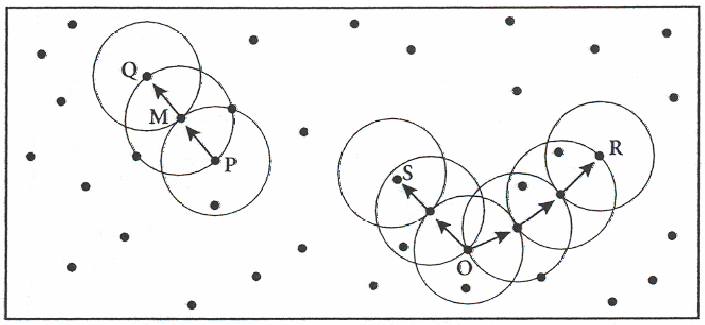


Figure 14 – Density reachability and density connectivity in density-based clustering. Adopted from [Ester96, Ester98].

OPTICS

OPTICS clustering was developed to overcome the difficulty of selecting appropriate parameter values for DBSCAN [Ankerst99]. The OPTICS algorithm finds clusters using the following steps:

a. Create an ordering of the objects in a database, storing the core-distance and a suitable reachability-distance for each object (see Figure 15a). OPTICS processes these distance parameter values at the same time. This ordering allows for clusters with higher density to be finished first.
b. Based on the ordering information produced by OPTICS, use another algorithm to extract clusters.
c. Extract density-based clusters with respect to any distance e' that is smaller than the distance e used in generating the order.

OPTICS has the same run-time complexity as that of DBSCAN, that is O(nlogn) if a spatial index is used.
The main advantage is that the user is not required to select the values of input parameters.

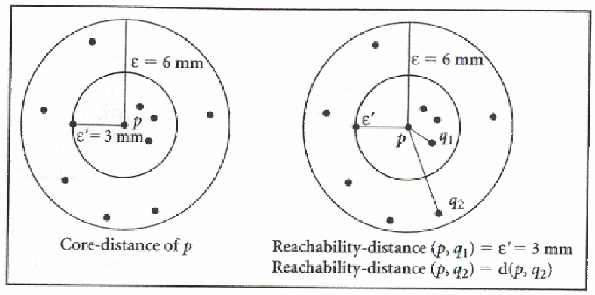


Figure 15a – OPTICS. The core distance of p is the distance e', between p and the fourth closest object. The reachability distance of q1 with respect to p is the core-distance of p (e'=3mm) since this is greater than the distance between p and q1. The reachability distance of q2 with respect to p is the distance between p and q2 since this is greater than the core-distance of p (e'=3mm). Adopted from [Ankerst99].

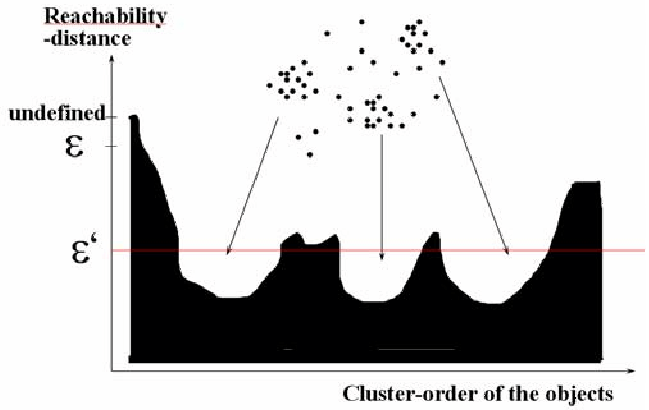


Figure 15b – Cluster ordering in OPTICS. Adopted from [Ankerst99].

DENCLUE

DENCLUE is a clustering method based on a set of density distribution functions. In this method the influence of each data object is formally modeled using a mathematical influence function, describing the impact of a data object within its neighborhood. The overall density of the data space is modeled as the sum of the influence functions of all data points. Clusters are then determined mathematically by identifying density attractors, which are local maxima of the overall density function [Hinneburg98].

The influence function may be a Euclidean distance function or a Gaussian influence function, such as:

$$f_{Gauss}(x,y) = e^{-\frac{d(x,y)^2}{2\sigma^2}}.$$

The density function at an object x(-F^d is defined as the sum of influence functions of all data objects on object x. Given n data objects, D={x_1,…,x_n} the density function at x that results from the Gaussian influence function is defined as:

$$f_{Gauss}(x) = \sum_{i=1}^{n} e^{-\frac{d(x,x_i)^2}{2\sigma^2}}.$$

From this density function, one can derive the *gradient* of the function and the *density attractor*, the local maxima of the density function. A hill-climbing algorithm guided by the gradient can be used to determine the density attractor of a set of data objects.

The main advantages of DENCLUE are that it allows description of arbitrarily shaped clusters.

Mutual Information clustering

Mutual information refers to a measure of correlation between the information content of two information elements [Michaels98]. For example, when clustering gene expression data mutual information measures the similarity in content, or shared information, between two gene expression patterns. The formula $M(A,B) = H(A) + H(B) - H(A,B)$ represents the mutual information M, that is shared by two temporal gene expression patterns A and B.

H refers to the *Shannon entropy*. $H(A)$ is a measure of the number of expression levels exhibited by the gene A expression pattern, over a given time period. The higher the value of H, the more expression levels gene A exhibits over a temporal expression pattern, and, thus, the more information the expression pattern contains. An H of zero means that a gene expression pattern over a time course was completely flat, or constant, and thus the expression pattern carries no information. $H(A,B)$ is a measure of the number of expression levels observed in *either* gene expression pattern A *or* B. Thus, the mutual information $M(A,B) = H(A) + H(B) - H(A,B)$ is defined as the number of expression levels that are observed in both gene expression patterns A and B, over a given time period [Michaels98].

It is important to note that mutual information cluster analysis is intended to detect different types of similarities between gene expression patterns than other types of cluster analysis, such as hierarchical. For example, two genes that respond to the same input very differently may be considered related by mutual information cluster analysis but not by hierarchical cluster analysis.

b) Categorical

Several density-based clustering methods have been proposed for data sets with categorical attributes.

CACTUS

CACTUS is presented in [Ganti99], introducing a novel formalization of a cluster for categorical attributes by generalizing a definition of a cluster for numerical data. CACTUS is a summary-based algorithm that utilizes summary information of the dataset. It characterizes the detected categorical clusters by building summaries. The algorithm relies on inter- and intra-attribute summaries that are assumed to fit into main memory for most categorical datasets. CACTUS consists of three phases: *summarization*, *clustering* and *validation*.

The authors assume the existence of a distinguishing number that represents the minimum size of the distinguishing sets – i.e. attribute value sets that uniquely occur within only one cluster. The distinguishing sets in CACTUS rely on the assumption that clusters are uniquely identified by a core set of attribute values that occur in no other cluster. While this assumption may hold true for many real world datasets, it is unnatural and unnecessary for the clustering process. The distinguishing sets are then extended to cluster projections.

CACTUS first computes cluster projections onto the individual attributes. The cluster projections on single attributes that CACTUS generates are combined in the extension phase to generate cluster candidates of higher dimensionality over multiple attributes. The cluster candidates of higher dimensionality are then validated against the original dataset. The proposed approach to this end selects as initial one dimensional candidates C_1 all cluster projections c_1 on the first attribute. Candidates in subsequent C_{k+1} are generated by combining each $(c_1; : : : ; c_k)$ in C_k with all cluster projections c_{k+1} on attribute A_{k+1}. If for all $1 \leq i \leq k$, $(c_i; c_{k+1})$ is a cluster projection on $(A_i;A_{k+1})$, $(c_1; : : : ; c_{k+1})$ is added to the candidate set C_{k+1}. Clearly, the candidates have to be validated by scanning the original dataset and counting the support of each candidate.

COOLCAT

COOLCAT is an entropy-based algorithm for categorical clustering [Barbara02]. The COOLCAT algorithm is based on the idea of entropy reduction within the generated clusters.

It first bootstraps itself using a sample of maximally dissimilar points from the dataset to create initial clusters. COOLCAT starts with a sample of data objects and identifies a set of k initial tuples such that the minimum pairwise distance among them is maximized. The remaining points are then added incrementally. Clusters are created by "cooling" them down, i.e. reducing their entropy. All remaining tuples of the data set are placed in one of the clusters such that, at each step, the increase in the entropy of the resulting clustering is minimized.

Naturally, this approach is highly dependent on the order of selection. To mitigate this dependency, the authors propose to remove the “worst fitting" points at defined times during the execution and re-clustering them.

STIRR

STIRR is introduced in [Gibson98] as an iterative algorithm. STIRR applies a linear dynamical system over multiple copies of a hypergraph of weighted attribute values, until a fixed point is reached. Each copy of the hypergraph contains two groups of attribute

values, one with positive and another with negative weights, which define the two clusters. The approach used in STIRR can be mapped to certain types of non-linear systems.

STIRR produces a dynamical system from a table of categorical data. STIRR begins with a table of relational data, consisting of k fields (or columns), each of which can assume one of many possible values. Figure 16 shows an example of the data representation. Each possible value in each field is represented by a node and the data is represented as a set of tuples – each tuple consists of nodes, with one node for each field. A configuration is an assignment of a weight w to each node. A normalization function N() is needed to rescale the weights of the nodes associated with each field so that their squares add up to 1.

	Attribute		
Tuple	a	b	c
1.	A	W	1
2.	A	X	1
3.	B	W	2
4.	B	X	2
5.	C	Y	3
6.	C	Z	3

is represented as

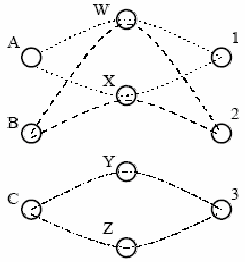


Figure 16 - Representation of a collection of tuples. Adopted from [Gibson98].

A dynamical system is the repeated application of a function f on some set of values. A fixed point of a dynamical system is a point u for which $f(u) = u$. So it is a point which remains the same under the repeated application of f.

The dynamical system for a set of tuples is based on a function f, which maps a current configuration of weights to a new configuration. A combiner function f defined below is used and each weight w_v is updated as follows:

To update the weight w_v:
For each tuple $\tau = \{v, u_1, \ldots, u_{k-1}\}$
containing v do
$x_\tau \leftarrow \oplus(u_1, \ldots, u_{k-1})$.
$w_v \leftarrow \sum_\tau x_\tau$.

Thus, w_v is updated by applying the function separately to the members of all tuples that contain v and adding the results. The function f is then computed by updating the weight of each w_v as above, and then normalizing the set of weights using N(). This yields a new configuration f(w). Iterating f defines a dynamical system on the set of weights configurations. The system runs through a specified number of iterations, returning the final set of configurations obtained. The weight sets, generally, converge to fixed points or to cycles through a finite set of values.

CLICK

CLICK finds clusters in categorical datasets based on a search method for k-partite maximal cliques [Peters04].

CLICK(Dataset $\mathcal{D}$, α, $minsup$)
AttributeValueRanking: $\mathcal{R} = \bigcup_{i=1}^{n} D_i$
Clique $C = \emptyset$
CliqueCollection $\mathcal{C} = \emptyset$

PreProcess($\mathcal{D}, \alpha, \Gamma(\mathcal{D}), \mathcal{R}$)
DetectMaxCliques($\Gamma(\mathcal{D}), \mathcal{C}, \mathcal{R}, C$)
PostProcess($\mathcal{D}$, $\mathcal{C}$, α, $minsup$)
return $\mathcal{C}$

Figure 17 – The CLICK algorithm. Adopted from [Peters04].

The basic Click approach consists of the three principal stages, shown in Figure 17, as follows:

- Pre-Processing: In this step, the k-partite graph is created from the input database D, and the attributes are ranked for efficiency reasons.
- Clique Detection: Given Γ(D), all the maximal k-partite cliques in the graph are enumerated.
- Post-Processing: the support of the candidate cliques within the original dataset is verified to form the final clusters. Moreover, the final clusters are optionally merged to partially relax the strict cluster conditions.

CLOPE

CLOPE is a clustering algorithm introduced in [Yang2002] for categorical and transactional data. This algorithm starts from a heuristic method of increasing the height-to-width ratio of the cluster histogram.

CLOPE proposes a novel global criterion function that tries to increase the intra-cluster overlapping of transaction items by increasing the height-to-width ratio of the cluster *histogram*. A parameter is used to control the tightness of the cluster. Different number of clusters can be obtained by varying this parameter. Let us take a small market basket database with 5 transactions *{(apple, banana), (apple, banana, cake), (apple, cake, dish), (dish, egg), (dish, egg, fish)}*. For simplicity, transaction *(apple, banana)* is abbreviated to *ab,* etc. For this small database, we want to compare the following two clustering (1) *{ {ab, abc, acd}, {de, def} }* and (2) *{ {ab, abc}, {acd, de, def} }*. For each cluster, we count the occurrence of every distinct item, and then obtain the height (H) and width (W) of the cluster. For example, cluster *{ab, abc, acd}* has the occurrences of a:3, b:2, c:2, and d:l, with H=2.0 and *W=4*. H is computed by summing the numbers of occurrences and dividing by the number of distinct items. Figure 18 shows the values of these results as histograms.

H=2.0, W=4 H=1.67, W=3 H=1.67, W=3 H=1.6, W=5
{ab, abc, acd} *{de, def}* *{ab, abc}* *{acd, de, def}*
clustering (1) clustering (2)

Figure 18 - Histograms of the two clusterings. Adopted from [Yang2002].

The qualities of these two clusterings are compared by analyzing the heights and widths of the clusters. Leaving out the two identical histograms for cluster *{de, def}* and cluster *{ab, abc},* the other two histograms are of different quality. The histogram for cluster *{ab, abc, acd}* has *H/W=0.5,* but the one for cluster *{acd, de, def}* has *H/W=0.32*. Clearly, clustering (1) is better since we prefer more overlapping among transactions in the same cluster. From this example, we can see that a larger height-to-width ratio of the histogram means better intra-cluster similarity. This intuition is the basis of CLOPE and defines the global criterion function using the geometric properties of the cluster histograms.

The advantages of CLOPE include that it has fast performance. CLOPE is also scalable when clustering large transactional databases with high dimensions. Disadvantages include that the accuracy of the results may suffer.

3.4 Grid-based methods

Grid-based clustering methods start by forming a grid structure of cells from the objects of the input data set. Each object is classified in a cell of the grid. The clustering is performed on the resulting grid structure.

WaveCluster

WaveCluster uses wavelet transformations to cluster data. It uses a wavelet transformation to transform the original feature space, finding dense regions in the transformed space [Sheikholeslami98]. A wavelet transform is a signal processing technique that decomposes a signal into different frequency subbands. The wavelet model can be applied to a data set with *n* dimensions by applying a one-dimensional wavelet *n* times. Each time that a wavelet transform is applied, data are transformed so as to preserve the relative distance between objects at different levels of resolution, although the absolute distance increases. This allows the natural clusters in the data to become more distinguishable. Clusters are then identified by searching for dense regions in the new domain.

Wavelets emphasize regions where the objects cluster, but suppress less dense regions outside of clusters. The clusters in the data stand out because they are more dense and "clear" the regions around them.

Figures 19 and 20 illustrate an example of a sample of two-dimensional feature space. Wavelet transformations can detect clusters at varying levels of accuracy. For example, figures 19a,b,c illustrate the resulting wavelet transformation at different resolutions, from a fine scale to a coarse scale.

WaveCluster is very fast with a computational complexity of O(n), where *n* is the number of objects in the data set.

Clustering with wavelet transforms does not require any prior classifications of objects to be taken as input. Furthermore, it can discover clusters with arbitrary shape. It does not need the number of clusters to be given as input by the user. It is insensitive to the order of the input and can effectively deal with outliers. It handles large data sets efficiently and can handle data sets with up to 20 dimensions. WaveCluster can result in the removal of outliers because the regions where outliers exist are much less dense than regions with clusters. Finally, WaveCluster produces clusters of high quality. The main drawback is that it is only applicable to low dimensional data.

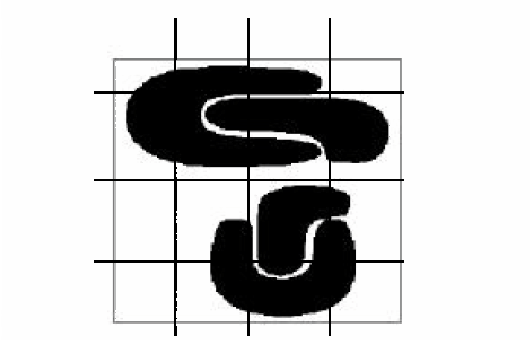


Figure 1: A sample 2-dimensional feature space.

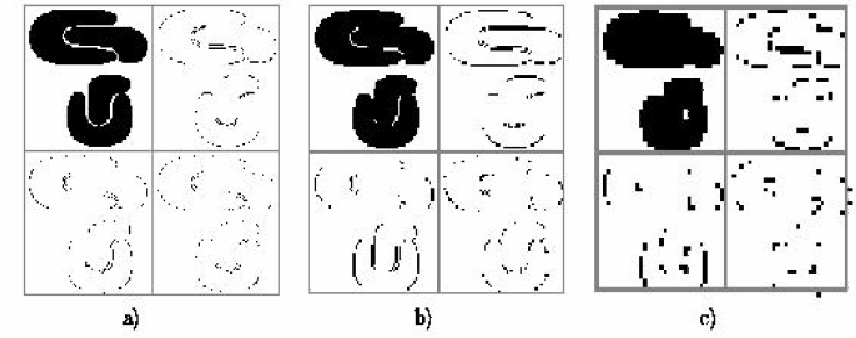


Figure 19 – WaveCluster. Adopted from [Sheikholeslami98, Han01].

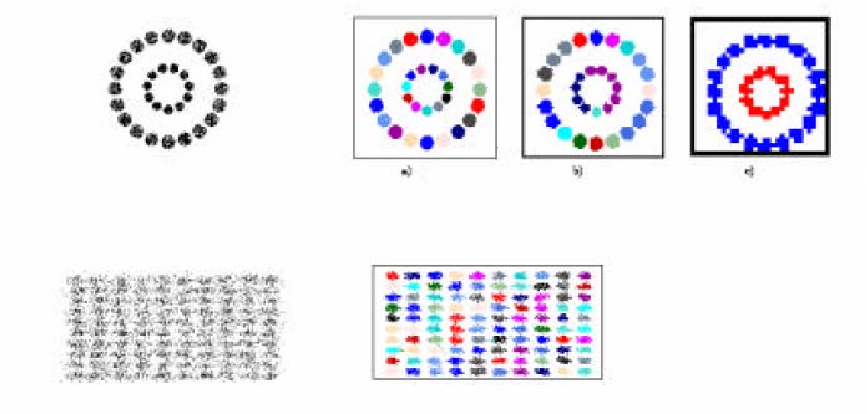


Figure 20 - WaveCluster. Adopted from [Sheikholeslami98, Han01].

STING

Sting is a combination of a grid-based and a hierarchical clustering method. Sting partitions the input data set into rectangular cells. There exist several levels of such rectangular cells corresponding to different levels of resolution and these cells form a hierarchical structure: each cell at a high level is partitioned to form a number of cells at the next lower level. Statistical information regarding the attributes in each grid cell - such as the mean, maximum and minimum values - is stored. Statistical parameters of higher level cells are computed from the parameters of lower-level cells [Wang97].

The parameters stored in each cell of each level of the grid include count, mean, standard deviation, minimum and maximum and the type of distribution that the attribute value in the cell follows, such as normal, uniform etc. The parameters of higher level cells are computed based on the parameters of lower level cells. The value of the distribution type for a higher-level cell in the grid can be computed based on the majority of the distribution types of its corresponding lower-level cells. The type of distribution for the bottom level, if known, may be assigned by the user.

Figures 21 and 22 show a hierarchical structure for STING clustering.

The main advantage of STING is its fast runtime, as its computational complexity is O(n) where n is the number of objects in the data set.

A major disadvantage is that all of the cluster boundaries are either horizontal or vertical and no diagonal boundary is detected. This lowers the quality and accuracy of the clusters.

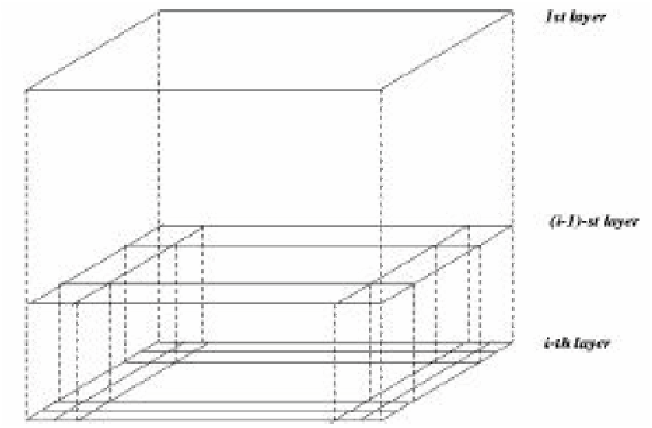


Figure 21 – Hierarchical structure for STING clustering. Adopted from [Wang97].

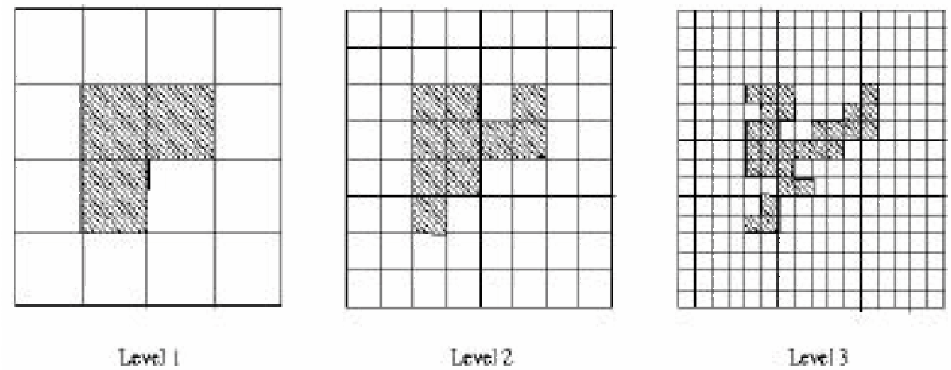


Figure 22 – Hierarchical structure for STING clustering. Adopted from [Wang97, Han01].

CLIQUE

The CLIQUE clustering algorithm combines grid-based and density-based clustering. CLIQUE attempts to find the sparse and the "crowded" units in space, where a unit is a cell of a rectangular grid splitting up the data objects. Figure 23 illustrates what a grid used in the CLIQUE method looks like. A cell of the grid is dense if the fraction of total data objects contained in it exceeds a user-specified value for an input parameter [Agrawal98].

A cluster in CLIQUE is defined as a maximal set of *connected dense units*.

CLIQUE performs the following steps:

1) Partition the n-dimensional data space into nonoverlapping rectangular units. Identify the dense units among these.
2) Intersect these dense units to form a candidate search space in which dense units of higher dimensionality may exist. The dense units are, then, examined to determine the clusters.
3) Generate a minimal description for each cluster, by determining the maximal region that covers the cluster of connected dense units.

CLIQUE has a runtime computational complexity of O(n). It scales linearly with the size of the input data set.

The main advantage of CLIQUE is that it is useful for clustering high-dimensional data in large databases. Furthermore, it is insensitive to the order of the objects in the input data set. The main disadvantage is that the accuracy of the CLIQUE clustering results is often not as good as that of other methods.

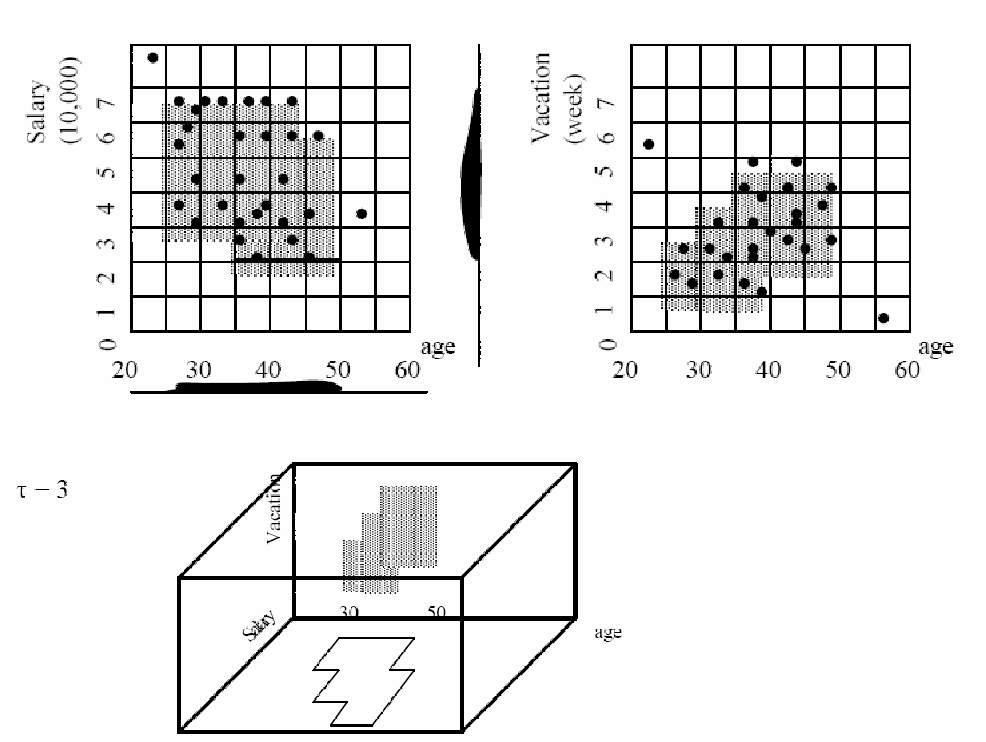


Figure 23 – CLIQUE. Adopted from [Agrawal98, Han01].

3.5 Model-based methods

Model-based clustering methods typically assume that the objects in the input data set match a model – which is often a statistical distribution. Then, the process tries to classify the objects such that they match the distribution the best. The model – or statistical distribution – may be given by the user beforehand as an input parameter and the model may change during the clustering process.

a) Categorical

Several clustering methods have been developed for data sets whose objects have categorical attributes.

COBWEB

COBWEB is a *conceptual* clustering method used for categorical data sets primarily. COBWEB creates a hierarchical clustering in the form of a classification tree. Figure 24 shows what a COBWEB classification tree looks like. Each node refers to a concept. The probabilistic description includes the probability of the concept and the probabilities of the attribute-value pairs. This is unlike decision trees, which label branches rather than nodes and use logical rather than probabilistic descriptions. The sibling nodes at a given level form a partition [Fisher87].

A classification tree differs from a decision tree. Each node in a classification tree refers to a concept and contains a probability for that concept which applies to the objects classified under that node. The sibling nodes at a level of the classification tree form a *partition* of the data set.

COBWEB uses a heuristic evaluation measure known as *category utility* (CU) to guide construction of the classification tree. CU is defined as:

$$\frac{\Sigma_{k=1}^{n} P(C_k)[\Sigma_i \Sigma_j P(A_i = V_{ij}|C_k)^2 - \Sigma_i \Sigma_j P(A_i = V_{ij})^2]}{n}$$

Given a partition, CU is the increase in the expected number of attribute-value pairs that can be guessed correctly. Ai=Vij is an attribute-value pair, Ck is the concept class and n is the number of nodes or concepts forming a partition. Category utility rewards intraclass similarity and interclass dissimilarity.

COBWEB incrementally incorporates objects into a classification tree by following the steps below for each new object:

a) Descend the classification tree along an appropriate path, temporarily placing the new object in each node and computing the category utility of the resulting partition. The placement that results in the highest category utility is the best node for the object.
b) Also compute the category utility of the partition that would result if the object was placed in a newly created node. The object may be placed in a newly created or an existing node, based on which partition results in the highest category utility value.

An important difference between COBWEB and earlier conceptual clustering systems is that it is incremental - COBWEB integrates an observation into an existing classification tree by classifying the observation along a path of `best' matching nodes. An advantage of COBWEB is that it is able to adjust the number of classes in a partition, without the user having to specify this input parameter beforehand. The main disadvantage of COBWEB is that it is based on the assumption that separate attributes are statistically independent of one another. However, in reality it is often the case that separate attributes are correlated.

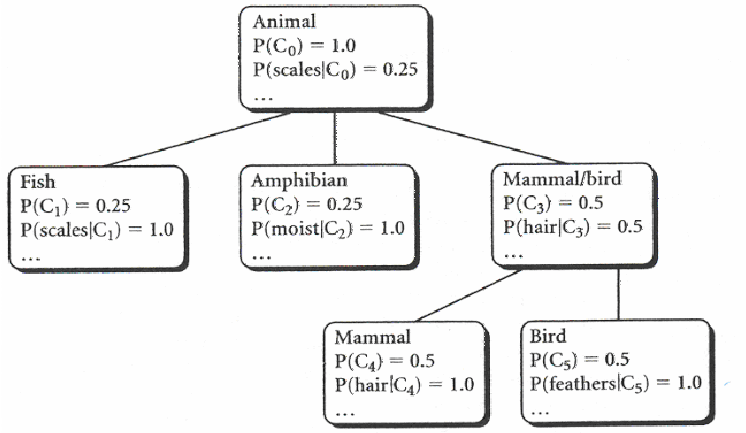


Figure 24 – A classification tree. Adopted from [Fisher87, Han01].

b) Mixed Categorical and Numerical

Several clustering methods have been developed that are able to deal with data sets having mixed categorical and numerical attributes.

AutoClass

AutoClass is a clustering algorithm that can work on categorical as well as numerical data sets [Stutz95]. AutoClass attempts to cluster a numerical or categorical data set by finding the most probable classifications of items in clusters, given some prior model about the structure of the clusters. Each data item Xi belongs to one and only one member of a set of J classes Cj. The goal is to maximize the probability $\pi(Xi \in Cj \mid model\ about\ Cj)$. **H** is a Hypothesis about the classification of items. **E** is the Evidence that is observed and that is relevant to the classification.

E is the attributes of a data item, that are given to us by the data set. For example, if each data item is a coin, the evidence **E** might be represented as follows for a coin **i**:
Ei = {"land tail"} meaning that in one trial the coin **i** landed to be tail.
If there were many attributes, then the evidence **E** might be represented as follows for a coin **i**:
Ei = {"land tail","land tail","land head"} meaning that in 3 separate trials the coin **i** landed as tail, tail and head.

H is a hypothesis about the classification of a data item. For example, if each data item is a coin, then **H** might state that a coin **i** belongs in the class "two headed coin". The problem with classification - that we will discuss later – is that we usually do not know the **H** for a data set. Thus, AutoClass needs to test many hypotheses.

$$\pi(H \mid E) = \frac{L(E \mid H)\pi(H)}{\pi(E)} = \frac{L(E \mid H)\pi(H)}{\sum_{H} L(E \mid H)\pi(H)}$$

AutoClass uses a Bayesian method for determining the optimal classes. AutoClass takes a prior distribution of each attribute in each cluster, symbolizing the prior beliefs of the user. It changes the classifications of items in clusters and changes the mean and variance of the distributions, until the mean and variance stabilize.
As a summary of the AutoClass process, for the Bayesian equation: $\pi(H \mid E) = \frac{J(EH)}{\pi(E)} = \frac{L(E \mid H)\pi(H)}{\pi(E)}$

The authors suggest replacing it with: $\pi(H \mid E) = \frac{J(EVT \mid S)}{\pi(E)} = \frac{L(E \mid VTS)\pi(VT \mid S)}{\pi(E)}$

Then ignore the π(E) denominator and try to find the models V,T for which this produces the maximum result. In other words, find the optimal set of models, along with the optimal number of classes, along with the optimal classification of data items in classes, that maximize the chance of observing the evidence E, for all data items.

The steps *in AutoClass* are the following:
Step 1: Estimate the probabilities of all items in the clusters. Calculate means and variances for all items' attributes' distributions in the cluster.
Step 2: Estimate the new model distribution, based on the results of *step 1*.
Step 3: Repeat steps 1 and 2 until convergence, i.e. until the model changes very little.

π(H) is a "prior" describing your belief in H before seeing evidence E.
π(H|E) is a "posterior" describing your belief in the world after observing evidence E.
L(E|H) is a "likelihood" embodying your theory of how likely it would be to see evidence E in world H.

If we had a few coins that might be "**two-headed**" or "**two-tailed**" or "**ordinary (one head/one tail)**", then the class **H** of each coin might be: "ordinary" or "two-headed" or "two-tailed". We usually do not know that these 3 classes exist. We want to classify each coin into **one** of these 3 classes.

If we tossed each coin **i** once, the possible evidence might be:
Ei = "land tail", or
Ei = "land head".

Now, the big question: What is the best class to classify a coin **i**, i.e. the one that is most likely to be correct, given that we have observed some evidence **Ei = "land tail"** for coin **i**? We would need to find the H for which the Bayesian equation is maximal:

$$\pi(H \mid E) = \frac{L(E \mid H)\pi(H)}{\pi(E)}$$

Given that **Ei = "land tail"** for coin **i.**
L(E|H) is
L(E|"two-tailed") = 1
L(E|"two-headed") = 0
L(E|"ordinary") = 0.5

π(H) is
π("two-tailed") = 0.5 * 0.5 = 0.25
π("two-headed") = 0.5 * 0.5 = 0.25
π("ordinary") = 0.5*0.5 + 0.5*0.5 = 0.50

Obviously, the Bayesian equation above is maximal for H = "two-tailed" and H = "ordinary".

Actually, it was so trivial that we did not even need AutoClass to classify the coins. Simply calculate (Given that **Ei = "land tail"** for coin **i**):
π(H|E) =
π("two-headed"|E) = 0
π("two-tailed"|E) = 0.5
π("ordinary"|E) = 0.5

But AutoClass might be useful for more complex examples.

The evidence **E** is usually given to us in the data set that we want to cluster. The bigger problem is: How to find **H**? **H** is what we are looking for in the end, i.e. the classifications of all data items. In practice this theory is difficult, because we do not know **H** beforehand. We do not know **π(H)** nor **L(E|H).** If we knew it, we would know how to classify all data items.

In theory, all we might need to do to find **π(H|E)** is, choose a set of states **H**, a set of prior expectations on the states **π(H),** a likelihood function describing what evidence is expected to be observed in those states **L(E|H)**, and collect some relevant evidence on the items.

Instead, usually we build models and we do all of our analysis conditional on the assumption that the world or set of states **H** is described by one of the models in our space.

Thus, replace the equation:

$$\pi(H \mid E) = \frac{J(EH)}{\pi(E)} = \frac{L(E \mid H)\pi(H)}{\pi(E)}$$

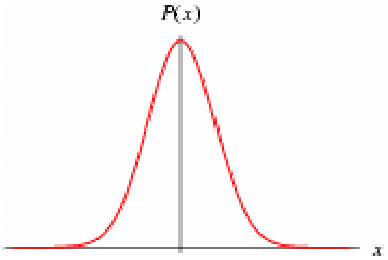
with:

$$\pi(H \mid E) = \frac{J(EVT \mid S)}{\pi(E)} = \frac{L(E \mid VTS)\pi(VT \mid S)}{\pi(E)}$$

We do not consider π(E). We try to find the models V,T that produce the maximum result for the above equation. Thus, find the V,T that maximize the joint J(EVT|S) = L(E|VTS)π(VT|S). L(E|VTS) is the likelihood approximation and p(VT|S) is the prior.

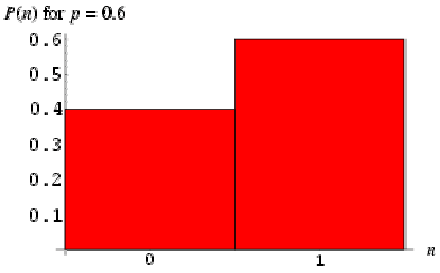
S - the world really is described by one of the models in our space **-** e.g. it is described by a single discrete attribute or many discrete attributes or many numerical attributes.
T - general form of the model. For example, is it a normal distribution?
V - some function that models the evidence that might be observed, given that an item belongs to a particular class. For example, in our coin example, the model for the class "ordinary coin" might be:
V = {L(heads), L(tails), L(unknown)} = {0.5, 0.5, 0}.

A model for a *continuous (numerical) attribute* might be a *normal* distribution like the following, representing the chances of observing any value of an attribute *i* within a specific class.



Normal distributions are a family of distributions that have the same general shape. They are symmetric with scores more concentrated in the middle than in the tails. Normal distributions are sometimes described as bell shaped. Examples of normal distributions are shown to the right. Notice that they differ in how spread out they are. The area under each curve is the same. The height of a normal distribution can be specified mathematically in terms of two parameters: the mean (μ) and the standard deviation (σ).

For a *discrete or categorical attribute* a model might be a Bernoulli distribution that looks as follows:



If we assume that a data item *i* is in class *c*, and we know that the distributions of the attributes in class *c* resemble this model, then what is the chance of observing the evidence Ei? This is really L(E|VTS) or L(E|H). Thus, we get the likelihood of the evidence E observed.

$$\pi(H \mid E) = \frac{J(EVT \mid S)}{\pi(E)} = \frac{L(E \mid VTS)\pi(VT \mid S)}{\pi(E)}$$

The coin might belong in either one of 3 classes and it has a single discrete attribute **-** we do not know which class it belongs to. **For each class**, we build a model **V={q1,q2,q3}** consisting of the likelihoods that in a toss of a coin **i,** the result will be heads or tails or unknown, ASSUMING THAT THE COIN **i** BELONGS TO THAT CLASS.

For class 1 the model V might be:
{q1, q2, q3} = {L(heads), L(tails), L(unknown)} = {0.5, 0.5, 0}.

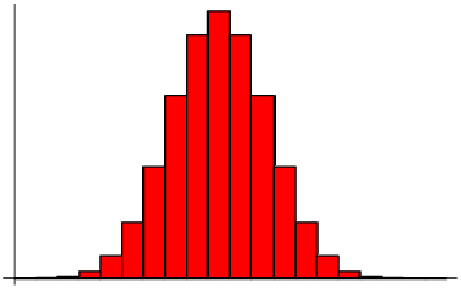
For class 2 the model V might be:
{q1, q2, q3} = {L(heads), L(tails), L(unknown)} = {0, 1, 0}....
For class 3 the model V might be:
{q1, q2, q3} = {L(heads), L(tails), L(unknown)} = {1, 0, 0}....

So if the Evidence for a coin i is, Ei = "land tail", then V suggests that the coin is likely to belong in class 2 (according to the distributions).

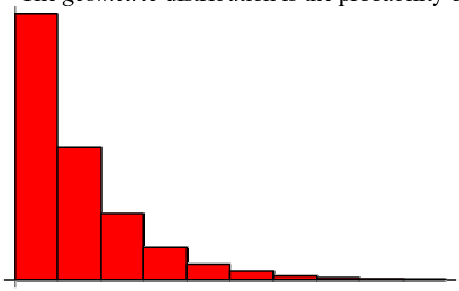
As can be seen, for all H, $$\sum_E L(E \mid H) = 1$$

This model for a coin is actually a **Bernoulli distribution**. The Bernoulli distribution is a discrete distribution having two possible outcomes labelled by $n = 0$ and $n = 1$ in which $n = 1$ ("success") occurs with probability p and $n = 0$ ("failure") occurs with probability $q=1-p$, where $0<p<1$. The distribution of heads and tails in coin tossing is an example of a Bernoulli distribution with $p=q=1/2$.

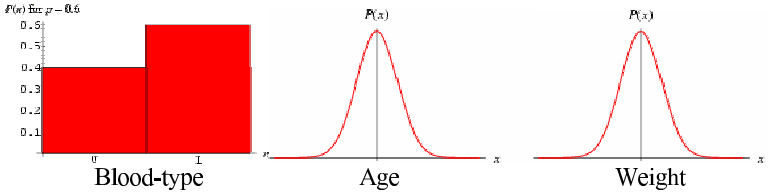
The *binomial* distribution gives the discrete probability distribution $P(n|N)$ of obtaining exactly n successes out of N Bernoulli trials . The result of each Bernoulli trial is true with probability p and false with probability $q=1-p$.



The *geometric* distribution is the probability of obtaining a *number n of failures before the first success.*



As a last example, let a data item **Xi = {age=28, blood-type=A, weight=73kg}**. This is the evidence E. Suppose blood-type is a discrete attribute that can be modeled with a Bernoulli distribution, while age and weight are continuous attributes that can be modelled with a normal (Gaussian) distribution.



Then AutoClass could suggest that **Xi** belongs better in class A or B or C, based on the distributions for the attributes for each of these classes. This is how to calculate to what extent a class' distributions fit the attributes of item **Xi :**

$$L(E_i \mid VTS) = \prod_k L(X_{ik} \mid VTS)$$

Advantages of AutoClass include that Bayesian theory is theoretically well-founded and empirically well-tested. The output is a mixture of several different answers. AutoClass can investigate different numbers of clusters and the algorithm does not expect the user to predefine the number of classes.

Disadvantages include that one has to be explicit about the space of models one is searching in. Using wrong models will produce wrong results. The AutoClass implementation can be slow. Furthermore, as with most modeling, our assumptions are almost certainly false. Then, the results won't be accurate. Thus, building models is an oversimplification and can be considered 'cheating'. At least this makes the analysis tractable.

Neural Networks and Self-Organizing Maps

Self-Organizing Maps are a type of clustering that involves neural networks, which are believed to resemble processing that occurs in the brain. Neural networks involve several layers of units that pass information from one unit to another, in an attempt to 'learn' the correct structure of clusters in a data set. SOMs assume that the units will eventually take on the clusters' structure in space. In this form of clustering several units compete for the current object. The unit that is closest to the current object becomes the winning or active unit. The weights of the winning unit are adjusted, as well as those of its nearest neighbors, so that the units will eventually take on the structure of the clusters in space [Li04].

A Self-Organizing Feature Map (SOM) is a clustering tool that is useful for visualizing high dimensional data in 2D space [Engelbrecht2002]. A SOM consists of a map of units and a set of input vectors known as the training set, as shown in Figure 25, where the circles symbolize the units. Each unit represents a weight vector that has the same dimension as the input vectors. The weight vectors can be initialized using various methods:

- Assign random values to each dimension of the weight vector. The initial random values are bounded by the range of the corresponding input parameter.
- Find the principal components of the set of input vectors and initialize the weight vectors to reflect these components.
- Initialize each weight vector to a randomly selected input vector.

SOM training is based on a competitive learning strategy [Li04]. For each input vector *v* in the training set, the Euclidean distance [Engelbrecht2002] is calculated to each unit in the SOM. Each unit competes to match *v*. The unit that is closest to *v* is known as the winning unit. The weights of the winning unit and those of its nearest neighbours are adjusted so as to reduce the Euclidean distance to the input vector. Adjusting the neighbors of the winning unit will allow for other vectors that are similar to *v* to be grouped together in the SOM. After training, similar patterns can be mapped to units that are close together in the SOM. This however, does not provide the cluster boundaries that specify the categories of the set of training vectors. To define the cluster boundaries, an additional step is required. To determine cluster boundaries, the distance between units in the SOM can be calculated and stored in a matrix, known as the unified distance matrix (U-matrix) [Engelbrecht2002]. Large values within the U-matrix show the position of cluster boundaries.

The main disadvantage of neural networks and self-organizing maps is the long processing time required, especially when dealing with large data sets.

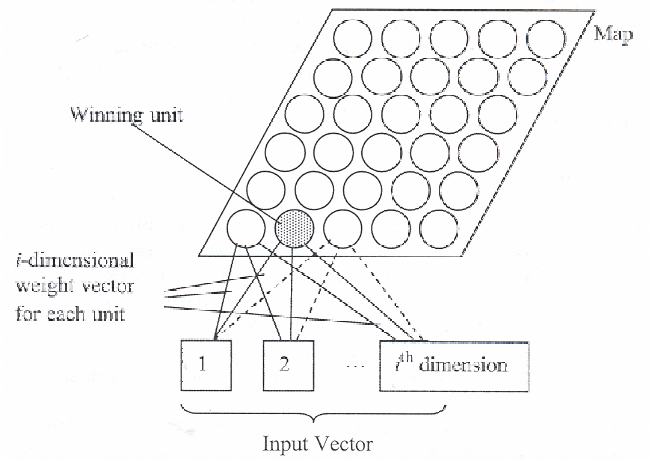


Figure 25 – SOM architecture. Adopted from [Li04].

Euclidean distance cluster analysis and Self-organizing maps applied to gene expression patterns

A sensible measure of similarity in the expression patterns of two genes is the *Euclidean distance* (angle, or dot product) [Wen98, Michaels98] which was previously presented in Sections 2.3 and 3.2. This measurement technique views a gene expression pattern over *n* time points as a point in *n*-dimensional space. Therefore, an *n*-dimensional vector represents a series of *n* measurements of a gene's expression level, over a time period. The similarity between two gene expression patterns is given by the Euclidean distance between the vectors, $\sqrt{\sum(a_i-b_i)^2}$.

Furthermore, Tamayo et al. proposed the application of *self-organizing maps (SOMs)* to the problem of clustering genes [Tamayo99]. SOMs are a type of mathematical cluster analysis, used to cluster points in multidimensional space into distinct groups. SOMs are particularly well suited for identifying underlying patterns in complex, multidimensional data, such as gene expression data. SOMs can be applied to gene expression data by regarding the gene expression pattern over *n* time points as a point in *n*-dimensional space, as described above.

In this algorithm, one chooses a geometry of nodes, which are randomly mapped into n-dimensional space. An iterative process of adjustment of the coordinates of all nodes follows, such that at each iteration nodes move towards a randomly chosen data point P. At each iteration the nodes closer to P move by a larger distance. Thus, the Euclidean distance gets decreased faster for nodes that are close than for nodes that are far. Neighboring nodes with a small Euclidean distance between them, correspond to similar gene expression patterns. Eventually, all the nodes get distributed over different clusters. Clustering of genes by using SOMs reliably clusters genes, since similar gene expression patterns will occur as neighbors in the SOM [Tamayo99].

Expectation-Maximization

EM assigns a probability distribution to each instance which indicates the probability of it belonging to each of the clusters. EM can decide how many clusters to create by cross validation, or you may specify apriori how many clusters to generate.

Assume there is a parameter that predicts the data and some of the data is missing. EM algorithm is a very general and useful iterative algorithm for parameter estimation by maximum likelihood *when some of the data are missing*. We are looking for 2 things: the missing data and the parameter.

The steps are the following:

Step 1: Replace the missing data by calculating the means. *In AutoClass, the missing data is π(H).*

Step 2: Estimate the parameter. *In AutoClass, the parameter is π(E|H).*

Step 3: Repeat steps 1 and 2 until convergence.

Suppose that Y=(y1,y2,y3,y4) follows a multinomial distribution with probability $\left(\frac{1}{2}-\frac{\theta}{2},\frac{\theta}{4},\frac{\theta}{4},\frac{1}{2}\right)$

We can only observe y1,y2,y3+y4 instead of (y1,y2,y3,y4) separately. The data are (y1,y2,y3+y4) = (38,34,125).

When there are no missing data, the parameter is

$$\theta = \frac{y2+y3}{y1+y2+y3}$$

Step 1:

$$E_{\theta[t]}(y3) = \frac{\theta_{[t]}/4}{\theta_{[t]}/4+1/2}\times 125$$

$$E_{\theta[t]}(y4) = \frac{1/2}{\theta_{[t]}/4+1/2}\times 125$$

Step 2: $$\theta_{[t+1]} = \frac{y2+E_{\theta[t]}(y3)}{y1+y2+E_{\theta[t]}(y3)}$$

Step 3: Repeat steps 1 and 2 until $\left|\theta_{[t]}-\theta_{[t+1]}\right| \le \varepsilon$

ACDC

ACDC works in a different way from algorithms we mentioned above. Most of the algorithms presented in the software clustering literature identify clusters by utilizing certain intuitive criteria such as the maximization of cohesion, the minimization of coupling or some combination of the two. ACDC cluster the software system in a way that will aid the process of understanding software [Tzerpos00]. ACDC performs the task of clustering in two stages. In the first one it creates a skeleton of the final decomposition by identifying subsystems using a pattern-driven approach. There are many patterns that have been used in ACDC. Figure 26 shows some of these patterns.

Depending on the pattern used the subsystems are given appropriate names. In the second stage ACDC completes the decomposition by using an extended version of a technique known as Orphan Adoption. Orphan Adoption is an incremental clustering technique based on the assumption that the existing structure is well established. It attempts to place each newly introduced resource (called an orphan) in the subsystem that seems "more appropriate". This is usually a subsystem that has a larger amount of connectivity to the orphan than any other subsystem. For more information see [Tzerpos00].

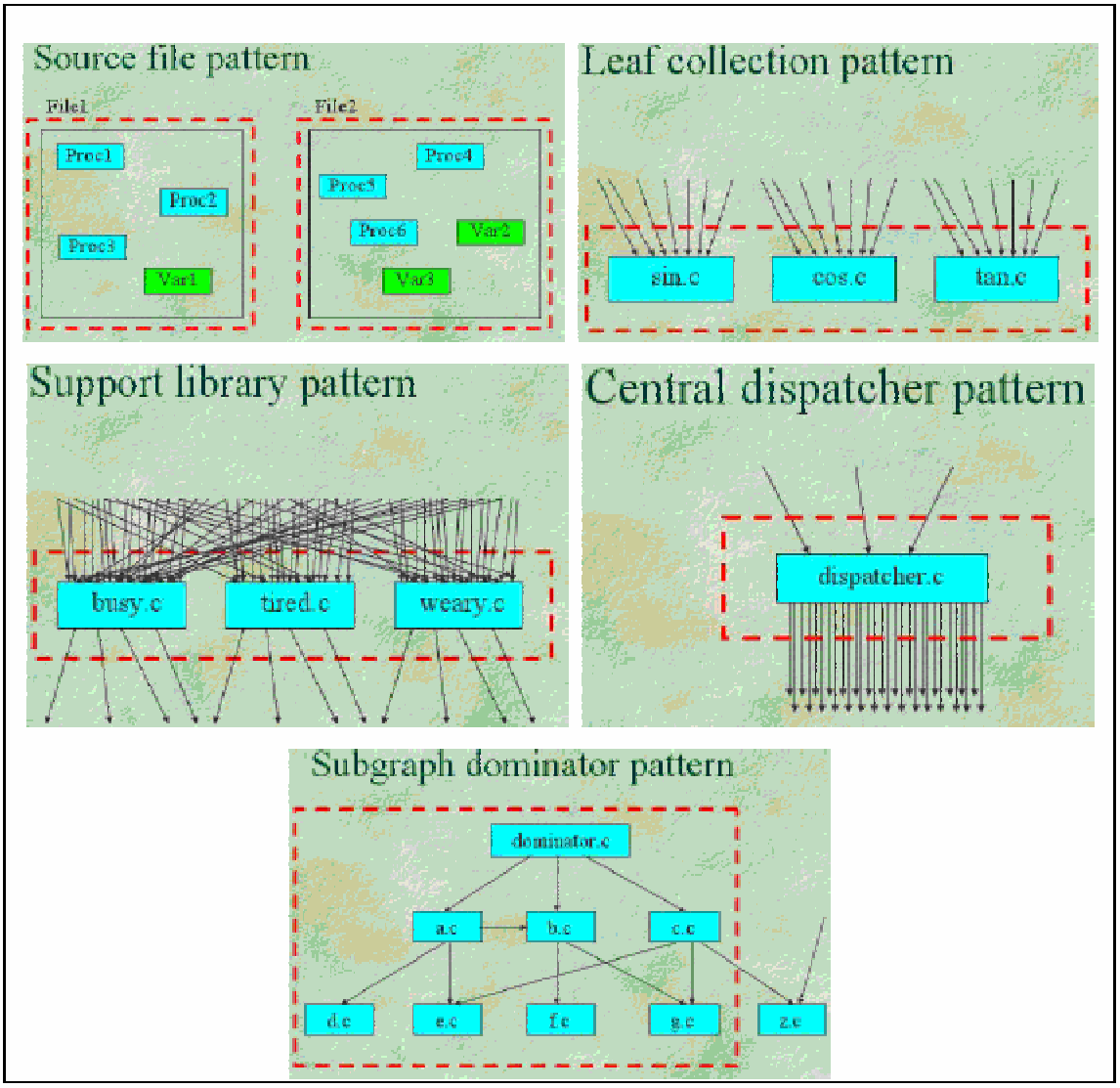


Figure 26 – Software patterns used in ACDC. Adopted from [Tzerpos00].

3.6 Supervised classification

Supervised classification methods base the knowledge discovery process on some instances for which the correct classification is known beforehand.

Support Vector Machines

Support Vector Machines were invented by Vladimir Vapnik [Vapnik98]. They are a method for creating functions from a set of labeled training data. The function can be a classification function – where the output is binary representing if an input data object belongs in a category - or the function can be a general regression function.

For classification, SVMs operate by finding a hypersurface in the space of possible inputs. This hypersurface will attempt to split the positive examples from the negative examples. The split will be chosen to have the largest distance from the hypersurface to the nearest of the positive and negative examples. This makes the classification correct for testing data that is near, but not identical to the training data. More information can be found in Burges' tutorial [Burges98] or in Vapnik's book [Vapnik98].

There are various ways to train SVMs. One particularly simple and fast method is Sequential Minimal Optimization.

The output of an SVM is an uncalibrated value, not a posterior probability of a class given an input.

Training an SVM on a large data set with many classes can be slow.

CCA-S

A data mining algorithm named Clustering and Classification Algorithm – Supervised (CCA-S) has been developed for detecting intrusions into computer network systems [Ye01].

CCA-S learns signature patterns of both normal and intrusive activities in the training data, and classifies the activities in the testing data as normal or intrusive based on the learned signature patterns of normal and intrusive activities. CCA-S differs from many existing data mining techniques in its ability in scalable, incremental learning.

CCA-S was tested and compared with two decision tree algorithms to obtain their performance for an intrusion detection problem. CCA-S produced better intrusion detection performance than these decision tree algorithms.

4 References

[Afshari99] Afshari, C.A. et al. Application of Complementary DNA Microarray Technology to Carcinogen Identification, Toxicology, and Drug Safety Evaluation. *Cancer research* **59,** 4759-4760 (1999).

[Agrawal98] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, Prabhakar Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. SIGMOD 1998.

[Aliz00] Alizadeh, A. et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature* **403**, 503-511 (2000).

[Alon99] Alon, U. et al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA* **Vol. 96,** 6745-6750 (1999).

[Alsabti98] K. Alsabti, S. Ranka, and V. Singh. ``An efficient *k*-means clustering algorithm". In *Proceedings of IPPS 11th International Parallel Processing Symposium*, 1998.

[Altman98] Russ B. Altman. Bioinformatics in support of molecular medicine. In C.G. Chute, Ed., 1998 AMIA Annual Symposium, Orlando, FL, 53-61 (1998).

[Andritsos04] P. Andritsos, P. Tsaparas, R. J. Miller, K. C. Sevcik. LIMBO: Scalable Clustering of Categorical Data. In 9th International Conference on Extending DataBase Technology (EDBT), March 2004.

[Ankerst99] Mihael Ankerst, Markus Breunig, Hans-Peter Kriegel, Jorg Sander. OPTICS: ordering points to identify the clustering structure. Proceedings of the 1999 ACM SIGMOD international conference on Management of data. Philadelphia, Pennsylvania, United States. Pages: 49 – 60. (1999).

[Barbara02] D. Barbara, Y. Li, J. Couto. COOLCAT: an entropy-based algorithm for categorical clustering. In Proceedings of CIKM'02, pp. 582-589, 2002.

[BenDor00] Amir Ben-Dor, Laurakay Bruhn, Nir Friedman, Iftach Nachman, Michel Schummer and Zohar Yakhini. Tissue Classification with Gene Expression Profiles. *Proceedings of the 4th Annual International Conference on Computational Molecular Biology*, 2000.

[BenDor99] Amir Ben-Dor, Ron Shamir, Zohar Yakhini. Clustering Gene Expression Patterns. *Journal of Computational Biology* **6(3/4):** 281-297 (1999).

[Bezdek81] Bezdek, J.C., 1981. Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York.

[Bittner00] Bittner, M., et al. Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature* **406**, 536-540 (2000).

[Brown00] Brown M.P.S. Grundy W.N., Lin D., Cristianini N., Sugnet C.W., Furey T.S., Manuel Ares, and Haussler D. Knowledge-based analysis of microarray gene expression data by using support vector machines. *PNAS* **97(1),** 262-267 (2000).

[Brown99] Brown, P.O. & Botstein, D. Exploring the new world of the genome with DNA microarrays. *Nature Genetics* **21** (suppl.), 33-37 (1999).

[Burges98] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery. Volume 2, Issue 2. Pages: 121–167. (June 1998).

[Butler01] Declan Butler. Are you ready for the revolution? *Nature* **409**, 758-760 (2001).

[Chu98] Chu, S., DeRisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P.O. and Herskowitz, I. The Transcriptional Program of Sporulation in Budding Yeast. *Science* **282**, 699-705 (1998).

[Cole99] Kristina A. Cole, David B. Krizman, and Michael R. Emmert-Buck, The genetics of cancer—a 3D model. Nature 21, 38-41 (1999).

[Dembele03] Dembele, D. and Kastner, P., Fuzzy C-means method for clustering microarray data, *Bioinformatics*, 19:973–980, 2003.

[DeRisi97] DeRisi, J.L., Iyer, V.R. and Brown, P.O. Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale. *Science* **278**, 680-686 (1997).

[DeRisi96] DeRisi, J. et al. Use of a cDNA microarray to analyse gene expression patterns in human cancer. *Nature Genetics* **14**, 457-460 (1996).

[Eisen99] Eisen, M.B. & Brown, P.O. DNA arrays for analysis of gene expression. *Methods Enzymol.* **303**, 179-205 (1999).

[Eisen98] Eisen, M.B. et al. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences USA* **95**, 14863-14868 (1998).

[Engelbrecht2002] Engelbrecht A.P., Computational Intelligence, An Introduction, pp. 61-71, John Wiley & Sons, Inc, 2002, ISBN 0-470-84870-7.

[Ester96] Ester, M., Kriegel, H-P., Sander, J., and Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceeding of 2nd Int. Conf.On Knowledge Discovery and Data Mining*, Portland (pp. 226–23).

[Ester98] Ester, M., Kriegel, H.-P., Sander, J.,Wimmer, M., and Xu, X. (1998). Incremental Clustering for Mining in a Data Warehousing Environment. In *Proceedings of 24th VLDB Conference*, New York, USA.

[Fasulo99] Fasulo D. (1999) An Analysis of Recent Work on Clustering Algorithms, Technical Report # 01-03-02, Department of Computer Science & Engineering, University of Washington.

[Fields89] Fields, S. and Song, O.K. A novel genetic system to detect protein-protein interactions. *Nature* **340**, 245-246 (1989).

[Fisher87] Fisher, D.H. Knowledge Acquisition via Incremental Conceptual Clustering. Machine Learning 2:139-172 (1987).

[Furey00] Furey, T.S. et al. Support Vector Machine Classification and Validation of Cancer Tissue Samples Using Microarray Expression Data. *Technical Report* **UCSC-CRL-00-04**, Department of Computer Science, University of California, Santa Cruz, 2000.

[Ganti99] V. Ganti, J. Gehrke, R. Ramakrishnan. CACTUS-clustering categorical data using summaries. In Proceedings of KDD '99, pp. 73-83.

[Gasch02] Gasch, A. and Eisen, M., Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering, *Genome Biology*, 3(11):research0059.1–research0059.22, 2002.

[Gasch00] Gasch, A.P. et al. Genomic expression programs in the response of yeast cells to environmental changes. *Molecular Biology of the Cell* **11**, 4241-4257 (2000).

[Gibson98] D. Gibson, J. Kleiberg, P. Raghavan. Clustering categorical data: an approach based on dynamic systems. In Proc of VLDB'98, pp. 311-323, 1998.

[Goebel99] Goebel, M. & Gruenwald, Le. A survey of data mining and knowledge discovery software tools. *ACM SIGKDD Explorations* **1**, 20-33 (1999).

[Golub99] Golub, T. R. et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* **286**, 531-537 (1999).

[Grambeier02] Grambeier J., Rudolph A. (2002) Techniques of Cluster Algorithms in Data Mining. Data Mining and Knowledge Discovery 6: 303-360.

[Guha00] Guha Sudipto, Rastogi Rajeev, Shim Kyuseok. (2000). ROCK: A Robust Clustering Algorithm for Categorical Attributes. Information Systems 25(5): 345-366.

[Guha98] Guha Sudipto, Rastogi Rajeev, Shim Kyuseok. (1998). CURE: An Efficient Clustering Algorithm for Large Databases. Proceedings of the 1998 ACM SIGMOD international conference on Management of data. Seattle, Washington, United States. Pages: 73 – 84. ISBN:0-89791-995-5

[Han01] Jiawei Han, Micheline Kamber "Data Mining: Concepts and Techniques", 2001, Academic Press.

[Hartigan75] Hartigan, J. A. Clustering algorithms. (John Wiley and Sons, New York, 1975).

[Hastie00] Hastie T, Tibshirani R, Eisen MB, Alizadeh A, Levy R, Staudt L, Chan WC, Botstein D, Brown P. 'Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biol.* **1(2)**, RESEARCH 0003.1-0003.21. (2000).

[He2002] Z. He, X. Xu, S. Deng. Squeezer: an efficient algorithm for clustering categorical data. Journal of Computer Science & Technology, 2002, 17(5): 611-624.

[Hess98] Hess, J. et al. Application of Differential cDNA Screening Techniques to the Identification of Unique Gene Expression in Tumours and Lymphocytes. *Current Opinion in Immunology* **10(2),** 125-130 (1998).

[Hinneburg98] Hinneburg, A. and Keim, D. (1998). An Efficient Approach to Clustering in Large Multimedia Databases with Noise. In *Proceedings of KDD Conference*.

[Hochbaum85] Hochbaum, D. S. and Shmoys, D. B.. A best possible heuristic for the k-center problem. Mathematics of Operational Research, 10(2):180-184, 1985.

[Huang99] Z. Huang, M.K.Ng. (1999). A fuzzy k-modes algorithm for clustering categorical data. IEEE Transaction on Fuzzy Systems, 1999, 7(4): 446-452.

[Huang98] Huang Z. (1998) Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. Data Mining and Knowledge Discovery 2(3): 283-304.

[Huang97] Huang, Z. (1997) Clustering Large Data Sets with Mixed Numeric and Categorical Values. Knowledge discovery and data mining: techniques and applications. World Scientific.

[Ito00] Ito, T. et al. Toward a protein-protein interaction map of the budding yeast: A comprehensive system to examine two-hybrid interactions in all possible combinations between the yeast proteins. *Proc. Natl. Acad. Sci. USA* **97**, 1143-1147 (2000).

[Jarvik75] Jarvik J. and Botstein D. Conditional-lethal mutations that suppress genetic defects in morphogenesis by altering structural proteins, *Proc Natl Acad Sci USA* **72(7)**. 2738-2742 (1975).

[Karypis99] G. Karypis, E.H. Han and V. Kumar. CHAMELEON: a hierarchical clustering algorithm using dynamic modeling. IEEE Computer Special Issue on Data Analysis and Mining. Volume 32 , Issue 8. Pages 68-75 (1999).

[Kaufmann90] L. Kaufmann and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

[Kaufmann87] L. Kaufmann and P. J. Rousseeuw. *Clustering by means of medoids*. In Y. Dodge, editor, Statistical Data Analysis based on the L 1 Norm and Related Methods, pages 405--416. Elsevier Science, 1987.

[kegg] Saccharomyces cerevisiae genes, according to the KEGG metabolic and regulatory pathways. http://www.genome.ad.jp/dbget-bin/get_htext?S.cerevisiae.kegg+-f+T+w+C/

[Khan99] Khan, J. et al. DNA microarray technology: the anticipated impact on the study of human disease. *Biochimica et Biophysica Acta* **1423**, M17-M28 (1999).

[Khan992] Khan, J. et al. Expression profiling in cancer using cDNA microarrays. *Electrophoresis* **20(2),** 223-229 (1999).

[Kononen98] Kononen J. et al. Tissue microarrays for high-throughput molecular profiling of tumor specimens. *Nature Medicine* **4(7),** 844-847 (1998).

[Kumar00] Kumar, A. et al. TRIPLES: a database of gene function in *Saccharomyces cerevisiae*. *Nucleic Acids Res.* **28**, 81-84 (2000).

[Lazzeroni00] Lazzeroni, L., and A. Owen. Plaid models for gene expression data. Technical report, Department of Statistics, Stanford University, 2000. http://www-stat.stanford.edu/research/list.html/

[Li04] Yun (Lillian) Li, H.S. Venter, J.H.P. Eloff. Categorizing vulnerabilities using data clustering techniques (2004). Information Security South Africa (ISSA 2004).

[Mancoridis99] S. Mancoridis, B.S. Mitchell, Y. Chen, and E. R. Gansner, Bunch: a clustering tool for the recovery and maintenance of software system structures, In Proceedings of International Conference on Software Engineering, 1999.

[Mjolsness99] Mjolsness, E., Castano, R., and Gray, A. Multi-Parent Clustering Algorithms for Large-Scale Gene Expression Analysis. *Technical report* **JPL-ICTR-99-5**, Jet Propulsion Laboratory Section 367, California Institute of Technology, October 1999.

[Mjolsness992] Mjolsness, E., Castano, R., Mann, T., Roden, J., Gray, A., and Wold, B. Clustering methods for the analysis of C.elegans gene expression array data. *Technical report* **JPL-ICTR-99-1**, Jet Propulsion Laboratory Section, California Institute of Technology, 1999.

[Michaels98] Michaels, G. et al. Cluster analysis and data visualization of large-scale gene expression data. *Pacific Symposium on Biocomputing (PSB)* **3**, 42-53 (1998).

[mips] MIPS --- Saccharomyces cerevisiae - The Yeast Genome. http://www.mips.biochem.mpg.de/proj/yeast/

[Ng94] Ng, R. and Han, J. Effecient and Effictive Clustering Methods for Spatial Data Mining. In *Proceeding's of the 20th VLDB Conference*, Santiago, Chile. (1994).

[Novick89] Novick P., Osmond BC, and Botstein D. Suppressors of yeast actin mutations. *Genetics* **121(4)**. 659-674 (1989).

[Perou00] Perou, C. et al. Molecular Portraits of Human Breast Tumours. Nature 406, 747-752 (2000).

[Perou99] Perou, C. et al. Distinctive Gene Expression Patterns in Human Mammary Epithelial Cells and Breast Cancers. *Proc. Natl. Acad. Sci. USA* **Vol. 96**, 9212-9217 (1999).

[Peters04] Markus Peters and Mohammed J. Zaki. CLICK: Clustering Categorical Data using K-partite Maximal Cliques.

[Petra00] Petra Ross-Macdonald. Functional analysis of the yeast genome. *Funct. Integr. Genomics* **1**, 99-113 (2000).

[Portnoy01] Leonid Portnoy, Eleazar Eskin and Sal Stolfo. Intrusion Detection with Unlabeled Data Using Clustering (2001). In ACM Workshop on Data Mining Applied to Security (DMSA 2001).

[Ross00] Ross, D. et al. Systematic variation in gene expression patterns in human cancer cell lines. *Nature Genetics* **24**, 227-235 (2000).

[Sandrock99] Sandrock, T.M. et al. Suppressor Analysis of Fimbrin (Sac6p) Overexpression in Yeast. *Genetics* **151**. 1287-1297 (1999).

[Scherf00] Scherf, U. et al. A gene expression database for the molecular pharmacology of cancer. *Nature Genetics* **24,** 236-244 (2000).

[Schwik00] Schwikowski, B. et al. A network of protein-protein interactions in yeast. *Nature Biotechnology* **18**, 1257-1261 (2000).

[sgd] SGD --- Saccharomyces Genome Database. http://genome-www.stanford.edu/Saccharomyces/

[Sgroi99] Sgroi Dc. et al. In vivo gene expression profile analysis of human breast cancer progression. *Cancer research* **59(22),** 5656-5661 (1999).

[Sheikholeslami98] WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. Gholamhosein Sheikholeslami, Surojit Chatterjee, Aidong Zhang. Proc. 24th Int. Conf. Very Large Data Bases, VLDB (1998).

[Sherman97] Sherman, Fred. Yeast Genetics. *The Encyclopedia of Molecular Biology and Molecular Medicine*, **Vol. 6.** 302-325 (1997).

[Slonim00] Donna K. Slonim, Pablo Tamayo, Jill P. Mesirov, Todd R. Golub, and Eric S. Lander. Class prediction and discovery using gene expression data. *Proceedings of the Fourth Annual Conference on Computational Molecular Biology (RECOMB),* 263-272, April, 2000.

[Spellman98] Spellman, P. et al. Comprehensive identification of Cell Cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell* **9**, 3273-3297 (1998).

[Straus99] Robert L. Strausberg and M.J. Finley Austin. Functional Genomics: technological challenges and opportunities. *Physiol Genomics* **1.** 25-32 (1999).

[Stutz95] Stutz J. and Cheeseman P. (1995) Bayesian Classification (AutoClass): Theory and results. Advances in Knowledge Discovery and Data Mining, 153-180, Menlo Park, CA, AAAI Press.

[Tamayo99] Tamayo, P. et al. Interpreting patterns of gene expression with self-organizing maps (SOMs): Methods and applications to hematopoietic differentiation. *Proc. Natl. Acad. Sci. USA* **96**, 2907-2912 (1999).

[Tzerpos00] V. Tzerpos and R. C. Holt. ACDC: An algorithm for comprehension-driven clustering. In Proceedings of the Seventh Working Conference on Reverse Engineering, pages 258.267, 2000.

[Uetz00] Uetz, P. et al. A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. *Nature* **403**, 623-627 (2000).

[Vapnik98] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[Wang99] Wang, K. et al. Monitoring Gene Expression Profile Changes in Ovarian Carcinomas using cDNA microarrays. *Gene* **229(1-2)**, 101-108 (1999).

[Wang97] Wei Wang, Jiong Yang, Richard Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining. Proceedings of the 23rd International Conference on Very Large Data Bases. Pp. 186-195. (1997)

[Wein97] Weinstein, J. et al. An information-intensive approach to the molecular pharmacology of cancer. *Science* **275,** 343-349 (1997).

[Wen98] Wen, X. et al. Large-scale temporal gene expression mapping of central nervous system (CNS) development. *Proc. Natl. Acad. Sci. USA* **95**, 334-339 (1998).

[Yang2002] Y. Yang, S. Guan, J. You. CLOPE: a fast and effective clustering algorithm for transactional data. In Proc of KDD'02, pp. 682-687, 2002.

[Ye01] Nong Ye, Xiangyang Li. A Scalable Clustering Technique for Intrusion Signature Recognition, Proceedings of the 2001 IEEE Workshop on Information Assurance and Security United States Military Academy, West Point, NY, 5-6 June, 2001.

[yeastcellcycle] Cell Cycle Regulated Yeast Genes. http://cellcycle-www.stanford.edu/

[yeaststress] Genomic Responses of Yeast to Diverse Stress Conditions. http://genome-www.stanford.edu/yeast_stress/

[ypd] YPD (Yeast Proteome Database) Quick Search. http://www.proteome.com/databases/YPD/YPDsearch-quick.html/

[Zhang97] Zhang, L. et al. Gene Expression Profiles in Normal and Cancer Cells. *Science* **276**, 1268-1272 (1997).

[Zhang96] Tian Zhang, Raghu Ramakrishnan, Miron Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *ACM SIGMOD*, Montreal, Canada. (1996).