Part IV

Clustering Mixed-type Data

There are not many algorithms available for clustering datasets containing mixture of numeric and categorical attributes. In Chapter 7 we present an efficient new algorithm developed by us for clustering large high dimensional mixed-type datasets. The algorithm works based on entropy calculations of clusters using different methods for numeric and categorical attributes.
Chapter 7

SMIC: A Subspace Preferred Mixed Type Data Clustering Technique

7.1 Introduction

Very often, real world databases contain both numeric and categorical attributes, requiring specialized algorithms to cluster such data. Some strategies [And73] for dealing with such problems are:

1. **Partitioning of attributes**: Two parallel but separate analyzes can be performed, one based on numeric attributes and the other based on categorical attributes. Relative weighing of the attribute types and the joint or interactive effects between attributes would be of importance. A systematic and meaningful method of integrating such separate analyses is required.
2. **Conversion of attributes:** Attributes can be converted from one type to another. The choice of how to homogenize the set of attributes should be influenced strongly by which attribute type is the most numerous.

3. **Disagreement indices:** It is necessary to equalize the attributes in some appropriate sense. The attributes can be equalized by attribute-by-attribute disagreement between data objects. When two data objects have identical responses on an attribute there is zero difference or disagreement between them. Within a finite dataset there is a maximum level of observed disagreement on any attribute. If the maximum disagreement is scored as one, then all disagreement on an attribute may be represented by a disagreement index ranging from zero to one. Disagreement on attributes of every type may be expressed in this manner.

Using the third approach we present an efficient algorithm for clustering large high dimensional datasets containing mixture of categorical and numeric attributes. Disagreement between data objects for each attribute is measured by entropy computation. The algorithm can be used for clustering categorical datasets as well. Although the algorithm can be used for clustering datasets with numeric attributes alone it is not recommended, since nature of a numeric dataset is very much different than nature of a categorical or mixed-type dataset and our algorithm is specifically designed for mixed-type datasets. Important features of the algorithm are:

- It provides a solution for the mixed-type attribute clustering problem.
- It produces good quality results efficiently.
- Use of subspace based similarity measure makes the algorithm suitable for clustering high dimensional datasets.
The algorithm is scalable as it uses an incremental algorithm to group similar objects together using only a single pass over the dataset. Then clusters are merged hierarchically to produce desired number of clusters.

Outliers can be handled efficiently.

### 7.2 Related Works

The proposed algorithm can cluster datasets with mixture of categorical and numeric attributes as well as datasets with categorical attributes alone. The number of works available on mixed-type data is small. Some of them are mentioned below. Related works on some categorical clustering algorithms were presented in chapter 6.

Huang [Hua98] extended the \textit{k-means} algorithm to the \textit{k-modes} algorithm to tackle the problem of clustering large categorical datasets in data mining. Further, Huang also combined the \textit{k-modes} algorithm with the \textit{k-means} algorithm resulting in the so-called \textit{k-prototypes} algorithm for clustering objects described by mixed numeric and categorical attributes. However, \textit{k-prototypes} also produce locally optimal results like \textit{k-means}.

A clustering algorithm for mixed-type data was proposed by Le [LH03]. The algorithm chooses \( k \) number of largest sets from non-expandable strongly connected sets, which had been built by using breadth first search algorithm. The remaining objects are assigned to some clusters by testing the minimum distance of the object with all clusters. Problem with this algorithm is that it may produce less than \( k \) number of clusters initially. Also, some clusters are reprocessed.

In [YTRC05], cluster ensemble approach based on divide-and-conquer technique is presented for clustering mixed type datasets. First, the original mixed dataset is divided into two sub-datasets: the pure categorical dataset and pure numeric dataset. Next, existing well established clustering algorithms designed
for different types of datasets are employed to produce corresponding clusters. Last, the clustering results on the categorical and numeric datasets are combined as a categorical dataset, on which the categorical data clustering algorithm is used to get the final clusters.

An algorithm for clustering mixed type data is presented in [HXd]. It uses a CF*-tree to pre-cluster datasets. Then the dense regions stored in the leaf nodes are treated as single points and $k$-prototype algorithm is used to cluster such points.

### 7.3 Problem Formulation

For each individual attribute of the dataset, entropy is calculated and normalized to the range $[0, 1]$. Different methods are used for calculating entropy for numeric and categorical attributes. Computing a dissimilarity measure of individual attributes based on normalized entropy values causes homogenization of the attributes. Then a subspace-based similarity measure is defined for an individual cluster. To achieve scalability in clustering large high dimensional datasets, an incremental method of clustering is to be used avoiding storage of the data objects in the main memory. Therefore, a cluster summary measure is defined based upon which the similarity of the cluster obtained by merging an object or another cluster to an existing cluster can be computed easily. A cluster structure is the ultimate data structure to be stored in the main memory.

The given set of $n$ objects $X = \{X_1, X_2, \cdots, X_n\}$ is described by $d$ attributes $A_1, A_2, \cdots, A_d$ that may be either numeric or categorical. For each $i$ ($1 \leq i \leq n$) and for each $j$ ($1 \leq j \leq d$), $x_{ij}$ represent the $j$-th component of object $X_i$ and $x_{ij}$ take on one of the possible values defined in domain $D_j$ of attribute $A_j$. An attribute $A_j$ is categorical if its domain $D_j$ is discrete valued, ordered or unordered while $A_j$ is numeric if its domain is continuous valued and ordered. An object $X_i$
will be represented by its object-id $i$ and an attribute $A_j$ will be represented by its attribute-id $j$ so that a set of objects $T \subseteq \{1, 2, 3, \ldots, n\}$ together with a set of attributes represent a subspace based cluster $C$. Each attribute $A_j$ of a cluster $C$ can be treated as a random variable taking a number of possible values defined in its domain $D_j$. So, entropy of each attribute and hence entropy of a cluster can be computed. The dataset will be partitioned into a set of disjoint clusters and a set of outliers.

### 7.3.1 Entropy

Suppose that a probabilistic experiment involves the observation of a discrete random variable $Y$. Let, $D_Y = \{y_1, y_2, \ldots, y_T\}$ is the set of $T$ possible values that $Y$ can take on and probability of $Y = y_i$ is $p_{y_i}, 1 \leq i \leq T$ so that

$$p_{y_1} + p_{y_2} + \cdots + p_{y_T} = 1.$$  \hfill (7.1)

It is assumed that all $p_{y_i}$ are strictly greater than zero. Then entropy, $H_Y$ of the random variable $Y$, is to be interpreted as the average uncertainty associated with the events ($Y = y_i$). It is defined in [Ash90] as:

$$H_Y = -\sum_{i=1}^{T} p_{y_i} \log_2(p_{y_i}).$$  \hfill (7.2)

$H_Y$ is a bounded variable. Its lower value is 0 and upper value is $\log_2(T)$. The joint entropy of $m$ independent random variables $Y_1, Y_2, \ldots, Y_m$ is obtained as

$$H(Y_1, Y_2, \ldots, Y_m) = H_{Y_1} + H_{Y_2} + \cdots + H_{Y_m}.$$  \hfill (7.3)

### 7.3.2 Entropy of a categorical attribute

Let the domain of a categorical attribute $A_j, 1 \leq j \leq d$ be represented as:

$$D_j = \{v_{j1}, v_{j2}, \ldots, v_{jd_j}\}$$  \hfill (7.4)

94
where, $v_{jk}$ denotes the $k$-th category and $d_j$ is the total number of categories in $D_j$
i.e. $d_j = |D_j|$. Let, $t_{jk}(C)$ denote the frequency of occurrence of attribute value $v_{jk}, 1 \leq k \leq d_j$ in a given cluster $C$ consisting of set of objects $T$, so that

$$t_{jk}(C) = |\{ t \mid t \in T, x_{ij} = v_{jk} \}|$$ (7.5)

Frequency of occurrence $t_{jk}$ divided by total number of objects in the cluster gives the probability $p_{jk}$ of category $v_{jk}$, that is

$$p_{jk}(C) = \frac{t_{jk}(C)}{n_C}$$ (7.6)

where $n_C = |T|$ denotes the total number of objects in cluster $C$. So, entropy of attribute $A_j$ for the cluster $C$, denoted by $H_j(C)$ is computed as

$$H_j(C) = -\sum_{k=1}^{d_j} p_{jk}(C) \log_2(p_{jk}(C)).$$ (7.7)

It is assumed here that $0 = \log_2(0)$. The maximum value that $H_j(C)$ can attain is $\log_2(d_j)$. This value is achieved when all $d_j$ categories of the attribute $A_j$ are present in the cluster with equal frequencies. This entropy is to be minimized for better clusters.

### 7.3.3 Entropy for a numerical attribute

A numeric attribute takes continuous values. Therefore, method of calculating entropy is somewhat different than for categorical attributes. Given a cluster $C$ with $n_C$ objects, a numeric attribute $A_j$ can be thought of as a random variable with $n_C$ possible values. Hence, entropy can be computed with $n_C$ probabilities as shown below. We assume that the dataset is preprocessed such that values taken by all numeric attributes are positive (nonzero) real numbers. Zero values, if present, can be replaced by a very small positive quantity less than all valid
values in the dataset. Let,

\[ \text{sum}_j(C) = \sum_{k \in T} x_{kj} \]  \hspace{1cm} (7.8)

represent the sum of the \( j \)-th attribute values of the set of objects \( T \) in the cluster. Probability of each value \( x_{kj}, k \in T \) taken by \( j \)-th attribute becomes \( \frac{x_{kj}}{\text{sum}_j(C)} \), and hence entropy \( H_j(C) \) is computed as:

\[ H_j(C) = -\sum_{k \in C} \frac{x_{kj}}{\text{sum}_j(C)} \log_2 \left( \frac{x_{kj}}{\text{sum}_j(C)} \right). \]  \hspace{1cm} (7.9)

The maximum value that \( H_j(C) \) can take is \( \log_2(n_C) \). This value is attained when all of the \( n_C \) values of the numeric attribute are the same. Which means that entropy is more when the data values are uniform and less when data values are more random. So, for better clusters entropy for numeric attributes need to be maximized. This is opposite to the entropy for categorical attributes where the entropy is to be minimized.

### 7.3.4 Dissimilarity measure of a cluster

Attributes are homogenized by computing dissimilarity measures for individual attributes based on normalized entropy values. Let, \( G_j(C) \) indicates dissimilarity of \( j \)-th attribute in the cluster \( C \). It is computed as,

\[ G_j(C) = \begin{cases} \frac{H_j(C)}{\log_2(n_C)} & \text{if } A_j \text{ is categorical} \\ 1 - \frac{H_j(C)}{\log_2(n_C)} & \text{if } A_j \text{ is numeric} \end{cases} \]  \hspace{1cm} (7.10)

The entropy computed for an attribute is divided by maximum possible entropy so that entropy of each attribute is normalized in the range \([0,1]\). Higher entropy of a categorical attribute indicates that the objects are more dissimilar, whereas higher entropy for numeric attributes indicate similar objects. So, to obtain a dissimilarity measure entropy of numeric attributes are subtracted from one.
Dissimilarity measure for a cluster $C$, $G(C)$ is obtained by summing the dissimilarity measures for each attribute. That is

$$G(C) = \sum_{j=1}^{d} G_j(C). \quad (7.11)$$

Dissimilarity measures need to be minimized for better clustering.

### 7.3.5 Subspace based similarity measure

We define the similarity measure, $S(C)$ of a cluster to be the count of attributes for which the dissimilarity measures are nearly equal to zero:

$$S(C) = |\{ j \mid G_j(C) \leq \epsilon, j \in \{1, 2, \cdots, d\} \}| \quad (7.12)$$

where, $\epsilon$ is a very small quantity (for example 0.0001).

### 7.3.6 Summary measures

A cluster summary measure,

$$F(C) = \{F_1(C), F_2(C), \cdots, F_d(C)\} \quad (7.13)$$

consists of attribute summary measures, $F_j(C)$ for each of the $d$ attributes. Summary measure of a new cluster, obtained by merging two existing clusters, can be computed easily from the summary measures of the existing clusters.

**Summary measure for a categorical attribute**

Let, $F_j(C)$ represents the summary measure of a categorical attribute $A_j$ for a given cluster $C$. The attribute value frequencies defined in Equation 7.5 provide the required summary measure. That is,

$$F_j(C) = \{t_{j1}(C), t_{j2}(C), \cdots, t_{jd}(C)\}. \quad (7.14)$$
Summary measure $F_j(C_p \cup C_q)$ for the merger of two clusters $C_p$ and $C_q$ is:

$$F_j(C_p \cup C_q) = \{t_{j1}(C_p) + t_{j1}(C_q), \ t_{j2}(C_p) + t_{j2}(C_q), \ \cdots, \ t_{jd}(C_p) + t_{jd}(C_q)\}$$

(7.15)

Summary measure for a numeric attribute

Summary measure for a numeric attribute $A_j$ in cluster $C$ contains only two entries which are nothing but the sum and entropy of the attribute values defined by Equations 7.8 and 7.9 respectively.

$$F_j(C) = \{\text{sum}_j(C), \ H_j(C)\}. \quad \text{(7.16)}$$

Given the summary measures for two clusters $C_p$ and $C_q$, summary measure $F_j(C_p \cup C_q)$ is calculated as shown below.

$$\text{sum}_j(C_p \cup C_q) = \text{sum}_j(C_p) + \text{sum}_j(C_q) \quad \text{(7.17)}$$

$$H_j(C_p \cup C_q) = Z_1\{H_j(C_p) - \log_2(Z_1)\} + Z_2\{H_j(C_q) - \log_2(Z_2)\}. \quad \text{(7.18)}$$

where, $Z_1 = \frac{\text{sum}_j(C_p)}{\text{sum}_j(C_p \cup C_q)}$ and $Z_2 = \frac{\text{sum}_j(C_q)}{\text{sum}_j(C_p \cup C_q)}$. Thus,

$$F_j(C_p \cup C_q) = \{\text{sum}_j(C_p \cup C_q), \ H_j(C_p \cup C_q)\} \quad \text{(7.19)}$$

### 7.3.7 Cluster structure

During the clustering process the cluster summary measure need to be maintained along with the list of objects, $T \subseteq \{1, 2, \cdots, n\}$. Total number of objects in the cluster ($n_C$) and similarity measure ($S$) can also be stored for computational efficiency. So creation of a new cluster $C$ means the creation of the following structure:

$$C = \{n_C, \ S, \ T, \ F\} \quad \text{(7.20)}$$
where $S$ and $F$ are given by Equations 7.12 and 7.13 respectively.

**Example:** A sample dataset shown in Table 7.1 consisting of 8 records defined over five categorical and two numeric attributes. Domains of the categorical attributes are: $D_1 = \{b, a\}$, $D_4 = \{u, y, l, t\}$, $D_5 = \{g, p, gg\}$, $D_6 = \{t, f\}$ and $D_7 = \{g, p, s\}$. Domains of numeric attributes are positive real numbers. Consider a cluster consisting of three records ($n_C = 3$) and $T = \{1, 4, 6\}$. The

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$A_4$</th>
<th>$A_5$</th>
<th>$A_6$</th>
<th>$A_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>19.40</td>
<td>0.75</td>
<td>u</td>
<td>g</td>
<td>t</td>
<td>s</td>
</tr>
<tr>
<td>b</td>
<td>21.17</td>
<td>0.25</td>
<td>y</td>
<td>p</td>
<td>f</td>
<td>g</td>
</tr>
<tr>
<td>b</td>
<td>17.50</td>
<td>22.00</td>
<td>l</td>
<td>gg</td>
<td>t</td>
<td>p</td>
</tr>
<tr>
<td>b</td>
<td>19.17</td>
<td>0.01</td>
<td>y</td>
<td>p</td>
<td>t</td>
<td>s</td>
</tr>
<tr>
<td>b</td>
<td>21.25</td>
<td>1.50</td>
<td>u</td>
<td>g</td>
<td>f</td>
<td>g</td>
</tr>
<tr>
<td>a</td>
<td>18.78</td>
<td>0.38</td>
<td>l</td>
<td>gg</td>
<td>t</td>
<td>s</td>
</tr>
<tr>
<td>b</td>
<td>33.67</td>
<td>1.25</td>
<td>u</td>
<td>g</td>
<td>f</td>
<td>g</td>
</tr>
<tr>
<td>b</td>
<td>26.75</td>
<td>4.50</td>
<td>y</td>
<td>p</td>
<td>f</td>
<td>g</td>
</tr>
</tbody>
</table>

Cluster structure for this cluster is shown in Table 7.2. Summary measures $F_1$, $F_4$, $F_5$, $F_6$ and $F_7$ for categorical attributes consists of two, four, three, two and three entries respectively as their domain sizes contain corresponding number of elements. The entries represent frequency of occurrence for each category in the domain. For example, the first entry(3) in $F_1$ indicates that the first category(b) of domain $D_1$ occurs three times, while the second entry(0) indicates no occurrence for the second category(a) of the domain. $F_2$ is a summary measure for numeric attribute. Its first entry stores the sum of the attribute values, $57.35 = 19.40 + 19.17 + 18.78$ and the second entry, 0.99992 represents the corresponding entropy measures for the attribute. Dissimilarity measures for the attributes are computed.
to be $G_1 = 0.9182$, $G_2 = 0.00002$, $G_3 = 0.3781$, $G_4 = 0.4591$, $G_5 = 1.0$, $G_6 = 0.0$ and $G_7 = 0.0$. Out of these values $G_2$, $G_6$ and $G_7$ are less than $0.0001$. So, similarity measure for the cluster $S = 3$.

<table>
<thead>
<tr>
<th>tobj</th>
<th>S</th>
<th>T</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>$F_4$</th>
<th>$F_5$</th>
<th>$F_6$</th>
<th>$F_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>57.35</td>
<td>1.14</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0.99992</td>
<td>0.6219</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.2: Summary measure

7.4 Proposed Algorithm

The proposed algorithm can cluster large, high dimensional datasets consisting of a mixture of categorical and numeric attributes. For each attribute, the attribute type ($\text{AttType}$) and domain size ($\text{Asize}$) should be provided as input. Attribute type provides the information for selecting the appropriate method for computing entropy for an attribute. Memory space is reserved for storing a summary measure based on the domain size. Summary measure of a numeric attribute has only two entries. Therefore domain size entered for all numeric attributes should be 2. Two phases of clustering are used. In the first phase, an incremental algorithm places objects, read sequentially from the hard disk, into existing clusters (initially none) based upon subspace similarity. Then, a hierarchical algorithm in the second phase reduces number of clusters by hierarchically merging them until required number of clusters are produced. Outliers handling is done in both the phases. The same incremental algorithm presented in $\text{CatSub}$ (refer Chapter 6) for clustering categorical data is also used here with new similarity measure, summary measure.
and cluster structure.

7.4.1 Incremental clustering

The purpose of this step is to form initial clusters with objects which are highly similar over a subset of attributes. Subspace based similarity measure of a cluster, $S$, introduced in Equation 7.12 provides the number of attributes which have dissimilarity measure nearly equal to zero. A cluster should contain at least $MinAtt$ number of such attributes, i.e. $S \geq MinAtt$ for any cluster, where $MinAtt$ is an input parameter. To achieve scalability each data object read sequentially from the hard disk is inserted on the fly in an existing cluster or a new cluster is created with the object. Inserting a new object in an existing cluster may decrease its $S$ value, but this decrease should be less than a given threshold, $\delta$, otherwise the object should not be inserted in the cluster. Minimum possible value for $\delta$ is zero and maximum possible value is total number of attributes($d$) minus $MinAtt$. To determine the cluster, $C$, where an object $x$ can be inserted, we define the following similarity function:

$$
\text{sim}(C, x) = \begin{cases} 
S(C') & \text{if } S(C) - S(C') \leq \delta \text{ and } S(C') \geq MinAtt \\
0 & \text{otherwise}
\end{cases}
$$

(7.21)

where, $C'$ indicates the cluster obtained if object $x$ is merged with $C$. The object will be inserted in the cluster returning the maximum nonzero sim value. A sequential search procedure is used to find the best cluster. Search space increases if outlier objects are allowed to create small clusters. To handle outliers three different lists of clusters - $ClusterList$, $CandidateList$ and $ExtraList$ are used. Initially all the lists are empty. If an object is not inserted in any of the clusters present in any of the three lists a new cluster is created with the object and the cluster is placed in the $CandidateList$. Maximum possible size for $CandidateList$ and $ExtraList$ are fixed at $maxSize$. When the $Candidate$
list becomes full, the oldest cluster in it is transferred to ExtraList to make room for the new cluster. If the ExtraList also becomes full with transferred clusters, the oldest cluster in it is removed and merged with Outliers cluster. Whenever an object gets inserted in a cluster, included in either CandidateList or ExtraList, the number of objects present in the cluster should be examined. If it collects MinObj (say, 4) objects then the cluster is transferred to ClusterList, which is the list of valid clusters. ClusterList can grow to any size. The threshold $\delta$ defined in Equation 7.21 takes three different forms - $\delta_1$, $\delta_2$ and $\delta_3$ for inserting an object in a cluster present in ClusterList, CandidateList and ExtraList respectively. The three thresholds take low, medium and high values in the range $[0, d]$, where $d$ is the number of attributes. An object read from the hard disk is first tried for insertion in a cluster in ClusterList with a very low value of threshold $\delta_1$. If it is not inserted, then CandidateList is tried with threshold $\delta_2$ assigned a medium value (say, less than 30% of $d$ with minimum value of 1). If not inserted again, ExtraList is tried with a very loose threshold $\delta_3$ allowing for much higher decrease in $S$ value. Maximum possible value is $\delta_3 = d - \text{Min.Att}$. If the object could not be inserted in a cluster in ExtraList also, a new cluster structure is created and inserted in CandidateList.

### 7.4.2 Hierarchical clustering

The first step may produce a large number of clusters. One may expect a reduced number of clusters or the number of clusters required (reqd) may be specified. Therefore, a bottom up hierarchical clustering technique is used to iteratively reduce number of clusters by merging the two least dissimilar clusters at a time until required number of clusters remain or maximum allowed dissimilarity measure ($G_{max}$) of a cluster is crossed. Dissimilarity between a pair of clusters is measured using $G(C)$, presented in Equation (7.11) as shown below.

\[
dissimilarity(C_p, C_q) = G(C_p \cup C_q) \tag{7.22}
\]
Final set of clusters are obtained when the hierarchical algorithm gets terminated. At this point an attempt is made for merging each of the sub clusters that remain in CandidateList and ExtraList with any one of the final clusters if maximum allowed dissimilarity threshold($G_{max}$) permits the merger, otherwise they are merged with Outliers cluster.

The algorithm is presented below.

---

Algorithm SMIC
---

**Inputs:** $X$, $n$, $d$, $MinAtt$, $MinObj$, $MaxSize$, $\delta_1$, $\delta_2$, $\delta_3$, $\{AttType_i, \, i = 1, 2, \cdots, d\}$, $\{Asize_i, \, i = 1, 2, \cdots, d\}$;

**Outputs:** The list of valid clusters found in ClusterList and Outliers cluster;

**Steps:**

01. Set $MaxSize=100$, $G_{max}=0.5*d$;
02. Initialize ClusterList, CandidateList and ExtraList to NULL;
03. FOR $i=1$ to $n$ DO
04.   { $x = X.getNextObject()$;
05.     ClusterList.findBestCluster($x$, $MinAtt$, $\delta_1$, $index$, $maxSim$);
06.     IF ($index != NULL$) THEN
07.       ClusterList[index].merge($x$, $i$, $maxSim$);
08.     ELSE
09.       {CandidateList.findBestCluster($x$, $MinAtt$, $\delta_2$, $index$, $maxSim$);
10.         IF ($index != NULL$) THEN
11.             {CandidateList[index].merge($x$, $i$, $maxSim$);
12.                 IF ($CandidateList[index].nC == MinObj$) THEN
13.                     Transfer cluster CandidateList[index] to ClusterList;
14.                 }
15.         ELSE
16.     }
17. }

103
16. \( \text{ExtraList.findBestCluster}(x, \text{MinAtt}, \delta3, \text{index}, \text{maxSim}); \)
17. \( \text{IF (index \neq \text{NULL}) THEN} \)
18. \( \{\text{ExtraList[index].merge}(x, i, \text{maxSim}); \)
19. \( \text{IF (ExtraList[index].nC == MinObj) THEN} \)
20. \( \text{Transfer cluster} \text{ExtraList[index]} \text{to ClusterList}; \)
21. \( \} \)
22. \( \text{ELSE} \)
23. \( \{C = \text{createClusterStructure}(x, i); \)
24. \( \text{Insert cluster} C \text{in CandidateList;} \)
25. \( \text{IF (sizeof(CandidateList) == MaxSize) THEN} \)
26. \( \{\text{transfer the oldest cluster in CandidateList to ExtraList;} \)
27. \( \text{IF (sizeof(ExtraList) == MaxSize) THEN} \)
28. \( \text{Delete the oldest cluster in ExtraList and merge it to Outliers;} \)
29. \( \} \)
30. \( \} \)
31. \( \} \)
32. \( \} \)
33. \( \} \) //end FOR

// Hierarchical Clustering

34. \( \text{NoOfClusters = sizeof(ClusterList);} \)
35. \( \text{WHILE (NoOfClusters > reqd) DO} \)
36. \( \{\text{ClusterList.findMergePair}(\text{index1}, \text{index2}, \text{minDissimilarity}); \)
37. \( \text{IF (minDissimilarity} \geq \text{Gmax } \text{BREAK;} \)
38. \( \text{ClusterList[index1].merge(ClusterList[index2]);} \)
39. \( \text{NoOfClusters} = \text{NoOfClusters} - 1; \)
40. \( \} \)
41. FOR each cluster $C$ in CandidateList DO 
42.   \{ClusterList.findMinDissimilar(C, index, minDissimilarity);
43.   IF (minDissimilarity <= Gmax ) THEN 
44.      ClusterList[index].merge(C);
45.   ELSE Outliers.merge(C);
46.   \}
47. FOR each cluster $C$ in ExtraList DO 
48.   \{ClusterList.findMinDissimilar(C, index, minDissimilarity);
49.   IF (MinDissimilarity <= Gmax ) THEN 
50.      ClusterList[index].merge(C);
51.   ELSE Outliers.merge(C);
52.   \}
53. FOR each cluster $C$ present in ClusterList DO 
54.   OUTPUT C.T;
55. END SMIC.

Details of the functions used can be derived from the description presented in Section 7.3. The function \textit{findBestCluster}(x, MinAtt, \delta, index, maxSim) returns the index of the best cluster where an object $x$ should be inserted. It also returns maxSim, the similarity value computed using \textit{Equation 7.21} when the object gets inserted in the best cluster. The computation is based upon first computing subspace based similarity measure $S$ as described in \textit{Equation 7.12}. Actual insertion of the object into the cluster is done by the function \textit{merge}(x, i, maxSim) which takes as input the object $x$, its serial number $i$ and the computed maxSim so that the cluster summary measure can be updated.
7.4.3 Complexity analysis

The algorithm requires only one pass through the dataset to produce a set of initial clusters. Number of comparisons required for each object depends upon the number of initial clusters \(c\) created. It is expected that large datasets also possess large clusters causing \(c\) to be smaller, as outliers are removed during clustering. Efficient implementation of the hierarchical clustering phase, as suggested in [And73], makes its complexity to be \(O(c^2)\). Therefore, the overall complexity becomes, \(O(nc + c^2)\), where \(n\) is the total number of objects in the dataset. For each cluster the cluster structure is to be stored in main memory. Size of the cluster structure depends upon the number of attributes \(d\). Therefore, space complexity is \(O(cd)\).

7.5 Experimental Validation

We perform experimental evaluation of the SMIC algorithm using some datasets available in the UCI Machine Learning Repository [BM98]. The selected datasets have labeled objects i.e. they are already classified into some classes(clusters). Accuracy of our algorithm is calculated with respect to those known clusters. Besides mixed-type datasets categorical datasets are also used for evaluating the algorithm. Experiments are conducted by implementing the algorithm in C++ on a 1.66 GHz HCL laptop with 512 MB RAM running LINUX operating system.

7.5.1 Accuracy calculation of clustering result

We use the clustering accuracy measure, \(r\) to evaluate the quality of the clustering algorithm. The clustering accuracy(exactness) measure is defined in [Hua98] as
follows:

\[ r = \frac{1}{n} \sum_{i=1}^{k} a_i \]  

(7.23)

where \( a_i \) is the number of data objects occurring in both cluster \( i \) and its corresponding class, \( k \) is the number of clusters and \( n \) is the number of objects in the dataset. Further, the clustering error \( e \) is defined as: \( e = 1 - r \).

### 7.5.2 Data sets

The real life datasets used to evaluate performance of our algorithm on clustering datasets with mixed categorical and numeric attributes are described below. The categorical datasets used in Chapter 6 for testing CatSub algorithm are also used here to test the performance of the algorithm on clustering datasets with categorical attributes alone. Accuracy obtained by CatSub for each dataset is also reported along with accuracies obtained by SMIC so that the two algorithms can be compared.

- **The credit approval dataset** contains mixed data. It has 690 instances each described by six numeric and nine categorical attributes. The instances are classified into two classes, approved(A) labeled as + and rejected(R) labeled as -. There are 37 instances having missing values on seven attributes. The dataset contains 307 approved instances and 383 rejected instances. We have removed 24 instances having missing values in numeric attributes as we have not used any method to deal with missing values in numeric attributes.

- **KDD CUP 1999 Corrected Network Intrusion Data.** The dataset [UoC99] contains 311029 data records, each representing a connection between two network hosts according to some well defined
network protocol and is described by 41 attributes (38 continuous or
discrete numerical attributes and 3 categorical attributes) such as duration
of connection, number of bytes transferred, number of failed login attempts
etc. Each record was labeled as either normal or one specific kind of
attack. There are 37 different attacks present in the dataset. Total number
of normal records is 60593, rest are attacks.

### 7.5.3 Result on credit approval dataset

The credit approval dataset contains both categorical and numeric attributes.
Clustering result and accuracy obtained for this dataset are shown in Tables 7.3
and 7.4. The result is obtained with parameter values: \( MinObj=4, MinAtt = 4, \\
\delta1=1, \delta2=4, \delta3=5, reqd=2. \) Accuracies reported by two other algorithms \( k-sets \) [LH03] and \( k\)-prototypes [Hua98] are also included for comparison.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Approved</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 )</td>
<td>277</td>
<td>92</td>
</tr>
<tr>
<td>( C_2 )</td>
<td>22</td>
<td>275</td>
</tr>
</tbody>
</table>

### 7.5.3 Result on credit approval dataset

The credit approval dataset contains both categorical and numeric attributes.
Clustering result and accuracy obtained for this dataset are shown in Tables 7.3
and 7.4. The result is obtained with parameter values: \( MinObj=4, MinAtt = 4, \\
\delta1=1, \delta2=4, \delta3=5, reqd=2. \) Accuracies reported by two other algorithms \( k-sets \) [LH03] and \( k\)-prototypes [Hua98] are also included for comparison.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMIC</td>
<td>0.83</td>
</tr>
<tr>
<td>( k)-sets</td>
<td>0.83</td>
</tr>
<tr>
<td>( k)-prototypes</td>
<td>0.81</td>
</tr>
</tbody>
</table>
7.5.4 Result on KDD CUP Corrected dataset

The algorithm is used to extract 40 clusters using the parameter values: \( \text{MinObj}=20, \text{MinAtt}=35, \delta_1=1, \delta_2=2, \delta_3=5, \text{reqd}=39 \). The result is shown in Table 7.5. Our aim is to extract pure clusters that contain either attack records or normal records. It can be seen from the table that most of the clusters are pure clusters. Some attack records in the dataset are so similar to normal records that it is very hard to separate them. The 40-th cluster consists of outliers. We have set \( \text{MinAtt} \) value to be 35 which means that objects in a cluster are similar over as many as 35 attributes out of 41 attributes. As most of the normal records are not similar over so many attributes they are separated into the Outliers cluster. Some attack records also remains merged with the outliers cluster. To separate them the outliers cluster may be clustered again with a lower \( \text{MinAtt} \) value. The clustering accuracy becomes \( r=0.94 \).

7.5.5 Result on soybean dataset

We used the \textit{SMIC} algorithm to find four clusters in the soybean small disease dataset. The result obtained is shown in Table 7.6, where \( C_i \)s are the cluster names produced by our algorithm, and \( D_1, D_2, D_3, \text{and} \ D_4 \) are the names of the original classes. There is one to one correspondence between the disease classes and the clusters obtained, which means that the \textit{SMIC} algorithm is able to cluster the dataset with 100% accuracy. Accuracies reported by some other algorithms, \( k \)-sets [LH03] and \( k \)-modes [Hua98] and SUBCAD [GW04] for the same dataset are also shown in Table 7.7. Parameter values used to run the program are: \( \text{MinObj}=4, \text{MinAtt}=18, \delta_1=1, \delta_2=5, \delta_3=10, \text{reqd}=4 \).
7.5.6 Result on zoo dataset

The misclassification matrix of the clustering result obtained with $MinObj=3$, $MinAtt=10$, $\delta_1=1$, $\delta_2=3$, $\delta_3=4$, $reqd=7$ is shown in Table 7.8. The column headings label the existing clusters in the dataset. A row-heading, $C_i$ indicates a cluster obtained by our algorithm with row sum giving the total number of objects present in the cluster. The accuracy achieved is $r = \frac{37+20+0+13+4+8+9}{101} = 0.90$. Accuracy figure obtained by CatSub was 0.89. Li et al. [LM004] reported an accuracy of 0.82 for their method.

7.5.7 Result on congressional voting dataset

We have used $reqd=2$ in order to extract the two known clusters in the congressional voting dataset. The result obtained by clustering all the 435 records of the dataset is presented in Table 7.9. Other parameter values used are: $MinObj=3$, $MinAtt=3$, $\delta_1=1$, $\delta_2=5$, $\delta_3=8$. The third cluster consists of outliers. Accuracy obtained is $r = \frac{155+218}{425} = 0.88$.

7.5.8 Result on Wisconsin breast cancer dataset

Result obtained for the breast cancer dataset is presented in Table 7.10. Corresponding parameter values used are: $MinObj=8$, $MinAtt=4$, $\delta_1=1$, $\delta_2=2$, $\delta_3=2$, $reqd=2$. Similar to the result obtained for CatSub (refer Chapter 6) the third cluster consist of outliers. We have considered it as a valid cluster because it signifies that the objects in Malignant class differ widely from one another, which separates them from the Benign class with somewhat similar objects. The dataset is clustered with an accuracy of 0.92 which is better than reported by other algorithms as shown in Table 7.11.
7.5.9 Result on mushroom dataset

The dataset is clustered in order to obtain minimum number of clusters such that all clusters are pure. The result (parameter values used: MinObj=8, MinAtt=14, δ1=1, δ2=4, δ3=8, reqd=19) with 19 clusters is shown in Table 7.12 that includes one impure cluster containing 32 edible and 72 poisonous mushrooms. The ROCK algorithm had reported 21 clusters shown in Table 7.13 with the same impure cluster. All pure clusters are produced by SMIC algorithm with 27 clusters (the result is not shown here).

7.5.10 Scalability test

Execution times of SMIC for each of the datasets used in the experiments are shown in Table 7.14. To ascertain the nature of scalability, execution times needed to extract 20 cluster in first 50000, 100000, 150000, 200000, 250000, 300000 and 311029 records of the KDD CUP Corrected dataset are evaluated and plotted in Figure 7.1. The graph shows that the execution time tend to increase almost linearly. The scalability of SMIC can be compared with that of CatSub presented in Figure 6.1. It is seen that CatSub is faster than SMIC. It is due to the fact that SMIC performs more computation in order to calculate the similarity measure and also it includes a hierarchical clustering phase.

7.5.11 Order dependency and parameter sensitivity

Incremental algorithms generally become order dependent. To test the order dependency of the algorithm we have repeated the above mentioned experiments several times, each time the records are shuffled randomly. Small difference in the results are noticed. It indicates minor order dependency for the algorithm.

The main input parameters are MinAtt and δ2. MinAtt has a wide range of input values. For example, MinAtt has the possible range [2, 22] for the
Mushroom dataset with 22 attributes. The same clustering result reported in Table 7.12 is obtained for any value of MinAtt in the range [2,16], with the other parameter $\delta_2$ set to 4. Acceptable results are obtained for $\delta_2$ in the range [2, 6]. It was observed in all the experiments that $\delta_2$ should be provided with a value which is less than 30% of the dimensionality of the dataset, minimum possible value being 1. In general value of $\delta_1$ should be set to 1. $\delta_3$ should be greater than or equal to $\delta_2$. A value greater than 2 can be used for MinObj, typical values are 4, 8. Larger values can be used if small clusters are not acceptable.

It is clear from the results presented that the proposed algorithm can produce good quality results for small or large datasets with categorical or mixed type attributes.
Table 7.5: Clustering result on KDD CUP Corrected dataset.

<table>
<thead>
<tr>
<th>Cluster-no</th>
<th>Normal</th>
<th>Attack</th>
<th>Cluster-no</th>
<th>Normal</th>
<th>Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11625</td>
<td>9571</td>
<td>21</td>
<td>8851</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>28910</td>
<td>22</td>
<td>0</td>
<td>249</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>131178</td>
<td>23</td>
<td>0</td>
<td>3812</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1451</td>
<td>24</td>
<td>0</td>
<td>119</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>61</td>
<td>25</td>
<td>0</td>
<td>259</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>239</td>
<td>26</td>
<td>0</td>
<td>143</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>359</td>
<td>27</td>
<td>0</td>
<td>124</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>281</td>
<td>28</td>
<td>0</td>
<td>37957</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>20</td>
<td>29</td>
<td>0</td>
<td>44</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>5279</td>
<td>30</td>
<td>0</td>
<td>337</td>
</tr>
<tr>
<td>11</td>
<td>53</td>
<td>107</td>
<td>31</td>
<td>614</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>457</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>525</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>22</td>
<td>33</td>
<td>0</td>
<td>180</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>2403</td>
<td>34</td>
<td>0</td>
<td>84</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>138</td>
<td>35</td>
<td>1888</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>3369</td>
<td>0</td>
<td>36</td>
<td>0</td>
<td>92</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>22</td>
<td>37</td>
<td>413</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>155</td>
<td>0</td>
<td>38</td>
<td>0</td>
<td>195</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>70</td>
<td>39</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>16164</td>
<td>40</td>
<td>33124</td>
<td>10020</td>
</tr>
</tbody>
</table>
Table 7.6: Clustering result on soybean dataset.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_1)</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(C_2)</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(C_3)</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>(C_4)</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 7.7: Accuracy on soybean dataset.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMIC</td>
<td>1.00</td>
</tr>
<tr>
<td>k-sets</td>
<td>1.00</td>
</tr>
<tr>
<td>CatSub</td>
<td>0.96</td>
</tr>
<tr>
<td>k-modes</td>
<td>0.95</td>
</tr>
<tr>
<td>SUBCAD</td>
<td>0.93</td>
</tr>
</tbody>
</table>

Table 7.8: The misclassification matrix of result on zoo dataset.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_1)</td>
<td>37</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(C_2)</td>
<td>0</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(C_3)</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(C_4)</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(C_5)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(C_6)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(C_7)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

114
Table 7.9: Clustering result on Congressional Voting dataset.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Republican</th>
<th>Democrat</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 )</td>
<td>155</td>
<td>43</td>
</tr>
<tr>
<td>( C_1 )</td>
<td>9</td>
<td>218</td>
</tr>
<tr>
<td>( C_2 )</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 7.10: Clustering result on breast cancer dataset.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Benign</th>
<th>Malignant</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 )</td>
<td>409</td>
<td>2</td>
</tr>
<tr>
<td>( C_2 )</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>( C_3 )</td>
<td>49</td>
<td>229</td>
</tr>
</tbody>
</table>

Table 7.11: Accuracy on breast cancer dataset.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMIC</td>
<td>0.92</td>
</tr>
<tr>
<td>CatSub</td>
<td>0.91</td>
</tr>
<tr>
<td>SUBCAD</td>
<td>0.87</td>
</tr>
</tbody>
</table>
Table 7.12: Clustering result on Mushroom dataset.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edible</td>
<td>192</td>
<td>0</td>
<td>288</td>
<td>0</td>
<td>32</td>
<td>1744</td>
<td>0</td>
<td>0</td>
<td>192</td>
<td>48</td>
</tr>
<tr>
<td>Poisonous</td>
<td>0</td>
<td>1722</td>
<td>0</td>
<td>36</td>
<td>72</td>
<td>0</td>
<td>1304</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7.13: Clustering result by ROCK on Mushroom dataset.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edible</td>
<td>96</td>
<td>0</td>
<td>704</td>
<td>96</td>
<td>768</td>
<td>0</td>
<td>1728</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Poisonous</td>
<td>0</td>
<td>256</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>192</td>
<td>0</td>
<td>32</td>
<td>1296</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 7.14: Execution time for the datasets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Objects</th>
<th>Attributes</th>
<th>Classes</th>
<th>Accuracy</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soybean small</td>
<td>47</td>
<td>35</td>
<td>4</td>
<td>1.00</td>
<td>0.01</td>
</tr>
<tr>
<td>Zoo</td>
<td>101</td>
<td>16</td>
<td>7</td>
<td>0.90</td>
<td>0.02</td>
</tr>
<tr>
<td>Congressional Votes</td>
<td>435</td>
<td>16</td>
<td>2</td>
<td>0.89</td>
<td>0.2</td>
</tr>
<tr>
<td>Credit approval</td>
<td>690</td>
<td>15</td>
<td>2</td>
<td>0.86</td>
<td>0.18</td>
</tr>
<tr>
<td>Wisconsin breast cancer</td>
<td>699</td>
<td>10</td>
<td>2</td>
<td>0.92</td>
<td>0.27</td>
</tr>
<tr>
<td>Mushroom</td>
<td>8124</td>
<td>22</td>
<td>2</td>
<td>0.99</td>
<td>4</td>
</tr>
<tr>
<td>KDD CUP Corrected</td>
<td>311029</td>
<td>41</td>
<td>2</td>
<td>0.93</td>
<td>575</td>
</tr>
</tbody>
</table>