INFORMATION-THEORETICAL METHODS IN CLUSTERING

A Dissertation Presented
by
Elena D. Cristofor

Submitted to the Office of Graduate Studies, University of Massachusetts Boston, in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY
June 2002
Computer Science Program
INFORMATION-THEORETICAL METHODS IN CLUSTERING

A Dissertation Presented
by
Elena D. Cristofor

Approved as to style and content by:

Dan A. Simovici, Professor
Chairperson of Committee

Richard L. Tenney, Professor
Member

William R. Campbell, Associate Professor
Member

Ivan Stojmenovic, Professor
Member

Dan A. Simovici, Program Director
Computer Science Program

Peter Fejer, Chairperson
Computer Science Department
ABSTRACT

INFORMATION-THEORETICAL METHODS IN CLUSTERING

June 2002

Elena D. Cristofor, B.S., Politehnica University
M.S., Politehnica University
M.S., University of Massachusetts Boston
Ph.D., University of Massachusetts Boston

Directed by Professor Dan A. Simovici

Clustering is the process of organizing a set of objects in a meaningful way. Objects characterized by categorical attributes present a special challenge because the domains of the attributes lack a natural ordering. This thesis introduces information-theoretical measures that facilitate the definition of a distance between the partitions determined by the attributes of a database. These measures are based on a concave, sub-additive function, which allows us to generalize the notion of entropy and conditional entropy. We present their properties and application to three clustering problems related to categorical databases. (1) In subspace clustering we want to find all sets of attributes that induce a partitioning of the database objects into a specified number of blocks. This can be done by imposing a threshold on the value of the entropy associated with a set of attributes. This threshold represents the maximum entropy of a partition having a prescribed number of clusters. We introduce a new algorithm for discovering all maximal sets of attributes that partition the set of objects in the required number of blocks. (2) Finding the median partition is the problem of discovering a partition as similar as possible to all partitions determined by the attributes of
an input database. We introduce a distance measure between partitions based on the notion of generalized conditional entropy. Discovering the median partition is computationally expensive, due to the large number of possible partitions, and to search more efficiently this space, we use a genetic algorithm approach. (3) Using a sample database, where each object is tagged with its appropriate class, we can estimate the influence of each attribute on the determination of an object’s class. We propose a new genetic algorithm for finding the natural clustering of the objects represented by the partition that preserves the same attribute influences as the estimated values. We discuss the type of databases for which our clustering method gives good results and we present a measure to assess this quality.
To my parents.
ACKNOWLEDGMENTS

It is my pleasure to thank Professor Dan Simovici, my advisor, for the discussions and fruitful suggestions that contributed to this work. I would also like to thank Professors Richard Tenney, William Campbell, and Elizabeth O’Neil for reading the manuscript and providing useful suggestions and Laurentiu Cristofor for his collaboration in the work on which Chapter 4 is based.
# Table of Contents

## 1 Introduction .................................................. 1

1.1 Motivation .................................................. 1

1.2 Background .................................................. 1

1.3 Similarity and dissimilarity measures ................. 2

1.4 Complexity of the clustering problem .............. 5

1.5 Clustering methods ....................................... 6

1.5.1 \(k\)-medoids or PAM .............................. 8

1.5.2 CLARA ............................................... 11

1.5.3 \(k\)-means ........................................... 11

1.6 Genetic algorithms .................................... 13

1.7 Overview of the thesis ................................ 16

## 2 Clustering applications ................................. 17

2.1 Clustering spatial data .................................. 17

2.1.1 CLARANS ............................................ 17

2.1.2 DBSCAN ............................................ 19

2.2 Subspace clustering .................................... 21

2.2.1 CLIQUE ............................................... 21

2.2.2 ENCLUS ............................................... 22

2.3 Clustering databases with categorical attributes . . . . . 24

2.4 Similarity and dissimilarity measures .............. 26

viii
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>Hierarchical algorithms</td>
<td>29</td>
</tr>
<tr>
<td>2.6</td>
<td>Median partition</td>
<td>31</td>
</tr>
<tr>
<td>2.7</td>
<td>Genetic algorithms for grouping problems</td>
<td>33</td>
</tr>
<tr>
<td>3</td>
<td>Generalized entropy</td>
<td>35</td>
</tr>
<tr>
<td>3.1</td>
<td>Impurity measures</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>Generalized entropy of sets and partitions</td>
<td>37</td>
</tr>
<tr>
<td>3.3</td>
<td>Dissimilarity measures</td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>Projection clustering</td>
<td>46</td>
</tr>
<tr>
<td>4.1</td>
<td>Preliminaries</td>
<td>46</td>
</tr>
<tr>
<td>4.2</td>
<td>Projection clustering algorithm</td>
<td>49</td>
</tr>
<tr>
<td>4.3</td>
<td>Experimental results</td>
<td>51</td>
</tr>
<tr>
<td>4.4</td>
<td>Conclusions and future research</td>
<td>60</td>
</tr>
<tr>
<td>5</td>
<td>Genetic algorithm for partitioning</td>
<td>62</td>
</tr>
<tr>
<td>6</td>
<td>Median partition</td>
<td>66</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>66</td>
</tr>
<tr>
<td>6.2</td>
<td>Properties of partitions with $k$ classes</td>
<td>68</td>
</tr>
<tr>
<td>6.3</td>
<td>Fitness measures</td>
<td>70</td>
</tr>
<tr>
<td>6.4</td>
<td>Experimental results</td>
<td>76</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Study of the GA results for different number of rows or chromosomes</td>
<td>77</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Study of the GA results for different fitness measures</td>
<td>80</td>
</tr>
</tbody>
</table>
6.4.3 Searching for a partition with a natural number of classes . 86
6.5 Conclusions .................................................. 88

7 Finding the natural clustering .................................. 90
  7.1 Introduction .................................................. 91
  7.2 Clustering using genetic algorithms ...................... 92
  7.3 Fitness measure ............................................. 93
  7.4 Training for clustering ..................................... 94
  7.5 Experimental results ..................................... 95
  7.6 Conclusions ............................................... 105

8 Conclusions .................................................. 108

References .................................................... 112
LIST OF TABLES

4.1 Projection clustering for 50% of the attributes .......................... 52
4.2 Projection clustering for 70% of the attributes .......................... 53
4.3 Characteristics of the set \{class, stalk-shape, veil-type\} ........... 56
4.4 Common sets between the results obtained for \(f_{\text{gini}}, f_{\text{ent}},\) and \(f_{ge}\) 59
4.5 Number of clusterable sets common to \(f_{\text{ent}}\) and \(f_{ge}\) for \(p = 0.5\) 60
4.6 Number of clusterable sets common to \(f_{\text{ent}}\) and \(f_{ge}\) for \(p = 0.7\) 61

5.1 Input parameters of the genetic algorithm ............................... 63

6.1 Measures for assessing the quality of the partition \(\pi\) ........... 71
6.2 Fitness measures ............................................................. 72
6.3 Alg-Rand, Alg-TO, and \(k\)-means on the first synthetically generated database ................................................................. 82
6.4 Alg-Rand, Alg-TO, and \(k\)-means on the second type of synthetically generated database .................................................. 85
6.5 Alg-Rand with \(N^j(\pi)\), Shannon entropy and various \(k\) on the second type of synthetically generated database ...................... 87
6.6 Characteristics of the partitions found by Alg-Rand with \(N^j(\pi)\) on the second type of synthetically generated database ............. 88

7.1 Estimated attribute influence for zoo.data ............................... 96
7.2 Estimated attribute influence for soybean-small.data ................. 97
7.3 Estimated attribute influence for house-votes-84.data ............... 98
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4</td>
<td>Estimated attribute influence for various datasets</td>
<td>98</td>
</tr>
<tr>
<td>7.5</td>
<td>Results for zoo.data</td>
<td>101</td>
</tr>
<tr>
<td>7.6</td>
<td>Results for soybean-small.data</td>
<td>103</td>
</tr>
<tr>
<td>7.7</td>
<td>Results for house-votes-84.data</td>
<td>105</td>
</tr>
<tr>
<td>7.8</td>
<td>Average time performance of the clustering genetic algorithms using $f_{\text{gini}}$, $f_{\text{ent}}$ and $f_{\text{gs}}$ as generators</td>
<td>107</td>
</tr>
</tbody>
</table>
## List of Figures

1.1 Initial representative objects selection for $k$-medoids ............. 8  
1.2 $k$-medoids algorithm ............................................. 9  
1.3 $k$-means algorithm ................................................. 12  
1.4 GA algorithm ....................................................... 13  

2.1 CLARANS algorithm .................................................. 19  
2.2 ENCLUS algorithm .................................................... 24  
2.3 $k$-modes algorithm .................................................. 25  
2.4 ROCK main procedure ................................................ 31  

4.1 Projection clustering algorithm .................................... 50  
4.2 Generators ........................................................... 51  
4.3 Number of sets for $p = 0.5$ ....................................... 54  
4.4 Number of sets for $p = 0.7$ ....................................... 55  
4.5 Generalized entropy threshold ..................................... 56  
4.6 Time for $p = 0.5$ .................................................... 57  
4.7 Time for $p = 0.7$ .................................................... 58  

5.1 Clustering genetic algorithm ....................................... 64  

6.1 Explanation of the convergence process for measure $O^f(\pi)$ ...... 75  
6.2 Classification rate for Alg-Rand with increasing number of chro- 
mosomes ................................................................. 78
6.3 Number of iterations for Alg-Rand with increasing number of chromosomes ........................................ 78
6.4 Time in secs for Alg-Rand with increasing number of chromosomes ........................................ 79
6.5 Classification rate for Alg-Rand with increasing number of chromosomes ........................................ 80
6.6 Number of iterations for Alg-Rand with increasing number of chromosomes ........................................ 80
6.7 Time in secs for Alg-Rand with increasing number of chromosomes ........................................ 81
7.1 Classification rates for zoo.data ................................................................. 102
7.2 Classification rates for soybean-small.data ................................................................. 104
7.3 Classification rates for house-votes-84.data ................................................................. 106
CHAPTER 1

Introduction

1.1 Motivation

The existence of a large amount of data available on the World Wide Web and the need to organize efficiently, search, and mine massive databases for knowledge, have brought a new interest in the development of more intelligent and efficient clustering algorithms. This thesis introduces new information-theoretical measures and presents their application in designing new clustering algorithms for categorical databases.

1.2 Background

Clustering represents a classical problem in computer science, that has been studied extensively and has been used in a broad range of applications. This continuous interest illustrates the complexity and usefulness of clustering. Among the disciplines in which clustering plays an important role are decision-making, machine-learning, data mining, document retrieval, image segmentation, and pattern classification.

The clustering problem can be formalized as follows: given a set of objects (patterns, n-dimensional points, tuples in a database), characterized by some properties (features, attributes), the goal is to group them in a meaningful way.
The final groups are called *clusters* or *classes*, and the distribution of the objects into classes is called *clustering* or *partitioning* of the input data. In a general sense, the goal is to place similar objects in the same class, and dissimilar objects in different classes. In this process, it is crucial to define an appropriate similarity or dissimilarity measure on the objects to be clustered.

The method used to distribute the objects to the final classes depends strongly on the domain of the application, such that the same input data may be clustered in different ways, depending on the choice of similarity/dissimilarity measure between objects, and on the choice of the clustering algorithm itself. The number of clusters in the final clustering can be preassigned by the user or it can be an internal parameter of the clustering algorithm, to be determined exclusively based on the input data.

Clustering is a difficult problem from a combinatorial point of view. The space of all possible clusterings grows significantly with the increase in the dimension of the input data, and the challenge is to find an efficient way to produce the final clusters.

Data clustering represents a non-supervised learning method. The input data are provided without any specification about the desired classes, and the clustering methods should discover the structure embedded in the data without any external help.

### 1.3 Similarity and dissimilarity measures

An important notion in the clustering process is the similarity or dissimilarity measure between the objects to be arranged into classes.
Let \( O \) be a set whose elements are referred to as objects. A similarity measure is a function \( \text{sim} : O \times O \rightarrow [0, 1] \) for which the following conditions hold:

\[
0 \leq \text{sim}(i, j) \leq 1 \\
\text{sim}(i, i) = 1 \\
\text{sim}(i, j) = \text{sim}(j, i)
\]

for every \( i, j \in O \).

The value \( \text{sim}(i, j) \) defines the degree of similarity between the objects \( i \) and \( j \), with the property that the closer the objects \( i \) and \( j \) are, the larger the value of \( \text{sim}(i, j) \) is.

Similarly, a dissimilarity measure is a function \( \text{dis} : O \times O \rightarrow [0, 1] \). The value \( \text{dis}(i, j) \) reflects the degree of dissimilarity between objects \( i \) and \( j \). For a dissimilarity measure the following conditions hold:

\[
0 \leq \text{dis}(i, j) \leq 1 \\
\text{dis}(i, i) = 0 \\
\text{dis}(i, j) = \text{dis}(j, i)
\]

A dissimilarity measure could be defined by starting with a similarity measure \( \text{sim} \) as \( \text{dis}(i, j) = 1 - \text{sim}(i, j) \), for every \( i, j \).

A semi-metric on the set \( O \) is a function \( d : O \times O \rightarrow R \), for which the following conditions hold:

\[
d(i, j) \geq 0 \\
d(i, i) = 0 \\
d(i, j) = d(j, i) \\
d(i, j) + d(j, k) \geq d(i, k)
\]

for every \( i, j, k \in O \). If \( d(i, j) = 0 \) implies \( i = j \), then we say that \( d \) is a metric (or a distance) on \( O \).
When the objects to be clustered are represented as \( n \)-dimensional points, the most commonly used dissimilarity measure is the \( L_p \) metric (Minkowski distance), defined as:

\[
L_p(i, j) = \left( \sum_{l=1}^{n} |x_l - y_l|^p \right)^{\frac{1}{p}}
\]

where \( 1 \leq p \leq \infty \), \( i = (x_1, x_2, \ldots, x_n) \) and \( j = (y_1, y_2, \ldots, y_n) \). When \( p = 1 \), \( L_p \) specializes to:

\[
L_1(i, j) = \sum_{l=1}^{n} |x_l - y_l|
\]

known as the Manhattan or city-block distance, and when \( p = 2 \) to:

\[
L_2(i, j) = \left( \sum_{l=1}^{n} |x_l - y_l|^2 \right)^{\frac{1}{2}}
\]

known as the Euclidean distance.

When the attributes characterizing the objects to be clustered have discrete domains, then the similarity measure can be defined as:

\[
d_A(i, j) = \sum_{l=1}^{n} \gamma_l
\]

where \( i = (x_1, x_2, \ldots, x_n) \), \( j = (y_1, y_2, \ldots, y_n) \), and

\[
\gamma_l = \begin{cases} 
1, & \text{if } x_l = y_l \\
0, & \text{otherwise}
\end{cases}
\]

This distance reflects the number of agreements between objects \( i \) and \( j \) with respect to their attribute values.

When the objects to be clustered are represented as sets, the Jaccard coefficient, defined as:

\[
J(i, j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}
\]
is a similarity measure, where $S_i$ and $S_j$ are the sets associated with objects $i$ and $j$.

More information about similarity/dissimilarity measures can be found in [Mir96], [KR90] and [JMF99].

### 1.4 Complexity of the clustering problem

The number of ways to partition a set of $n$ objects into $k$ non-empty classes, is a Stirling set number of the second type, and can be computed from the formula:

$$S(n, k) = \frac{1}{k!} \sum_{i=0}^{k-1} (-1)^i \binom{k}{i} (k - i)^n.$$  

For example, there is only one way to partition the objects into a single group or into $n$ disjoint groups, so $S(n, 1) = 1$ and $S(n, n) = 1$, and there are $S(n, 2) = 2^{n-1} - 1$ ways to partition the objects into two non-empty groups.

If we have a set of $n$ objects and we want to partition them into $k$ classes of specified cardinalities $n_1, n_2, \cdots, n_k$, such that $n_1 + n_2 + \cdots + n_k = n$, the number of possible partitions is given by:

$$\frac{n!}{n_1! n_2! \cdots n_k!}$$

Thus, the number of ways to partition a set of $n$ elements into $k$ or fewer classes is given by:

$$\sum_{n_1 + n_2 + \cdots + n_k = n} \frac{n!}{n_1! n_2! \cdots n_k!} = k^n$$

The number of partitions of $n$ objects having exactly $k$ classes is of the order of $\frac{k^n}{k!}$, and the number of partitions having $k$ or fewer classes is equal to $k^n$. These results show that to check exhaustively all possible clusterings is not feasible, except for very small values of $n$ and $k$. 

5
Another important property of a clustering is the \textit{diversity} ([Mir96]) of the partition of the \( n \) objects into \( k \) clusters. The diversity is minimum when all the objects belong to the same cluster, and it is maximum when all clusters contain only one object. Given a partition \( \pi \) of the set of objects, having class cardinalities \( n_1, n_2, \cdots, n_k \), two measures have been used in the literature to assess the degree of diversity. The Shannon entropy given by:

\[
H(\pi) = - \sum_{i=1}^{k} \frac{n_i}{n} \log \left( \frac{n_i}{n} \right)
\]

and the Gini index (or qualitative variance) given by:

\[
G(\pi) = 1 - \sum_{i=1}^{k} \left( \frac{n_i}{n} \right)^2 = \sum_{i=1}^{k} \frac{n_i}{n} \left( 1 - \frac{n_i}{n} \right).
\]

The entropy measures the average value of the quantity \( -\log \frac{n_i}{n} \) (the amount of information associated with the cluster of cardinality \( n_i \)) and the Gini index measures the average value of the quantity \( 1 - \frac{n_i}{n} \) (the error of predicting if an object belongs to the cluster of cardinality \( n_i \)). Both measures have their minimum 0 when \( n_i = n \) for some \( i \), and have their maximum when \( n_1 = n_2 = \cdots = n_k = \frac{n}{k} \).

1.5 Clustering methods

Clustering methods fall into two main categories: \textit{partitioning} methods ([Boc96, RF01, MSD81]), producing a single clustering of the input data, and \textit{hierarchical} methods, producing a series of nested clusterings ([Fis96, BRM96, Gor96]).

In \textit{partitioning} methods, the goal is to find \( k \) clusters \( C_1, C_2, \cdots, C_k \) of the \( n \) input objects, that optimize a certain criterion. The most commonly used
criterion is the minimization of the quantity:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, m_i),$$

where $m_i$ represents the centroid (or medoid) of the cluster $C_i$, and $d(x, m_i)$ is the Euclidean distance between object $x$ and centroid $m_i$. The centroids $m_i$ may or may not be part of the input objects. The general strategy is the following: starting with $k$ initial clusters, the objects are moved from one cluster to another to improve the value of the criterion function $E$. Intuitively, the method attempts to minimize the distance between each object and the centroid of the cluster to which the object is assigned.

The hierarchical methods can be further subdivided into agglomerative (the clustering process starts with each object in its own cluster, and the clusters are subsequently merged until some stopping criterion is satisfied) and divisive (all objects are in a single cluster and then clusters are split until some stopping criterion holds). Usually, the stopping criterion is a desired number of clusters.

From a different point of view, clustering methods can be categorized into methods that produce a hard clustering, where the objects are grouped into disjoint classes, or fuzzy clustering, where each object has associated degrees of membership in all classes. A fuzzy clustering can be transformed into a hard clustering by assigning each object to the class for which it has the largest degree of membership.

A key issue in all clustering methods is the type of data and the existence or not of a natural way to measure the similarity or dissimilarity between the objects. In general, partitioning algorithms use an input parameter that specifies the number of classes in the final clustering. This value has a great impact on the quality of the clustering. Hierarchical methods produce a hierarchy of nested
clusterings, that can range from the clustering with one class containing all the objects, to the clustering with \( n \) classes, each containing one object. The user decides at which hierarchy level to stop the clustering process.

### 1.5.1 \( k \)-medoids or PAM

An example of a classical partitioning method is the \( k \)-medoids algorithm (also known as PAM, Partitioning Around Medoids). This method selects \( k \) objects called representative objects (medoids) out of the given objects \( x_1, \ldots, x_n \), and then assigns the remaining objects to the nearest representative object. The representative objects should be located in the center of the clusters they define. The quality of the clustering is measured by the average dissimilarity between each object and the representative object of its cluster. The procedure for selecting the first \( k \) representative objects is presented in Figure 1.1.

1. Let \( i = 0 \); the first representative object \( r_1 \) is chosen such that \( \sum_{x_i \in \{x_1, x_2, \ldots, x_n\}} dis(x_i, r_1) \) is minimum.
2. \( i = i + 1 \); the representative objects selected so far are \( r_1, r_2, \ldots, r_i \).
3. For each object \( x_h \not\in \{r_1, r_2, \ldots, r_i\} \), consider \( x_h \) as a candidate for the \( r_{i+1} \) representative object. For each object \( x_l \not\in \{r_1, r_2, \ldots, r_i\} \), let \( r_j \) be the closest representative object to \( x_l \). Compute the influence of \( x_l \) in the decision of having \( r_{i+1} = x_h \), given by \( \text{gain}(x_h) = \max\{dis(x_l, r_j) - dis(x_l, x_h), 0\} \). Compute the total gain of having \( r_{i+1} = x_h \), given by \( \text{gain}(x_h) = \sum_{x_l} \text{gain}(x_l, x_h) \). Choose the object \( x_h \) for which \( \text{gain}(x_h) \) is maximum.
4. If \( i + 1 < k \) go to step 2, otherwise exit.

Figure 1.1: Initial representative objects selection for \( k \)-medoids
The measure \(\inf(x_i, x_h) = \max\{\text{dis}(x_i, r_j) - \text{dis}(x_i, x_h), 0\}\) has the following significance. If object \(x_i\) is closer to \(x_h\) than to its currently nearest representative object \(r_j\), then by choosing \(r_{i+1} = x_h\) the objective function (average of dissimilarities between objects and their closest representative object) will decrease with the quantity \(\text{dis}(x_i, r_j) - \text{dis}(x_i, x_h)\). We want to choose as \(r_{i+1}\) the object that will determine the most significant decrease in the objective function, so we choose that \(x_h\) which maximizes the quantity \(\sum x_i \inf(x_i, x_h)\).

The pseudocode for the \(k\)-medoids algorithm is presented in Figure 1.2.

1. Select the representative objects \(r_1, r_2, \cdots, r_k\) as explained in the previous procedure, and assign the remaining objects to the nearest representative object.
2. For all objects \(x_h\) in the input data (not among \(r_1, r_2, \cdots, r_k\)), and all representative objects \(r_i\), compute the cost of replacing the representative object \(r_i\) with object \(x_h\), denoted by \(\text{Cost}(r_i, x_h)\).
3. Select the objects \(r_i\) and \(x_h\) that correspond to \(\min_{i,h} \text{Cost}(r_i, x_h)\).
4. If \(\text{Cost}(r_i, x_h) < 0\), then replace representative the object \(r_i\) with the object \(x_h\), and go back to step 2. Else, for each object \(x_h\) (not among \(r_1, r_2, \cdots, r_k\)), find the most similar representative object and assign it to the appropriate cluster.

Figure 1.2: \(k\)-medoids algorithm

In Figure 1.2 \(\text{Cost}(r_i, x_h)\) stands for the expression:

\[
\sum_{x_i \notin \{r_1, r_2, \cdots, r_k\}} \text{Cost}(r_i, x_h, x_i),
\]

where \(\text{Cost}(r_i, x_h, x_i)\) is defined as follows. If \(x_i\) belongs to the cluster defined by
\( r_i \) and \( r_j \) is the second most similar representative object, then \( \text{Cost}(r_i, x_h, x_l) \) is:

\[
\begin{align*}
\text{Cost}(r_i, x_h, x_l) &= \begin{cases} 
\text{dis}(x_l, r_j) - \text{dis}(x_l, r_i), & \text{if } \text{dis}(x_l, x_h) \ge \text{dis}(x_l, r_j) \\
\text{dis}(x_l, x_h) - \text{dis}(x_l, r_i), & \text{if } \text{dis}(x_l, x_h) < \text{dis}(x_l, r_j)
\end{cases}
\end{align*}
\]

In this situation, changing the representative object \( r_i \) with the object \( x_h \) has the following impact on the cluster assignment of object \( x_l \). The object \( x_l \) will be reassigned from the cluster represented by \( r_i \) to the new cluster represented by \( x_h \), if \( \text{dis}(x_l, x_h) \ge \text{dis}(x_l, r_j) \); otherwise \( x_l \) will continue to belong to the cluster associated to \( r_j \).

If \( x_l \) belongs to the cluster defined by the representative object \( r_j \), then \( \text{Cost}(r_i, x_h, x_l) \) is:

\[
\begin{align*}
\text{Cost}(r_i, x_h, x_l) &= \begin{cases} 
\text{dis}(x_l, x_h) - \text{dis}(x_l, r_j), & \text{if } \text{dis}(x_l, x_h) \le \text{dis}(x_l, r_j) \\
0, & \text{if } \text{dis}(x_l, x_h) > \text{dis}(x_l, r_j)
\end{cases}
\end{align*}
\]

In this case, if \( \text{dis}(x_l, x_h) \le \text{dis}(x_l, r_j) \), then the object \( x_l \) will be reassigned from the cluster represented by \( r_j \) to the new cluster represented by \( x_h \), otherwise \( x_l \) will continue to be associated with cluster \( r_j \).

In general, the dissimilarity measure is defined as the Euclidean distance. In steps 2 and 3 we have \( k(n - k) \) pairs for the choice of objects \( r_i, x_h \) and \( n - k \) possible choices of \( x_j \). Thus, the complexity of one iteration of this algorithm is \( O(k(n - k)^2) \), where \( n \) is the total number of points and \( k \) is the number of classes. Experimental results show that \( k \)-medoids runs satisfactory only for small datasets ([KR90, NH94]).

This method can be applied when we are interested in the final clustering of the input data or in the representative objects themselves, as a summary of the data.
1.5.2 CLARA

CLARA (CLustering LARge Applications, introduced in [KR90]) is a method for clustering large datasets, obtained from the classical $k$-medoids algorithm in the following way: a sample of points is extracted from the input data, clustered using the $k$-medoids algorithm, and then each point of the data is assigned to the nearest cluster. This procedure is repeated several times and the solution with the best overall objective function is retained as the final clustering. Usually, we look to minimize the objective function representing the total dissimilarity of the objects belonging to each cluster:

$$E = \sum_{i=1}^{k} \sum_{x \in clust(r_i)} dis(x, r_i).$$

where $r_1, r_2, \cdots, r_k$ are the representative objects and $clust(r_i)$ is the cluster associated with $r_i$.

1.5.3 $k$-means

Another classical clustering method, similar to $k$-medoids, is the $k$-means algorithm ([Mac67]). Given $n$ objects $\{x_1, \cdots, x_n\} \in \mathbb{R}^p$ in a $p$-dimensional space, where each $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip}) \in \mathbb{R}^p$, the method partitions them into $k$ clusters $C_1, \cdots, C_k$. Each cluster $C_i$ has an associated center or centroid $c_i = (c_{i1}, c_{i2}, \cdots, c_{ip}) \in \mathbb{R}^p$ computed as:

$$c_{ij} = \frac{\sum_{x \in C_i} x_{ij}}{|C_i|}.$$

The centroids are similar to the representative objects in the $k$-medoids method with the difference that they do not necessarily correspond to an existing object. $k$-means consists of alternating two steps: a step in which the clusters are updated by assigning each object to the cluster associated with the nearest centroid, and
another step in which the centroids are recomputed to reflect the centers of the newly formed clusters. The pseudocode of the $k$-means algorithm is presented in Figure 1.3.

1. Randomly choose a set of $k$ tentative centroids $c_1, c_2, \ldots, c_k$.
2. *Updating of the clusters.* Assign each object to the cluster associated with the nearest centroid. Exit if there is no change in the assignment of objects.
3. *Updating of the centroids.* Recompute the centers of the clusters formed in the previous step. Go to step 2.

Figure 1.3: $k$-means algorithm

Each time an object is reassigned to a different cluster, the centroids of the two clusters can be recomputed as follows. If $x_i$ is moved from cluster $C_1$ to cluster $C_2$, then the centroids of the new clusters are given by:

$$c_{1j} = \frac{|C_1| \cdot c_{1j} - x_{ij}}{|C_1| - 1}$$

$$c_{2j} = \frac{|C_2| \cdot c_{2j} + x_{ij}}{|C_2| + 1}$$

The objective function minimized by this method is the sum of squares of distances between objects and the centroids of the associated clusters. When an object is reassigned to the cluster associated with the nearest centroid, the distance between the object and the centroid of the cluster to which it belongs decreases and thus, the objective function also decreases.
1.6 Genetic algorithms

When the space of all possible solutions to a problem is very large, searching for the best solution by using exhaustive methods will not be efficient and we need to use more intelligent methods. Genetic algorithms (GAs) are among such techniques.

GAs represent a method of solving problems using biological principles of evolution and heredity. GAs are maintaining a population \( P(t) = \{K_1, K_2, \cdots, K_m\} \) of chromosomes, representing possible solutions to the problem, that will evolve into another population \( P(t+1) \) by means of selection and transformation of the fittest chromosomes. The pseudo-code for a GA algorithm is given in Figure 1.4.

\[
\begin{aligned}
&\text{Let } t = 0. \\
&\text{Randomly initialize the population of chromosomes } P(0). \\
&\text{Compute fitness of chromosomes in population } P(0). \\
&\text{while (stopping criteria not satisfied)} \\
&\quad t = t + 1. \\
&\quad \text{Select } P(t) \text{ from } P(t - 1). \\
&\quad \text{Perform transformations of the selected chromosomes.} \\
&\quad \text{Compute fitness of chromosomes in population } P(t).
\end{aligned}
\]

Figure 1.4: GA algorithm

The success of the genetic algorithms rely on two factors, the diversity of the population, which influences the search for the optimal solution and the selective pressure, which influences the creation of the new population. The selective pressure represents the degree of representation in the selection of the most fit individuals from the population. There is a strong influence among these two
factors. An enhancement of the selection pressure might decrease the diversity of the population, leading the GA to a premature convergence to a local optimum solution. On the other hand, a weak selection pressure might lead to a longer search for the best solution.

Encoding in a chromosome a solution to the problem at hand depends on the application. Traditionally, a chromosome is represented as a string, a sequence of 0’s and 1’s, but in clustering problems another encoding is needed. If we have \( n \) objects to be clustered into \( k \) classes, then a possible solution can be encoded as a sequence of numbers between 1 and \( k \). This representation is known as the group number encoding. All the objects associated with the same number belong to the same cluster. For example, if \( n = 10 \) and \( k = 2 \), then the chromosome \( K = 1211122112 \) represents a partition with 2 groups, where the objects 1, 3, 4, 5, 8, 9 belong to one cluster and the objects 2, 6, 7, 10 belong to the second cluster.

Each chromosome in the population is evaluated according to some criterion, dependent on the domain of the application. This criterion represents a way to quantify the degree of fitness of the chromosome with respect to the problem we are trying to solve. The greater this value, the more fit is the chromosome as a solution of the problem.

The selection of the fittest chromosomes can be done in different ways. A method called proportional selection or roulette wheel works in the following way. Each chromosome \( K_i \) is assigned a fitness value, denoted by \( f(K_i) \), which measures the effectiveness of the chromosome in solving the problem. We can associate with each chromosome a probability of being selected, computed based on this fitness, as:

\[
p(K_i) = \frac{f(K_i)}{\sum_{i=1}^{m} f(K_i)} \in [0, 1]
\]
and a cumulative probability

\[ p_c(K_i) = \sum_{i=1}^i p(K_i). \]

The roulette wheel strategy consists in spinning the wheel for \( m \) times and at each step proceeding as follows:

- randomly generate a number \( r \in [0, 1] \)
- if \( r < p_c(K_1) \), then select the first chromosome \( K_1 \); otherwise select the first \( i \) index such as \( p_c(K_{i-1}) < r \leq p_c(K_i) \).

This selection ensures that the fittest chromosomes will have a greater contribution in the creation of the new offspring because they are selected most often. The average fit chromosomes will have an even contribution and the worst fit chromosomes will disappear from the population because they will never be selected.

A different selection method called \textit{tournament selection} consists in repeating for a number of iterations the following procedure: a number of \( r \) chromosomes are randomly selected and the best of them is retained. The competition between two chromosomes \( i \) and \( j \) can also be based on a simulating annealing idea, where the winner is selected according to the formula \( \frac{1}{1+e^{\frac{f(K_i)-f(K_j)}{T}}} \), where \( T \) is a parameter of the selection procedure called temperature.

As a result of the selection procedure we have a number of chromosomes that will undergo external or internal transformations.

The external transformation is done by the crossover operator. Two chromosomes are crossed over and produce the new offspring. The method to generate the offspring can follow different patterns. The classical operators are the \textit{single}-
site and the two-sites crossover. In the first crossover operator randomly generates a position and forms the offspring by combining the first part from the first chromosome with the second part from the second chromosome and vice-versa. The second crossover operator randomly generates two positions and changes the middle part between the two chromosomes.

The internal transformation is done by applying the mutation operator. This operator changes the encoding of the chromosome by randomly selecting a number of positions and modifying the values at these positions.

1.7 Overview of the thesis

The remainder of the thesis is structured as follows. Chapter 2 presents different clustering applications and recent research done in these directions. Chapter 3 introduces new information-theoretical measures based on a generalization of the notions of entropy and conditional entropy. Chapter 4 presents the first application of the informational-theoretical measures to the problem of subspace clustering ([SCC00a, SCC00b]). A genetic algorithm approach for clustering problems is discussed in chapter 5. Chapter 6 ([CS01, CS02a]) presents the problem of finding the median partition as a summarization of a set of partitions determined by the attributes of a categorical database. Chapter 7 presents a classification problem, solved using a clustering genetic algorithm and information-theoretical measures ([CS02b]). Chapter 8 concludes the thesis with an overview of its results and suggestions for future research.
CHAPTER 2

Clustering applications

Clustering provides a way to gain insight into the structure of the input data, can be used as an intermediary step for further analysis and processing of the data, or as input data for algorithms that will not work efficiently with the high dimensional data available originally. In this chapter we present a number of clustering applications and the associated recent research.

2.1 Clustering spatial data

An important application of clustering methods is related to the analysis of large spatial databases.

2.1.1 CLARANS

CLARANS (introduced in [NH94] as a variation of the k-medoids algorithm) is designed to cluster large spatial data represented by n points into k clusters. The method is based on randomized search. We denote by $G_{n,k}$ a graph in which a node is defined by a set of k representative objects $\{r_1, \ldots, r_k\}$ belonging to the input dataset $\{x_1, \ldots, x_n\}$. Two nodes $S_1 = \{r_1, \ldots, r_k\}$ and $S_2 = \{w_1, \ldots, w_k\}$ are neighbors in this graph if $|S_1 \cap S_2| = k - 1$ (they have $k - 1$ common objects). Thus, in $G_{n,k}$ each node has $k(n - k)$ neighbors (since we can choose $k$ positions
for the missing object and we have \( n - k \) candidates for that position). Also, each node has an associated cost equal to the total dissimilarity between every object and the representative object of its cluster. If \( S_1 = \{ r_i \} \cup S \) and \( S_2 = \{ r_j \} \cup S \), then \( \text{cost}(S_1) - \text{cost}(S_2) = \text{Cost}(r_i, r_j) = \sum_{x_i \not\in \{ r_i, r_j \}} \text{Cost}(r_i, r_j, x_i) \) represents the cost of replacing the representative object \( r_i \) with the representative object \( r_j \). \( \text{Cost}(r_i, r_j, x_i) \) is the cost of replacing \( r_i \) with \( r_j \) from the point of view of object \( x_i \) as defined in the previous chapter.

The \( k \)-means method represents a search for a node with the minimum cost in this graph \( G_{n,k} \). At each step, all the neighbors of the current node are examined, and the current node is replaced with the node that provides the deepest descent in the cost value. The CLARA method tries to examine fewer neighbors and restricts the search on subgraphs \( G_{|S_{\text{sample}}|} \) of smaller sizes, where \( S_{\text{sample}} \) is the set of objects in the sample drawn from the \( n \) input objects. If \( M \) is the minimum node in the original graph \( G_{n,k} \) and \( M \) is not included in \( G_{|S_{\text{sample}}|} \), then \( M \) will never be found in the search using \( G_{|S_{\text{sample}}|} \). Many samples should be collected to eliminate this deficiency.

The CLARANS algorithm searches in the original graph, but does not check every neighbor of a node. CLARANS has two parameters: the maximum number of neighbors to be examined, \( \text{maxneighbor} \), and the number of local minima to be obtained, \( \text{numlocal} \). The pseudocode for the algorithm is represented in Figure 2.1.

When the maximum number of neighbors for a particular node has been reached, the neighbor yielding the smaller cost is considered a local minima. The algorithm stops after the maximum number of local minima have been computed and produces the representative objects associated with the node yielding the smallest cost. The higher is the value of the maximum neighbors examined,
1. Let \( \text{step} = 1 \), \( \text{mincost} \) = maximum double value, and \( S_{\text{curr}} \) = a randomly chosen node in \( G_{n,k} \).
2. Let \( i = 1 \).
3. Chose a random neighbor \( S \) of \( S_{\text{curr}} \) and compute the difference in cost between the two nodes \( \text{cost}(S) - \text{cost}(S_{\text{curr}}) \).
4. If \( S \) has a lower cost, then \( S_{\text{curr}} = S \) and go to step 2. Else, \( i = i + 1 \).
5. If \( i < \text{maxneighbor} \), then go to step 3.
6. If \( \text{cost}(S_{\text{curr}}) < \text{mincost} \), then set \( \text{mincost} = \text{cost}(S_{\text{curr}}) \), \( \text{bestnode} = S_{\text{curr}} \), and \( \text{step} = \text{step} + 1 \).
7. If \( \text{step} > \text{numlocal} \), then output \( \text{bestnode} \) and exit. Else, go to step 2.

Figure 2.1: CLARANS algorithm

the closer is CLARANS to \( k \)-medoids and the longer it takes to search for the minimum cost node.

2.1.2 DBSCAN

[EKS96] introduced a new clustering algorithm for spatial databases, called DBSCAN. This algorithm uses the notion of density in the definition of the clusters, thus allowing the discovery of arbitrarily shaped clusters. The main idea is that each object belonging to a cluster, contains in its neighborhood of a given radius, a minimum number of objects. The radius \( \epsilon \) of the thinnest cluster in the final clustering, and the minimum number of points \( N_{\text{min}} \) in the clusters (the density of the thinnest cluster) are two parameters of the DBSCAN algorithm. The neighborhood of an object \( x \) is defined as \( N_{\epsilon}(x) = \{x_i \mid \text{dis}(x, x_i) \leq \epsilon \} \). For the objects that are located at the border of their associated cluster, the requirement is that
they are close to an object whose neighborhood contains the minimum number of points specified by the user. An object \( x_1 \) is \textit{directly density-reachable} from an object \( x_2 \), if \( x_1 \in N_\epsilon(x_2) \) and \( |N_\epsilon(x_2)| \geq N_{\text{min}} \). An object \( x_1 \) is \textit{density-reachable} from an object \( x_2 \), if there is a series of objects \( x_{i_1} = x_1, x_{i_2} \cdots x_{i_l} = x_2 \) such that each \( x_{i_k} \) is directly density-reachable from \( x_{i_{k-1}} \), for \( k = 2 \) to \( l \). An object \( x_1 \) is \textit{density-connected} from an object \( x_2 \), if there is another object \( x_3 \) such that both pairs \( (x_1, x_3) \) and \( (x_2, x_3) \) are density-reachable. Thus, using these notions, a cluster is defined as a maximal set of density-connected objects. This notion of density-connectivity is parameterized in terms of the values of \( \epsilon \) and \( N_{\text{min}} \).

The algorithm starts with an arbitrary object \( x \) and retrieves all objects that are density-reachable from it with respect to the values of \( \epsilon \) and \( N_{\text{min}} \). This set forms a cluster in the final clustering. If \( x \) is located on the border of its associated cluster, then there is no other object density-reachable from it, thus no cluster is formed in this case, and the algorithm continues with another selection of an unprocessed object.

Experimental results ([EKS96]) showed that the DBSCAN algorithm outperforms the CLARANS algorithm, also introduced a heuristic method for determining the parameters \( \epsilon \) and \( N_{\text{min}} \) of the algorithm that lead to the best clustering.

An improvement of the DBSCAN clustering algorithm, called OPTICS, is presented in [ABK99]. OPTICS determines a density-based clustering of the data without requiring input parameters to define the clustering, by incrementally producing density-based clusterings of the input data, corresponding to different values of the parameters \( \epsilon \) and \( N_{\text{min}} \) used in DBSCAN. The article also presents a visualization method for displaying the resulting clustering.
2.2 Subspace clustering

Another important clustering application consists of partitioning the set of rows of a database according to subsets of its attributes ([AGG98, CWY99, AWY99]). The main idea is to find a subset of attributes that offers a good clustering of the rows. Looking for a subset of attributes with good clustering helps improving the final data clustering and also increases the computational efficiency of the search in the space of all possible clusterings.

2.2.1 CLIQUE

An algorithm called CLIQUE, introduced in [AGG98], determines an automatic clustering of high dimensional data. The data is partitioned into non-overlapping rectangular units, obtained by dividing every dimension into $\epsilon$ intervals of equal length ($\epsilon$ is an input parameter and every attribute of the database represents a dimension). A unit is the cross-product of one interval from each dimension and has associated a selectivity measure equal to the fraction of objects from the input data contained in the unit. A unit is dense if its selectivity is greater than a threshold value, also specified by the user. Two units $u_1$ and $u_2$ are connected if they have a common interval in one dimension or there is another unit $u_3$ such that both pairs, $(u_1, u_3)$ and $(u_2, u_3)$, are connected. Using these notions, a $k$-cluster is defined as a maximal set of connected dense units in $k$-dimensions.

The algorithm has three steps:

1. Identification of subspaces that contain clusters.
2. Identification of clusters.
3. Generation of a minimal description for the clusters.
The first step involves checking all subsets in an incremental way, starting with subsets containing one attribute, and at each iteration incrementing the cardinality of the sets by one. This approach is similar to the one used by the Apriori algorithm ([AMS96]). To generate a candidate \( k \)-subspace, we combine two dense \((k-1)\)-subspaces that have \( k-2 \) common units in some dimensions. To prune the candidates, the algorithm eliminates subspaces with lower selectivity. In the second step, the dense units resulting from the first step are considered to be the vertices of a graph in which an edge connects two dense units that have a common interval. A depth first search algorithm applied to this graph, provides the final clusters as the set of maximally connected dense units. Finally, the algorithm computes a minimal description of the clusters as a DNF formula representing a disjunction of the conjunctions describing the regions which comprise all the points in each dimension.

2.2.2 ENCLUS

A different subspace clustering algorithm, called ENCLUS, was introduced in [CWY99]. As in CLIQUE, ENCLUS uses the same division of the dimensions (determined by the attributes), into a number of equal size intervals (the size \( \Delta \) of the intervals is specified by the user), and the same notion of a unit. The algorithm scans the data and computes the number of objects in each unit. The density of unit \( u \), denoted by \( d(u) \), represents the percentage of the data contained in \( u \). For a subset \( X \) of attributes, having \( U \) units in total (\( U \) ways to form cross-products with the intervals in each dimension), we can compute its entropy as:

\[
H(X) = - \sum_{u \in U} d(u) \log d(u).
\]
The ENCLUS algorithm has the following steps:

1. Find subspaces with *good clustering* by using the entropy measure.

2. Identify the clusters.

3. Present the results to the user.

A threshold value $w$, imposed on the value of the entropy, determines the *goodness* of a cluster. A subspace $X$ with $H(X) \leq w$ is considered to have a *good clustering*. A subspace $X = \{A_1, A_2, \cdots, A_p\}$ is *correlated* if $H(X) \neq \sum_{i=1}^{p} H(A_i)$. To measure the degree of correlation a quantity called *interest* is defined as:

$$
\text{int}(\{A_1, \cdots, A_p\}) = \sum_{i=1}^{p} H(A_i) - H(A_1, \cdots A_p).
$$

The algorithm uses a *downward closure* property, stating that if a $k$-dimensional subspace has good clustering, so do all $(k-1)$-dimensional projections of it, and an *upward closure* property, specifying that if a set of dimensions $X$ is correlated, so is every superset of $X$. The downward closure property is a pruning property, since if a subspace does not have this property, all its supersets are crossed out because they will not have the property either. The upward closure property is a constructive property, if a subspace satisfies the property, all its supersets also satisfy the property. The ENCLUS algorithm focuses only on the minimally correlated subspaces. Therefore, the upward closure also becomes a pruning property. Subspaces having good clustering and being minimally correlated are called *significant* subspaces. The pseudocode of the algorithm is given in Figure 2.2.

[CWY99] also presents a different approach to subspace clustering, based on the idea that subspaces of higher dimensionality may be more interesting if
1. Let $k = 1$, $S = \emptyset$.
2. Let $X_k$ be the collection of $k$-dimensional subspaces.
3. For each subspace $X \in X_k$
4. Compute its entropy $H(X)$.
5. If $H(X) < w$, then
6. If $\text{int}(X) > \epsilon$, then $S = S \cup X$.
7. Else $C_k = C_k \cup X$.
8. Compute in $X_{k+1}$ all candidate $(k+1)$-subspaces using $C_k$.
9. If $C_{k+1}$ is empty exit and return $S$.
10. $k = k + 1$. Go to step 3.

Figure 2.2: ENCLUS algorithm

they have a stronger correlation of their subspaces. To measure this increase in correlation, a measure called interest gain is defined as:

$$\text{gain}({A_1, \cdots, A_p}) = \text{int}({A_1, \cdots, A_p}) - \max_i \text{int}({A_1, \cdots, A_p} - \{A_i\})$$

For the one-dimensional subspaces, the interest gain is 0. The new goal is to mine subspaces with entropy less than $w$ and interest gain greater than $\epsilon$. These are called interesting subspaces.

2.3 Clustering databases with categorical attributes

[hua97] introduces a clustering algorithm for categorical data. This new algorithm, called $k$-modes, extends the $k$-means algorithm to categorical data (domain finite and unordered). Two dissimilarities measures are proposed. If $X$, $Y$ are two rows of a database, the dissimilarity between $X = \{x_1, x_2, \cdots, x_m\}$ and $Y = \{y_1, y_2, \cdots y_m\}$ is defined as $\text{dis}_1(X,Y) = \sum_{j=1}^{m} \gamma(x_j, y_j)$, where $\gamma(x_j, y_j)$
is 0 if \( x_j = y_j \), and 1 otherwise. The smaller the number of mismatches, the
more similar the objects are. The measure \( dis_1(X, Y) \) gives equal importance
to each value of an attribute. If we want to take into account the frequencies of each value, then we can define the dissimilarity between \( X \) and \( Y \) as
\[
dis_2(X, Y) = \sum_{j=1}^{m} \frac{n_{x_j} + n_{y_j}}{n_{x_j} n_{y_j}} \gamma(x_j, y_j),
\]
where \( n_{x_j} \) is the number of objects in the data set that have value \( x_j \) for attribute \( j \), and \( n_{y_j} \) is the number of objects in the data set that have value \( y_j \) for attribute \( j \). This dissimilarity measure
gives more importance to rare categories than to frequent ones. Given a set
\( H = A_1 A_2 \cdots A_m \) of categorical attributes, a \textit{mode} is a vector \( Q = [q_1, \cdots q_m] \)
that minimizes the sum \( D(Q, H) = \sum_{i=1}^{n} d(X_i, Q) \), where \( d \) can be either \( dis_1 \)
or \( dis_2 \), and \( \{X_1, X_2, \cdots , X_n\} \) is the set of objects of the database. The general
description of the \textit{k-modes} algorithm is given in Figure 2.3.

1. Select \( k \) initial modes, one for each cluster.
2. Allocate an object to the cluster with the nearest mode and update
the mode of the cluster after each addition.
3. After all objects are allocated to clusters, the dissimilarity between
each object and the current modes is tested again. If an object is found
such that its nearest mode belongs to another cluster, then the object is
reallocated to that cluster and the modes are recomputed.

Figure 2.3: \textit{k-modes} algorithm

As for the \textit{k-means} algorithm, the final clusters depend on the initial choice
of the \( k \) modes.

[HKK97] introduced a clustering algorithm for market basket data (databases
with boolean attributes). In this context the attributes are called items and the
objects are called transactions. This approach is based on clustering related items

25
using association rules ([AMS96]) and clustering related transactions using the clusters of items. The frequent itemsets obtained using the Apriori algorithm ([AMS96]) are used to form a hypergraph, in which vertices correspond to the itemsets, an edge corresponds to all the association rules involving the items from a frequent itemset and the weight of the edge is the average of the confidences of these association rules. This graph is partitioned using HMETIS ([GRV97]) such that the sum of the weight of the edges across partitions is minimized. The idea behind the hypergraph partitioning criterion is the following: items that appear in association rules with greater confidence should be grouped in the same cluster. Given these new sets of items $C_i$ (corresponding to the edges in the new hypergraph) and a transaction $T$, the algorithm computes the ratio $\frac{|\text{m}C_i|}{c_i}$, for all the transactions in the database. A transaction belongs to the cluster $C_i$ yielding the greatest ratio value.

2.4 Similarity and dissimilarity measures

An important issue in the clustering methods is the choice of the similarity or dissimilarity measures between the objects to be clustered. [DMR98] discusses this idea in the context of clustering the rows of a database with boolean attributes and proposes several new measures. The internal measure of similarity between attributes $A$ and $B$ is defined purely in terms of the values in the $A$ and $B$ columns. This kind of measure is used in general in the classical clustering algorithms. The external measure of similarity between attributes $A$ and $B$ is defined in terms of the values in certain other columns called probe columns. Given a binary relation $\rho$ with $n$ rows over the attributes $H$ and a selection condition $\theta$ on the rows of $\rho$, $\rho_\theta$ denotes the fraction of rows of $\rho$ that satisfy $\theta$, and $\rho_A$
is the shorthand notation for $\rho_{A=1}$. An association rule on the relation $\rho$ is an expression $X \rightarrow Y$ with $X, Y \in H$, where $\sup(X \rightarrow Y) = \rho_{X \cup Y}$ is the support and $\text{conf}(X \rightarrow Y) = \frac{\rho_{X \cup Y}}{\rho_X}$ is the confidence of the rule.

An internal measure of similarity $\text{sim}_I$ between attributes $A$ and $B$ is defined as the relative size of the symmetric difference between the rows with $A = 1$ and rows with $B = 1$:

$$\text{sim}_{I,\text{sd}}(A, B) = \frac{\rho_{A=1 \wedge B=0} \vee \rho_{A=0 \wedge B=1}}{\rho_{A=1 \wedge B=1}} = \frac{\rho_A + \rho_B - 2\rho_{AB}}{\rho_A + \rho_B - \rho_{AB}},$$

or in data mining contexts as the confidences of the rules $A \rightarrow B$ and $B \rightarrow A$:

$$\text{sim}_{I,\text{conf}}(A, B) = (1 - \text{conf}(A \rightarrow B)) + (1 - \text{conf}(B \rightarrow A)).$$

Both $\text{sim}_{I,\text{sd}}(A, B)$ and $\text{sim}_{I,\text{conf}}(A, B)$ are metrics on the set of all attributes.

As external measures of similarity, the following measures are proposed. The similarity between attributes $A$ and $B$ is the similarity between the relations $\rho^A = \sigma_{A=1}(\rho)$ and $\rho^B = \sigma_{B=1}(\rho)$, projected on a set $P \subset H$ of probe attributes. These relations can be viewed as defining two multivariate distributions $g_A$ and $g_B$ on $\{0, 1\}^P$. Given $\overline{x} \in \{0, 1\}^P$, the value $g_A(\overline{x})$ is the relative frequency of $\overline{x}$ in the relation $\rho^A$. The Kullback-Leibler distance, known also as relative entropy or cross entropy, can be used as the distance between these distributions:

$$\text{sim}_{E,\text{ce},P}(A, B) = \sum_{\overline{x}} g_A(\overline{x}) \log \frac{g_A(\overline{x})}{g_B(\overline{x})}.$$  

The symmetric version $\text{sim}_{E,\text{ce},P}(A, B) + \text{sim}_{E,\text{ce},P}(B, A)$ could also be used. The $\text{sim}_{E,\text{ce},P}(A, B)$ measure has $2^{\mid P \mid}$ elements to sum, and for larger sets $P$ will be difficult to compute it efficiently. A solution is to consider a single attribute $D \in P$ at a time and define $\text{sim}_{E,F,P}(A, B) = \sum_{D \in P} F(A, B, D)$, where $F(A, B, D)$ measures how closely $A$ and $B$ are with respect to $D$. Different choices for the function $F$ are discussed, $F(A, B, D) = \mid \rho^A_D - \rho^B_D \mid$ being the most robust. The associated
The external measure of the similarity is: \( \text{sim}_{E,F,P}(A, B) = \sum_{D \in P} |\rho_D^A - \rho_D^B| \). The measure \( \text{sim}_{E,F,P}(A, B) \) is a pseudometric, is symmetric, satisfies the triangular inequality but the distance can be 0 even if the two attributes are not identical. A reformulation in terms of confidences is \( \text{sim}_{E,F,P}(A, B) = \sum_{D \in P} |\text{conf}(A \rightarrow D) - \text{conf}(B \rightarrow D)| \). The internal measure of similarity that uses the confidence measure can be rewritten as \( \text{sim}_{I,\text{conf}} = \text{sim}_{E,F,\{A,B\}}(A, B) \).

If \( P = \{D_1, D_2, \cdots, D_k\} \), we can construct the vectors \( v_{A,P} = [\rho_{D_1}^A, \cdots, \rho_{D_k}^A] \) and \( v_{B,P} = [\rho_{D_1}^B, \cdots, \rho_{D_k}^B] \), and alternative ways to define the external measure of similarity are:

1. Using the \( L_p \) metric we can define \( \text{sim}_{E,L_p,P} \) as the \( L_p \) distance between \( v_{A,P} \) and \( v_{B,P} \).

2. \( P \) can be a set of boolean formulae \( \theta_i \) and \( \text{sim}_{E,\text{diff},P} = \sum_i |fr(r_A, \theta_i) - fr(r_B, \theta_i)| \).

3. External measures can also be defined by using internal measures. Given:

\[
    v_{A,P} = [\text{sim}_I(A, D_1), \cdots, \text{sim}_I(A, D_k)]
\]

and

\[
    v_{B,P} = [\text{sim}_I(B, D_1), \cdots, \text{sim}_I(B, D_k)]
\]

then we can construct external measures from internal measures by taking \( \text{sim}_{E,\text{sim}_I,P}(A, B) = d(v_{A,P}, v_{B,P}) \) for any suitable distance \( d \) between the two vectors.
2.5 Hierarchical algorithms

ROCK (RObust Clustering using Links), is a hierarchical clustering algorithm for categorical attributes databases introduced in [GRS00]. Traditional algorithms using distances between points are not appropriate for categorical attributes and a new measure is introduced, based on the notion of links. A pair of objects are neighbors if their similarity exceeds a certain threshold. The similarity measure can be any traditional similarity measure, such as the $L_p$ distance metric or the Jaccard coefficient. The number of links between a pair of objects is given by the number of common neighbors. The main idea of this algorithm is to first merge the clusters/objects with the largest number of links between them. The classical similarities between a pair of objects are local properties, involving only knowledge about the two objects in question, the link notion incorporates global knowledge about the other objects in the neighborhood of the two objects.

For example, in the context of market basket data where we are interested in the similarity between transactions, the Jaccard coefficient can be used to define the similarity between transactions $T_1$ and $T_2$, as $sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$, where $|T_i|$ is the number of items in transaction $T_i$.

If $sim(p_i, p_j)$ is a classical similarity function that capture the closeness between the pair of objects $p_i$ and $p_j$, and the threshold for the similarity (a user defined parameter) is $\theta$, then a pair of points are neighbors if $sim(p_i, p_j) > \theta$.

$\text{link}(p_i, p_j)$ denotes the number of common neighbors between $p_i$ and $p_j$. The goal is to maximize the sum of $\text{link}(p_q, p_p)$ for $p_q, p_p$ in the same cluster and in the same time minimize the sum of $\text{link}(p_q, p_s)$ for $p_q, p_s$ in different clusters. This
leads to maximization of the following criterion:

$$E_i = \sum_{i=1}^{k} n_i \cdot \sum_{p_q, p_r \in C_i} \frac{\text{link}(p_q, p_r)}{n_i^{1+2f(\theta)}}.$$  

where $C_i$ is a cluster of size $n_i$. Assuming that $C_i$ has approximately $n_i^{f(\theta)}$ neighbors in $C_i$, the expected number of links between the pairs of points in $C_i$ is given by the expression $n_i^{1+2f(\theta)}$.

The general description of the algorithm is the following:

1. A random sample is extracted from the input data.

2. This data is clustered using the idea of links between objects.

3. The rest of the input data is assigned to the appropriate clusters found during the previous step.

Let $C_i$ and $C_j$ be two clusters, let $\text{link}[C_i, C_j] = \sum_{p_q \in C_i, p_r \in C_j} \text{link}(p_q, p_r)$ be the number of cross links between the two clusters. Then the goodness measure $g(C_i, C_j)$ for merging the two clusters is:

$$g(C_i, C_j) = \frac{\text{link}[C_i, C_j]}{(n_i + n_j)1+2f(\theta) - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}}.$$

The pair of clusters for which the above measure reaches a maximum value is the best pair of clusters to be merged.

Initially, each object sits in a separate cluster. To each cluster $C_i$, the algorithm associates a local heap $q[i]$ containing every cluster $C_j$ such that $\text{link}[i, j]$ is different than zero (they have sufficient common neighbors). The clusters $C_j$ in $q[i]$ are ordered in decreasing order of the goodness measure with respect to $i$, that is, in decreasing order of $g(i, j)$. A global heap $Q$ is also maintained, containing all the clusters, ordered in decreasing order of their best goodness.
measure \( g(j, \max(q[j])) \). At each step in the algorithm, the best cluster \( C_i \) in \( Q \) and the best cluster in \( q[i] \) represent the best clusters to be merged. The algorithm iterates the procedure from Figure 2.4, until \( k \) clusters remain in \( Q \).

1. Extract the best cluster \( C_u \) from \( Q \).
2. Merge \( C_u \) and the first cluster \( C_v \) from \( q[u] \) (representing the best cluster \( C_v \) to be merged with \( C_u \)). Clusters \( C_u \) and \( C_v \) are merged into a new cluster \( C_w \) containing \( |C_u| + |C_v| \) objects.
3. For every cluster heap that contains \( C_u \) or \( C_v \), the following adjustments are necessary. \( C_u \) or \( C_v \) are replaced with \( C_w \), and the heap is updated.
4. A new local heap is created for \( C_w \), and \( C_w \) is added to \( Q \).

Figure 2.4: ROCK main procedure

2.6 Median partition

Given a set (profile) of partitions \( \pi = (\pi_1, \pi_2, \cdots, \pi_n) \), [BL95] describes various procedures to find the consensus partition, a partition \( \pi^* \) that summarizes the partitions in the profile \( \pi \) in some useful way. An axiomatic approach to this problem is based on imposing some constraints (axioms) to the consensus partition. For example the constraint that two objects belong to the same class in the consensus partition if they belong to the same class in all the partitions in the profile; for this approach the partition determined by the intersections of all the partitions in the profile is a consensus partition. A constructive approach specifies a way to construct the consensus partition. A combinatorial optimization approach is one where a criteria to measure the dissimilarity of two
partitions $\text{dis}(\pi, \pi')$ is available and we want to find a partition that minimizes $\sum_{\pi_i \in \pi} \text{dis}(\pi_i, \pi')$. In this last case the resulting partition is called a median partition. The dissimilarity between two partitions can be defined as the cardinality of the symmetrical distance between the equivalence relations associated with the two partitions.

[Reg83b] presents this problem of finding a median partition, called in their article the central partition, that is, a partition of a set of objects that reflects as well as possible the similarity between them. Each partition of the set of objects defines a characteristic matrix $C(\pi) = (c_{ij}^\pi) \in \mathbb{R}^{n \times n}$, where $c_{ij}^\pi = 1$ if objects $i$ and $j$ belong to the same class of $\pi$, and 0 otherwise. Between two matrices $C(\pi) = (c_{ij}^\pi)$ and $C(\sigma) = (c_{ij}^\sigma)$ associated with two partitions $\pi$ and $\sigma$, we can define a distance measure as $d(C(\pi), C(\sigma)) = \sum_{i=1}^n \sum_{j=1}^n (c_{ij}^\pi - c_{ij}^\sigma)^2$. Each partition in the profile of partitions $\pi = (\pi_1, \pi_2, \cdots, \pi_p)$ determines a matrix $C(\pi_i)$. Let $C(\pi) = \sum_{i=1}^p C(\pi_i)$. The median partition is a partition $\pi^*$ with the characteristic matrix $C(\pi^*) = (x_{ij}) \in \mathbb{R}^{n \times n}$ that minimizes the quantity $\sum_{i=1}^p d(C(\pi^*), C(\pi))$.

It can be shown (see [Reg83b]), that this problem is equivalent with maximizing the quantity $E(X) = \sum_{i=1}^n \sum_{j=1}^n (\sum_{i=1}^p c_{ij}^\pi - \frac{1}{2})x_{ij}$. The algorithm for finding the median partition, called “transfer objects algorithm”, starts with a randomly generated partition $\pi^*$ and constructs a new partition in the neighborhood of $\pi^*$, by transferring an object from one class to another. At each iteration, the algorithm selects the object that maximizes the quantity $E(X)$. Because $x_{ij} = 1$ if objects $i$ and $j$ belong to the same class $c$ of the partition $\pi$, the quantity $E(X)$ can be rewritten as $E(X) = \sum_c \sum_{i,j \in c} (\sum_{i=1}^p c_{ij}^\pi - \frac{1}{2})$. If object $i$ is selected to migrate from class $c_1$ to class $c_2$, then the difference in the values of the quantity $E(X)$ is given by $\Delta E(X) = \sum_{j \in c_2} (\sum_{i=1}^p c_{ij}^\pi - \frac{1}{2}) - \sum_{j \in c_1} (\sum_{i=1}^p c_{ij}^\pi - \frac{1}{2})$. If we denote by $A_i^c$ the quantity $\sum_{j \in c} (c_{ij}^\pi - \frac{1}{2})$, called the attraction of object $i$ by the class $c$, then $\Delta E(X) = A_i^{c_2} - A_i^{c_1}$. If object $i$ is the one that
maximizes $\Delta E(X)$, then the partition $\pi$ and its associated matrix $X$ are updated to reflect the change of the class of object $i$: $A_i^{c_1} = A_i^{c_1} - \sum_{j \in c_1} \sum_{t=1}^{P} (c_{ij} - \frac{1}{2})$ and $A_i^{c_2} = A_i^{c_2} + \sum_{j \in c_2} \sum_{t=1}^{P} (c_{ij} - \frac{1}{2})$.

The median partition and its properties are presented also in [Reg83a].

### 2.7 Genetic algorithms for grouping problems

Genetic algorithms are also used in clustering applications. The GAs have the advantage of combining in an optimal way the exploitation of the good regions in the search space with the exploration of the unseen portions of the space. They perform a global search, whereas the other clustering methods perform a localized search, by looking only in the neighborhood of the optimal points found at one moment. A presentation of the genetic algorithms design for grouping problems is given in [Fal99].

[Mic99] introduces a modification of the classical genetic algorithm to solve partitioning problems, where we have a set of $n$ objects and we want to partition them into $k$ categories. In these problems, the chromosomes are encoded using the group number encoding, where a partition of the $n$ objects into $k$ classes is represented as a n-string of integers $(i_1, i_2, \cdots, i_n)$ with $i_j \in \{1, 2, \cdots, k\}$. Crossover and mutation operators have been designed to include knowledge about the domain of the application. In a structural crossover, one randomly chosen class from the first chromosome is copied into the second chromosome, and vice versa. A structural mutation performs a swap between two randomly chosen elements of the string representation of a chromosome. An edge-based crossover uses the idea that objects in the same class of the partition form a clique in the graph having as vertices the $n$ objects and edges between any two objects in
the same class. An edge-based crossover generates new offspring by combining edges from the two parent chromosomes, trying to meet as well as possible the requirement that there is an edge in the offspring only if that edge belongs in at least one of the parents.

Other chromosomal representation for the problem of partitioning $n$ objects into $k$ classes, uses a string of $n + k - 1$ integers, such that integers $\{1, 2, \cdots n\}$ represent the objects and integers $\{n + 1, \cdots n + k - 1\}$ represent separators between the classes of the partition. For this type of representation special care is given to the manipulation of separators, for example the mutation operator will swap two elements only if they are not separators.

Other genetic algorithms (introduced in [Fal99] and also presented in [Mic99]), designed for clustering problems, use a representation of the chromosomes that has two parts, an object part where the group number encoding is used to map each object to the class to which it belongs and a group part, that enumerates the group numbers that appear in the object part. The crossover operator works similarly to the classical two-point crossover with the difference that it is applied only to the group part. After new groups are determined for the offspring, the object part is modified to reflect the new partitioning. The mutation operator also works with the group part of the chromosome and randomly selects and eliminates some groups.

More information about clustering applications involving genetic algorithms can be found in [EM97], [Col98] and [CG00].
CHAPTER 3

Generalized entropy

Traditionally, measures like the Shannon entropy or the Gini index are used for assessing the clustering quality. Both measures and many others, can be obtained with the help of a concave, sub-additive function, which allows us to generalize the notion of entropy and conditional entropy. In this chapter, we introduce these new information-theoretical measures and their properties. They will play an essential role in the clustering methods we present in subsequent chapters.

3.1 Impurity measures

Let $\mathbb{R}$ be the set of reals. The $k$-dimensional simplex is the set:

$$\text{SIMPLEX}_{k-1} = \{(p_1, \ldots, p_k) \in \mathbb{R}^k \mid p_i \geq 0 \text{ and } p_1 + \cdots + p_k = 1\}.$$ 

A function $f : \mathbb{R} \to \mathbb{R}$ is concave on a set $R \subseteq \mathbb{R}$ if

$$f(\alpha x + (1 - \alpha)y) \geq \alpha f(x) + (1 - \alpha)f(y),$$

for $\alpha \in [0, 1]$ and $x, y \in R$. The function $f$ is sub-additive (supra-additive) on $R$ if

$$f(x + y) \leq f(x) + f(y) \quad (f(x + y) \geq f(x) + f(y)),$$

for $x, y \in R$. 

35
In [CHH99] a concave impurity measure is defined as a real-valued function $i : \text{SIMPLEX}_{k-1} \to \mathbb{R}$ that satisfies the following conditions:

(i) $i(\alpha p + (1 - \alpha)q) \geq \alpha i(p) + (1 - \alpha)i(q)$, for any $\alpha \in [0, 1]$, and $p, q \in \text{SIMPLEX}_{k-1}$, with equality if and only if $p = q$;

(ii) if $p = (p_1, \ldots, p_k)$, then $i(p) = 0$ if $p_i = 1$ for some $i$, $1 \leq i \leq k$.

The corresponding frequency-weighted impurity measure is the real-valued function $I : \mathbb{N}^k \to \mathbb{R}$ given by $I(n_1, \ldots, n_k) = Ni(n_1/N, \ldots, n_k/N)$, where $N = \sum_{i=1}^{k} n_i$.

Both the Gini index and the Shannon entropy can be generated using a simple one-argument concave function $f : [0, 1] \to \mathbb{R}$, such that $f(0) = f(1) = 0$.

**Definition 3.1** A generator is a concave, sub-additive function $f : [0, 1] \to \mathbb{R}$, such that $f(0) = f(1) = 0$ and

$$f(\theta p_1) + \cdots + f(\theta p_n) \leq \theta(f(p_1) + \cdots + f(p_n)) + f(\theta),$$  \hspace{1cm} (3.1)

for every $(p_1, \ldots, p_n) \in \text{SIMPLEX}_{n-1}$ and $\theta \in [0, 1]$.

The impurity measure induced by $f$ is $i(p_1, \ldots, p_k) = f(p_1) + \cdots + f(p_k)$, for every $(p_1, \ldots, p_k) \in \text{SIMPLEX}_{k-1}$. \hfill \Box

It is easy to verify that such functions as $f_{\text{gin}}(p) = p - p^2$ (the Gini index), $f_{\text{ent}}(p) = -p \log p$ (the Shannon entropy), $f_{\text{ge}}(p) = -(p - p^2) \cdot \log(p - p^2)$, or $f_{\text{peak}}$ given by:

$$f_{\text{peak}}(p) = \begin{cases} p, & \text{if } 0 \leq p \leq 0.5 \\ 1 - p, & \text{if } 0.5 < p \leq 1 \end{cases}$$

are generators.
Since $f$ is concave it satisfies Jensen's inequality $f(p_1) + \cdots + f(p_k) \leq kf(\frac{1}{k})$, for every $(p_1, \ldots, p_k) \in \text{SIMPLEX}_{k-1}$, and this implies that the largest value of the sum $f(p_1) + \cdots + f(p_k)$ is achieved if and only if $p_1 = \cdots = p_k = \frac{1}{k}$. Therefore, for the impurity measure generated by the function $f$ we have $0 \leq i(p_1, \ldots, p_k) \leq kf(\frac{1}{k})$, for $(p_1, \ldots, p_k) \in \text{SIMPLEX}_{k-1}$.

### 3.2 Generalized entropy of sets and partitions

Let $\text{PART}(R)$ be the set of all partitions of a set $R$. For any two partitions $\pi, \sigma \in \text{PART}(R)$, we write $\pi \leq \sigma$ if every class of $\sigma$ is a union of classes of $\pi$. We denote by $\omega_R$ the one-class partition, and by $\alpha_R$ the partition with one-element classes. $|\pi|$ represents the number of classes in the partition $\pi$.

Let $\mathcal{U}$ be a set whose elements are referred to as attributes. We assume that for every element $A$ of $\mathcal{U}$ there is a set denoted by $\text{Dom}(A)$ referred to as the domain of $A$ such that $|\text{Dom}(A)| \geq 2$.

A table is a function $T : R \times H \rightarrow \bigcup D_H$, where $R$, a finite set, is referred to as the set of rows of $T$, $H$ is a finite subset of $\mathcal{U}$ (called the heading of $T$), $D_H = \{\text{Dom}(A) \mid A \in H\}$, and $T(r, A) \in \text{Dom}(A)$, for every $r \in R$ and $A \in H$.

The projection of the table $T : R \times H \rightarrow \bigcup D_H$ on the set of attributes $L \subseteq H$ is the table $T[L] : R \times L \rightarrow \bigcup D_L$ given by $T[L](r, A) = T(r, A)$, for every $r \in R$ and $A \in L$. If $r$ is a row of the table $T$, then the set $\{T(r, A) \mid A \in H\}$ is the content of the row $r$. This definition of a table allows the existence of multiple rows having the same content. The projection of the row $r$ of the table $T$ on $L$ is the set $r[L] = \{T(r, A) \mid A \in L\}$. For $A \in H$, $Q_T[A] = \{T(r, A) \mid r \in R\}$ represents the content of the column associated with the attribute $A$ in the table $T$. 

37
The active domain of a set of attributes \( L \) in the table \( T \) is the set \( \text{aDom}_T(L) = r|L| \). The cardinality of the active domain of \( L \) in the table \( T \) is denoted by \( a_L^T \), or, when there is no risk for confusion, just by \( a_L \).

Let \( T : R \times H \longrightarrow \bigcup D_H \) be a table with \( n \) attributes and \( H = A_1 A_2 \ldots A_n \). Every subset of attributes \( L \subseteq H \) determines a partition \( \pi_L^T = \{ D_1, \ldots, D_{a_L} \} \) of the set \( R \) of rows in table \( T \), where \( \text{aDom}_T(L) = \{ v_1, \ldots, v_{a_L} \} \) and \( D_i = \{ r \in R \mid T(r, L) = v_i \} \), for \( 1 \leq i \leq a_L \). Clearly, if \( L = \{ A_{i_1}, \ldots, A_{i_\ell} \} \), then \( \pi_L^T = \bigcap \{ \pi_{A_{i_j}}^T \mid 1 \leq j \leq \ell \} \).

**Definition 3.2** Let \( f \) be a generator, \( R \) be the set of rows of a table \( T \), \( L \) be a subset of \( R \) and let \( \pi = \{ B_1, \ldots, B_n \}, \sigma = \{ C_1, \ldots, C_m \} \) be two partitions of \( R \).

The **\( f \)-entropy of partition \( \pi \)** is the quantity:

\[
\mathcal{H}_f(\pi) = f \left( \frac{|B_1|}{|R|} \right) + \cdots + f \left( \frac{|B_n|}{|R|} \right).
\]

The **\( f \)-impurity of a subset \( L \) of \( R \) relative to a partition \( \pi \)** is the quantity:

\[
\text{IMP}_f(\pi)(L) = |L| \left( f \left( \frac{|L \cap B_1|}{|L|} \right) + \cdots + f \left( \frac{|L \cap B_n|}{|L|} \right) \right).
\]

The **specific \( f \)-impurity of a subset \( L \) of \( R \) relative to a partition \( \pi \)** is the quantity:

\[
\text{imp}_f(\pi)(L) = f \left( \frac{|L \cap B_1|}{|L|} \right) + \cdots + f \left( \frac{|L \cap B_n|}{|L|} \right).
\]

The **\( f \)-entropy of \( \pi \) relative to \( \sigma \)** is the quantity:

\[
\mathcal{H}_f(\pi|\sigma) = \sum_{j=1}^{m} \frac{|C_j|}{|R|} \text{IMP}_f(\pi)(C_j) = \frac{1}{|R|} \sum_{j=1}^{m} |C_j| \sum_{i=1}^{n} f \left( \frac{|B_i \cap C_j|}{|C_j|} \right).
\]

This quantity is also called the **conditional \( f \)-entropy of \( \pi \) relative to \( \sigma \)**.
Note that Jensen’s inequality implies that the largest value of the $f$-impurity of a set $L \subseteq R$ relative to a partition $\pi = \{B_1, \ldots, B_n\}$ of $R$ is $|L| \cdot n \cdot f\left(\frac{1}{n}\right)$, and the largest value of the $f$-entropy of the partition $\pi$ is $n \cdot f\left(\frac{1}{n}\right)$.

The conditional $f$-entropy $\mathcal{H}^f(\pi | \sigma)$ is the average value of the specific $f$-impurity of the classes of the partition $\sigma$ relative to the partition $\pi$.

Note that $\mathcal{H}^f(\pi | \omega_R) = \mathcal{H}^f(\pi)$ for every $\pi \in \text{PART}(R)$. We have $\sigma \leq \pi$, if and only if $\mathcal{H}^f(\pi | \sigma) = 0$. Thus, for any partition of the set $R$ we have $\mathcal{H}^f(\pi | \alpha_R) = 0$ and $\mathcal{H}^f(\omega_R | \pi) = 0$.

**Theorem 3.1** Let $\pi = \{B_1, \ldots, B_n\} \in \text{PART}(R)$ and $K, L$ two disjoint subsets of the set $R$. Then, we have:

$$\text{IMP}^f_\pi(K \cup L) \geq \text{IMP}^f_\pi(K) + \text{IMP}^f_\pi(L).$$

**Proof.**

The disjointness of $K$ and $L$ allows us to write

$$\text{imp}^f_\pi(K \cup L) = \sum_{t=1}^{n} f\left(\frac{|(K \cup L) \cap B_t|}{|K \cup L|}\right) = \sum_{t=1}^{n} f\left(\frac{|K \cap B_t| + |L \cap B_t|}{|K \cup L|}\right).$$

Since $\frac{|K \cap B_t| + |L \cap B_t|}{|K \cup L|}$ is a convex combination of $\frac{|K \cap B_t|}{|K|}$ and $\frac{|L \cap B_t|}{|L|}$, the concavity of $f$ implies

$$f\left(\frac{|K \cap B_t| + |L \cap B_t|}{|K \cup L|}\right) \geq \frac{|K|}{|K| + |L|} f\left(\frac{|K \cap B_t|}{|K|}\right) + \frac{|L|}{|K| + |L|} f\left(\frac{|L \cap B_t|}{|L|}\right),$$

so $\text{imp}^f_\pi(K \cup L) \geq \frac{|K|}{|K| + |L|} \text{imp}^f_\pi(K) + \frac{|L|}{|K| + |L|} \text{imp}^f_\pi(L)$, which gives immediately the desired inequality.

**Corollary 3.2** Let $\pi = \{B_1, \ldots, B_n\} \in \text{PART}(R)$ and $K, L$ two subsets of $R$ such that $K \subseteq L$, then:

$$\text{IMP}^f_\pi(K) \leq \text{IMP}^f_\pi(L).$$
Proof.

Let $H = L - K$. By Theorem 3.1, since $K, H$ are disjoint, we have:

$$\text{IMP}_\pi^f(L) = \text{IMP}_\pi^f(K \cup H) \geq \text{IMP}_\pi^f(K) + \text{IMP}_\pi^f(H),$$

so $\text{IMP}_\pi^f(L) \geq \text{IMP}_\pi^f(K)$.

**Theorem 3.3** Let $\pi = \{B_1, \ldots, B_n\}$ and $\sigma = \{C_1, \ldots, C_m\}$ be two partitions of a set $R$. If $\pi \leq \sigma$, then for every subset $K$ of $R$ we have:

$$\text{IMP}_\sigma^f(K) \leq \text{IMP}_\pi^f(K).$$

**Proof.**

Since $\pi \leq \sigma$ every block $C_j$ of $\sigma$ is the union of some blocks of the partition $\pi$, $C_j = \bigcup \{B_h \mid h \in H_j\}$, where $H_1, \ldots, H_m$ is a partition of the set $\{1, \ldots, n\}$. The sub-additivity of $f$ implies

$$\text{imp}_\sigma^f(K) = \sum_{j=1}^m f\left(\frac{|K \cap C_j|}{|K|}\right) \leq \sum_{j=1}^m \sum_{h \in H_j} f\left(\frac{|K \cap B_h|}{|K|}\right) = \text{imp}_\pi^f(K),$$

This yields immediately the inequality of the theorem.

**Theorem 3.4** Let $f$ be a generator and let $\pi = \{B_1, \ldots, B_n\} \in \text{PART}(R)$ and $\sigma = \{C_1, \ldots, C_m\} \in \text{PART}(R)$. We have:

$$\mathcal{H}^f(\pi) \geq \mathcal{H}^f(\pi | \sigma) \geq \mathcal{H}^f(\pi \wedge \sigma) - \mathcal{H}^f(\sigma).$$

**Proof.**

To prove the first inequality observe that we can write

$$\frac{|B_i|}{|R|} = \sum_{j=1}^m c_j \frac{|B_i \cap C_j|}{|C_j|},$$


where \( c_j = \frac{|C_j|}{|R|} \) for \( 1 \leq j \leq m \) and \((c_1, \ldots, c_m) \in \text{SIMPLEX}_{m-1}\). The concavity of \( f \) implies
\[
f \left( \frac{|B_i|}{|R|} \right) \geq \sum_{j=1}^{m} c_j f \left( \frac{|B_i \cap C_j|}{|C_j|} \right),
\]
which, in turn, yields
\[
\mathcal{H}^f(\pi | \sigma) = \sum_{i=1}^{n} \sum_{j=1}^{m} c_j f \left( \frac{|B_i \cap C_j|}{|C_j|} \right) \leq \sum_{i=1}^{n} \sum_{j=1}^{m} f \left( \frac{|B_i|}{|R|} \right) = \mathcal{H}^f(\pi).
\]

For a fixed \( j \) let \( b_i = \frac{|B_i \cap C_j|}{|C_j|} \) and \( \theta = \frac{|C_j|}{|R|} \). Since \((b_1, \ldots, b_n) \in \text{SIMPLEX}_{n-1}\) we have
\[
\sum_{i=1}^{n} f \left( \frac{|B_i \cap C_j|}{|R|} \right) \leq |C_j| \sum_{i=1}^{n} f \left( \frac{|B_i \cap C_j|}{|C_j|} \right) + f \left( \frac{|C_j|}{|R|} \right).
\]
Summing these inequalities for \( 1 \leq j \leq m \) and observing that the classes of the partition \( \pi \land \sigma \) have the form \( B_i \cap C_j \) we obtain the second inequality of the theorem.

**Theorem 3.5** If \( \pi_1, \pi_2, \sigma \in \text{PART}(R) \) are such that \( \pi_1 \leq \pi_2 \), then:
\[
\mathcal{H}^f(\pi_1 | \sigma) \geq \mathcal{H}^f(\pi_2 | \sigma).
\]

If \( \pi, \sigma_1, \sigma_2 \in \text{PART}(R) \) are such that \( \sigma_1 \leq \sigma_2 \), then:
\[
\mathcal{H}^f(\pi | \sigma_1) \leq \mathcal{H}^f(\pi | \sigma_2).
\]

**Proof.**

Let \( \pi_1, \pi_2 \in \text{PART}(R) \) be such that \( \pi_1 \leq \pi_2 \). If \( \pi_2 = \{B_1, \ldots, B_n\} \), then for each class \( B_i \) of \( \pi_2 \) there exists a set of classes \( \mathfrak{B}_i = \{B_{i_1}, \ldots, B_{i_{k_i}}\} \) of \( \pi_1 \) such that \( B_i = B_{i_1} \cup \cdots \cup B_{i_{k_i}} \). Therefore, \( B_i \cap C_j = \bigcup\{B_{i_\ell} \cap C_j| 1 \leq \ell \leq k_i\} \), and the sub-additivity of \( f \) implies
\[
f \left( \frac{|B_i \cap C_j|}{|C_j|} \right) \leq \sum_{\ell=1}^{k_i} f \left( \frac{|B_{i_\ell} \cap C_j|}{|C_j|} \right).
\]
In turn, this implies $\mathcal{H}^f(\pi_1|\sigma) \geq \mathcal{H}^f(\pi_2|\sigma)$.

Let now $\sigma_1, \sigma_2 \in \text{PART}(R)$ be two partitions such that $\sigma_1 \leq \sigma_2$. Denote by $\mathcal{C}_j$ the subset of $\sigma_1$ that contains the classes included in the class $C_j$ of $\sigma_2$, where $\mathcal{C}_j = \{C_{j_1}, \ldots, C_{j_{h_j}}\}$. Note that we can write

$$\frac{|B_i \cap C_j|}{|C_j|} = c_1 \frac{|B_i \cap C_{j_1}|}{|C_{j_1}|} + \cdots + c_{h_j} \frac{|B_i \cap C_{j_{h_j}}|}{|C_{j_{h_j}}|},$$

where $c_p = \frac{|C_{j_p}|}{|C_j|}$ for $1 \leq p \leq h_j$. The concavity of the function $f$ implies

$$f \left( \frac{|B_i \cap C_j|}{|C_j|} \right) \geq \sum_{p=1}^{h_j} \frac{|C_{j_p}|}{|C_j|} f \left( \frac{|B_i \cap C_{j_p}|}{|C_{j_p}|} \right),$$

which implies the second inequality of the theorem.

**Corollary 3.6** If $\pi_1, \pi_2, \sigma \in \text{PART}(R)$ are such that $\pi_1 \leq \pi_2$, then:

$$\mathcal{H}^f(\pi_1) \geq \mathcal{H}^f(\pi_2).$$

**Proof.**

This statement follows from Theorem 3.5 by taking $\sigma = \omega_M$.

**Corollary 3.7** For all partitions $\pi_1, \ldots, \pi_n \in \text{PART}(R)$ we have:

$$\frac{1}{n} \sum_{i=1}^{n} \mathcal{H}^f(\pi_i) \leq \mathcal{H}^f \left( \bigwedge_{i=1}^{n} \pi_i \right) \leq \sum_{i=1}^{n} \mathcal{H}^f(\pi_i).$$

**Proof.**

The first inequality follows from Corollary 3.6; the second follows from Theorem 3.4.

**Corollary 3.8** Let $X, X', Y, Y'$ be sets of attributes of the table $T$. For every generator $f$ we have:
1. $X \subseteq X'$ implies $\mathcal{H}(\pi_X|\pi_Y) \leq \mathcal{H}(\pi_{X'}|\pi_Y)$, for every set of attributes $Y$;

2. $Y \subseteq Y'$ implies $\mathcal{H}(\pi_X|\pi_Y) \geq \mathcal{H}(\pi_X|\pi_{Y'})$, for every set of attributes $X$.

Proof.

$X \subseteq X'$ implies $\pi_{X'} \leq \pi_X$, $Y \subseteq Y'$ implies $\pi_{Y'} \leq \pi_Y$ and thus, the inequalities follow from Theorem 3.5.

Corollary 3.9 If $X, X'$ are two sets of attributes of the table $T$, then $X \subseteq X'$ implies $\mathcal{H}(\pi_X) \leq \mathcal{H}(\pi_{X'})$.

Proof.

$X \subseteq X'$ implies $\pi_{X'} \leq \pi_X$ and this statement follows immediately from Theorem 3.5.

Corollary 3.10 For every sets of attributes $X, Y \in H$ we have $\mathcal{H}(\pi_{XY}) \leq \mathcal{H}(\pi_X) + \mathcal{H}(\pi_Y)$.

Proof.

$\pi_{XY} = \pi_X \land \pi_Y$ and the inequality follows immediately from Theorem 3.4.

Theorem 3.11 Let $\pi_1, \pi_2, \sigma \in \text{PART}(R)$ and let $f$ be a generator. Then,

$$\mathcal{H}(\pi_1 \land \pi_2|\sigma) \leq \mathcal{H}(\pi_1|\pi_2 \land \sigma) + \mathcal{H}(\pi_2|\sigma).$$

Proof.

Suppose that $\pi_1 = \{B_1, \ldots, B_n\}$, $\pi_2 = \{D_1, \ldots, D_k\} \in \text{PART}(R)$, and $\sigma = \{C_1, \ldots, C_m\} \in \text{PART}(R)$. Applying the property of generators described by Equation (3.1) for

$$\left( \frac{|B_1 \cap D_t \cap C_j|}{|D_t \cap C_j|}, \ldots, \frac{|B_n \cap D_t \cap C_j|}{|D_t \cap C_j|} \right) \in \text{SIMPLEX}_{n-1}$$

43
and \( \theta = \frac{|D_0 \cap C_1|}{|C_1|} \), we obtain the inequalities
\[
\sum_{i=1}^{n} f\left( \frac{|B_i \cap D_{\ell} \cap C_j|}{|C_j|} \right) \leq \frac{|D_{\ell} \cap C_j|}{|C_j|} \cdot \sum_{i=1}^{n} f\left( \frac{|B_i \cap D_{\ell} \cap C_j|}{|D_{\ell} \cap C_j|} \right) + f\left( \frac{|D_{\ell} \cap C_j|}{|C_j|} \right)
\]
for \( 1 \leq \ell \leq k \) and \( 1 \leq j \leq m \). Summing these inequalities for \( \ell \) and \( j \) yields the desired inequality since the classes of the partition \( \pi_2 \land \sigma \) have the form \( D_{\ell} \cap C_j \).

\[ \blacksquare \]

**Corollary 3.12** Let \( \pi_1, \ldots, \pi_n, \sigma \in \text{PART}(M) \) and let \( f \) be a generator. Then,
\[
\mathcal{H}^f\left( \bigwedge_{i=1}^{n} \pi_i | \sigma \right) \leq \sum_{i=1}^{n} \mathcal{H}^f\left( \pi_i \bigwedge_{j=i+1}^{n} \pi_j \land \sigma \right).
\]

**Proof.**

The inequality of the corollary follows immediately from Theorem 3.5.

\[ \blacksquare \]

### 3.3 Dissimilarity measures

A **dissimilarity on a set** \( S \) is a function \( d : S \times S \rightarrow \mathbb{R} \) such that \( d(x, y) \geq 0 \), \( d(x, x) = 0 \) and \( d(x, y) = d(y, x) \) for every \( x, y \in S \).

The dissimilarity \( d \) is said to be **definite** if for all \( x, y \in S \), \( d(x, y) = 0 \) implies \( x = y \). If a dissimilarity satisfies the triangular inequality \( d(x, y) + d(y, z) \geq d(x, z) \) for \( x, y, z \in S \), then we obtain the familiar notion of metric on \( S \).

If \( f \) is a generator, then the mapping \( d^f : \text{PART}(R) \times \text{PART}(R) \rightarrow \mathbb{R} \) given by \( d^f(\pi, \sigma) = \mathcal{H}^f(\pi | \sigma) + \mathcal{H}^f(\sigma | \pi) \) for \( \sigma, \pi \in \text{PART}(R) \), is clearly a definite dissimilarity on \( \text{PART}(R) \). When \( \pi \) is close to \( \sigma \), meaning that their classes have many elements in common, then both \( \mathcal{H}^f(\pi | \sigma) \) and \( \mathcal{H}^f(\sigma | \pi) \) are close to 0, so \( d^f(\pi, \sigma) \) is close to 0.

Thus, the information-theoretical measures allow us to define a class of dissimilarities between partitions of a set \( R \), parameterized by the generator function.
This dissimilarity measure plays an important role in the clustering methods discussed in chapters 6 and 7.
CHAPTER 4

Projection clustering

Clustering categorical databases poses challenges that are quite distinct from numerical data clustering (see [AGG98] and [CWY99]). Categorical data can be encoded numerically; however, this transformation introduces artificial total orderings on the domains of the attributes involved, which have the potential to create artificial agglomeration, meaningless from a practical point of view. We present a new clustering algorithm for categorical databases for which no artificial transformation of the input database is necessary. This clustering approach is based on identifying projections (sets of attributes) that have significant concentration of values, determined by limiting the value of the generalized entropy associated with the set of attributes. Each projection represents a clustering of the set of tuples into a number of clusters. These clusters are represented by the sets of tuples, formed by grouping on the attributes belonging to the projection. It is desired that the tuples from a cluster do not differ too much on the remaining attributes; this requirement can be quantified by using the value of a generalization of conditional entropy.

4.1 Preliminaries

Let \( T : R \times H \rightarrow \bigcup D_H \) be a table, \( X \) be a set of attributes, \( X \subseteq H \), and assume that \( \text{aDom}_T(X) \) consists of \( N \) elements, \( \text{aDom}_T(X) = \{x_1, \ldots, x_N\} \). Let
\( \pi_X = \{B_1, B_2, \ldots, B_N\} \) be the partition determined by the set of attributes \( X \). Each block \( B_i \) of \( \pi_X \) corresponds to a value \( x_i \) of the active domain of \( X \) and \( x_i \) appears under \( X \) with the frequency \( \frac{|B_i|}{|R|} \) for \( 1 \leq i \leq N \).

Let \( d_{T,X} \) be a distance defined on \( \text{aDom}_T(X) \). The average distance between the elements of \( \text{aDom}_T(X) \) is

\[
E(d_{T,X}) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \{d_{T,X}(x_i, x_j)|B_i| \cdot |B_j|\}}{|R|^2}.
\]

The next result connects one of the generalizations of entropy with the average distance and serves as a foundation for the applications of information-theoretical methods to clustering.

Let \( d_{T,X} \) be the distance on \( \text{aDom}_T(X) \) defined by:

\[
d_{T,X}^{01}(x, x') = \begin{cases} 
0 & \text{if } x = x' \\
1 & \text{otherwise}
\end{cases}
\]

for \( x, x' \in \text{aDom}_T(X) \).

**Theorem 4.1** The average distance \( E(d_{T,X}^{01}) \) equals \( \mathcal{H}_{I^{g_{aw}}}(\pi_X) \).

**Proof.**

The definition of the distance \( d_{T,X}^{01} \) implies that the average distance \( E(d_{T,X}^{01}) \) can be written as:

\[
E(d_{T,X}^{01}) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \{|B_i| \cdot |B_j| \mid 1 \leq i, j \leq N, i \neq j\}}{|R|^2} = \frac{|R|^2 - \sum_{i=1}^{N} \{|B_i|^2 \mid 1 \leq i \leq N\}}{|R|^2} = \mathcal{H}_{I^{g_{aw}}}(X),
\]

which gives the desired equality.  

Theorem 4.1 suggests that by limiting the entropy of the partition determined by a set of attributes \( X \), we limit the average distance between the projections
of the tuples of the table on $X$ and, therefore, increase our chances to find large clusters of tuples based on their similarity on $X$.

A projection of a table $T$ on a set of attributes $X$, which determines a partition $\pi_X$ of the set of rows, presents an interest for us if it satisfies the following criterion:

- The $f$-entropy $\mathcal{H}^f(\pi_X)$, referred to as the \textit{internal entropy} of $X$ is limited. This ensures that the values of the $X$-projections congregate towards a small number of clusters.

It is desired also that the following criterion is satisfied:

- The impurity of each of the blocks of the partition $\pi_X$ relative to the partition $\pi_{H-X}$ is small. This ensures that the tuples created by clustering the $X$-projection do not diverge excessively on the remaining attributes of the table. Such sets of attributes can be identified by limiting the value of the conditional $f$-entropy $\mathcal{H}^f(\pi_{H-X}|\pi_X)$, referred to as the \textit{external entropy} of $X$.

Our algorithm will enforce only a limitation of the value of the internal entropy. Experimental results show that some of the projections found by our algorithm are also characterized by a small value of their external entropy.

\textbf{Definition 4.1} Let $T : R \times H \rightarrow \bigcup D_H$ be a table.

An \textbf{$(f, \alpha)$-clusterable set of attributes in $T$} is a set $X \subseteq H$ such that $\mathcal{H}^f(\pi_X) \leq \alpha$ and $X$ is maximal with this property, that is, $X \subseteq Y$ implies $\mathcal{H}^f(\pi_Y) > \alpha$.

The \textbf{clusters determined by the $(f, \alpha)$-clusterable set} $X$ are the groups of records that have equal values on $X$. 

48
4.2 Projection clustering algorithm

We present a new algorithm for identifying all clusterable sets of attributes of a given categorical database. The algorithm is based on the monotonicity of the $f$-entropy shown in Corollary 3.9. Namely, if $U$ is a set of attributes such that $\mathcal{H}^f(\pi_U) > \alpha$, then for every set $V$ such that $U \subseteq V$ we have $\mathcal{H}^f(\pi_V) > \alpha$. Thus, once a set $U$ is disqualified from being $(f, \alpha)$-clusterable, no superset of $U$ can be $(f, \alpha)$-clusterable.

Note that if $X \subseteq X'$, then, by Theorem 3.8, we have $\mathcal{H}^f(\pi_{H-X|X}) \geq \mathcal{H}(\pi_{H-X'|X'})$. Thus, in order to achieve a small value for the external entropy, we focus on the maximal $(f, \alpha)$-clusterable sets of attributes.

It is easy to see that if an attribute set partitions the set of tuples in $k$ blocks, then its maximum entropy would be $1 - \frac{1}{k}$ for the $f_{\text{gini}}$-entropy, $\log(k)$ for the $f_{\text{ent}}$-entropy, $\frac{1-w}{n} \log\left(\frac{n-w}{n}\right)$ for the $f_{\text{ge}}$-entropy, and in general $kf\left(\frac{1}{k}\right)$ for an $f$-entropy. Given a specific value $k$, we compute the maximum $f$-entropy of a partition with $k$ blocks, and then search for all maximal sets of attributes that have the entropy smaller than the maximum value computed previously. Thus, we obtain all sets of attributes that can be clustered in approximately $k$ blocks.

Our algorithm has three input parameters: the number of blocks $k$ in the partition determined by the sets of attributes, a percentage of the attributes $p$ used in the clustering process, and the type of generator function $f$.

We consider from the input database only the first $pN$ attributes, ordered by the values of their internal entropy, where $N$ is the total number of attributes. The output of the algorithm consists of all sets of attributes with internal entropy smaller than $kf\left(\frac{1}{k}\right)$. The parameter $p$ allows us to eliminate attributes with larger entropy.
We use three collections of sets of attributes: Candidates (potential clusterable sets to be processed at the current step in the algorithm), KAttrSets (clusterable sets of cardinality \( k \)), and MaxAttrSets (clusterable sets that are maximal). When the algorithm ends, the collection MaxAttrSets will contain all the maximal \((f, kf(\frac{1}{k}))\)-clusterable sets.

The pseudo-code of the algorithm is presented in Figure 4.1.

```
Candidates = KAttrSets = MaxAttrSets = empty;
retain from the set of attributes only pN attributes
    with the smallest internal entropies and discard the rest;
add to Candidates all one-attribute sets;
for step = 1 to N do
    scan dataset and compute for each set S from Candidates
        the internal entropy of S;
    for each set of attributes S from Candidates do
        if \( \mathcal{H}^f(\pi_S) \leq kf(\frac{1}{k}) \) do
            add S to KAttrSets;
            add S to MaxAttrSets;
            apriori-gen(Candidates, KAttrSets);
        if Candidates is empty
            break;
    remove non maximal sets from MaxAttrSets;
    scan dataset and compute for each set S from MaxAttrSets
        the external entropy of S
```

Figure 4.1: Projection clustering algorithm
Once we found all clusterable sets of cardinality $k$ in the collection $\text{KAttrSets}$, we apply the apriori-gen technique from [AMS96] for identifying those sets of cardinality $k + 1$ that can be included in $\text{Candidate}$. The apriori-gen procedure consists of generating all attribute sets of cardinality $k + 1$, starting from the sets of cardinality $k$, by combining two sets $S, S'$ that have $k - 1$ attributes in common. For example if $S = \{A_1, A_2, \cdots, A_{k-1}, A_i\}$ and $S' = \{A_1, A_2, \cdots, A_{k-1}, A_j\}$, with $i \neq j$, then a candidate set of cardinality $k + 1$ is obtain as:

$$SS' = \{A_1, A_2, \cdots, A_{k-1}, A_i, A_j\}.$$  
This procedure generates in an efficient way all candidates attribute sets of cardinality $k + 1$.

### 4.3 Experimental results

![Generators](image)

**Figure 4.2: Generators**

We ran our algorithm on the database $\text{agaricus-lepiota.data}$ from the UCI Machine Learning Repository ([BM98]), experimenting using three different gen-
erators \( f_{\text{ent}}, f_{\text{gini}}, \) and \( f_{\text{ge}} \), for different values of \( k \) and \( p \). Figure 4.2 presents the graph of the generators used in our experiments.

The point where \( f_{\text{ent}}(p) = f_{\text{ge}}(p) \) is \( p = \frac{1}{2} \). For \( 0 \leq p \leq 0.5 \), we have \( f_{\text{gini}}(p) \leq f_{\text{ge}}(p) \leq f_{\text{ent}}(p) \), and for \( 0.5 < p \leq 1 \) we have \( f_{\text{gini}}(p) \leq f_{\text{ent}}(p) \leq f_{\text{ge}}(p) \). \( f_{\text{gini}} \) and \( f_{\text{ge}} \) have a symmetric variation; penalizing more significantly the clusters containing a percent between 20% to 80% of the total rows. \( f_{\text{ent}} \) has an asymmetric variation, penalizing more the clusters having between 20% to 50% of the total elements. For all three generators, we obtain small values of the generalized entropy, when the attribute partition has clusters of very small or very large cardinality.

The results (time in seconds and the number of clusterable sets) obtained for \( p = 0.5 \) and \( p = 0.7 \) are summarized in Tables 4.1 and 4.2.

<table>
<thead>
<tr>
<th>( k )</th>
<th>( f_{\text{gini}} )</th>
<th>( f_{\text{ent}} )</th>
<th>( f_{\text{ge}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>no. sets</td>
<td>time</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>26</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>39</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>40</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>45</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>31</td>
<td>57</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>40</td>
<td>41</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>83</td>
<td>36</td>
<td>46</td>
</tr>
<tr>
<td>30</td>
<td>117</td>
<td>15</td>
<td>77</td>
</tr>
<tr>
<td>40</td>
<td>119</td>
<td>1</td>
<td>98</td>
</tr>
<tr>
<td>50</td>
<td>119</td>
<td>1</td>
<td>112</td>
</tr>
</tbody>
</table>

Table 4.1: Projection clustering for 50% of the attributes
<table>
<thead>
<tr>
<th>$k$</th>
<th>$f_{\text{gini}}$</th>
<th>$f_{\text{ent}}$</th>
<th>$f_{\text{eq}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>no. sets</td>
<td>time</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>42</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>65</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>35</td>
<td>102</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>60</td>
<td>123</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>89</td>
<td>141</td>
<td>40</td>
</tr>
<tr>
<td>9</td>
<td>141</td>
<td>186</td>
<td>57</td>
</tr>
<tr>
<td>10</td>
<td>203</td>
<td>178</td>
<td>74</td>
</tr>
<tr>
<td>20</td>
<td>922</td>
<td>368</td>
<td>496</td>
</tr>
<tr>
<td>30</td>
<td>1927</td>
<td>374</td>
<td>1125</td>
</tr>
<tr>
<td>40</td>
<td>2578</td>
<td>473</td>
<td>1854</td>
</tr>
<tr>
<td>50</td>
<td>3632</td>
<td>274</td>
<td>2537</td>
</tr>
</tbody>
</table>

Table 4.2: Projection clustering for 70% of the attributes

Figures 4.3 and 4.4 present the plot of the number of clusterable sets, for all generator functions considered in our experiments: $f_{\text{gini}}, f_{\text{ent}},$ and $f_{\text{eq}}$.

The graph of the number of sets as a function of the anticipated number of blocks ($k$) is unimodal and reaches its maximum for a number of blocks that depends on $p$, the fraction of attributes retained from the table, and on the generator function involved in the experiment. After this maximum is reached, the number of sets decreases. For $p = 0.5$, the maximum number of sets is reached when $k = 30$ for $f_{\text{gini}}$ and $f_{\text{ge}}$, and when $k = 20$ for $f_{\text{ent}}$. For $p = 0.7$, all algorithms obtained a maximum number of sets for $k = 40$. Figure 4.5 illustrated how the threshold value for the generalized entropy varies with the number of clusters $k$. We see that as $k$ increases, the threshold value increases also, so we expect that the number of clusterable sets to increase with the increase in
Figure 4.3: Number of sets for $p = 0.5$

$k$. However, since we are interested in the maximal sets having the entropy less than the threshold, there is a point at which the set containing all the attributes qualifies as a clusterable set. After this point an increase in $k$ has no effect in the output of the algorithm, since only one set will be the result, the set containing all attributes. We can see this effect for $p = 0.5$ and for the generator $f_{\text{gini}}$. Even if this extreme situation is not reached, the number of clusterable sets produced by our algorithm decrease after a certain value of $k$, since larger and larger sets qualify as clusterable sets, and all their subsets are pruned from the selection of clusterable sets.
Figures 4.6 and 4.7 present the graphs of the time as a function of $k$. The time used by our algorithm grows until the moment when the algorithm reaches its saturation point, by finding the set of all attributes. This effect is visible from the experiments with $p = 0.5$ and $f_{\text{gini}}$, when the set of all attributes was generated as a clusterable set. The algorithms using $f_{\text{ent}}$ and $f_{\text{ge}}$ have similar time performance, and they perform faster than the algorithm using $f_{\text{gini}}$.

An example of an $(f, k f \left( \frac{1}{k} \right))$-clusterable set of attributes, is the set $S = \{ \text{class, stalk-shape, veil-type} \}$, obtained using the parameters $p = 0.5$, $k = 4$, for all three generators $f_{\text{gini}}$, $f_{\text{ent}}$ and $f_{\text{ge}}$. The internal entropy of this set of attributes, computed with $f_{\text{ent}}$ is $\mathcal{H}^{\text{ent}}(\pi_S) = 1.978$, representing 98% of the maximum pos-
Table 4.3: Characteristics of the set \{class, stalk-shape, veil-type\}

Possible entropy of a partition with 4 classes, equal to 2. The external entropy, computed also using $f_{\text{ent}}$ is $\mathcal{H}^{f_{\text{ent}}} (\pi_{H-S} | \pi_S) = 2.231$, representing only 43% of
the maximum value for the conditional entropy, equal to 5.169. The partition determined by this set has 4 clusters, formed by grouping on the values of the attributes. These clusters contain 2592, 2016, 1900 and 161 rows, respectively, and are presented in the table 4.3. The attributes $H - S = \{\text{bruises}, \text{gill-attachment}, \text{gill-spacing}, \text{gill-size}, \text{stalk-surface-above-ring}, \text{stalk-surface-below-ring}, \text{veil-color}, \text{ring-number}, \text{ring-type}\}$ are considered in the computation of external entropy. The third column represents the intersections between the cluster $B$ of the partition $\pi_S$ and the clusters of the partition $\pi_{H-S}$. The values in this column are given in the following form: the number of groups with similar cardinality times the group’s cardinality. The last column gives the impurity of the cluster $B$ with
respect to the partition \( \pi_{H-S} \). By summing the values in this last column we get the value of the external entropy \( \mathcal{H}_{}(\pi_{H-S}|\pi_S) = 2.231 \). Note, that the clusters associated with the set \( S \) are pure with respect to the rest of the attributes \( H-S \).

Table 4.4 presents the clusterable sets common for all three generators used in our experiments. The number of clusters of the partitions determined by the clusterable sets presented in Table 4.4, corresponds to the value of the parameter \( k \) used in the computation of the threshold for the internal entropy. For example, when the parameters \( p = 0.7 \) and \( k = 4 \) were used, for all generators our algorithm produced the set \textit{class}, \textit{stalk-shape}, \textit{veil-type}. This set has the following groups of values, corresponding to the attributes \textit{class}, \textit{stalk-shape}, and \textit{veil-type}.

Figure 4.7: Time for \( p = 0.7 \)
<table>
<thead>
<tr>
<th>Attributes set &amp; clusters cardinalities</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>p, k = 0.5, k = 2</td>
<td></td>
</tr>
<tr>
<td>class, veil-type</td>
<td>4802 3916</td>
</tr>
<tr>
<td>stalk-shape, veil-type</td>
<td>4608 3516</td>
</tr>
<tr>
<td>p = 0.7, k = 2</td>
<td></td>
</tr>
<tr>
<td>class, stalk-shape, veil-type</td>
<td>2592 2016 1900 1616</td>
</tr>
</tbody>
</table>

Table 4.4: Common sets between the results obtained for $f_{gini}$, $f_{ent}$, and $f_{ge}$ respectively: 2592 values of \{edible, tapering, partial\}, 2016 values of \{poisonous, tapering, partial\}, 1900 values of \{poisonous, enlarging, partial\}, and 1616 values of \{edible, enlarging, partial\}. The attribute veil-type has only one value in the database, and will be very often attached to other attributes, in the process of forming the maximal clusterable sets.

The results obtained for $f_{ent}$ and $f_{ge}$ share the most numerous common sets. They have 1510 common sets across all experiments. Tables 4.5 and 4.6 present for each experiment the number of sets obtained by using $f_{ent}$ as generator, by
using $f_{ge}$ as generator, and also present the number of common sets between the results obtained in the two cases.

<table>
<thead>
<tr>
<th>$k$</th>
<th>no. sets for $f_{ent}$</th>
<th>no. sets for $\geq$</th>
<th>no. common sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>24</td>
<td>28</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>39</td>
<td>37</td>
<td>23</td>
</tr>
<tr>
<td>8</td>
<td>52</td>
<td>46</td>
<td>29</td>
</tr>
<tr>
<td>9</td>
<td>53</td>
<td>46</td>
<td>28</td>
</tr>
<tr>
<td>10</td>
<td>49</td>
<td>56</td>
<td>27</td>
</tr>
<tr>
<td>20</td>
<td>71</td>
<td>63</td>
<td>34</td>
</tr>
<tr>
<td>30</td>
<td>67</td>
<td>68</td>
<td>24</td>
</tr>
<tr>
<td>40</td>
<td>45</td>
<td>43</td>
<td>17</td>
</tr>
<tr>
<td>50</td>
<td>17</td>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 4.5: Number of clusterable sets common to $f_{ent}$ and $f_{ge}$ for $p = 0.5$

The results obtained for $f_{gini}$ and $f_{ent}$ share 30 sets in common, and for $f_{gini}$ and $f_{ge}$ only 10 sets.

### 4.4 Conclusions and future research

We introduced a notion of the internal and external entropy associated with a set of attributes, and designed and implemented a new algorithm for finding all maximal clusterable subsets of attributes having an entropy value below a certain threshold. The value of the threshold is determined based on the number of clusters that we require from the attribute subspaces. Experimental results proved that the resulting subsets have approximately the required number of
<table>
<thead>
<tr>
<th>$k$</th>
<th>no. sets for $f_{ent}$</th>
<th>no. sets for $\geq$</th>
<th>no. common sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>33</td>
<td>18</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>47</td>
<td>29</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>77</td>
<td>49</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td>102</td>
<td>72</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>145</td>
<td>101</td>
<td>55</td>
</tr>
<tr>
<td>9</td>
<td>180</td>
<td>125</td>
<td>63</td>
</tr>
<tr>
<td>10</td>
<td>200</td>
<td>132</td>
<td>84</td>
</tr>
<tr>
<td>20</td>
<td>419</td>
<td>344</td>
<td>216</td>
</tr>
<tr>
<td>30</td>
<td>596</td>
<td>503</td>
<td>285</td>
</tr>
<tr>
<td>40</td>
<td>612</td>
<td>539</td>
<td>276</td>
</tr>
<tr>
<td>50</td>
<td>453</td>
<td>411</td>
<td>189</td>
</tr>
</tbody>
</table>

Table 4.6: Number of clusterable sets common to $f_{ent}$ and $f_{ge}$ for $p = 0.7$

clusters, they have a small internal entropy, and some of them have also a small external entropy.

Further investigation of the classes of generator functions is necessary to determine the types of clusterable sets that are favored by these classes of generators. Future research can be directed to the investigation of the existence of a set of properties characterizing the clusterable sets that are induced by the generator function used in their computation.

The definition of clusters can be further refined as groups of records determined by certain sets of attributes that have limited internal and external impurities, and techniques that allow the user to choose a covering of the set of records by $(f, \alpha, \beta)$-clusters can be developed, where $\alpha$ represents the threshold for the internal entropy and $\beta$ represents the threshold for the external entropy.

61
CHAPTER 5

Genetic algorithm for partitioning

We present next, a new partitioning algorithm that searches for the best partition of a set of objects satisfying a certain criterion. This method, based on a genetic algorithm approach, will be used in designing algorithms for two clustering applications. The first application, presented in details in chapter 6, consists of finding the median partition of the set of partitions determined by the attributes of a categorical database. The second application, discussed in chapter 7, consists of finding the natural clustering of a set of objects in a categorical database, whose clusters share as much as possible common characteristics of their attributes. This chapter presents the common core of the two clustering algorithms, and important notions related to the genetic algorithm approach.

Definition 5.1 A $k$-chromosome on a table $T : R \times H \rightarrow \bigcup D_H$ is a function $K : R \rightarrow \{1, \ldots, k\}$. An element of the set $\{1, \ldots, k\}$ is called a class identifier.

Each $k$-chromosome encodes a partition into $k$ classes of the set $R$ of objects (tuples) of the table $T$. We denote by $\pi_K$ the partition of $R$ determined by the $k$-chromosome $K$; $\pi_K = \{C_1, \ldots, C_k\}$, where $C_j = \{r \in R \mid K(r) = j\}$ is the class associated with the class identifier $j$, for $1 \leq j \leq k$. 

62
<table>
<thead>
<tr>
<th>$M$</th>
<th>cardinality of the chromosomal population</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>number of rows in the table $T$</td>
</tr>
<tr>
<td>$n$</td>
<td>number of columns in the table $T$</td>
</tr>
<tr>
<td>$k$</td>
<td>number of classes in the target partition</td>
</tr>
<tr>
<td>$r$</td>
<td>percent of chromosomes that are used for crossover</td>
</tr>
<tr>
<td>$m$</td>
<td>percent of chromosomes that undergo a mutation</td>
</tr>
<tr>
<td>$N_{\text{max}}$</td>
<td>maximum number of consecutive iterations without improvement</td>
</tr>
<tr>
<td>$\mu$</td>
<td>margin error for the fitness function</td>
</tr>
</tbody>
</table>

Table 5.1: Input parameters of the genetic algorithm

The input parameters of the genetic algorithm are summarized in table 5.1.

The chromosomal population $K_1, K_2, \cdots, K_M$ consists of $k$-chromosomes; each $k$-chromosome $K_i$ can be regarded as a sequence of length $N = |R|$ representing a possible assignment of the rows of the table $T$ to the $k$ classes of the partition $\pi_{K_i}$. Initially, the chromosomes $K_1, K_2, \cdots, K_M$, are randomly generated using values between 1 and $k$.

The idea of the genetic evolution is to modify the chromosomes in the current population by using mutation and crossover as genetic operators, such that in the new population we have chromosomes that will be increasingly closer to the partition we are searching for.

We use the classical single-point crossover operator. Starting from two chromosomes, a random crossing point, called a crossover site, is selected as a number $l$, between 1 and $N$. The offspring will contain the first $1$ to $l$ positions from the first parent chromosome, and the last $l+1$ to $N$ positions from the second parent chromosome, and vice versa. We use the classical mutation operator, which involves changing randomly a number of max\{$1, 0.1N\}$ positions in the chromosome.
that undergoes a mutation. The new value for each position is chosen randomly from 1 to $k$.

Figure 5.1 presents the pseudocode of the algorithm.

```
initialize the population of genetic algorithm.
while (true)
    compute the fitness of all chromosomes in the population.
    if (best fitness $\geq \mu$ or there has been no relative
        improvement in best fitness value for $N_{max}$ iterations)
        output the partition of $K_{best}$ and exit;
    copy fittest $(1 - r - m)M$ chromosomes to the new population.
    select probabilistically $\max\{2, rM\}$ chromosomes to cross over.
    apply crossover operator to the selected chromosomes,
    replace parent chromosomes with the offspring in the current
    population, and copy the offspring to the new population.
    select with uniform probability $\max\{1, mM\}$ chromosomes from
    the current population to undergo mutations.
    apply mutation operator to the selected chromosomes and
    copy the modified chromosomes to the new population.
    replace old population by the new one.
```

Figure 5.1: Clustering genetic algorithm

For each chromosome $K_i$, we compute the value of its fitness $\text{fitness}(K_i)$. The
value of this quantity depends on the particular application. Thus, specific details
related to the fitness measure are presented in the future chapters, dedicated to
specific clustering applications. For the moment, we assume that each chromo-
some has associated a fitness value that expresses how close is a chromosome to
the partition we are searching for. The larger is the values of the fitness, the closer is the chromosome to the target partition. We denote by $K_{\text{best}}$ the chromosome which has the largest fitness($K_i$) value; its fitness is denoted by $\text{fitness}_{\text{best}}$.

If $\text{fitness}_{\text{best}} \geq \mu$ or there is no improvement in the best fitness for $N_{\text{max}}$ consecutive iterations, then the algorithm will halt, and the resulting partition will correspond to the chromosome $K_{\text{best}}$. Otherwise, a new population is generated. We use an elitist strategy ([Mic99]), according to which the fittest max{$M - 3, (1 - r - m)M$} chromosomes are copied directly in the new population, ensuring thus, that we never lose the best chromosome from the old population.

Next, a number of max{$2, rM$} chromosomes from the old generation are selected probabilistically to be used in generating new offspring by application of the crossover operator. As selection method we use a roulette wheel strategy or fitness-proportionate selection (explained in chapter 1), in which the chromosomes having greater values of their fitness have also a greater chance of being selected. After the crossover transformation, the parent chromosomes are replaced by their offspring, and the newly generated offspring is also copied into the new population.

Finally, a number of max{$1, mM$} chromosomes are selected with uniform probability and they undergo mutations. The mutation operator is not biased towards the fittest chromosomes, and therefore, the chromosomes that will suffer a mutation are selected with uniform probability.
CHAPTER 6

Median partition

In a database with categorical attributes, each attribute defines a partition whose classes can be regarded as natural clusters of rows. In this chapter we focus on finding a partition of the rows of a given database, that is as close as possible to the partitions associated to each attribute. We evaluate the closeness of two partitions by using a generalization of the classical conditional entropy. From this perspective, we wish to construct a partition (referred to as the median partition) such that the sum of the dissimilarities between this partition and all the partitions determined by the attributes of the database is minimal. Then, the problem of finding the median partition is an optimization problem, over the space of all partitions of the rows of the database, for which we give an approximative solution. To search more efficiently the large space of possible partitions we use a genetic algorithm where the partitions are represented by chromosomes. Our genetic algorithm obtains better clustering results than the classical k-means (presented in chapter 1) and transfer objects (presented in chapter 2) algorithms.

6.1 Introduction

Given a set of $N$ objects, characterized by $n$ attributes, we want to group these objects such that objects which are similar to be placed in the same group and objects which are dissimilar to be placed in different groups.
Research in clustering has been focus more on data characterized by numerical or quantitative attributes, for which natural distances (like Euclidean or Manhattan distance) between the objects to be clustered are available. Clustering categorical or qualitative attributes presents an extra challenge, since a natural ordering of the attributes values is not available. Previous research in this direction include algorithms introduced in [VJR99, GRS00, Hua97].

We focus on the case of categorical attributes databases and we investigate the application of information-theoretical methods described in chapter 3 to this clustering problem. The objects that we cluster are represented by tuples in a table and the goal of the genetic algorithm is to construct a partition of the set of rows whose classes are regarded as clusters of the rows. We require that this partition satisfy the following two conditions:

1. it must summarize the partitions generated by the attributes of the table.

2. the number of classes of the partition should not exceed a prescribed upper limit.

The first condition ensures that the resulting partition represents a clustering of the rows, while the second condition represents a restriction of the number of classes in this clustering.

In literature, the partition satisfying the first condition is called median partition or consensus partition and was analyzed in [BL95, Reg83b, Reg83a]. Research concerning the application of genetic algorithms to grouping problems was presented in [Hol92, Mic99, Fal99, Mit97, JB91].

Finding the median partition is an NP-complete problem as it was shown by [BL95]. That is why we focus in this paper on finding an approximative solution using a genetic algorithm approach. To determine how well a partition
summarizes the attribute partitions of a table, we use a measure based on a generalization of the conditional entropy, that was introduced also in chapter 3.

6.2 Properties of partitions with \( k \) classes

Let \( \pi_k = \{B_1, B_2 \ldots B_k\} \in \text{PART}(R) \) with \(|\pi_k| = k\) classes and let \( \pi_{k-1} \) be the partition with \( k - 1 \) classes, obtained from \( \pi_k \) by joining the classes \( B_i \) and \( B_j \), and keeping the rest of the classes unmodified. The difference between the entropies of the two partitions is given by \( \mathcal{H}^f(\pi_k) - \mathcal{H}^f(\pi_{k-1}) = f \left( \frac{|B_i|}{|R|} \right) + f \left( \frac{|B_j|}{|R|} \right) - f \left( \frac{|B_i \cap B_j|}{|R|} \right) \geq 0 \) (since the generator \( f \) is sub-additive). Thus, by joining two classes of a partition we obtained another partition with a smaller value of its entropy. Similarly, by splitting a class of a partition we obtained another partition with a greater value of the entropy. The quantity \( \mathcal{H}^{f_{\text{gini}}}(\pi_k) - \mathcal{H}^{f_{\text{gini}}}(\pi_{k-1}) = 2 \frac{|B_i||B_j|}{|R|^2} \). Therefore, for the Gini index as generator, the maximum decrease in the entropy is obtained when the largest classes of the partition \( \pi_k \) are united. For the Shannon entropy as generator, the decrease in entropy is \( \mathcal{H}^{\text{ent}}(\pi_k) - \mathcal{H}^{\text{ent}}(\pi_{k-1}) = \frac{1}{|R|} (1 + \frac{1}{r})^{|B_i|} \), where \( r = \frac{|B_i|}{|R|} \). If we fix \( B_i \) as being the largest class of \( \pi_k \), then the maximum value of the expression \((1 + \frac{1}{r})^r\) is obtained for the largest possible value of \( r \), thus for the largest value of \(|B_j|\). Similarly, to obtain the greatest decrease in entropy we have to fuse the largest classes of the partition \( \pi_k \).

Given a set \( R \), \( \alpha_R \) has the largest value of the entropy \( \mathcal{H}^f(\alpha_R) = |R|f \left( \frac{1}{|R|} \right) \) and the largest number of classes \(|\alpha_R| = |R|\) and \( \omega_R \) has the smaller value of the entropy \( \mathcal{H}^f(\omega_R) = 0 \) and the smallest number of classes \(|\omega_R| = 1\). We can obtain any partition \( \pi \) with \( k \) classes, by starting from \( \alpha_R \) and successively joining two classes. We obtain in this way a sequence of partitions \( \pi^0 = \alpha_R, \pi^1, \pi^2 \ldots \pi^k = \pi \) such that each \( \pi^i \) has the same classes as \( \pi^{i-1} \) with the exception of two classes.
which are united in $\pi^i$. If we want the partition $\pi$ to have the smallest possible value of the entropy among all partitions with $k$ classes, in the series of partitions obtained above, we must always to unite the largest possible classes, to ensure at each step the largest decrease in the value of the entropy. Thus, in this process, we always add one more element to the largest class obtained so far. Therefore, the partition $\pi$ having $|\pi| = k$ classes and the smallest possible value of $\mathcal{H}^f(\pi)$ has $k - 1$ classes of cardinality 1 and one class of cardinality $|R| - k + 1$.

The partition $\pi$ having $|\pi| = k$ classes and the largest possible value of the entropy, has classes of equal cardinality $\frac{|R|}{k}$.

We conclude that the entropy of a partition $\pi$ with $|\pi| = k$ classes is bounded by:

$$ kf \left( \frac{1}{|R|} \right) + f \left( \frac{|R - k + 1|}{|R|} \right) \leq \mathcal{H}^f(\pi) \leq kf \left( \frac{1}{K} \right).$$

Given an entropy value $\mathcal{H}^f_{\text{target}} \leq |R| f\left(\frac{1}{|R|}\right)$, we can estimate an upper and lower bound for the number of classes $k = |\pi|$ of a partition $\pi$ which has entropy $\mathcal{H}^f(\pi) = \mathcal{H}^f_{\text{target}}$.

For the Gini index as generator, the inequality

$$ \mathcal{H}^f_{\text{target}} = \mathcal{H}^f(\pi) \leq kf \left( \frac{1}{K} \right) $$

implies $k \geq \lceil \frac{1}{1 - \mathcal{H}^f_{\text{target}}} \rceil$ and the inequality

$$ \mathcal{H}^f_{\text{target}} = \mathcal{H}^f(\pi) \geq kf \left( \frac{1}{|R|} \right) + f \left( \frac{|R - k + 1|}{|R|} \right) $$

implies $k \leq \lceil |R| \sqrt{\mathcal{H}^f_{\text{target}}} + 1 \rceil$.

For the Shannon entropy as generator, we obtain the boundaries $2^{\mathcal{H}^f_{\text{target}}} \leq k \leq |R| + 1 - \lfloor e^{\mathcal{H}^f_{\text{target}}} \rfloor$, where $k_0$ is the first integer that makes the inequality

$$ \frac{|R|}{e^{\mathcal{H}^f_{\text{target}}}} \leq (|R| - k + 1)(|R| - k + 1) $$

true.
For a generator $f$, the upper bound of the value $k$, obtained from the inequality $\mathcal{H}'_{\text{target}} \leq kf \left( \frac{1}{k} \right)$ represents the number of classes of a partition $\pi$ with equal cardinality classes having the value of the entropy $\mathcal{H}'(\pi) = \mathcal{H}'_{\text{target}}$. The lower bound represents the number of classes of a partition with one class containing the majority of the elements. Since the last situation will be very unlikely to occur in practical clustering problems, more attention should be given to the upper bound estimate.

### 6.3 Fitness measures

Let $T : R \times H \rightarrow \bigcup D_H$ be a table. We are searching for the median partition, that is a partition $\pi$ such that $|\pi| \leq k$, and $\pi$ minimizes the sum $\sum_{A \in H} d^f(\pi_A, \pi)$.

Let $\pi \in \text{PART}(R)$, and denote:

$$\mathcal{K}^f(\pi) = \sum_{A \in H} \mathcal{H}^f(\pi_A | \pi),$$

and

$$\mathcal{L}^f(\pi) = \sum_{A \in H} \mathcal{H}^f(\pi | \pi_A).$$

In the median partition problem, we are searching for a partition $\pi$ which minimizes the distance $d^f(\pi) = \sum_{A} \left[ \mathcal{H}^f(\pi_A | \pi) + \mathcal{H}^f(\pi | \pi_A) \right] = \mathcal{K}^f(\pi) + \mathcal{L}^f(\pi)$.

We have $d^f(\omega_R) = \mathcal{K}^f(\omega_R)$, because $\pi_A \leq \omega_R$ and $\mathcal{H}^f(\omega_R | \pi_A) = 0$ for all partitions $\pi_A$. Also $\mathcal{K}^f(\omega_R) = \sum_{A} \mathcal{H}^f(\pi_A)$, so $d^f(\omega_R) = \sum_{A} \mathcal{H}^f(\pi_A)$.

Similarly, we have $d^f(\alpha_R) = \mathcal{L}^f(\alpha_R)$, because $\pi_A \geq \alpha_R$ and $\mathcal{H}^f(\alpha_R | \pi_A) = 0$ for all partitions $\pi_A$. Given that $\pi_A = \{B_1^A, \ldots, B_i^A\}$ and $\alpha_R = \{D_1, \ldots, D_{|R|}\}$ we have $\mathcal{H}^f(\alpha_R | \pi_A) = \sum_{i=1}^{i} \left[ \frac{|B_i^A|}{|R|} \sum_{j=1}^{|R|} f \left( \frac{|B_i^A \cap D_j^i|}{|B_i^A|} \right) \right]$. There are $|B_i^A|$ non-empty intersections in the inner sum, so $\mathcal{H}^f(\alpha_R | \pi_A) = \sum_{i=1}^{i} \left[ \frac{|B_i^A|^2}{|R|} f \left( \frac{1}{|B_i^A|} \right) \right]$. Thus, $d^f(\alpha_R) = \sum_{i=1}^{i} \left[ \frac{|B_i^A|^2}{|R|} f \left( \frac{1}{|B_i^A|} \right) \right]$. 

70
\[ \alpha_R \leq \pi \leq \omega_R, \mathcal{H}^f(\pi_A|\pi) \leq \mathcal{H}^f(\pi_A|\omega_R), \text{ and } \mathcal{H}^f(\pi|\pi_A) \leq \mathcal{H}^f(\alpha_R|\pi_A). \] Thus, we have an upper bound for the value \( d^f(\pi) \), namely \( d^f(\pi) \leq d^f(\omega_R) + d^f(\alpha_R) \).

We propose the quantities summarized in table 6.3 as measures of the closeness between the partitions represented by the chromosomes and the attribute partitions.

<table>
<thead>
<tr>
<th>Quality measures for ( \pi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{K}^f(\pi) = \sum_{A \in H} \mathcal{H}^f(\pi_A</td>
</tr>
<tr>
<td>( \mathcal{M}^f(\pi) = \mathcal{K}^f(\pi) + \mathcal{L}^f(\pi) )</td>
</tr>
<tr>
<td>( \mathcal{N}^f(\pi) = \sum_{A \in H} \frac{\mathcal{H}^f(\pi</td>
</tr>
<tr>
<td>( \mathcal{P}^f(\pi) = \begin{cases} \mathcal{K}^f(\pi) &amp; \text{if } \mathcal{H}^f(\pi) \leq \mathcal{H}^f_{\text{avg}} \ \mathcal{L}^f(\pi) &amp; \text{otherwise} \end{cases} )</td>
</tr>
<tr>
<td>( \mathcal{Q}^f(\pi) =</td>
</tr>
</tbody>
</table>

Table 6.1: Measures for assessing the quality of the partition \( \pi \)

Based on these measures, we define the fitness measure associated with each chromosome \( K \) whose partition is \( \pi_K \), as in table 6.3.

Note that if a chromosome has a small value for \( \mathcal{K}^f(\pi_K) \), then its associated fitness value \( f^f_K(\pi) \) will be large. This property holds for all proposed fitness measures.

The measure \( \mathcal{K}^f(\pi) \) leads the convergence process toward a partition \( \pi \) whose classes fit as well as possible inside the classes of all attribute partitions.

The measure \( \mathcal{M}^f(\pi) \) requires the partition \( \pi \) to have classes that fit as well as possible inside the classes of all attribute partitions and conversely the classes
<table>
<thead>
<tr>
<th>Fitness Measures</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{fitness}_K^f(K) = \begin{cases} \mu, &amp; \text{if } K^f_T(\pi_K) = 0 \ (K^f_T(\pi_K))^{-1}, &amp; \text{otherwise} \end{cases}$</td>
<td></td>
</tr>
<tr>
<td>$\text{fitness}_M^f(K) = \begin{cases} \mu, &amp; \text{if } M^f_T(\pi_K) = 0 \ (M^f_T(\pi_K))^{-1}, &amp; \text{otherwise} \end{cases}$</td>
<td></td>
</tr>
<tr>
<td>$\text{fitness}_N^f(K) = \begin{cases} \mu, &amp; \text{if } N^f_T(\pi_K) = 0 \ (N^f_T(\pi_K))^{-1}, &amp; \text{otherwise} \end{cases}$</td>
<td></td>
</tr>
<tr>
<td>$\text{fitness}_O^f(K) = \begin{cases} \mu, &amp; \text{if } O^f_T(\pi_K) = 0 \ (O^f_T(\pi_K))^{-1}, &amp; \text{otherwise} \end{cases}$</td>
<td></td>
</tr>
<tr>
<td>$\text{fitness}_P^f(K) = \begin{cases} \mu, &amp; \text{if } P^f_T(\pi_K) = 0 \ (P^f_T(\pi_K))^{-1}, &amp; \text{otherwise} \end{cases}$</td>
<td></td>
</tr>
<tr>
<td>$\text{fitness}_Q^f(K) = \begin{cases} \mu, &amp; \text{if } Q^f_T(\pi_K) = 0 \ (Q^f_T(\pi_K))^{-1}, &amp; \text{otherwise} \end{cases}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.2: Fitness measures

of the attribute partitions should fit as well as possible inside the classes of the partition $\pi$.

For a given partition $\pi$ and a given database, $K^f(\pi)$ and $L^f(\pi)$ do not have the same range of possible values, but $\frac{3\mathcal{H}^f(\pi_A)}{2 + 3\mathcal{H}^f(\pi)}$ and $\frac{3\mathcal{H}^f(\pi_A)}{2 + 3\mathcal{H}^f(\pi)}$ are both between 0 and 1. The measure $N^f(\pi)$ uses the scaled versions of the conditional entropies.

The introduction of the measure $O^f(\pi)$ is explained by the following results.

For any partition $\pi$ by summing over all partitions $\pi_A$ we get the following inequalities

$$\frac{1}{2} \sum_A \mathcal{H}^f(\pi_A) - \frac{1}{4} n \mathcal{H}^f(\pi) \leq \mathcal{H}^f(\pi) \leq \sum_A \mathcal{H}^f(\pi_A)$$

and

$$\frac{1}{2} \sum_A \mathcal{H}^f(\pi_A) \leq L^f(\pi) \leq n \mathcal{H}^f(\pi).$$

Let $\mathcal{H}^f_{\text{avg}}$ be the average entropy of the partitions generated by the attributes of the table, given by $\mathcal{H}^f_{\text{avg}} = \frac{\sum_A \mathcal{H}^f(\pi_A)}{n}$. The previous inequalities can be rewritten as

$$\mathcal{H}^f_{\text{avg}} - \mathcal{H}^f(\pi) \leq \frac{2\mathcal{H}^f(\pi)}{n} \leq 2\mathcal{H}^f_{\text{avg}}$$

and

The introduction of the measure $O^f(\pi)$ is explained by the following results.
\( \mathcal{H}^f(\pi) - \mathcal{H}^f_{\text{avg}} \leq \frac{2\mathcal{K}^f(\pi)}{n} \leq 2\mathcal{H}^f(\pi) \). We have two cases depending on the relative values of \( \mathcal{H}^f(\pi) \) and \( \mathcal{H}^f_{\text{avg}} \):

1. If \( \mathcal{H}^f_{\text{avg}} \geq \mathcal{H}^f(\pi) \), then we have \( \max \left\{ \frac{\mathcal{K}^f(\pi)}{n}, \mathcal{H}^f_{\text{avg}} - \frac{2\mathcal{K}^f(\pi)}{n} \right\} \leq \mathcal{H}^f(\pi) \leq \mathcal{H}^f_{\text{avg}} \), \( \mathcal{K}^f(\pi) \leq n\mathcal{H}^f_{\text{avg}} \), and \( \mathcal{L}^f(\pi) \leq n\mathcal{H}^f(\pi) \leq n\mathcal{H}^f_{\text{avg}} \).

   To have the blocks of the partition \( \pi \) fit as well as possible inside the blocks of the partitions \( \pi_A \), we seek to minimize the average \( f \)-impurity of the blocks of \( \pi \), relative to each of these partitions, that is, we seek to minimize \( \mathcal{K}^f(\pi) \). The inequality \( 0 \leq \mathcal{H}^f_{\text{avg}} - \mathcal{H}^f(\pi) \leq \frac{2\mathcal{K}^f(\pi)}{n} \), guarantees that if \( \mathcal{K}^f(\pi) \) is small, then \( \mathcal{H}^f(\pi) \) is close to \( \mathcal{H}^f_{\text{avg}} \).

   For the quantity \( \mathcal{M}^f(\pi) = \mathcal{K}^f(\pi) + \mathcal{L}^f(\pi) \) we have also the following lower and upper bound \( \frac{n}{2} \{ \mathcal{H}^f_{\text{avg}} - \mathcal{K}^f(\pi) \} \leq \mathcal{K}^f(\pi) + \mathcal{L}^f(\pi) \leq 2n\mathcal{H}^f_{\text{avg}} \).

2. If \( \mathcal{H}^f_{\text{avg}} < \mathcal{H}^f(\pi) \), then we have \( \max \left\{ \frac{\mathcal{K}^f(\pi)}{n}, \mathcal{H}^f_{\text{avg}} \right\} \leq \mathcal{H}^f(\pi) \leq \mathcal{H}^f_{\text{avg}} + \frac{2\mathcal{K}^f(\pi)}{n} \), \( \mathcal{K}^f(\pi) \leq n\mathcal{H}^f_{\text{avg}} \), and \( \mathcal{L}^f(\pi) \leq n\mathcal{H}^f(\pi) \).

   To have the blocks of the attribute partitions \( \pi_A \), fit as well as possible inside the blocks of the partition \( \pi \) found by the genetic algorithm, we seek to minimize the average \( f \)-impurity of the blocks of attribute partitions \( \pi_A \), relative to the partition obtained by the genetic algorithm, that is, we seek to minimize \( \mathcal{L}^f(\pi) \). The inequality \( 0 \leq \mathcal{H}^f(\pi) - \mathcal{H}^f_{\text{avg}} \leq \frac{2\mathcal{K}^f(\pi)}{n} \), insures that the value of \( \mathcal{H}^f(\pi) \) will get close to the value \( \mathcal{H}^f_{\text{avg}} \).

   In this case the lower and upper bound of the quantity \( \mathcal{M}^f(\pi) = \mathcal{K}^f(\pi) + \mathcal{L}^f(\pi) \) are \( \frac{n}{2} \{ \mathcal{H}^f(\pi) - \mathcal{H}^f_{\text{avg}} \} \leq \mathcal{K}^f(\pi) + \mathcal{L}^f(\pi) \leq n\mathcal{H}^f_{\text{avg}} + n\mathcal{H}^f(\pi) \).

To conclude, by minimizing \( \mathcal{K}^f(\pi) \) and \( \mathcal{L}^f(\pi) \) we obtain a partition \( \pi \) which has the value \( \mathcal{H}^f(\pi) \) as close as possible to the value \( \mathcal{H}^f_{\text{avg}} \). This observation suggest the following convergence criteria for the genetic algorithm:
If $\mathcal{H}_{avg}^f \geq \mathcal{H}^f(\pi)$, the genetic algorithm will minimize $\mathcal{K}^f(\pi)$, favoring partitions with many blocks of smaller sizes. Otherwise ($\mathcal{H}_{avg}^f < \mathcal{H}^f(\pi)$), the genetic algorithm will minimize $\mathcal{L}^f(\pi)$, favoring partitions with fewer blocks of bigger sizes. This alternation between minimization of $\mathcal{K}^f(\pi)$ and $\mathcal{L}^f(\pi)$ corresponds to the measure $\mathcal{O}^f(\pi)$ from table 6.3.

Similarly, the measure $\mathcal{P}^f(\pi)$ uses the idea of alternating between the minimization of $\mathcal{K}^f(\pi)$ and $\mathcal{L}^f(\pi)$, but the choice depends on the relative values of the two quantities. Namely, if $\mathcal{K}^f(\pi) < \mathcal{L}^f(\pi)$ the genetic algorithm will minimize $\mathcal{L}^f(\pi)$, otherwise will minimize $\mathcal{K}^f(\pi)$.

When the quantity $\mathcal{K}^f(\pi)$ is very small, the partition $\pi$ has many classes with smaller sizes, that fit well inside the classes of attribute partitions and therefore the classes of attribute partitions might not fit that well inside the classes of $\pi$, leading to a greater value of the quantity $\mathcal{L}^f(\pi)$. Thus, we want to favor partitions for each the difference between the two quantities is small in absolute value. The measure $\mathcal{Q}^f(\pi)$ is based on this observation.

For any given database, we have $\mathcal{H}_{avg}^f \leq \mathcal{K}^f(\pi_H) \leq n\mathcal{H}_{avg}^f$. Thus, the average entropy of the attribute partitions has always a smaller value than the entropy of the intersection partition. We denote by $k_H = |\pi_H|$. The blocks of $\pi_H$ fit perfectly inside the blocks of all attribute partitions $\pi_A$, so $\pi_H$ is a candidate for the partition that we search. But, we are not interested in this special partition since it might have a large number of blocks (close to the total number of rows $|R|$) and thus, the corresponding clustering of the data is not interesting. We are interested to find a partition $\pi$ that is as close as possible to all attribute partitions and has the maximum possible number of blocks $k$ less than $k_H$. In this process of minimizing $\mathcal{K}^f(\pi)$ and $\mathcal{L}^f(\pi)$, the entropy $\mathcal{H}^f(\pi)$ of the partition $\pi$ will get closer and closer to the value $\mathcal{H}_{avg}^f$. 

74
The interpretation of the measure $\mathcal{O}^f(\pi)$ is illustrated in Figure 6.1.

\[
\mathcal{H}^f(\pi) \quad \mathcal{H}_{\text{avg}}^f \quad \mathcal{H}^f(\pi) ~ \mathcal{H}^f(\pi_H)
\]

\[
\begin{align*}
&\min \mathcal{H}^f(\pi) \quad \text{increasing the number} \\
&\text{of classes or splitting larger} \\
&\text{classes}
\end{align*} \quad \begin{align*}
&\min \mathcal{L}^f(\pi) \quad \text{decreasing the number} \\
&\text{of classes or merging} \\
&\text{larger classes}
\end{align*}
\]

Figure 6.1: Explanation of the convergence process for measure $\mathcal{O}^f(\pi)$

If $\mathcal{H}^f(\pi) \geq \mathcal{H}_{\text{avg}}^f$, the partition $\pi$ has too many blocks or very small blocks and the minimization of $\mathcal{L}^f(\pi)$ has the effect of decreasing the number of blocks or respectively joining blocks of the partition $\pi$. Otherwise ($\mathcal{H}^f(\pi) \leq \mathcal{H}_{\text{avg}}^f$), $\pi$ has too few blocks or blocks with large sizes and the minimization of $\mathcal{K}^f(\pi)$ has the effect of increasing the number of blocks or splitting the blocks of the partition $\pi$.

For databases for which the value $\mathcal{H}_{\text{avg}}^f$ represents a good approximation of the entropy of the partition that we are searching, then the measures $\mathcal{O}^f(\pi)$ or $\mathcal{P}^f(\pi)$ should be used in the convergence process of the genetic algorithm in preference to the other proposed measures. The partition $\pi$ associated with the minimum possible value of the quantity $\mathcal{M}^f(\pi)$ has the entropy $\mathcal{H}^f(\pi)$ close to the value of the average entropy of all attribute partitions $\mathcal{H}_{\text{avg}}^f$. Thus, the number of blocks $k = |\pi|$ of the partition $\pi$ can be estimated from the inequality

\[
kf \left( \frac{1}{|\pi|} \right) \leq \mathcal{H}_{\text{avg}}^f \leq k f \left( \frac{1}{k} \right)
\]

as explained in the previous section. For the Gini index generator we get the limits for the number of classes: $|R| \sqrt{\mathcal{H}_{\text{avg}}^f} + 1 \geq k \geq \frac{1}{1-\mathcal{H}_{\text{avg}}^f}$. If the partition $\pi$ has well represented classes (with similar cardinalities) then the lower bound limit is a better estimation of $k$. On the other hand if $\pi$ has almost all elements grouped into one class, then the upper
bound limit is a better estimate. If we do not know in advance what to assume about the nature of the partition $\pi$ that we searched, we have to apply the genetic clustering algorithm searching for successive values of $k$, starting from $\frac{1}{1-\frac{\overline{\delta}}{\delta_{\text{avg}}}}$ and increasing $k$ by 1. The quantity $\mathcal{M}^f(\pi)$ will have a minimum for the value of $k$ reflecting the natural number of classes. If the value $\mathcal{M}^f(\pi)$ always increases as we increment the number of classes, then the value $k = \frac{1}{1-\frac{\overline{\delta}}{\delta_{\text{avg}}}}$ represents the natural number of classes.

### 6.4 Experimental results

We studied the clustering found by three algorithms: our genetic algorithm, denoted by Alg-Rand, the algorithm introduced by [Reg83b] and discussed in chapter 1, denoted by Alg-TO, and the classical k-means algorithm, denoted by k-means. We used the k-means algorithm implemented in the WEKA package ([WF00]), that uses the distance $d_A$ defined in chapter 1.

To prove the better quality of the partition obtained by our method, we used synthetically generated databases for which we know in advance the partition embedded in the data, denoted by reference partition.

The generation of these databases follows the pattern: for each row number $\text{rowid} \in [1, N]$, a number $i \in [1, c]$ is randomly generated and saved at position $\text{rowid}$ in the reference partition and in all attribute partitions, but one. The exception attribute $A_c \in H$, randomly chosen, receives at position $\text{rowid}$ a different value $j \in [1, c], j \neq i$. To ensure that the reference and attribute partitions have exactly $c$ classes, the first values for $i$ are $1, 2, \ldots c$.

To assess the quality of the resulting partition, we compute a quantity denoted as classification rate. We identify a one-to-one mapping of the classes from the
resulting partition $\pi$ and the classes from the reference partition $\pi_{\text{ref}}$. At each step, we maintain a list of classes from $\pi_{\text{ref}}$ that have not been yet mapped to any class of $\pi$. Initially, this list contains all the classes from $\pi_{\text{ref}}$. Then, for each class of $\pi$ we compute the cardinalities of its intersections with the classes in $\pi_{\text{ref}}$, given by the number of tuples they have in common. The pairs consisting of a class from $\pi$ and an available class from $\pi_{\text{ref}}$, are processed in decreasing order of their intersection's cardinality. Each time we retain and map the pair yielding the largest number of tuples in common. Once a class from $\pi_{\text{ref}}$ is mapped to a class from $\pi$, it is removed from the list of available classes. When the unique mapping between partitions has been completed, the classification rate is computed as the fraction of all tuples, represented by the sum of tuples shared by the classes mapped to each other.

### 6.4.1 Study of the GA results for different number of rows or chromosomes

We generated a synthetic database with a fixed number of rows $N = 50$ and columns $n = 5$, and embedded a $k = 5$ class partition in the data. To see how the algorithm behave on average when the population of chromosomes increases relative to the size of the database, we ran the genetic algorithm with 10 different values for the random number generator seed and with the number of chromosomes equal to 1, 2, up to 10 times the number of rows. The other parameters for the genetic algorithm are: crossover rate of 0.8, mutation rate of 0.1, the Shannon entropy used as generator, 100 consecutive iterations without improvement and the fitness threshold $\mu = 10^4$. The fitness measure used in these experiments is $\text{fitness}_{k}$. The results are shown in Figures 6.2, 6.3, and 6.4.

77
Figure 6.2: Classification rate for Alg-Rand with increasing number of chromosomes

Figure 6.3: Number of iterations for Alg-Rand with increasing number of chromosomes

Alg-Rand performed extremely well and found the reference partition or a close approximation of it, leading to classification rates (the number of rows classified in the partition found by the algorithm in the same way as in the reference partition) between 0.94 and 1. By increasing the size of the chromosomal population, the
Figure 6.4: Time in secs for Alg-Rand with increasing number of chromosomes. The number of iterations required by the algorithm decreases significantly. The time required by the algorithm is linear in the number of chromosomes.

We fixed the number of chromosomes to $N = 50$ and we generated databases with $c = 5$ columns, $k = 5$ class partitions embedded in the data and a number of rows equal respectively to 1, 2 up to 10 times the number of chromosomes. The other parameters for the genetic algorithm remained as in the previous experiment. The fitness measure used in these experiments is $fitness_k^t$. The results are shown in Figures 6.5, 6.6, and 6.7.

With the increase in the number of rows, the classification rate decreases since the search space becomes more complex, the number of iterations and the time increases. The increase in the number of iterations and the time is linear in the size of the database.
Figure 6.5: Classification rate for Alg-Rand with increasing number of chromosomes

Figure 6.6: Number of iterations for Alg-Rand with increasing number of chromosomes

6.4.2 Study of the GA results for different fitness measures

We generated a database with $N = 100$ rows, $n = 10$ columns, and $c = 5$ classes in the attribute and reference partitions. On this database, we ran our
Figure 6.7: Time in secs for Alg-Rand with increasing number of chromosomes
genetic algorithm Alg-Rand, the algorithm Alg-TO and the algorithm k-means
implemented in the WEKA package, searching for a partition with \( k = 5 \) classes.
We executed 10 runs for each algorithm, corresponding to different values for the
seed of the random number generator and we computed the average performance
of the partitions found. For the genetic algorithm we used the fitness measures
based on the quantities presented in table 6.3, 100 chromosomes, a crossover
rate of 0.8, a mutation rate of 0.1, the Gini index and respectively the Shannon
entropy as generators, and 100 consecutive iterations without improvement. The
results are summarized in the table 6.3. The results table contains the entropy
of the intersection partition \( \mathcal{H}^J(\pi_H) \), the average value of the entropies of all
attribute partitions \( \mathcal{H}^J_{\text{avg}} \), the entropy of the reference partition \( \mathcal{H}^J(\pi_{\text{ref}}) \) and its
associated value \( \mathcal{M}^J(\pi_{\text{ref}}) \). For the genetic algorithms we present the average
value of the following measures: \( \mathcal{H}^J(\pi) \), \( \mathcal{M}^J(\pi) \), \( d(\pi, \pi_{\text{ref}}) \) (the distance between
\( \pi \) and \( \pi_{\text{ref}} \)), the classification rate \( \text{cr} \), and the number of iterations \( \text{nits} \). For
the Alg-TO and the k-means algorithms we present only the average value of the
classification rate.

81
<table>
<thead>
<tr>
<th>Alg-Rand with Gini index</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{H}^I_{\text{gini}}(\pi_H) = 0.99$</td>
<td>$\mathcal{H}^I_{\text{gini}}(\pi_{\text{ref}}) = 0.793$</td>
<td>$\mathcal{H}^I_{\text{avg}} = 0.793$</td>
</tr>
<tr>
<td>$\mathcal{H}^I_{\text{gini}}(\pi_{\text{ref}}) = 0.793$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{M}^I(\pi_{\text{ref}}) = 3.561$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>fitness $^I_K$</th>
<th>$\mathcal{H}^I_{\text{gini}}(\pi)$</th>
<th>$\mathcal{M}^I(\pi)$</th>
<th>$d(\pi, \pi_{\text{ref}})$</th>
<th>cr</th>
<th>nitr</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{K}^I(\pi)$</td>
<td>0.784</td>
<td>4.461</td>
<td>0.122</td>
<td>0.911</td>
<td>798</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{M}^I(\pi)$</td>
<td>0.757</td>
<td>4.331</td>
<td>0.111</td>
<td>0.909</td>
<td>704</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{N}^I(\pi)$</td>
<td>0.776</td>
<td>4.224</td>
<td>0.093</td>
<td>0.94</td>
<td>690</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{O}^I(\pi)$</td>
<td>0.787</td>
<td>4.216</td>
<td>0.090</td>
<td>0.938</td>
<td>719</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{P}^I(\pi)$</td>
<td>0.786</td>
<td>4.127</td>
<td>0.080</td>
<td>0.945</td>
<td>745</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{Q}^I(\pi)$</td>
<td>0.793</td>
<td>3.820</td>
<td>0.036</td>
<td>0.99</td>
<td>706</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alg-Rand with Shannon entropy</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{H}^I_{\text{ent}}(\pi_H) = 6.643$</td>
<td>$\mathcal{H}^I_{\text{ent}}(\pi_{\text{ref}}) = 2.298$</td>
<td>$\mathcal{H}^I_{\text{ent}}(\pi_{\text{ref}}) = 2.298$</td>
</tr>
<tr>
<td>$\mathcal{H}^I_{\text{avg}} = 2.300$</td>
<td>$\mathcal{M}^I(\pi_{\text{ref}}) = 10.480$</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{M}^I(\pi_{\text{ref}}) = 10.480$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>fitness $^I_K$</th>
<th>$\mathcal{H}^I_{\text{ent}}(\pi)$</th>
<th>$\mathcal{M}^I(\pi)$</th>
<th>$d(\pi, \pi_{\text{ref}})$</th>
<th>cr</th>
<th>nitr</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{K}^I(\pi)$</td>
<td>2.300</td>
<td>11.102</td>
<td>0.029</td>
<td>0.992</td>
<td>816</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{M}^I(\pi)$</td>
<td>2.054</td>
<td>13.003</td>
<td>0.170</td>
<td>0.867</td>
<td>769</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{N}^I(\pi)$</td>
<td>2.274</td>
<td>11.660</td>
<td>0.068</td>
<td>0.972</td>
<td>780</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{O}^I(\pi)$</td>
<td>2.274</td>
<td>12.365</td>
<td>0.117</td>
<td>0.924</td>
<td>811</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{P}^I(\pi)$</td>
<td>2.279</td>
<td>12.273</td>
<td>0.106</td>
<td>0.934</td>
<td>851</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{Q}^I(\pi)$</td>
<td>2.273</td>
<td>12.337</td>
<td>0.120</td>
<td>0.905</td>
<td>960</td>
<td></td>
</tr>
</tbody>
</table>

Alg-TO Average Classification Rate 0.616
k-means Average Classification Rate 0.824

Table 6.3: Alg-Rand, Alg-TO, and k-means on the first synthetically generated database
The genetic algorithms using fitness measures based on the quantities from table 6.3 lead in all cases to a partition close to the reference partition, in approximately the same number of iterations. The genetic algorithm using the Gini index as generator obtained the best partition in the case of the measure \( Q^I(\pi) \), for which \( M^I(\pi) = 3.820 \) is the closest value to \( M^I(\pi_{\text{ref}}) = 3.561 \), and \( H_{\text{gini}}^I(\pi) = 0.793 \) is the closest value to \( H_{\text{avg}}^I = 0.793 \). For this quality measure the average classification rate was 0.99, leading to a very good approximation of the searched partition, in which 99 elements have been classified in average as in the reference partition. When the Shannon entropy was used as generator, the best partition was found by using the measure \( K^I(\pi) \); in this case also the associated value \( M^I(\pi) = 11.102 \) is the closest value to \( M^I(\pi_{\text{ref}}) = 10.480 \), and \( H_{\text{entr}}^I(\pi) = 2.300 \) is the closest value to \( H_{\text{avg}}^I = 2.298 \). For the measure \( K^I(\pi) \), the genetic algorithm has also a high value of the average classification rate equal to 0.992, meaning that 99 elements have been grouped as in the reference partition.

We are searching for a partition \( \pi \) having well represented classes, that is we do not have any reason to assume that some classes have significantly more elements than the others, and we can estimate the number of classes \( k \) of this partition using the value of the average entropy of the attribute partitions. When the Gini index is used as generator, the value of \( k \) is given by \( \left\lceil \frac{1}{1 - 3H_{\text{avg}}^I} \right\rceil = \left\lceil 4.85 \right\rceil = 5 \), and when the Shannon entropy is used, we have \( k = \left\lceil 2^{3H_{\text{avg}}^I} \right\rceil = \left\lceil 2^{2.3} \right\rceil = 5 \). This estimation corresponds to the real number of classes of the reference partition.

The algorithm Alg-TO converged toward a partition that is far from the reference partition, having in average a classification rate of 0.616. Alg-TO favors partitions with fewer than \( k \) classes and thus, with larger class cardinalities.
The k-means algorithm from the WEKA package converged to better partitions than Alg-TO, having an average classification rate of 0.824, but still did not outperformed our genetic algorithms.

We can conclude that on this database, our genetic algorithms produce better clusterings than the classical “transfer points” and k-means algorithms.

A second type of synthetically generated databases are characterized by attribute partitions with a predominant class. The generation process differs from the one described previously, in the fact that after the first \(c\) positions have being filled with the sequence \(1, 2, \ldots, c\), the number \(i\) to be filled in the reference and all attribute partitions, is set to 1 for any two other rows. Thus, the reference partition and the partitions associated with the attributes of the database have the class 1 with the largest cardinality value.

The parameters of the genetic algorithms and the setting of the experiments remain the same as in the previous ones. The results are summarized in table 6.4.

The partitions found by the genetic algorithms using both the Gini index and Shannon entropy as generators, are very close to the reference partition for the quality measures \(M_I^f(\pi)\) and \(N_I^f(\pi)\). The best partition is found in case of the Gini index for \(M_I^f(\pi)\), when \(M_I^f(\pi) = 4.380\) is the closest value to \(M_I^f(\pi_{\text{vert}}) = 3.353\), and in case of the Shannon entropy for \(N_I^f(\pi)\), when \(M_I^f(\pi) = 11.580\) is the closest value to \(M_I^f(\pi_{\text{vert}}) = 9.600\). The best classification rate for the genetic algorithms is in average 0.895 for the Gini index and 0.927 for the Shannon entropy.

For this database, the algorithm Alg-TO performed better than in the previous experiment, but still the average classification rate of 0.82 is smaller than the average classification rate obtained by the genetic algorithm using \(M_I^f(\pi)\) and \(N_I^f(\pi)\) as quality measures. The better performance of Alg-TO for this database
### Alg-Rand with Gini index

<table>
<thead>
<tr>
<th></th>
<th>(\mathcal{H}<em>g^{l</em>{gini}}(\pi_H) = 0.99)</th>
<th>(\mathcal{H}<em>g^{l</em>{gini}} = 0.660)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\mathcal{H}<em>g^{l</em>{gini}}(\pi_{\text{ref}}) = 0.623)</td>
<td>(\mathcal{M}^l(\pi_{\text{ref}}) = 3.353)</td>
</tr>
<tr>
<td>(f_{itk})</td>
<td>(\mathcal{H}<em>g^{l</em>{gini}}(\pi))</td>
<td>(\mathcal{M}^l(\pi))</td>
</tr>
<tr>
<td>(\mathcal{K}^l(\pi))</td>
<td>0.767</td>
<td>6.921</td>
</tr>
<tr>
<td>(\mathcal{M}^l(\pi))</td>
<td>0.614</td>
<td>4.380</td>
</tr>
<tr>
<td>(\mathcal{N}^l(\pi))</td>
<td>0.599</td>
<td>4.478</td>
</tr>
<tr>
<td>(\mathcal{O}^l(\pi))</td>
<td>0.660</td>
<td>6.986</td>
</tr>
<tr>
<td>(\mathcal{P}^l(\pi))</td>
<td>0.650</td>
<td>4.861</td>
</tr>
<tr>
<td>(\mathcal{Q}^l(\pi))</td>
<td>0.648</td>
<td>5.087</td>
</tr>
</tbody>
</table>

### Alg-Rand with Shannon entropy

<table>
<thead>
<tr>
<th></th>
<th>(\mathcal{H}<em>g^{l</em>{\text{ent}}}(\pi_H) = 6.643)</th>
<th>(\mathcal{H}<em>g^{l</em>{\text{ent}}} = 1.911)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\mathcal{H}<em>g^{l</em>{\text{ent}}}(\pi_{\text{ref}}) = 1.810)</td>
<td>(\mathcal{M}^l(\pi_{\text{ref}}) = 9.600)</td>
</tr>
<tr>
<td>(f_{itk})</td>
<td>(\mathcal{H}<em>g^{l</em>{\text{ent}}}(\pi))</td>
<td>(\mathcal{M}^l(\pi))</td>
</tr>
<tr>
<td>(\mathcal{K}^l(\pi))</td>
<td>2.156</td>
<td>15.796</td>
</tr>
<tr>
<td>(\mathcal{M}^l(\pi))</td>
<td>1.656</td>
<td>12.181</td>
</tr>
<tr>
<td>(\mathcal{N}^l(\pi))</td>
<td>1.757</td>
<td>11.580</td>
</tr>
<tr>
<td>(\mathcal{O}^l(\pi))</td>
<td>1.879</td>
<td>13.696</td>
</tr>
<tr>
<td>(\mathcal{P}^l(\pi))</td>
<td>1.863</td>
<td>13.745</td>
</tr>
<tr>
<td>(\mathcal{Q}^l(\pi))</td>
<td>1.896</td>
<td>15.285</td>
</tr>
</tbody>
</table>

Alg-TO Average Classification Rate 0.82
k-means Average Classification Rate 0.833

Table 6.4: Alg-Rand, Alg-TO, and k-means on the second type of synthetically generated database
is explained by the tendency of the algorithm to converge toward partitions with fewer classes of larger cardinality.

Again the $k$-means algorithm from the WEKA package has a better classification rate 0.833 than the one of Alg-TO, and is still not as good as the genetic algorithm using the measures $M^f(\pi)$, and $N^f(\pi)$.

Note that the quality measures $N^f(\pi)$ and $P^f(\pi)$ lead to good classification rates regardless of the type of database.

6.4.3 Searching for a partition with a natural number of classes

As we discussed in section 6.2, we can obtain an estimate of the natural number of classes in the input database, by using the value of $H^f_{\text{avg}}$. In the second experiment we work with a database having $H^f_{\text{avg}} = 1.911$. For this value the lower bound estimate of the number of classes gives $k = \lceil 2^{1.911} \rceil = 4$. Using the same database we executed 10 runs of the algorithm Alg-Rand, searching for partitions with $k \in \{3, 4, 5, 6, 7, 8, 9, 10\}$ classes.

We observe from table 6.5 that the value of the quantity $M^f(\pi)$ we search to minimize reaches a minimum for values of $k$ around the real number of classes embedded in the data. A detailed description of the partitions found by Alg-Rand in one of the 10 experiments is given in table 6.6.

The best partition found by Alg-Rand using the quality measure $N^f(\pi)$ and the Shannon entropy is the one obtained for $k = 5$, characterized by the smallest value of $M^f(\pi)$. In this case Alg-Rand obtained exactly the reference partition, as the classification rate of 1 indicates. Even if we started the algorithm by asking for more that the real number of classes, the partitions found by the algorithm have discovered the major classes of the reference partition and the additional classes have smaller cardinalities. This show that our algorithms are robust and
Alg-Rand with $\mathcal{N}^f(\pi)$ and Shannon entropy

<table>
<thead>
<tr>
<th>$\mathcal{H}_{\text{ent}}(\pi_H) = 6.643$</th>
<th>$\mathcal{H}_{\text{ent}} = 1.911$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{H}<em>{\text{ent}}(\pi</em>{\text{ref}}) = 1.810$</td>
<td>$\mathcal{M}^f(\pi_{\text{ref}}) = 9.600$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\mathcal{H}_{\text{ent}}(\pi)$</th>
<th>$\mathcal{M}^f(\pi)$</th>
<th>$d(\pi, \pi_{\text{ref}})$</th>
<th>$c_r$</th>
<th>nitrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.292</td>
<td>13.384</td>
<td>0.242</td>
<td>0.827</td>
<td>581</td>
</tr>
<tr>
<td>4</td>
<td>1.555</td>
<td>12.373</td>
<td>0.163</td>
<td>0.894</td>
<td>663</td>
</tr>
<tr>
<td>5</td>
<td>1.757</td>
<td>11.580</td>
<td>0.109</td>
<td>0.927</td>
<td>850</td>
</tr>
<tr>
<td>6</td>
<td>1.885</td>
<td>10.712</td>
<td>0.051</td>
<td>0.967</td>
<td>998</td>
</tr>
<tr>
<td>7</td>
<td>1.952</td>
<td>11.102</td>
<td>0.068</td>
<td>0.959</td>
<td>1178</td>
</tr>
<tr>
<td>8</td>
<td>2.051</td>
<td>11.632</td>
<td>0.093</td>
<td>0.946</td>
<td>1080</td>
</tr>
<tr>
<td>9</td>
<td>2.148</td>
<td>12.456</td>
<td>0.133</td>
<td>0.919</td>
<td>1349</td>
</tr>
<tr>
<td>10</td>
<td>2.295</td>
<td>13.615</td>
<td>0.192</td>
<td>0.876</td>
<td>1345</td>
</tr>
</tbody>
</table>

Table 6.5: Alg-Rand with $\mathcal{N}^f(\pi)$, Shannon entropy and various $k$ on the second type of synthetically generated database

they are able to discover the real clustering of the data even if they are asked for more than the necessary number of classes.

When the number of classes in the natural clustering is not known, we have to run a couple of experiments with increasing values of $k = |\pi|$, and to consider both the shape of the partition $\pi$ and its associated value $\mathcal{M}^f(\pi)$. The partition with the smaller value of $\mathcal{M}^f(\pi)$ and the fewest number of small cardinality classes candidates for the best approximation of the median partition.
\[
\begin{array}{|c|c|c|c|c|}
\hline
 & \pi_{\text{ref}} \\
\hline
k & \text{class card} & \mathcal{H}_{\text{intr}}(\pi) & \mathcal{M}(\pi) & \text{cr} \\
\hline
5 & 57 17 10 9 7 & 1.810 & 9.600 & \\
\hline
\end{array}
\]

Alg-Rand with \( \mathcal{N}(\pi) \) and Shannon entropy

\[
\begin{array}{|c|c|c|c|c|}
\hline
k & \text{class card} & \mathcal{H}_{\text{intr}}(\pi) & \mathcal{M}(\pi) & \text{cr} \\
\hline
3 & 57 33 10 & 1.322 & 13.047 & 0.84 \\
4 & 58 19 16 7 & 1.602 & 11.546 & 0.9 \\
5 & 57 17 10 9 7 & 1.810 & 9.600 & 1 \\
6 & 56 17 16 10 1 & 1.724 & 11.365 & 0.92 \\
7 & 56 17 10 9 7 1 & 1.882 & 10.196 & 0.99 \\
8 & 57 17 10 8 7 1 & 1.855 & 9.932 & 0.99 \\
9 & 53 16 10 9 7 4 1 & 2.074 & 11.856 & 0.95 \\
10 & 50 10 9 9 8 7 7 & 2.286 & 13.441 & 0.85 \\
\hline
\end{array}
\]

Table 6.6: Characteristics of the partitions found by Alg-Rand with \( \mathcal{N}(\pi) \) on the second type of synthetically generated database

6.5 Conclusions

We demonstrated the use of the notion of generalized entropy for designing a genetic algorithm to find clusters in a given data and experimented with different fitness measures for the evaluation of the chromosomal population. Our experiments showed that, with very few exceptions, the algorithms converged to the clustering that we embedded in the synthetically generated databases, and, when they didn’t converge, then they found a clustering that was close to the hidden one. The number of iterations required by our algorithms scales linearly with the size of the database.
Using the dissimilarity measure $\mathcal{K}^f(\pi) + \mathcal{L}^f(\pi)$, it is possible to determine the number of clusters embedded in the data in the absence of any apriori knowledge. As shown in this paper the information-theoretical measures proved successful in finding the natural clustering of the data.
CHAPTER 7

Finding the natural clustering

In a categorical database, each attribute determines a clustering of the set of
tuples, into a number of clusters, equal to the number of distinct values. Thus,
each attribute determines a partition of the set of tuples. These attribute parti-
tions can be regarded as clusterings of the input data, corresponding to different
criteria. We are interested to combine these criteria, and obtain an unifying
clustering, whose clusters will reflect as much as possible the same relationships
between tuples as they are in the individual attribute partitions. When the
attribute partitions play an equal role in the determination of the unifying clus-
tering, we are searching for the median partition of the set of attribute partitions,
as discussed in the previous chapter. In this chapter, we focus on finding a differ-
ent clustering, denoted as a natural clustering of the input data. The attribute
partitions have a different importance, or relevance, in the process of constructing
the structure of the natural clustering. The importance of each attribute can be
specified by a domain expert, or it can be estimated from a sample database. We
present a clustering algorithm based on an information-theoretical definition of
dissimilarities between partitions, that uses for finding an acceptable approxima-
tive natural clustering, a genetic algorithm approach. We tested our method on
databases for which the clustering of the rows is known in advance and we show
that our proposed method finds the natural clustering of the data with a good
classification rate, better than that of the classical algorithm $k$-means.

90
7.1 Introduction

Clustering a set of objects with categorical attributes presents a special challenge since natural distances (like Euclidean or Manhattan distances) between objects are not available. As we shall see, the Hamming distance (that gives the number of disagreements between the values associated with two objects) is a blunt instrument that generates clusterings of poor quality. Previous research in clustering categorical data include [VJR99, GRS00, Hua97, HKK97].

The application of genetic algorithms to grouping problems was investigated in [Hol92, Mic99, Fal99, Mit97, JB91, EM97]. Our contribution, which results in better performing algorithms, is centered around a measure of dissimilarity between partitions of a set of objects, introduced in chapter 3. To search for the best clustering of the data we use a genetic algorithm approach, where the chromosomes represent possible clustering solutions and the fitness measure captures the dissimilarity between the partition associated with a chromosome and the partitions determined by the attributes of the input database. The proposed clustering process involves two phases:

1. Initially, we estimate the influence of each attribute on the decision to place an object in a specific cluster. This estimation can be obtained from a domain expert or by using a training set of objects.

2. In the second phase, our algorithm searches for a clustering of the entire set of objects such that the attributes of the objects influence the clustering to the extent obtained in the first phase.

To verify the quality of the clustering obtained through our techniques and to expedite the first phase, we tested our algorithms on databases for which a classification of the set of objects is already known. For a broad class of databases,
which we characterize in the paper, our approach yields results that are superior to clusterings obtained with classical clustering methods (such as k-means).

A clustering of the tuples of a table can be regarded as a partition of the set of tuples. Also, each set of attributes $X$ of a table $T$ determines a partition $\pi^T_X$ of the set of tuples such that two tuples belong to the same block of the partition if they have equal projections on $X$. A dissimilarity between partitions of a set is defined using a generalization of conditional entropy. This dissimilarity helps us estimate the influence of the attributes on the clustering.

We obtain good results on databases for which there is a strong relationship between attributes and the “natural” clustering of the data, and we propose a method to determine if the given database is suitable or not to be well-classified by our approach.

### 7.2 Clustering using genetic algorithms

Let $T : R \times H \rightarrow \bigcup D_H$ be a table. A training table for $T$ is a table $T_s : R_s \times (H \cup \{C\}) \rightarrow \bigcup D_H \cup D_C$, where $C$ is an additional attribute that specifies the class of each row of $R_s$.

If $R_s \subseteq R$, then $T_s$ is referred to as a sample table.

The training table can be used to determine the influence of each attribute on the clustering. We adopt as a measure of this influence the weight:

$$w(A, \pi^T_C) = \mathcal{H}^I(\pi^T_A | \pi^T_C) + \mathcal{H}^I(\pi^T_C | \pi^T_A).$$

The weight $w(A, \pi^T_C)$ captures the dissimilarity between the partitions $\pi^T_A$ and $\pi^T_C$ and thus, the influence of attribute $A$ on the classes of the natural clustering. Attributes with small values of $w(A, \pi^T_C)$ have a significant impact on determining the structure of the natural clustering. Small values for $w(A, \pi^T_C)$ indicate that
each class of the partitions $\pi_K^T_A$ and $\pi_C^T$ has a high degree of “purity” relative to the other partition.

Our goal is to find a partition $\pi$ of the set of rows $R$, that has no more classes that a certain prescribed limit $k$, such that the weights of the attributes $w(A, \pi) = \mathcal{H}l(\pi_A^T | \pi) + \mathcal{H}l(\pi | \pi_A^T)$ are as close as possible to the estimated weights $w(A, \pi_K^T)$. We use the GA introduced in chapter 5.

7.3 Fitness measure

The fitness of a chromosome $K$ is based on the similarity between the weights $w(A, \pi_K^T)$ of attributes $A \in H$ and the estimated weights $w(A, \pi_C^T)$, where $\pi_K^T$ is the partition determined by each chromosome $K$. The closer the weights $w(A, \pi_K^T)$ are to $w(A, \pi_C^T)$, the closer the partition $\pi_K^T$ is to the natural clustering of the data. Thus, we seek to determine a partition $\pi_K^T$ of the set of rows $R$ of the table $T$ such that:

$$\frac{w(A_1, \pi_K^T)}{w(A_1, \pi_C^T)} = \frac{w(A_2, \pi_K^T)}{w(A_2, \pi_C^T)} = \cdots = \frac{w(A_n, \pi_K^T)}{w(A_n, \pi_C^T)}$$

We denote by

$$g(\pi_K^T, \pi_C^T) = \frac{\sum_{A \in H} w(A, \pi_K^T)}{\sum_{A \in H} w(A, \pi_C^T)}$$

a global dissimilarity measure between the weights $w(A, \pi_K^T)$ and $w(A, \pi_C^T)$, for all attributes in $H$.

The fitness measure is based on the quantity $W^f(\pi_K^T) = \sum_{A \in H} | w(A, \pi_K^T) - w(A, \pi_C^T) \cdot g(\pi_K^T, \pi_C^T) |$. The smaller the quantity $W^f(\pi)$, the closer the partition $\pi_K^T$ is to the natural clustering of the data. Chromosomes representing partitions that are closer to the natural clustering should have larger values of their fitness.
Thus, we need to convert smaller values of the quantity \( W^f(\pi_K^T) \) into larger values of the associated fitness. This conversion can be done by defining the fitness measure as follows:

\[
\text{fitness}_{\text{new}}^f(K) = \begin{cases} 
\mu, & \text{if } W^f(\pi_K^T) = 0 \\
\left(W^f(\pi_K^T)\right)^{-1}, & \text{otherwise}
\end{cases}
\]

for every chromosome \( K \), whose associated partition is \( \pi_K^T \). \( \mu \) represents the fitness threshold, specified by the user.

### 7.4 Training for clustering

Our clustering approach is suitable for databases that have a strong relationship between the attribute partitions and the natural clustering of the data. To evaluate this relationship, we compute the weights \( w(A, \pi_A^{T_s}) \) using the training table \( T_s \). The value \( w(A, \pi_A^{T_s}) \) consists of the sum of two quantities \( \mathcal{H}^f(\pi_C^{T_s}|\pi_A^{T_s}) \) and \( \mathcal{H}^f(\pi_A^{T_s}|\pi_C^{T_s}) \). We will focus on the first quantity \( \mathcal{H}^f(\pi_C^{T_s}|\pi_A^{T_s}) \), that represents the generalized conditional entropy between the partitions \( \pi_A^{T_s} \) and \( \pi_C^{T_s} \). Since \( 0 \leq \mathcal{H}^f(\pi_C^{T_s}|\pi_A^{T_s}) \leq \mathcal{H}^f(\pi_C^{T_s}) \), the closer \( \mathcal{H}^f(\pi_C^{T_s}|\pi_A^{T_s}) \) is to 0, the closer the partition \( \pi_A^{T_s} \) is to the partition \( \pi_C^{T_s} \). Conversely, the closer \( \mathcal{H}^f(\pi_C^{T_s}|\pi_A^{T_s}) \) is to \( \mathcal{H}^f(\pi_C^{T_s}) \), the more distant the partition \( \pi_A^{T_s} \) is from the partition \( \pi_C^{T_s} \). Thus, when the value of the expression \( \mathcal{H}^f(\pi_C^{T_s}) - \mathcal{H}^f(\pi_C^{T_s}|\pi_A^{T_s}) \) is closer to 0, the importance of attribute \( A \) in the process of finding the natural clustering of the data is smaller. Another interpretation of the value \( \mathcal{H}^f(\pi_C^{T_s}) - \mathcal{H}^f(\pi_C^{T_s}|\pi_A^{T_s}) \) is the following: if attribute \( A \) has no relationship with the natural classes, then knowing the class of the attribute \( A \) brings no useful information in the process of determining the class of the partition \( \pi_A^{T_s} \), and in this case, the value of the quantity \( \mathcal{H}^f(\pi_C^{T_s}) - \mathcal{H}^f(\pi_C^{T_s}|\pi_A^{T_s}) \) is close to 0.
Given a training database, we compute for each attribute the values $\mathcal{H}^f(\pi_C^T) - \mathcal{H}^f(\pi_C^T | \pi_A^T)$ and the quantity $\text{ind}^f_{T_A} = \max_{A \in B} \frac{\mathcal{H}^f(\pi_C^T) - \mathcal{H}^f(\pi_C^T | \pi_A^T)}{\mathcal{H}^f(\pi_C^T)}$. This last quantity is an indicator of how suitable is the given database to the proposed clustering algorithm. If the value $\text{ind}^f_{T_A}$ is very small, than there is no point in applying our algorithm, since there is no relationship between the attribute partitions and the partition associated with the natural clustering of the data.

### 7.5 Experimental results

We tested the proposed clustering algorithm on different categorical databases from the UCI Machine Learning Repository (see [BM98]). These databases have an attribute, denoted target attribute, whose values represent the classes of the natural clustering of the rows. We extracted a sample database, from the input database, by randomly choosing a certain number of rows, computed in terms of an input parameter of our algorithm, which specifies the percent of the original database to be used for sampling. In all our experiments, we used a sample database of 40% of the input database.

To test the proposed algorithm we first analyzed the datasets and then we continued with the search for the natural clustering only on the suitable databases.

Using the sample database, we estimate the relationship between the partition determined by each attribute and the partition determined by the target attribute. As explained in the previous section, for each attribute $A$ different than the target attribute, we compute the value $\mathcal{H}^f(\pi_C^T) - \mathcal{H}^f(\pi_C^T | \pi_A^T)$ and the value $w(A, \pi_C^T) = \mathcal{H}^f(\pi_A^T | \pi_C^T) + \mathcal{H}^f(\pi_C^T | \pi_A^T)$. The larger is the quantity $\mathcal{H}^f(\pi_C^T) - \mathcal{H}^f(\pi_C^T | \pi_A^T)$, the stronger is the relationship between attribute $A$ and the target attribute.
The first experiment involved the database zoo.data, containing records for 101 animals, characterized by 17 attributes and classified in 7 classes. We removed the name attribute, since it has unique values for each row in the database, and it does not bring any useful information to the clustering process.

| attribute and its value $\mathcal{H}_{\text{gini}}(\pi_C) - \mathcal{H}_{\text{gini}}(\pi_C | \pi_A)$ | hair  | feathers | 0.224 |
|---------------------------------------------------------------|-------|----------|--------|
| eggs                                                         | 0.264 | milk     | 0.281  |
| airborne                                                     | 0.112 | aquatic  | 0.110  |
| predator                                                    | 0.03  | toothed  | 0.203  |
| backbone                                                    | 0.117 | breathes | 0.161  |
| venomous                                                     | 0.031 | fins     | 0.185  |
| legs                                                        | 0.465 | tail     | 0.095  |
| domestic                                                    | 0.016 | catsize  | 0.115  |

Table 7.1: Estimated attribute influence for zoo.data

The results of the analysis process for zoo.data are summarized in table 7.1. Note that attribute legs plays the most important role in the classification of animals and that attribute domestic has the weakest influence on determining the class of an animal. For this dataset, the value $\text{ind}_{\text{gini}}^{\text{zoo.data}} = \frac{0.465}{0.777} = 0.59$ reflects a strong relationship between attribute partitions and the target partition.

Next, we analyzed soybean-small.data, documenting 4 soybean diseases, having 47 rows and 35 attributes, including also the target attribute. The estimated attribute influences in the clustering process are summarized in table 7.2. Attributes leafspots-halo, leafspots-marg, leafspot-size, leaf-shred, leaf-malf, leaf-mild, stem, fruit-spots, seed, mold-growth, seed-discolor, seed-size, shriveling have only one value in the sample database, so they play no role in the clustering pro-
<table>
<thead>
<tr>
<th>attribute and its value $\mathcal{H}_{\mathcal{L}^\text{ini}}(\pi_T^0 \mid \pi_T^A)$</th>
<th>date</th>
<th>0.296</th>
<th>plant-stand</th>
<th>0.157</th>
<th>precip</th>
<th>0.232</th>
</tr>
</thead>
<tbody>
<tr>
<td>temp</td>
<td>0.309</td>
<td>hail</td>
<td>0.040</td>
<td>crop-hist</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>area-damaged</td>
<td>0.264</td>
<td>severity</td>
<td>0.117</td>
<td>seed-tmt</td>
<td>0.018</td>
<td></td>
</tr>
<tr>
<td>germination</td>
<td>0.101</td>
<td>plant-growth</td>
<td>0</td>
<td>leaves</td>
<td>0.271</td>
<td></td>
</tr>
<tr>
<td>leafspots-halo</td>
<td>0</td>
<td>leafspots-marg</td>
<td>0</td>
<td>leafspot-size</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>leaf-shread</td>
<td>0</td>
<td>leaf-malf</td>
<td>0</td>
<td>leaf-mild</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>stem</td>
<td>0</td>
<td>lodging</td>
<td>0.046</td>
<td>stem-cankers</td>
<td>0.507</td>
<td></td>
</tr>
<tr>
<td>canker-lesion</td>
<td>0.507</td>
<td>fruiting-bodies</td>
<td>0.196</td>
<td>external decay</td>
<td>0.158</td>
<td></td>
</tr>
<tr>
<td>mycelium</td>
<td>0.121</td>
<td>int-disco</td>
<td>0.136</td>
<td>sclerotia</td>
<td>0.136</td>
<td></td>
</tr>
<tr>
<td>fruit-pods</td>
<td>0.209</td>
<td>fruit-spots</td>
<td>0</td>
<td>seed</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>mold-growth</td>
<td>0</td>
<td>seed-discolor</td>
<td>0</td>
<td>seed-size</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>shrveling</td>
<td>0</td>
<td>roots</td>
<td>0.212</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.2: Estimated attribute influence for soybean-small.data

cess and in table 7.2 we see that their influence value is 0. Attributes stem-cankers and canker-lesion play the most important role in determining the type of soybean disease. The value $\text{ind}_{\text{soybean-small.data}} = \frac{0.507}{0.685} = 0.74$ indicates also a strong connection between attributes and the target attribute.

We also analyzed the database house-votes-84.data, documenting Congressional Voting Records from year 1984, and having 435 votes on 16 issues. Each row is characterized by a classification label of democrat or republican. The results of the analysis are summarized in table 7.3. Attribute el-salvador-aid has the greatest importance in the democrat/republican classification and attribute adoption-of-the-budget-resolution the smallest importance. For the house-votes-84.data database, we computed also the value of the quantity $\text{ind}_{\text{house-votes-84.data}} =$
attribute and its value $\mathcal{H}^{\text{init}}_g(T_r^C) - \mathcal{H}^{\text{init}}_g(T_r^C | \pi_A^r)$

<table>
<thead>
<tr>
<th>attribute</th>
<th>score</th>
<th>attribute</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>handicapped infants</td>
<td>0.063</td>
<td>water project cost sharing</td>
<td>0.069</td>
</tr>
<tr>
<td>adoption of the budget resolution</td>
<td>0.006</td>
<td>physician fee freeze</td>
<td>0.299</td>
</tr>
<tr>
<td>el-salvador aid</td>
<td>0.403</td>
<td>religious groups in schools</td>
<td>0.223</td>
</tr>
<tr>
<td>anti satellite test ban</td>
<td>0.093</td>
<td>aid to nicaraguan contras</td>
<td>0.120</td>
</tr>
<tr>
<td>mx-missile</td>
<td>0.165</td>
<td>immigration</td>
<td>0.177</td>
</tr>
<tr>
<td>synfuels corporation cutback</td>
<td>0.011</td>
<td>education spending</td>
<td>0.054</td>
</tr>
<tr>
<td>superfund right to sue</td>
<td>0.221</td>
<td>crime</td>
<td>0.105</td>
</tr>
<tr>
<td>duty-free exports</td>
<td>0.147</td>
<td>export administration act south-africa</td>
<td>0.089</td>
</tr>
</tbody>
</table>

Table 7.3: Estimated attribute influence for house-votes-84.data

\[
\frac{0.403}{0.483} = 0.83
\]

which reflects like in the previous cases, a strong connection between attributes of the database and the target attribute.

Other categorical datasets from the UCI Machine Learning Repository do not have a strong relationship between attribute partitions and the natural clustering of the data. Table 7.4 summarizes the results obtained for these databases.

<table>
<thead>
<tr>
<th>dataset</th>
<th>NROWS</th>
<th>NCOLS</th>
<th>k</th>
<th>\text{ind}^{\text{init}}_{g, \text{dataset}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>car.data</td>
<td>1728</td>
<td>7</td>
<td>4</td>
<td>0.17</td>
</tr>
<tr>
<td>soybean-large.data</td>
<td>307</td>
<td>36</td>
<td>19</td>
<td>0.22</td>
</tr>
<tr>
<td>breast-cancer.data</td>
<td>286</td>
<td>10</td>
<td>2</td>
<td>0.13</td>
</tr>
<tr>
<td>flags.data</td>
<td>194</td>
<td>27</td>
<td>8</td>
<td>0.43</td>
</tr>
</tbody>
</table>

Table 7.4: Estimated attribute influence for various datasets
The second phase of our algorithm represents the actual search for the partition reflecting the natural clustering of the data. In this phase, we applied the proposed clustering method based on a genetic algorithm approach, to the input databases, from which we eliminated the \textit{target} attribute. We denote by $\pi_{\text{target}}$ the partition associated with the \textit{target} attribute as it appears in the input database.

For all the following experiments, the parameters of the genetic algorithm are as follows. The crossover and mutation rate are set to 0.8 and 0.1 respectively. The number of consecutive iterations without improvement is set to 500 and the fitness threshold $\mu$ is set to $10^4$.

To assess the quality of the resulting clustering, we compute a quantity denoted as \textit{classification rate}. We identify a one-to-one mapping of the classes from the resulting partition $\pi$ and the classes from the target partition $\pi_{\text{target}}$. At each step, we maintain a list of classes from $\pi_{\text{target}}$ that have not been yet mapped to any class of $\pi$. Initially, this list contains all the classes from $\pi_{\text{target}}$. Then, for each class of $\pi$ we compute the cardinalities of its intersections with the classes in $\pi_{\text{target}}$, given by the number of tuples they have in common. The pairs consisting of a class from $\pi$ and an available class from $\pi_{\text{target}}$, are processed in decreasing order of their intersection’s cardinality. Each time we retain and map the pair yielding the largest number of tuples in common. Once a class from $\pi_{\text{target}}$ is mapped to a class from $\pi$, it is removed from the list of available classes. When the unique mapping between partitions has been completed, the classification rate is computed as the fraction of all tuples, represented by the sum of tuples shared by the classes mapped to each other.

For each dataset, we executed 30 runs of our genetic algorithm using the Gini index, the Shannon entropy, and $f_{ge}$ as generators, and of the \textit{k}-means clustering
algorithm implemented in the WEKA package ([WF00]). Each experiment used a
different seed value for the random number generator. We searched for partitions
\( \pi_k \) with \( |\pi_k| = k \) classes with \( k = 7 \) for zoo.data, \( k = 4 \) for soybean-small.data
and \( k = 2 \) for house-votes-84.data.

The results for the zoo.data with 100 chromosomes are presented in table 7.5,
where we specify the average classification rate obtained over all 30 runs, and
characteristics of the best partition found, the number of classes, the associated
classification rate and the number of elements classified as in the target attribute.

The best classification rate obtained for the Gini index is 0.86, obtained for a
partition that grouped together 40 out of 41 mammals, all 20 birds, 12 out of 13
fishes, all 8 insects, 2 out of 4 amphibians, 4 out of 10 invertebrates and 1 out of
5 reptiles. For the Shannon entropy the best partition grouped together 39 out
of 41 mammals, all 20 birds, all 13 fishes, all 10 invertebrates, 7 out of 8 insects,
and 1 out of 5 reptiles, leading to a classification rate of 0.89. When \( f_{gs} \) was
used as generator, the best partition found has correctly classified 38 out of 41
mammals, 12 out 13 fishes, all 20 birds, 5 out of 10 invertebrates, all 8 insects, all
4 amphibians and 2 out of 5 reptiles, leading to a classification rate of 0.88. The
best clustering obtained by \( k \)-means classified correctly 90 out of 101 animals,
leading to a classification rate of 0.90.

The best result obtained is similar across all algorithms. Overall, in these
experiments, our approach found a more robust clustering. The average classifi-
cation rate is 0.815 for the Gini index, 0.806 for the Shannon entropy, 0.803 for
\( f_{gs} \), and 0.724 for \( k \)-means. Thus, on average our algorithm finds a more robust
clustering of the input data. Figure 7.1 presents the classification rates obtained.

We repeated the previous tests for soybean-small.data, with the number of
chromosomes set to 50. The results for the genetic algorithms using the Gini

100
<table>
<thead>
<tr>
<th>Gini index as generator</th>
<th>Shannon entropy as generator</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average classification rate:</strong> 0.815</td>
<td><strong>Average classification rate:</strong> 0.806</td>
</tr>
<tr>
<td><strong>Best Partition Characteristics</strong></td>
<td><strong>Best Partition Characteristics</strong></td>
</tr>
<tr>
<td>Cardinalities: 42 20 16 14 5 3 1</td>
<td>Cardinalities: 39 20 20 11 9 2</td>
</tr>
<tr>
<td>Classification rate: 0.86</td>
<td>Classification rate: 0.89</td>
</tr>
<tr>
<td>Elements classified as in ( \pi_{\text{target}} ):</td>
<td>Elements classified as in ( \pi_{\text{target}} ):</td>
</tr>
<tr>
<td>40 out of 41 12 out of 13</td>
<td>39 out of 41 13 out of 13</td>
</tr>
<tr>
<td>20 out of 20 4 out of 10</td>
<td>20 out of 20 10 out of 10</td>
</tr>
<tr>
<td>8 out of 8 2 out of 4</td>
<td>7 out of 8 1 out of 5</td>
</tr>
<tr>
<td>1 out of 5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( f_{ge} ) as generator</th>
<th>( k )-means</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average classification rate:</strong> 0.803</td>
<td><strong>Average classification rate:</strong> 0.724</td>
</tr>
<tr>
<td><strong>Best Partition Characteristics</strong></td>
<td><strong>Best Partition Characteristics</strong></td>
</tr>
<tr>
<td>Cardinalities: 38 20 14 12 8 5 4</td>
<td>Cardinalities: 41 19 14 8 7 6 6</td>
</tr>
<tr>
<td>Classification rate: 0.88</td>
<td>Classification rate: 0.90</td>
</tr>
<tr>
<td>Elements classified as in ( \pi_{\text{target}} ):</td>
<td>Elements classified as in ( \pi_{\text{target}} ):</td>
</tr>
<tr>
<td>38 out of 41 12 out of 13</td>
<td>41 out of 41 13 out of 13</td>
</tr>
<tr>
<td>20 out of 20 5 out of 10</td>
<td>19 out of 20 7 out of 10</td>
</tr>
<tr>
<td>8 out of 8 4 out of 4</td>
<td>6 out of 8 4 out of 4</td>
</tr>
<tr>
<td>2 out of 5</td>
<td>1 out of 5</td>
</tr>
</tbody>
</table>

Table 7.5: Results for zoo.data

index, the Shannon entropy, and \( f_{ge} \) as generators, and for the \( k \)-means algorithm are presented in Table 7.6.
Figure 7.1: Classification rates for *zoo.data*

For all genetic algorithms the best classification rate was 0.97 (only one row was misclassified) and was obtained in the majority of the 30 experiments. The best $k$-means clustering has a classification rate of 1. The average classification rate is 0.960 for Gini index, 0.948 for Shannon entropy, 0.955 for $f_{ge}$, and 0.846 for $k$-means. Again, we can conclude that on average our algorithm performed better than $k$-means. Figure 7.2 presents the classification rate values obtained in all 30 experiments.


<table>
<thead>
<tr>
<th>Gini index as generator</th>
<th>Shannon entropy as generator</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average classification rate:</strong> 0.960</td>
<td><strong>Average classification rate:</strong> 0.948</td>
</tr>
<tr>
<td><strong>Best Partition Characteristics</strong></td>
<td><strong>Best Partition Characteristics</strong></td>
</tr>
<tr>
<td>Cardinalities: 16 11 10 10</td>
<td>Cardinalities: 17 11 10 9</td>
</tr>
<tr>
<td>Classification rate: 0.97</td>
<td>Classification rate: 0.97</td>
</tr>
<tr>
<td>Elements classified as in $\pi_{\text{target}}$:</td>
<td>Elements classified as in $\pi_{\text{target}}$:</td>
</tr>
<tr>
<td>10 out of 10 10 out of 10</td>
<td>10 out of 10 10 out of 10</td>
</tr>
<tr>
<td>10 out of 10 16 out of 17</td>
<td>9 out of 10 17 out of 17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$f_g$ as generator</th>
<th>$k$-means</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average classification rate:</strong> 0.955</td>
<td><strong>Average classification rate:</strong> 0.846</td>
</tr>
<tr>
<td><strong>Best Partition Characteristics</strong></td>
<td><strong>Best Partition Characteristics</strong></td>
</tr>
<tr>
<td>Cardinalities: 17 11 10 9</td>
<td>Cardinalities: 17 10 10 10</td>
</tr>
<tr>
<td>Classification rate: 0.97</td>
<td>Classification rate: 1</td>
</tr>
<tr>
<td>Elements classified as in $\pi_{\text{target}}$:</td>
<td>Elements classified as in $\pi_{\text{target}}$:</td>
</tr>
<tr>
<td>10 out of 10 10 out of 10</td>
<td>10 out of 10 10 out of 10</td>
</tr>
<tr>
<td>9 out of 10 17 out of 17</td>
<td>10 out of 10 17 out of 17</td>
</tr>
</tbody>
</table>

Table 7.6: Results for soybean-small.data

Last, we repeated our experiments on house-votes-84.data, with the number of chromosomes set to 100. The results are presented in table 7.7.

On this dataset the best classification rate obtained using the Gini index was 0.933, and resulted in a partition which grouped together 162 out of the 168 republicans, and 244 out of 267 democrats, so only 29 rows were misclassified. The best partition for Shannon entropy as generator had a classification rate of 0.935, it grouped together 161 out of the 168 republicans and 246 out of 267
democrats, leading to only 28 rows misclassified. The genetic algorithm using $f_{gs}$ as generator classified correctly in the best partition obtained, 152 out of 168 republicans and 256 out of 267 democrats, having a classification rate of 0.937. The best partition found by the $k$-means algorithm had a classification rate of 0.87, since 54 rows are misclassified. The average classification rate is 0.923 for Gini index, 0.920 for Shannon entropy, 0.866 for $f_{gs}$, and 0.894 for $k$-means. We can conclude again that our algorithm performed better than $k$-means.
<table>
<thead>
<tr>
<th>Gini index as generator</th>
<th>Shannon entropy as generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average classification rate: 0.923</td>
<td>Average classification rate: 0.920</td>
</tr>
<tr>
<td>Best Partition Characteristics</td>
<td>Best Partition Characteristics</td>
</tr>
<tr>
<td>Cardinalities: 250 185</td>
<td>Cardinalities: 253 182</td>
</tr>
<tr>
<td>Classification rate: 0.933</td>
<td>Classification rate: 0.935</td>
</tr>
<tr>
<td>Elements classified as in $\pi_{\text{target}}$:</td>
<td>Elements classified as in $\pi_{\text{target}}$:</td>
</tr>
<tr>
<td>162 out of 168 244 out of 267</td>
<td>161 out of 168 246 out of 267</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$f_{ge}$ as generator</th>
<th>$k$-means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average classification rate: 0.894</td>
<td>Average classification rate: 0.866</td>
</tr>
<tr>
<td>Best Partition Characteristics</td>
<td>Best Partition Characteristics</td>
</tr>
<tr>
<td>Cardinalities: 272 163</td>
<td>Cardinalities: 226 209</td>
</tr>
<tr>
<td>Classification rate: 0.937</td>
<td>Classification rate: 0.87</td>
</tr>
<tr>
<td>Elements classified as in $\pi_{\text{target}}$:</td>
<td>Elements classified as in $\pi_{\text{target}}$:</td>
</tr>
<tr>
<td>152 out of 168 256 out of 267</td>
<td>162 out of 168 220 out of 267</td>
</tr>
</tbody>
</table>

Table 7.7: Results for house-votes-84.data

Figure 7.3 presents the classification rate values obtained in all 30 experiments.

The time performance for the genetic algorithms using the generators $f_{\text{Gini}}$, $f_{\text{ent}}$ and $f_{ge}$, computed in average over all 30 experiments are summarized in Table 7.8.

7.6 Conclusions

We proposed a clustering method based on a genetic algorithm that uses information theoretical measures to determine the quality of the resulting clustering.
Figure 7.3: Classification rates for house-votes-84.data

Our algorithm involves two phases. The first phase consists of estimating the degree of influence of each attribute on the natural classification of the rows of the database; the second phase consists of searching for a partition that provides the same degree of influence of the attributes. We showed that the generalized entropy is a powerful tool in assessing the similarity between two partitions, and that we obtained better results than the classical k-means algorithm, for generators as the Gini index, the Shannon entropy, and $f_{gs}$. Our proposed method pro-
<table>
<thead>
<tr>
<th>Database</th>
<th>$f_{gini}$</th>
<th>$f_{ent}$</th>
<th>$f_{ge}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>zoo.data (101 rows, 17 attributes)</td>
<td>265</td>
<td>225</td>
<td>323</td>
</tr>
<tr>
<td>soybean-small.data (47 rows, 35 attributes)</td>
<td>61</td>
<td>71</td>
<td>69</td>
</tr>
<tr>
<td>house-votes-84.data (435 rows, 17 attributes)</td>
<td>435</td>
<td>269</td>
<td>705</td>
</tr>
</tbody>
</table>

Table 7.8: Average time performance of the clustering genetic algorithms using $f_{gini}$, $f_{ent}$ and $f_{ge}$ as generators

...duces best results for databases for which there is a strong relationship between attributes and the natural clustering of the data. To determine this relationship we proposed a measure also based on a generalized conditional entropy. This measure can be used as an indication of how useful is an attribute in the process of determining the class associated with a tuple. Attributes that have small influence in the natural clustering can be ignored from the clustering process, and thus we have a method of reducing the dimensionality of the data without diminishing the quality of the clustering. In our experiments we did not ignore the weakest influence attributes, but this can be done easily, by adding a minor modification to the proposed algorithm.
CHAPTER 8

Conclusions

This work introduces new information-theoretical measures and applies their properties in designing algorithms for three clustering applications.

The Gini index and the Shannon entropy are measures used in many clustering applications. Both measures can be obtained using a concave, sub-additive function. By imposing an additional constraint to this function we obtain a class of functions, called generators, that can be used to generalize the notions of entropy and conditional entropy. These generalizations preserve the same properties as those of the Shannon entropy and conditional entropy measures.

We used the generalized entropy and conditional entropy to cluster categorical attributes databases. A categorical attribute has a discrete domain and determines a partition of the set of rows of the database; the rows having the same value of the attribute represent a block of the this partition. The value of the generalized entropy associated to an attribute partition reflects the clustering property of the attribute. We introduced a dissimilarity between two partitions using the notion of generalized conditional entropy.

The first application of these information-theoretical measures is related to the problem of subspace or projection clustering. Given a database we are interested to find all projections represented by subsets of attributes with good clustering quality. A projection has good clustering quality if its associated gen-
eralized entropy has a value smaller than a threshold value. This threshold is computed based on the maximum entropy of a partition with \( k \) blocks, where \( k \) is specified by the user. The algorithm also receives as input, a value specifying the percent of the attributes to be used in the computation of the clusterable projections. The required number of attributes are retained in increasing order of their entropy, such that the attributes with larger values of their entropies are eliminated. Given a number of blocks and a percent of attributes to be considered, our algorithm will return all subsets of attributes, of maximum cardinality, having approximately the specified number of blocks. To see how the clusters in a projection returned by the algorithm diverge when we look on the rest of the attributes, not belonging to the projection, we computed a measure based on the notion of generalized conditional entropy. If the entropy of the partition determined by the rest of the attributes conditioned on the partition determined by the projection, is small, than we have a clusterable set having also a global clustering quality. Experimental results show that the clusterable sets returned by our algorithm have indeed the number of clusters required by the user, and some of the sets do not diverge too much on the remaining attributes.

The second clustering application is related to the problem of finding the median partition associated with a set of partitions determined by the attributes of a categorical database. A median partition is a partition that is as similar as possible to all attribute partitions. We defined a dissimilarity between partitions using the generalized conditional entropy and we searched for a partition with the minimum sum of dissimilarity between itself and each attribute partition. To search efficiently for the median partition we designed a clustering method using a genetic algorithm approach. A chromosome represents a partition of the set of rows. The new generation of chromosomes is created by applying crossover and mutation operators to the current population of chromosomes. We designed
several fitness measures, based on the notion of generalized conditional entropy. We have experimented with synthetical databases and showed that our algorithm found the median partition or close approximations of it. We also showed that by looking on the shape of the median partitions resulted for different number of clusters $k$, and by checking their values of the total dissimilarity between the median partition and all attribute partition, it is possible to find the optimal value of $k$, representing the number of clusters of the real median partition embedded in the data.

The last application of the information-theoretical measures is a classification problem. In a categorical database, each attribute determines a clustering of the set of tuples, into a number of clusters, equal to the number of distinct values. Thus, each attribute determines a partition of the set of tuples. These attribute partitions can be regarded as clusterings of the input data, corresponding to different criteria. We are interested to combine these criteria, and obtain an unifying clustering, whose clusters will reflect as much as possible the same relationships between tuples as they are in the individual attribute partitions. When the attribute partitions play an equal role in the determination of the unifying clustering, we are searching for the median partition of the set of attribute partitions. In this application, we focused on finding a different clustering, denoted as a natural clustering of the input data. The attribute partitions have a different importance, or relevance, in the process of constructing the structure of the natural clustering. The importance of each attribute can be specified by a domain expert, or it can be estimated from a sample database. We present a clustering algorithm based on an information-theoretical definition of dissimilarities between partitions, that uses for finding an acceptable approximative natural clustering, a genetic algorithm approach. We tested our method on databases for which the clustering of the rows is known in advance and we show that our pro-
posed method finds the natural clustering of the data with a good classification rate, better than that of the classical algorithm \( k \)-means.
REFERENCES


112


114


[NH94] Raymond T. Ng and Jiawei Han. “Efficient and Effective Clustering Methods for Spatial Data Mining.” *Proc. VLDB*, pp. 144–155, 1994.


