Chapter 5

MLSCLUS: A Multi Level Subspace CLUStering Algorithm

In Chapter 3, we developed AOSS method to store very high dimensional huge datasets to be used mainly for subspace clustering. Although AOSS method is more efficient than the old record based method of storage, to handle the very high dimensional huge datasets with many missing values, it will fail to produce the best results in some cases, as illustrated below.

- Not many missing values are found. AOSS method stores the data in various AOSS Attribute tables by splitting the original record-based database to save on the space required by the many missing value attributes, that are most common in the very high dimensional datasets. We know that in each of the entries of the AOSS Attribute tables, we store the record-ids along with the respective attribute values. If missing values are not common, then in the AOSS method the additional space required to store the record-ids along with the attribute values in all the AOSS attribute tables will be significantly large.

- Real datasets contain a combination of all types of attributes. Majority of the existing subspace clustering algorithms use a uniform threshold value for all the
attributes. By using a single threshold value for all attributes, we tend to give equal importance to all the attributes. In reality, the threshold should not be uniform. Certain attributes are exceptional and need to be handled in a different manner. These exceptional attributes either have a much lower threshold requirement, or need a very high threshold value depending on whether they are rare or very frequent. Thus a uniform threshold for all attributes might lead to either generation of uninteresting subspace clusters for frequently occurring attributes and at the same time miss out some interesting subspace clusters of rare attributes.

- Applications with Huge datasets need more scalability. Existing methods are not efficient when the dataset is very large. This problem was addressed in chapter 4 and the SAMCLIQ [36] algorithm designed for this purpose. The SAMCLIQ algorithm succeeded in reducing the number of passes, but since it used the old record based method of storage and the level wise apriori based method for generation of dense units it fails just like the other methods when the dimensionality of the subspace clusters is high. This was also observed from the experimental results of SAMCLIQ, in section 4.4

In this chapter, we present algorithms to address some of the above issues. They differ from the CLIQUE, SAMCLIQ and AOSSCLIQUE methods addressed in the previous chapters of this thesis in the following respects -
Based on the AOSS method presented in chapter 3,

- Use different threshold values for the different attributes,
- Use the concept of maximal dense units for identification of subspace clusters,
- Mine the knowledge about the subspace clusters at different levels.

All these features are required in order to do a thorough analysis of data in any application area like census data analysis, and classification of web documents involving huge datasets with a very large number of attributes.

The remainder of the chapter is organized as follows. In Section 5.1, we discuss the use of the maximal frequent itemsets concept for finding all the dense units in first step of subspace clustering algorithm. Section 5.2 presents the details of the algorithm MADUGEN designed to find all the maximal dense units in a given dataset using uniform threshold for all attributes. In subsection 5.2.1 of this section we present the experimental results obtained using MADUGEN. In section 5.2.2 we present an algorithm MADUGENMT (MAximal Dense Unit GENeration with Multiple Thresholds), to find the maximal dense units using different threshold values for different attributes. We present in section 5.3 AOMLSCLUS, the Attribute Oriented Multi Level Subspace CLUSTERing algorithm, which uses concept of maximal dense units to identify sub subspace clusters in very high dimensional huge datasets,
consisting of attributes with varied threshold requirements and uses a variable number of intervals instead of using the same value for all attributes. The experimental results of AOMLSCLUS and AOMADUGENMT are reported in section 5.3.1. AOMADUGENMT is again a variation of CLIQUE implemented using MADUGENMT in step 1 of CLIQUE. Due to the use of variable intervals for different attributes, AOMLSCLUS can be enhanced further even for processing categorical attributes. In case of categorical attributes, the number of intervals will be equal to the number of unique categorical values of that attribute.

5.1 Use of Maximal Frequent Itemsets in Subspace Clustering

The apriori algorithm used for finding the frequent itemsets has the following main drawback –

- It employs a bottom-up search that enumerates every single itemset. Hence, in order to produce a frequent itemset of length k, it must produce all $2^k$ of its subsets since they too must be frequent. This exponential complexity of the algorithm restricts it to discovering only short patterns in medium sized datasets. To address this problem, the concept of maximal frequent itemsets [33] [34] [35] was introduced.
Definition 5.1: A frequent itemset is a set of items appearing together in a number of database records meeting a user-specified threshold. For example, if X is a k-itemset (an itemset consisting of k items), then X is frequent iff all the items found in X occur in at least minsupport number of records, where minsupport is equal to threshold multiplied by total number of records in the dataset.

Definition 5.2: If X is a frequent itemset and no superset of X is frequent, then we say that X is a maximal frequent itemset.

Any frequent itemset Y which is not a maximal frequent itemset, will be a subset of some maximal frequent itemset X of the dataset. Hence the set of all maximal frequent itemsets present in a dataset, concisely represents all the frequent itemsets present in that dataset.

The first step of the subspace clustering algorithm, needs to find all the dense units in order to identify the subspaces containing clusters. CLIQUE [1] uses a level wise apriori based algorithm to generate the dense units and suffers from the same drawback as the apriori algorithm. The efficiency of this step can be significantly improved if we apply the maximal frequent itemset concept to the dense units. The concept of dense units has been explained in chapter 2, but we redefine them again.
considering our AOSS representation and then we proceed to define maximal dense unit.

**Definition 5.3** A k-dimensional unit \( u^k \) is defined as the collection of the units from each of k distinct attributes. It has the form \( u^k = \{u_1, u_2, \ldots, u_k\} \) where \( u_i \) is the \( <A_i, I_i> \) pair of the \( i^{th} \) attribute \( A_i \) present in \( u^k \), \( I_i \) is an integer value representing the interval to which the attribute belongs to.

**Definition 5.4** The frequency count of a unit \( u^k \) in the original database DB is equal to the number of record-ids common to all the AOSS attribute tables of the k attribute units present in \( u^k \).

**Definition 5.5** The minimum support value msv of a unit \( u^k \) in the original database DB \( msv(u^k, DB) = N \cdot \min \{ \tau_i \text{ of } A_i \in u^k, i = 1 \text{ to } k \} \) where \( N \) is the number of records in DB and \( \tau_i \) is the density threshold value of the \( i^{th} \) attribute of \( u^k \), expressed as a percentage of records expected in each unit of the attribute for it to be dense.
Definition 5.6 A k-dimensional unit $u^k$ is said to be a dense unit in the original database DB, if the frequency count of this unit in the original database DB is greater than or equal to the minimum support value $msv(u^k, DB)$ of the unit in DB.

Definition 5.7 If $X$ is a k-dimensional dense unit and no m-dimensional superset of $X$ where $m > k$, is dense, then we say that $X$ is a maximal dense unit.

In order to improve the efficiency of the first step we can find all the maximal dense units and use them to find the subspace clusters present in the dataset. In the next section we present an algorithm for the same.

5.2 MADUGEN: A MAXimal Dense Unit GENeration algorithm.

In this section, we present the MADUGEN algorithm to find all the maximal dense units present in a dataset containing attributes with a uniform threshold value. We have designed this algorithm using the AOSS method of representing the data. It is based on the GenMax [34] algorithm used to find the maximal frequent itemsets for association rule mining.
Notations and Terminology used –

*unit* - is a pair of integers representing the attribute-id and unit-id respectively.

*iset* - it is a collection of attribute-id and unit-id pairs(units) belonging to the one dimensional dense units. An iset of length k consists of k such units.

*len* – used to keep track of number of units in iset.

*pset(*$u^k$*)* - pset of a k-dimensional unit $u^k$ denotes the possible set of $u^k$ and consists of all the units from the one-dimensional dense units $D[1]$, which are candidate units for forming higher dimensional dense units with $u^k$ as the base unit. The units in $D[1]$ are sorted in ascending order based on attribute-id as primary key and unit-id as secondary key.

*plen* - is used to keep track of the number of units in $pset(u^k)$.

*cset(*$u^k$*)* - cset of a unit $u^k$ denotes the combine set of $u^k$ and is a subset of $pset(u^k)$ consisting of only those units from pset which form $k+1$ dimensional dense unit when combined with $u^k$. This helps in pruning those units from pset which are not candidates for forming higher dimensional dense units.

*ccnt* - ccnt used to keep track of the number of units in the $cset(u^k)$.

*mduset* - consists of all the maximal dense units from the given data set.

*mducnt* – used to keep track of the number of maximal dense units obtained in the data set.

*threshold value* and *one-dimensional dense units* have the same interpretation as used in chapter 2.
**MADUGEN Algorithm**

*Input:*

- D[1] – details of one dimensional dense units obtained in the dataset,
- threshold value,
- record count (total number of data records in the dataset),

*Output:*

- mduset – set of maximal dense units found,
- mducnt – number of maximal dense units found in the dataset.

*Processing method:*

1. Start
2. initialize mduset to empty and mducnt to 0
3. for each subspace unit u in D[1]
   a. initialize iset to u
   b. call findpset(D[1], u) // findpset used to find the pset of unit u.
   c. call findmdu(mduset, iset, len, pset, plen, threshold) // findmdu is a recursive function to find maximal dense units.
4. // endfor
5. Stop // end of MADUGEN algorithm

The details of the algorithm used for findpset and findmdu are described in the following pages.
Algorithm for findpset

Inputs

- $D[1]$ – one dimensional dense units
- unit $u$ – consisting of its attribute-id and unit-id

Output

- pset of $u$ – set of all units in $D[1]$, whose attribute-id is greater than attribute-id of unit $u$.
- plen - number of units in pset.

Processing method

1. for each unit $u_1$ in $D[1]$
   
   if attribute-id of $u_1 >$ attribute-id of $u$
   
   add unit $u_1$ to the pset

2. return pset

Algorithm for findmdu

Inputs

- iset
- len
- pset
- plen
- threshold
record-count

Output

- mduset — set of all maximal dense units.

Processing method

1. if pset is not empty
2. begin
3. call findcset(iset, len, pset, plen, mduset, threshold, record-count) // finds the combine set - cset, detailed algorithm for findcset described after this algorithm.
4. if cset is empty
5. call addiset(mduset, iset, len) // detailed algorithm for addiset described after algorithm for findcset.
6. else
7. begin
8. for all the units cu in cset
   with attribute-id = attribute-id of first unit in cset do
9. begin //for
10. // form new isets by extending iset with cu.
11. iset = iset + cu
12. cset = cset - cu
13. end //for
14. len = len + 1 // length of iset increased by 1
15. for all the newisets obtained in step 8 do
16. begin //for
17. call findmdu( mduset, iset, len, cset, cnt, threshold, record-count)
   // pset = cset in above step 17.
18. end // for
19. end // else of if cset empty
20. end // if pset not empty
21. call addiset( mduset, iset, len)
22. Stop // end of findmdu.

Algorithm for findcset

Inputs
- iset
- len
- pset
- plen
- mduset
- threshold
- record-count

Output
- cset
CHAPTER 5. MLSCLUS: A MULTI-LEVEL SUBSPACE CLUSTERING ALGORITHM

- cent

Processing method:

1. for each unit u in pset
2.  newiset = iset + u // extend iset by adding u
3.  check if newiset exists in mduset
4.  if exists goto step 1 // continue with next unit u from pset
5.  if not  find frequency count of newset in the dataset
6.  if frequency count > threshold * record-count
7.    add u to cset
8.    increment cent
9.    stop

Algorithm for addiset

Inputs

- iset
- len
- mduset

Output

- mduset
Processing method

1. add iset to mduset.
2. increment mducle
3. stop

5.2.1 Experimental results

In this subsection we present the comparison of time requirements of the
ROSCLIQUE, AOSSCLIQUE and AOMADUCLIQUE. AOMADUCLIQUE is the
implementation which uses MADUGEN algorithm in step one of CLIQUE to find the
maximal dense units and uses only the maximal dense units to find the subspace
clusters, instead of using all the dense units. We compared the performance by varying
the dimension of the clusters from 5 to 12 using dataset of size 50,000 with 100
attributes. The values for the threshold and number of intervals, was set to 0.15 and 10
respectively. For generating synthetic data the method discussed in section 3.4.1 was
used. The experiments were run on a 3.00GHz Pentium 4 processor running linux. The
results obtained are shown in figure 5.1.
5.2.2 MADUGENMT: MADUGEN algorithm with multiple threshold values.

In this section we discuss, how we have modified MADUGEN algorithm to handle attributes with varying threshold values and also discuss the experimental results obtained using MADUGENMT.

In all the discussions earlier we have seen that the downward closure property is satisfied by all the dense units i.e if a particular k-dimensional unit $u^k$ is dense then all its subsets are also dense. This property is no longer satisfied when the attributes have different threshold values. In MADUGEN, $D_1$ is used to generate the maximal dense
units, but here if we use only $D_1$ we fail to generate all the maximal dense units, that is all those units possible with the low threshold value attributes. Hence we use the seed set $SS$, which is generated using $D_1$ and $C_1$. The seed set $SS$ is generated as follows-

**Seed Set Generation** -

1. sort all the 1-dimensional candidate units $C_1$ based on the threshold values of the attributes in ascending order.

2. $D_1$ in this multiple threshold value case consists of all the units whose frequency count is greater than or equal to the minimum support value calculated using the threshold value of the attribute representing the respective units. Again in $D_1$ all the units are kept sorted in ascending value of the threshold values of the attributes.

3. Using the sorted order of attributes in $C_1$, find the first attribute unit $fu$ in $C_1$ which belongs to $D_1$ and insert it into seedset $SS$.

4. for each subsequent attribute unit $su$ in $C_1$ which comes after $fu$, whose frequency count is greater than or equal to the $msv(fu, DB)$, insert $su$ into $SS$.

All the maximal dense units generated using $SS$ ensure that these dense units satisfy the sorted closure property. The sorted closure property ensures that we do not miss out any dense subsets of the low threshold attributes. Another variation that is required is that in step 6 of findcset function, in MADUGEN we had just one threshold value to decide whether to add it to cset, but in this case we will be having a maximum
of k different threshold values if iset is of length k. From these k different values, we use the threshold value of the first unit in iset as that will be the lowest threshold value.

In MADUGEN, findpset function uses D[1] the set of all one-dimensional dense units D_1 to find the units in pset, but in MADUGENMT we use the seed set SS obtained as explained above and pset is obtained as follows:

Processing method // for pset(u^k)of MADUGENMT

1. for each unit u_1 in SS
   if attribute-id of u_1 is not contained in any unit of u^k
      add unit u_1 to the pset

2. return pset

**MADUGENMT Algorithm**

*Input:*

- C[1] – details of one dimensional candidate units of all attributes after finding their frequency counts
- D[1] – details of one dimensional dense units obtained in the dataset
- thresholdarray // storing threshold values of all attributes
- record count (total number of data records in the dataset)

*Output:*

- mduset – set of maximal dense units found,
- mducnt – number of maximal dense units found in the dataset.
CHAPTER 5. MLSCLUS: A MULTI-LEVEL SUBSPACE CLUSTERING ALGORITHM

Processing method:

1. Start
2. initialize mduset to empty and mducnt to 0
3. find seed set SS as explained earlier in this section under Seed Set Generation
4. sort D[1] in ascending order of threshold values of attributes
5. for each subspace unit u in D[1]
   a. initialize iset to u
   b. call findpset(SS, u)  // findpset used to find the pset of unit u.
   c. call findmdu(mduset, iset, len, pset, plen, thresholdarray, record-count)
      // findmdu is a recursive function to find maximal dense units.
6. // endfor
7. Stop // end of MADUGENMT algorithm

5.3 AOMLSCLUS: An Attribute Oriented Multi Level Subspace CLUStering Algorithm.

The subspace clustering algorithms that we have discussed so far find only the subspace clusters, which are found in the original dataset. In this section, we define the subspace clustering problem for attributes with different threshold and interval values, which also identifies all the sub subspace clusters found in the subspace clusters of the original dataset. We also report some experimental results obtained.
Problem statement:

Given a set of say \( n \) records \( R_1, R_2, \ldots, R_n \) each record \( R_j, j = 1 \) to \( n \) having \( d \) attributes where \( R_j = \{ <A_i, V_i>, i = 1 \) to \( d \} \) where \( A_i \) is the \( i \)th attribute and \( V_i \) is the value of the \( i \)th attribute and given user input interval values \( \xi_i \) for \( i = 1 \) to \( d \), for the different \( d \) attributes the problem is to find the clusters in all the subspaces of the original data space and also identify those sub clusters which are not found in the original data space by the one dimensional dense units, but are found in the subspace clusters identified in the original data space.

The algorithm for this problem has been named as AOMLSCLUS, and is implemented using the AOSS data representation, the MADUGENMT algorithm and the findthreshold algorithm which we have designed to find the threshold values of the attributes in the dataset. The details of the findthreshold algorithm are discussed in the next subsection. We used the AOSS method discussed in chapter 3 to store the data records. All the record details are stored in the AOSS record table. The record details include the attribute-id and the attribute-record id pairs for all attributes of the record containing non-missing values. The attribute values along with the record id’s are stored in independent AOSS attribute tables. The Attribute-details table stores the information about all the attributes of the dataset. The Attribute-details table includes following information about each attribute –

- Attribute-id
- Attribute-name
• Attribute-filename

• Attribute-type(Numeric/Categorical)

• Attribute-Num-Intervals( number of intervals for the attribute)

The number of intervals is input by user for each attribute. Our implementation of the algorithm splits the attribute values into equal sized intervals and works for numeric attributes. In order to provide for variable interval sizes, the lower and upper range for each of the intervals can also be stored in case of numeric attributes and all distinct values of the categorical attributes can be stored to handle the categorical attributes.

This problem can be decomposed into the following three main parts –

1. Identification of the maximal dense units in the original dataset DB.

2. Identification of the maximal sub dense units in the various maximal dense units identified in DB.

3. Presentation of the details regarding the maximal dense and maximal sub dense units to the user.

The details of the above three steps are explained in the following subsections-
Identification of the maximal dense units in the original dataset DB

The maximal dense units found in the original dataset are found using the MADUGENMT algorithm. The different threshold values for all the attributes are found using the following algorithm-

findthreshold algorithm

Inputs

- frequency counts of each attribute in all its units

// number of units for each attribute is equal to the number of intervals for that attribute.

Output

- threshold value for each attribute

Processing

1. for each attribute repeat following steps -

2. find the average frequency count, \( \text{avgfcount} \)

   // \( \text{avgfcount} = \text{sum of frequency counts in all its units divided by the number of units for that attribute.} \)

3. find new average frequency count, \( \text{navgfcount} \) by considering only those units whose frequency counts are greater than the \( \text{avgfcount} \)

4. find the standard deviation, \( \text{stddev} \)
CHAPTER 5. MLSCLUS: A MULTI-LEVEL SUBSPACE CLUSTERING ALGORITHM

// stddev = squareroot of ( sumof squares (difference between navgfcount and frequency count) of each unit)

5. threshold of the attribute = \( \frac{\text{navgcount} - \text{stddev}}{\text{sum of frequency count in all units of that attribute}} \)

6. stop

It has been observed that, those attributes, which do not contain any significant clusters do not show much variation in their frequency count values in the various units and as a result have a low value for the standard deviation. But in the other case, where there are clusters there is a wide variation in the frequency count values of the different units, and have a much higher value for the standard deviation. By finding the new average frequency count in step 3 of findthreshold and using it to find the threshold value, we try to eliminate the low frequency count units of that attribute by setting a higher threshold value for that attribute. If we just use the average frequency count obtained in step 2, we get more or less the same threshold value for all attributes having the same number of intervals. Using the threshold values obtained as above, we find all maximal dense units and use these to find the subspace clusters in DB. The user can also be given the freedom to input their own threshold values for the attributes at the time of entering the attribute details.

We define below certain terms, which are used in the next subsection.
Definition 5.8 The minimum support value msv of unit $u^p$ in a dense unit $u^k$ is defined as

$$msv(u^p, u^k) = M \times \min\{ \tau_i \mid A_i \in u^p, i = 1 \text{ to } p \}$$

where $M$ is the frequency count of the $u^p$ unit in unit $u^k$ and $\tau_i$ is the density threshold value of the $i^{th}$ attribute of $u^p$, expressed as a percentage of records expected in each unit of the attribute for it to be dense.

Definition 5.9 We define the frequency count of a unit $u^p$ in another unit $u^k$ as equal to the number of record-ids common to all the AOSS attribute tables of the $(k+p)$ attribute units. This is used to find the sub dense units.

Definition 5.10 We define a sub dense unit as one which is not dense in the original dataset DB but is dense within the maximal dense unit identified in DB. A unit $u^p$ is dense in $u^k$ if frequency count of $u^p$ in $u^k$ is greater than or equal to $msv(u^p, u^k)$.

Definition 5.11 If $X$ is a $k$-dimensional sub dense unit in unit $u^k$ and no $m$-dimensional superset of $X$ where $m > k$, is sub dense in $u^k$, then we say that $X$ is a maximal sub dense unit of $u^k$. 
Identification of the maximal sub dense units in the various maximal dense units identified in DB.

In order to find the sub subspace clusters in the subspace clusters identified above, we need to first find all maximal sub dense units found in the maximal dense units identified in above step. We have designed algorithm findsubmdu which uses the findpset and findmdu functions of MADUGENMT algorithm for this purpose. The details are as follows –

findsubmdu algorithm

Inputs

- mduset // output of step 1 obtained using MADUGENMT
- mducnt

Output

- mdusubset // maximal dense units found in the mduset of step 1
- mdusubcnt // number of sub dense units identified

Processing

1. for each unit mdu belonging to mduset
2. record-count = frequency count of mdu
3. initialize iset to mdu
4. call findpset(SS, mdu) // findpset used to find the pset of unit mdu.
5. call findmdmu(mduset, iset, len, pset, plen, thresholdarray, record-count)
   // findmdmu is a recursive function to find maximal dense units.
6. endfor
7. stop // end of findsubmdmu

Presentation of the details regarding the maximal dense and maximal sub dense units to the user

We display the details of the maximal dense units found in the original database DB along with the maximal sub dense units found in them both in descending order of their dimensionality. In order to obtain a concise description for the cluster and sub clusters represented by the maximal dense units and the maximal sub dense units respectively the logic used in step 2 and step 3 of CLIQUE discussed in section 2.2 of chapter 2 can be used.

Clusters obtained from k-dimensional maximal dense units -

\( u_1^k \) -

sub clusters details obtained from maximal sub dense units found in \( u_1^k \)

(d-k)-dimensional sub dense units -
1-dimensional sub dense units –

\[ u_2^k \]

sub clusters details obtained from maximal sub dense units found in \( u_2^k \)

(d-k)-dimensional sub dense units –

and so on ...

Cluster details obtained from (k-1)-dimensional maximal dense units -

\[ u_1^{k-1} \]

sub cluster details obtained from maximal sub dense units found in \( u_1^{k-1} \)

(d-(k-1))-dimensional sub dense units –
1-dimensional sub dense units –

\( u_2^{k-1} \)

sub cluster details obtained from maximal sub dense units found in \( u_2^{k-1} \)

\((d-(k-1))-dimensional sub dense units – \)

... 

1-dimensional sub dense units –

and so on till the one dimensional dense units.

Clusters obtained from one-dimensional maximal dense units–

\( u_1^1 \)

sub clusters details obtained from maximal sub dense units found in \( u_1^1 \)

\((d-1))-dimensional sub dense units –

... 

1-dimensional sub dense units –
CHAPTER 5. MLSCLUS: A MULTI-LEVEL SUBSPACE CLUSTERING ALGORITHM

$u_2^1$ -

sub clusters details obtained from maximal sub dense units found in $u_2^1$

(d-1)-dimensional sub dense units –

.

.

.

1-dimensional sub dense units –

and so on for each of the one-dimensional dense units.

Example 5.3: Consider a census database having four attributes namely age as a numeric attribute and sex, educational_qualifications and marital_status as categorical attributes.

Each of these attributes have say, the following number of intervals –

for age it will be equal to $\lceil (110 - 1)/10 \rceil = 11$,

for sex it will be 2 having values male and female
for educational_qualifications it will be 6 with values non-SSC, SSC, HSSC, GRADUATE, POST_GRADUATE, DOCTORATE and for marital_status it is equal to 2 having values married and unmarried.

Assume that k = 2, for this database and the threshold values input for age, sex, marital status and educational qualifications are 0.2, 0.6, 0.7 and 0.15 respectively.

The presentation of the dense units and their sub units for a sample of data will be as shown below -

2-dimensional cluster details

<edu_qual, GRADUATE> <Age, 21–30> -
  <sex, female>
<edu_qual, POSTGRADUATE> <Age, 31–40> -
  <sex, male>
  <marital_status, married>

1-dimensional cluster details

<Age, 41-50> -
  <sex, male> <edu_qual, DOCTORATE>
  <marital_status, married>
5.3.1 Experimental results

We present here the details of the clusters identified using ROSCLIQUE, AOSSCLIQUE, AOMADUCLIQUE, AOMADUMTCLIQUE and AOMLSCLUS using a synthetic dataset. AOMADUCLIQUE and AOMADUMTCLIQUE are the implementations, which use MADUGEN and MADUGENMT algorithm in step one of CLIQUE respectively to find the maximal dense units and use only the maximal dense units to find the subspace clusters, instead of using all the dense units. We compared the results obtained using each of them using a synthetic dataset of size 50,000 with 100 attributes containing three 9-dimensional clusters. The value for the number of intervals was set to 10. For generating synthetic data the method discussed in section 3.4.1 was used. The experiments were run on a 3.00GHz Pentium 4 processor running linux. The results obtained are reported below.

<table>
<thead>
<tr>
<th>Method Used</th>
<th>Threshold values used</th>
<th>No. of correct clusters found</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROSCLIQUE</td>
<td>0.13</td>
<td>One 9-dimensional (out of 3)</td>
</tr>
<tr>
<td>AOSSCLIQUE</td>
<td>0.13</td>
<td>Three 9-dimensional (all 3)</td>
</tr>
<tr>
<td>AOMADUCLIQUE</td>
<td>0.13</td>
<td>Two 9-dimensional (out of 3)</td>
</tr>
<tr>
<td>AOMADUMTCLIQUE</td>
<td>0.13, 0.16, 0.17, 0.20</td>
<td>Three 9-dimensional (all 3)</td>
</tr>
<tr>
<td>AOMLSCLUS</td>
<td>0.13, 0.16, 0.17, 0.20</td>
<td>Three 9-dimensional (all 3)</td>
</tr>
</tbody>
</table>
Besides the three 9-dimensional main clusters, AOMLSCLUS also reported the sub clusters found within the dataset. In case of AOMADUMTCLIQUE and AOMLSCLUS the threshold values obtained using findthreshold algorithm discussed in section 5.3 were used.

5.4 Summary

In this Chapter, we proposed the subspace clustering problem for mixed data types for finding the subspace clusters in the database by fixing different values of threshold for the different attributes. We also extended the algorithm to find the significant subspace clusters found in the various subspace clusters existing in the database at various levels. This problem can be used to study the patterns found in a typical census data, university/college enrollment data, grading data which contains information about marks, grade awarded, course name, course credits etc.,