3.1 Introduction

Data mining process involves a number of sub-processes such as data preprocessing, mining the data, and the interpretation and visualization of the mined patterns. Some of these sub-processes are iterative in nature and at times some others loop amongst themselves. Among these, the data preprocessing step is of much interest because this step prepares the data according to the input specification of the data mining algorithms. The necessary tasks in data preprocessing are data integration, creating a single data matrix from data tables obtained from various resources, creating attributes, excluding irrelevant attributes, discretization of the values of attributes. Discretization transforms the infinitely many continuous values of an attribute into a significantly small numbers of finite intervals. A technique for discretization based on Rough Set Theory (RST) and clustering has been presented in this chapter.

The methods from clustering such as PAM and DBSCAN, and concepts of rough set theory presented in detail in Section 2.6, and Section 1.3 respectively. Therefore rest of the chapter is organized as follows: Section 3.2 discusses the discretization process; Section 3.3 describes the proposed method for discretization
followed by the experimental details in Section 3.4. Results and analysis of the experiments are presented in Section 3.5 and finally we conclude in Section 3.6.

3.2 Discretization

The task of discretization is well studied and a number of techniques have been developed for numerical variable. Discretization is usually achieved in two steps. In first step the number of discrete intervals is determined and in the second step the width or the boundaries of the intervals are obtained. Only a few discretization algorithms like Class-Attribute Interdependence Maximization (CAIM) [Kurgan2004], 4cDiscreteAlg [WWW2003] etc. compute the number of intervals. Often the user must specify the number of intervals or provide a heuristic rule for this purpose [Ching1995]. Given the range of all the values of a continuous attribute, the width or the boundaries of the intervals are obtained in the second step. Discretization algorithms are commonly divided into two categories - unsupervised and supervised [Kurgan2004].

Unsupervised: The algorithms for unsupervised discretization make no consideration for the class attribute if any. The representative algorithms of this category are Equal-Width and Equal-Frequency discretization [Catlett1991, Dougherty1995, Kerber1992]. The Equal-Width discretization algorithm initially determines the minimum and maximum values of the attribute to be discretized, and then divides the range into the pre-defined numbers discrete intervals of equal width. The Equal-Frequency [Catlett1991, Dougherty1995, Kerber1992] algorithm sorts all
the values of the attribute in ascending order, then divides the range into pre-defined number of intervals in such a way that every interval contains the same number of the sorted values.

**Supervised:**

The supervised discretization algorithms consider the interdependence between class labels and the attribute values. Some of the algorithms categorized in this group are Maximum Entropy [Wong1987], and Patterson and Niblett [Paterson1987], as incorporated in the decision tree algorithm C4.5 [Quinlan1993], Information Entropy Maximization (IEM) [Fayyad1993] and other information gain or entropy-based algorithms [Dougherty1995], [Wu1996], statistics-based algorithms such as ChiMerge [Kerber1992] and Chi2 [Liu1997], class-attribute interdependency algorithms such as Class-Attribute Dependent Discretizer (CADD) [Ching1995], and clustering-based algorithms like K-Means discretization[Tou1974].

There are two more criteria to classify the discretization methods namely – global-local and static-dynamic [Dougherty1995].

**Global-vs-local technique:**

A global technique simultaneously considers all attributes and generates a conversion function for the entire dataset while a local method only restricts it to a single attribute [Kerber1992], [Quinlan1993].
**Static-vs-dynamic:**

A static technique assumes that all attributes are independent and consequently presets the number of bins as a constant [Catlett1991], [Holte1993], while in a dynamic method a relevant number of bins are determined by measuring the interdependencies of attributes. This technique requires a substantial involvement of domain experts.

Quantization methods [Linde1980] are also used to design discretization algorithms, e.g. the Adaptive Quantizer Algorithm [Chan1991]. The large number of attribute values in the database slows down the process of better discovery and makes inductive learning inefficient [Wong1975]. Therefore one of the main goals of any discretization process is to reduce the number of discrete intervals of the values of a continuous attribute significantly. The algorithm should also maximize the interdependence between discretized attribute values and class labels, since this minimizes the information loss as a result of discretization. A satisfactory trade off between these two goals needs to be achieved [Kurgan2004].

The class-attribute interdependence maximization (CAIM) algorithm [Kurgan2004] discretizes an attribute using class-interdependency as defined in [Ching1995]. CAIM algorithm uses the following heuristic formula to estimate the number of intervals so that the user need not specify the number of discrete intervals:

\[ N_{Fi} = \frac{M}{3C} \]  

\[ \text{......... (3.1)} \]
where, $N_{Fi}$ is the number of intervals for attribute $Fi$, $M$ is the number of examples and $C$ is the number of classes [Wong1987]. CAIM algorithm gives good results in terms of Class-Attribute Interdependence Redundancy (CAIR) value [Wong1975] and the number of intervals. It is also reported in the literature that the classificational accuracies and the numbers of rules generated when used for Cover Learning using Integer Linear Programming 4 (CLIP4) [Cios2001], [Cios2004], and C5.0 [WWW2003], is good. Although CAIM algorithm computes the number of intervals using a heuristic formula without any intervention of the user, it does not find the natural intervals with reference to the density, which may be desirable.

Supervised discretization algorithms discretize the attribute values without much loss of information and number of intervals is less as compared to unsupervised algorithms. Also the unsupervised techniques like equal width, equal frequency does not provide natural intervals. In this chapter two algorithms have been presented to discretize the continuous values of attributes of a dataset. The first algorithm corresponds to supervised categories as it takes into account the class label/labeled dataset. While the second may be placed in the unsupervised category of discretization algorithm as this must either operate on unlabeled data or it must not take into account the class label of the labeled dataset. The goal of proposed technique has been to obtain natural intervals of numeric values by combining rough set theoretic concepts with clustering. Employing the principles of DBSCAN for clustering the natural intervals are obtained and the approximations of the class attribute dependent concepts in RST help refine the intervals. Natural interval is an interval without any significant substructure (as quantified by the measure). RST is
applicable usually to the labeled data but to implement the proposed method to
unlabeled data Partition Around Medoid (PAM) clustering algorithm has been used to
group similar objects and assign temporary labels to each group.

**CAIR Value:**

Consider a data set consisting of $M$ records, where each record belongs to only
one of $S$ classes. $A$ indicates any of the continuous attributes from the mixed-mode
data. Let a discretization scheme discretizes the continuous domain of attribute $A$, into
$n$ discrete intervals bounded by the pairs of values:

$$D = \{(d_0, d_1], (d_1, d_2], \ldots, (d_{n-1}, d_n]\} \quad (3.2)$$

Where $d_0$ is the minimal value and $d_n$ is the maximal value of attribute $A$, and the
values are arranged in ascending order.

In $D$, each value of attribute $A$ must exist in only one of the $n$ intervals. With
the change in discretization scheme, the membership of each value of $A$ in a certain
interval may or may not change to another interval. The class attribute and the
variable of attribute $A$ for discretization can be treated as two random variables, thus a
two-dimensional frequency matrix called the Quanta Matrix can be set up as shown in
Table 3.1.

In Table 3.1, $q_{ir}$ is the total number of continuous values belonging to the $i^{th}$
class that are within interval $(d_{r-1}, d_r]$. $M_{ir}$ is the total number of attribute-values
belonging to the $i^{th}$ class and $M_{ar}$ is the total number of continuous values of attribute
$F$ that are within the interval $(d_{r-1}, d_r]$, for $i = 1, 2, \ldots, S$ and $r = 1, 2, \ldots, n$. 
Table 3.1: Quanta Matrix

<table>
<thead>
<tr>
<th>Class</th>
<th>Interval</th>
<th>Class wise no. of values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>([d_0, d_1])</td>
<td></td>
</tr>
<tr>
<td></td>
<td>([d_{i-1}, d_i])</td>
<td></td>
</tr>
<tr>
<td></td>
<td>([d_{n-1}, d_n])</td>
<td></td>
</tr>
<tr>
<td>C_1</td>
<td>q_{11}</td>
<td>M_{1+}</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>C_i</td>
<td>q_{i1}</td>
<td>M_{i+}</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>C_s</td>
<td>q_{s1}</td>
<td>M_{s+}</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

The estimated joint probability of the occurrence that values of attribute \(A\) are within the interval \(D_r = (d_{r-1}, d_r]\) and belong to class \(C_i\) is given by:

\[
p_{ir} = p(C_i, D_r | A) = \frac{q_{ir}}{M} \tag{3.3}
\]

The estimated class marginal probability \(p_{ir}\) that attribute values of \(A\) belong to class \(C_i\) and the estimated interval marginal probability \(p_{sr}\) that values of attribute \(A\) are within the interval \(D_r = (d_{r-1}, d_r]\) are as follows:

\[
p_{ir} = p(C_i) = \frac{M_{ir}}{M} \tag{3.4}
\]
The Class-Attribute Mutual Information between the class variable $C$ and $D$ corresponding a discretization scheme for attribute $A$ given a two dimensional frequency matrix is given by,

$$I(C, D | A) = \sum_{i=1}^{s} \sum_{r=1}^{n} p_{ir} \log_2 \frac{p_{ir}}{p_{+r} p_{+i}}$$ ........... (3.6)

The Shannon's entropy, $H$, is defined as:

$$H(C, D | A) = \sum_{i=1}^{s} \sum_{r=1}^{n} p_{ir} \log_2 \frac{1}{p_{ir}}$$ ........... (3.7)

Given (3.6) and (3.7), the Class-Attribute Interdependence Redundancy (CAIR) criterion [Wu1996], $R$, is defined as,

$$R(C, D | A) = \frac{I(C, D | A)}{H(C, D | A)}$$ ........... (3.8)

CAIR is built on the notion of class-attribute joint entropy and class-attribute mutual information. Higher the CAIR value, higher the interdependence between the class labels and the discrete intervals The CAIR criterion is used to measure the interdependence between classes and the discretized attribute (the larger its value, the better correlated are the class labels and the discrete intervals). It is also independent...
of the number of class labels and the number of unique values of the continuous attribute.

3.3 Proposed Method

The proposed method for discretization comprises of two phases. In the first phase the values of the attribute are partitioned into natural intervals using density based clustering approach and in the second phase the intervals are refined using RST tools to have good discretization results.

Phase 1:

In this phase the values of all the continuous attributes are partitioned using concepts of DBSCAN clustering algorithm to obtain the natural interval of the attribute values.

The characteristic of intervals obtained from the first phase may vary in terms of density and range. Therefore three threshold values MaxPoint, MinPoint, and MaxLength are used to characterize the intervals. MaxPoint and MinPoint are used as controls on the number of distinct attribute values. MaxLength is applied to limit the range of a normal interval. The outcomes of the clustering phase are categorized as, normal, large or small.

1. Normal: An interval \( I \) is said to be normal if

\[
( MinPoint \leq \text{Card}(I) \leq \text{MaxPoint}) \ \text{AND} \ (\text{Range}(I) \leq \text{MaxLength}).
\]

2. Large: An interval \( I \) is said to be large if

\[
( \text{Card}(I) > \text{MaxPoint}) \ \text{OR} \ (\text{Range}(I) > \text{MaxLength}) \ \text{OR both}.
\]
3. **Small:** An interval $I$ is said to be small if

$$\text{Card}(I) < \text{MinPoint}. $$

**Phase 2:**

To achieve good discretization, the partition of attribute values or discretized intervals needs to be refined by reorganizing the large or small intervals. This phase aims to optimize the number of intervals by splitting the large, and merging the small intervals. The objective of refining the set of discretized intervals is either to maintain the significance of the attribute or possibly enhance it.

### 3.3.1 Significance of Attribute

Let the numeric values of the attribute $a_i$ is considered for discretization, and let D be the class attribute. The significance of $a_i$ measures the ability of discrete values of $a_i$ that can predict the classes correctly. Therefore the rough set concept $\text{POSa}_i(D)$ as defined in Section 1.3, is a measure of the significance of an attribute $a_i$. While discretizing the values of an attribute $a_i$, the cardinality of $\text{POSa}_i(D)$ is aim to be maximized. Therefore the cardinality of $\text{POSa}_i(D)$ is aim to be maximized, towards the refinement of the set of the discrete intervals.

To maximize the cardinality of $\text{POSa}_i(D)$ the refinement is implemented in such a way that the maximum number of objects are correctly classified by each of the discretized values of the attribute $a_i$ just as they are classified by D. A rough membership function $f(a_i, c_p, I)$ is proposed to achieve this objective, corresponding to each of the intervals $I$ for the given/selected attribute with respect to class labels.
Let the dataset $U$ include objects of $m$ classes, say $\{c_1, c_2, c_3 \ldots c_m\}$ and let the $k$ distinct values of an attribute $a_i$ in ascending order be $\{v_{i1}, v_{i2}, v_{i3} \ldots v_{ik}\}$ i.e. the interval $[v_{i1}, v_{ik}]$. The rough membership function of any arbitrary interval $I= (v_{i1}, v_{ij}]$ of the values of attribute $a_i$ for a class $c_p$ is defined as

$$f(a_i, c_p, I) = \frac{\text{Card} \ (a_{i,1}X_{c_p})}{\text{Card} \ (X_{a_{i,1}})} \quad \text{........... (3.9)}$$

where $X_{a_{i,1}}=\{x \mid x \in U, a_i(x) \in I\}$ and, $a_{i,1}X_{c_p}=\{x \mid a_i(x) \in I, D(x)=c_p\}$.

For a given conditional attribute $a_i$, the positive region $\text{POS}_{a_i}(D)$ is given by

$$\text{POS}_{a_i}(D) = \bigcup\{ a_iX : X \in [x]_D \} \quad \text{......... (3.10)}$$

For a specific interval $I= [v_{i1}, v_{ij}]$

$$\text{POS}_{a_i,j}(D) = a_{i,j}X_{c_1} \cup a_{i,j}X_{c_2} \cup \ldots a_{i,j}X_{c_p} \ldots \cup a_{i,j}X_{c_m} \quad \text{......... (3.11)}$$

Where for $I=\{ I_1, I_2, I_3, \ldots \}$, $a_{i,1}X_{c_p}$ may further be written as,

$$a_{i,1}X_{c_p} = a_{i,1}X_{c_r} \cup a_{i,1}X_{c_r} \ldots \quad \text{......... (3.12)}$$

Therefore, by maximizing $f(a_i, c_p, I)$ each of the $a_{i,1}X_{c_p}$, $a_{i,1}X_{c_p}$, $\ldots$ is maximized and hence $\text{POS}_{a_i}(D)$ is maximized.

### 3.3.2 Refinement Criteria

In order to refine the outcome of phase I, the resulting small intervals need to merge while large intervals split. Two criteria each for merging and splitting are proposed to automate the process of merging and splitting. The merge operation is aimed to reduce the number of resulting intervals. The merging criterion ought to
avoid obtaining a large resulting interval after the merge operation. Analogously, the 
splitting operation is aimed to reduce the inconsistency of an interval with respect to 
class labels and thus enhance the significance of the attribute. Splitting criterion must 
avoid obtaining two small intervals after the split operation. The threshold values 
MaxPoint, MinPoint, MaxLength along with two function $g$ and $h$ are used for the two 
criteria. The function $g$ is defined to compute the cardinality of the two intervals to be 
merged and to compute the range of two intervals the function $h$ has been defined 
below. Let $I_j = \{v_{j1}, v_{j2}, ..., v_{jn}\}$ and $I_k = \{v_{k1}, v_{k2}, ..., v_{kr}\}$ be the distinct values in the 
intervals $I_j$ in $I_k$ respectively. Then,

$$g(I_j, I_k) = |I_j| + |I_k|$$  \hspace{1cm} \text{(3.13)}

$$h(I_j, I_k) = \max\{v_{j1}, v_{j2}, ..., v_{jn}, v_{k1}, v_{k2}, ..., v_{kr}\}$$
$$- \min\{v_{j1}, v_{j2}, ..., v_{jn}, v_{k1}, v_{k2}, ..., v_{kr}\}$$  \hspace{1cm} \text{(3.14)}

The distance between two intervals $I_1$, $I_2$ can be measured using any one of the 
following formulas,

1. Minimum distance

$$d(I_1, I_2) = \min\{|p - p'|, p \in I_1, p' \in I_2\}$$  \hspace{1cm} \text{(3.15)}

2. Maximum distance

$$d(I_1, I_2) = \max\{|p - p'|, p \in I_1, p' \in I_2\}$$  \hspace{1cm} \text{(3.16)}
3. **Mean distance**

\[ d(I_1, I_2) = |\overline{p} - \overline{p}'|, \quad \overline{p} \text{ and } \overline{p}' \text{ are the mean of } I_1 \text{ and } I_2 \text{ respectively} \]

\[ \cdots \cdots (3.17) \]

4. **Average Distance**

\[ d(I_1, I_2) = \frac{1}{n_1n_2} \sum_{p \in I_1} \sum_{p' \in I_2} |p - p'| \quad n_1 = |p| \text{ and } n_2 = |p'| \]

\[ \cdots \cdots (3.18) \]

Consider three arbitrarily located contiguous intervals \( I_{j-1}, I_j, I_{j+1} \). If \( I_j \) be a small interval then the suitability of intervals \( I_{j-1} \) and \( I_{j+1} \) need to be checked before merge. The two Boolean functions \( MR_C \) for merging criterion and \( SP_C \) for splitting criterion are defined to implement the merge and split operations. The functions \( g, h \) and \( d \) are used to define the \( MR_C \).

\[ MR_C(I_j, I_k) = \begin{cases} 
\text{TRUE, if } & [g(I_j, I_k) < \text{MaxPoint AND } h(I_j, I_k) < \text{MaxLegth AND } d(I_j, I_k) < d(I_j, I_{k_l})] \forall k_l \neq k \\
\text{FALSE otherwise} 
\end{cases} \]

\[ \cdots \cdots (3.19) \]

\[ SP_C(I_j) = \begin{cases} 
\text{TRUE, if } & [l I_j \geq \text{MinPo int OR Range } (I_j) > \text{MaxLegth }] \\
\text{FALSE otherwise} 
\end{cases} \]

\[ \cdots \cdots (3.20) \]
3.3.3 Algorithm for Labeled Dataset

Given a dataset of size $N$, with $m$ classes $c_1, c_2, \ldots, c_m$, and $n$ attributes. Let $A = \{A_1, A_2 \ldots A_n\}$. The base algorithm as presented below is applied to each of the continuous attributes of $A$. The algorithm employs the function $\text{Refine}$ to refine the set of intervals obtained by the DBSCAN. The function $\text{Cut_Point}$ is applied to find the best splitting point in a large interval which is eligible for split. The rough membership function $f(A_i, c_p, l)$ is used to maximize the dependency of $C_p$ on $a_i$ while splitting the interval $I$. In order to obtain a trade off between number of objects and balanced split of a large interval the search for the cut point proceeds from middle value $k/2$ down to 1 or from $k/2$ to $k$ thus reducing the computational effort by half in each iteration. The proposed discretization algorithm for labeled data has given in the Figure 3.1.

Step 1. Select distinct values of $A_i$ in an array $\text{Distinct}$.

Step 2. Sort($\text{Distinct}$).

Step 3. Call $\text{DBSCAN}(\text{Distinct}, Eps, \text{MinPts})$ to obtain intervals $I_1$, $I_2$, $\ldots$, $I_r$.

Step 4. Refine($I_1, I_2, \ldots, I_r$) to get the optimal intervals for the specified parameters $\text{MinPoint, MaxPoint and MaxLength}$.

Step 5. Assign interval id to the result of step 4.

Figure 3.1: Discretization Algorithm for labeled data
The algorithm has 5 steps where steps 2, 3 and 4 involve computation. In the first step distinct value of the attribute is obtained by scanning the database for the relevant attribute. These values are sorted in the second step. In the third step DBSCAN algorithm is used to obtain the natural intervals of the attribute values. In the fourth step the refinement of the intervals is obtained to have optimum number of intervals and with minimum loss of information. In the fifth step interval id assigned to each value of the attribute in accordance with result of step 4.

\[
\text{Refine}_I (I_1, I_2, \ldots, I_n)
\]

While (no change in no. of intervals) do

For each interval \( I_j \)

If \( \text{SP}_C(I_j, \text{MinPoint}, \text{MaxLength}) = \text{True} \) then

\[
\text{Temp} = \text{Cut\_Point} (I_j) \quad \text{\( I_j = \{v_{j1}, v_{j2}, v_{j3}, \ldots v_{jk}\} \)}
\]

Replace the interval \( I_j \) with two intervals \( I_{j1} = (v_{j1}, \text{Temp}] \) and \( I_{j2} = (\text{Temp}, v_{jk}] \)

Elseif \( |I_j| < \text{MinPoint} \) then

If for \( I_k \) either neighbour of \( I_j \) \( \text{MR\_C} (I_j, I_k, \text{MaxPoint}, \text{MaxLength}) \) = True then

Merge \( I_j \) to an interval \( I_k \)

Endif

Endif

Endfor

Endwhile

\[\text{Figure 3.2: Refinement function}\]
Cut_Point \( (I_j) \)

1. \( I = [v_{i1}, v_{ik/2}] \) // \( v_{ik/2} \) is the middle term of \( \{ v_{i1}, v_{i2}, v_{i3}, \ldots v_{ik} \} \)
2. \( \text{MAXRMV} = \max (\{ f(A_j, c_p, I) \}) \forall c_p, \)
3. for each \( v_{ij}, j = k/2 \) to 2
   3.1. \( I = [v_{i1}, v_{ij}] \)
   3.2. \( \text{Temp} = \max (\{ f(A_i, c_p, I) \}) \forall c_p; \)
   3.3. if \( \text{Temp} > \text{MAXRMV} \) then
       \( \text{MAXRMV} = \text{Temp}; \)
       else
       break;
4. if \( j < k/2 \) then return \( v_{ij} \) as cut point for the interval
   else
   for each \( v_{ij}, j = k/2 \) to \( k-1 \)
   4.1. \( I = [v_{ij}, v_{ij}] \)
   4.2. \( \text{Temp} = \max (\{ f(A_i, c_p, I) \}) \forall c_p; \)
   4.3. if \( \text{Temp} > \text{MAXRMV} \) then
       \( \text{MAXRMV} = \text{Temp}; \)
       else
       return \( v_{ij} \) as cut point for the cluster

**Figure 3.3: Cut Point function**

The refinement function has been presented in Figure 3.2. The function \( \text{Refine}_I \) refines the intervals yielded by the DBSCAN. In this function the size and density measures of every interval is checked. If an interval is large then this interval is check for splitting based on the splitting criteria \( \text{SP}_C \). If an interval is eligible for splitting then function \( \text{Cut_Point} \) is applied to determine the best suitable cut point for the interval. Using this cut point the large interval is split into two intervals. If an
interval is small then the function $\text{Refine\_I}$ determine the best nearest interval to which the small interval may be merged based on the result of criteria $MR\_C$. Figure 3.3 exhibits the details of the function $\text{Cut\_Point}$.

### 3.3.4 Algorithm for Unlabeled Dataset

Given an unlabeled dataset of $N$ objects and $n$ attributes say $A = \{A_1, A_2, ..., A_n\}$ with no class attribute. The base algorithm employs a modified function $\text{Refine\_ul}$ for the refinement Step 4. Since the dataset is unlabeled before splitting a large interval PAM is applied to partition and temporarily label the relevant objects i.e. the objects which have attribute value in the considered interval.

Here PAM is used to label the data objects for which the class attributes is not considered, due to the following salient features of k-medoid method. Firstly, unlike many other partitioning methods, the k-medoid methods are robust to the presence of outliers. Secondly, clusters found by k-medoid methods do not depend on the order in which the objects are examined. Furthermore, they are invariant with respect to translations and orthogonal transformations of data objects.

In Figure 3.4 the refinement function for unlabeled data has been presented. The basic functionality of the $\text{Refine\_ul}$ is same as that of $\text{Refine\_I}$. Function $\text{Refine\_ul}$ employs use PAM to temporarily label the objects during the split process to a large interval.
Refine_ul(I_1, I_2, \ldots, I_i)

while (no change in no. of intervals) do

For each interval I_j

If SP-C(I_j, MinPoint, MaxLength) = True then

Select the objects from the dataset which have value in the interval I_j for attribute A_i.

Partition the selected object into two parts using PAM.

Assign the cluster id 1 and 2 as class label to the objects corresponding to each cluster.

Temp= Cut_Point ({v_{j1}, v_{j2}, v_{j3} \ldots v_{jk}})

Replace the interval I_j with two intervals I_{j1} = [v_{j1}, Temp] and I_{j2} = [Temp, v_{jk}]

elseif |I_j| < MinPoint then

If for I_k' neighbour of I_j MR_C(I_j, I_k', MaxPoint, MaxLength) = True
then merge I_j to an interval I_k'

Endif

Endif

Endfor

Endwhile

Figure 3.4: Refinement function for unlabeled data
3.3.5 Complexity

The running time of the algorithms applied to the labeled data and the unlabeled differ at Step 4 corresponding to the complexities of the functions Refine_l and Refine_ul. In Step 1 and Step 2 the distinct values of the attribute maybe selected in the sorted order thus the complexity of the two steps is $O(N \log N)$ where $N$ is the number of objects in the dataset. DBSCAN is known to have the complexity $O(N \log N)$ which may be in worst situation for the above algorithm, i.e. when the attribute values for each object is distinct. The complexity of the Refine_l function is bounded by $k \times N/2$, where $k$ is the number of intervals of an attribute and the running time of the function Cut_Point is bounded by $N/2$. If $n$ is the number of attribute then the total complexity of the algorithm is bounded by,

$$n \times (N \log N + N \log N + k \times N/2 + N)$$

$$= n \times (N \log N)$$

The number of attributes $n$ is normally small in comparison to $N$. The preprocessing of the dataset for selecting relevant attributes further reduces the value of $n$ to be smaller compared to $N$. Therefore, the running time of the proposed algorithm for labeled data is bounded by $N \log N$.

The complexity of the refinement function Refine_ul for unlabeled data applies clustering algorithm PAM. The complexity of PAM is known to be that of $O(N^2)$. Thus the complexity of the proposed method is bounded by $k \times N^2$. However,
the values of \( k \) and \( n \) are small compared to \( N \) hence the complexity of the algorithm is \( O(N^2) \).

### 3.4 Experimental Detail

The experiments were carried out on the following four datasets obtained from the UC Irvine ML repository [WWW] whose descriptions of the data sets is presented in Table 3.2.

1. Iris Plants dataset (iris),
2. Johns Hopkins University Ionosphere dataset (ion),
3. Statlog Project Heart Disease dataset (hea),
4. Pima Indians Diabetes dataset (pid).

**Table 3.2: Data Set Description**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Iris</th>
<th>Ion</th>
<th>Hea</th>
<th>Pid</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Examples</td>
<td>150</td>
<td>351</td>
<td>270</td>
<td>768</td>
</tr>
<tr>
<td>No. of Classes</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>No. of Attributes</td>
<td>4</td>
<td>34</td>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td>No. of Cont. Attributes</td>
<td>4</td>
<td>32</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>All-Cont/Mix-mode</td>
<td>All-Cont</td>
<td>Mix-mode</td>
<td>Mix-mode</td>
<td>All-Cont</td>
</tr>
</tbody>
</table>
The proposed technique has been compared with seven discretization algorithms namely equal-width, equal-frequency, Patterson-Niblett, IEM, Maximum Entropy, CADD and CAIM. The first two algorithms equal-width, equal frequency are categorized as unsupervised algorithms which require the user to specify the number of discrete intervals. In these experiments, the heuristic formula $N_F = M(3C)$ to estimate the number of intervals is applied. The supervised algorithms apply their own criteria to generate an appropriate number of discrete intervals. To evaluate the result of discretization for the unlabeled dataset, iris and pid were discretized by the proposed method disregarding their respective class attributes. Here iris and pid data has been selected for evaluating of the discretization for the unlabeled dataset as minimum and maximum example.

In the experiment the values of the three thresholds values $MinPoint$, $MaxPoint$ and $MaxLength$ are fixed manually. The values have been fixed independently for different data sets. Here the value of $MinPoint$ has been set to 1/10 of all the total number of objects. The value of $MaxPoint$ is set to 1/2 of all the number of objects. Similarly the value of $MaxLength$ is set equal to half of the range of the attribute values.

The two parameters considered to measure the quality of discretization are the CAIR value and the number of intervals. The CAIR criterion is used to measure the interdependence between classes and the discretized attribute.
Chapter 3

3.5 Result and Analysis

The CAIR values for each of the numeric attributes for all the four datasets were computed and the average CAIR value yielded by applying each of the eight discretization schemes on the four datasets has been presented in Table 3.3. Similarly to evaluate the number of intervals on executing each of the eight discretization schemes, the total number of intervals obtained as the outcome for all the numeric attributes corresponding each of the four datasets has been recorded and exhibited in Table 3.4.

Table 3.3: Comparison of the Eight Discretization Schemes for Labeled Data using CAIR Value

<table>
<thead>
<tr>
<th>Discretization Method</th>
<th>Iris</th>
<th>Ion</th>
<th>Hea</th>
<th>Pid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal Width</td>
<td>0.40</td>
<td>0.098</td>
<td>0.087</td>
<td>0.058</td>
</tr>
<tr>
<td>Equal frequency</td>
<td>0.41</td>
<td>0.095</td>
<td>0.079</td>
<td>0.052</td>
</tr>
<tr>
<td>Patterson-Niblett</td>
<td>0.35</td>
<td>0.192</td>
<td>0.088</td>
<td>0.052</td>
</tr>
<tr>
<td>IEM</td>
<td>0.52</td>
<td>0.193</td>
<td>0.118</td>
<td>0.079</td>
</tr>
<tr>
<td>Max. Entropy</td>
<td>0.30</td>
<td>0.100</td>
<td>0.081</td>
<td>0.048</td>
</tr>
<tr>
<td>CADD</td>
<td>0.51</td>
<td>0.130</td>
<td>0.098</td>
<td>0.057</td>
</tr>
<tr>
<td>CAIM</td>
<td>0.54</td>
<td>0.168</td>
<td>0.138</td>
<td>0.084</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>0.56</td>
<td>0.237</td>
<td>0.128</td>
<td>0.107</td>
</tr>
</tbody>
</table>

Discretization Using Rough Set Theory and Clustering 64
Table 3.4: Comparison of the Eight Discretization Schemes for Labeled Data using number of intervals

<table>
<thead>
<tr>
<th>Discretization Method</th>
<th>Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iris</td>
</tr>
<tr>
<td>Equal Width</td>
<td>16</td>
</tr>
<tr>
<td>Equal frequency</td>
<td>16</td>
</tr>
<tr>
<td>Patterson-Niblett</td>
<td>48</td>
</tr>
<tr>
<td>IEM</td>
<td>12</td>
</tr>
<tr>
<td>Max. Entropy</td>
<td>16</td>
</tr>
<tr>
<td>CADD</td>
<td>16</td>
</tr>
<tr>
<td>CAIM</td>
<td>12</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>12</td>
</tr>
</tbody>
</table>

Form the observations recorded in Table 3.3. It is noted that the proposed method achieved the highest class-attribute interdependency for three out of four data sets and yielded the second highest for the hea dataset. From table 3.2 observing the number of classes and the number of continuous attributes, amidst the seven discretization scheme the performance of CAIM & IEM are observed to be the better...
than rest, therefore the performance of proposed method has been compared with CAIM & IEM. It is note worthy that the total number of intervals produced by CAIM and IEM is approximately equals Number of Classes * Number of Continuous attributes. It is observed that the proposed algorithm was able to generate same number of intervals for the *iris* data as that of CAIM and IEM which is lowest. For *ion* data the number of intervals is second lowest but it is marginally higher than the lowest value in comparison to results of the other algorithms. For *hea* dataset the result shows second lowest number of intervals, only one more than the lowest number corresponding to the IEM algorithm. Though for *pid* dataset the number of intervals is significantly higher than the results achieved by CAIM and IEM, it is significantly less than the other results.

The performance of the proposed method is comparable for the labeled as well as unlabeled data. A comparison of the classificational performance of the two proposed schemes for the labeled and unlabeled data was carried out on the basis of results of the experiments on iris and pid datasets each. Table 3.5, presents the CAIR value and the number of intervals corresponding to the two schemes for this purpose. It is observed that the CAIR value for both schemes for each of the datasets is significantly close. However, the number of intervals varies for pid dataset. This may be attributed to the number of objects being more in the pid dataset therefore the number of distinct numeric values for each continuous attributes being more in number or the range of the distinct values being large. The numbers of intervals for the iris data are equal.
Table 3.5: Comparison of the Discretization by proposed scheme for labeled and unlabeled data

<table>
<thead>
<tr>
<th>Evaluation Parameter</th>
<th>Discretization Scheme</th>
<th>Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Iris</td>
</tr>
<tr>
<td>CAIR Value</td>
<td>Labeled</td>
<td>0.56</td>
</tr>
<tr>
<td></td>
<td>Unlabeled</td>
<td>0.53</td>
</tr>
<tr>
<td>Numbers of Intervals</td>
<td>Labeled</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Unlabeled</td>
<td>12</td>
</tr>
</tbody>
</table>

3.6 Conclusion

By the proposed method in the first phase the natural intervals of the values of the continuous attributes were obtained. The rough set theoretic membership function using $\text{POS}_{a_i}(D)$ when applied in the refinement step was able to maximize the mutual class-attribute interdependency. The refinement step was also able to yield possibly minimum number of intervals. The proposed method also satisfies the other criteria to compare the discretization methods in following ways:

- Supervised vs. Unsupervised: Since the DBSCAN is unsupervised the first phase is unsupervised while refinement process by fixing the minimum and maximum
points in an interval and maximum length of intervals is supervised. The method proposed in this paper for discretization is semi-supervised.

- **Global vs. Local:** In the discretization phase the natural intervals of attribute values are obtained without the aid of other features and thus the method is local, while in the refinement phase the consideration of the class labels takes a global view of the dataset.

- **Static vs. Dynamic:** In the first phase the values for each continuous attribute is clustered independently from other attributes while in the second phase refinement of the intervals of attribute values using class interdependency makes the process dynamic in nature.

Although the computational effort for the search algorithm for cut point has been reduced to half of N, the size of dataset, by implementing binary search for the cut point can further reduce the complexity of search step.

In the next chapter a method for granulation as data preprocessing has been proposed. The algorithm for granulation proposed in the next chapter has been founded on partitioning the data space based on attribute values. The discretization method proposed in this chapter has been applied to deal with numeric attributes in the proposed method of granulation.