DB-HReduction: A Data Preprocessing Algorithm for Data Mining Applications

XIAOHUA HU
College of Information Science and Technology, Drexel University
Philadelphia, PA 19104, U.S.A.
thu@cis.drexel.edu
(Received and accepted August 2002)
Communicated by N. Cercone

Abstract—Data preprocessing is an important and critical step in the data mining process and it has a huge impact on the success of a data mining project. In this paper, we present an algorithm DB-HReduction, which discretizes or eliminates numeric attributes and generalizes or eliminates symbolic attributes very efficiently and effectively. This algorithm greatly decreases the number of attributes and tuples of the data set and improves the accuracy and decreases the running time of the data mining algorithms in the later stage. © 2003 Elsevier Science Ltd. All rights reserved.

Keywords—Data mining, Data preprocessing, Data reduction, Horizontal reduction.

1. INTRODUCTION

Data preprocessing is an important and critical step in the data mining process, and it has a huge impact on the success of a data mining project. The purpose of data preprocessing is to cleanse the dirty/noise data, extract and merge the data from different sources, and then transform and convert the data into a proper format. Data preprocessing has been studied extensively in the past decade [1], and many commercial products such as Informatica [2] and Data Joiner [3] have been applied successfully in many applications. Most of the studies and commercial systems focus on data cleaning, extraction, and merging, even though some provide limited transformation capability, but they cannot meet the requirements of a lot of complicated data mining tasks. A typical data set in data mining application tends to be high dimensional (hundreds even thousands of feature variables) with both numerical and symbolic type and has millions of tuples. Many actual applications, such as telephone billing, text categorization, and supermarket transactions, may collect hundreds to thousands of feature variables. Nonetheless, not all of the feature variables inherent in these applications are useful for sophisticated data analysis, for example, for data mining. One reason for this phenomenon is because most of the time, the data are collected without “mining” in mind. In addition, the existence of numeric data and the primitive symbolic values of symbolic attributes create a huge data space determined by the numeric data and primitive symbolic values. In order to mine the knowledge pattern from the
data efficiently, it is essential to reduce the data set before the mining algorithm can be mined. There are two directions to reduce the data set. One is to reduce the dimensions (attributes) of the data set by eliminating all those unnecessary attributes, and the other is to reduce the number of the tuples in the data set by discretizing the numeric attributes and generating the symbolic attribute values to high-level concept; a lot of tuples will be combined into one after the discretization and generalization, thus reducing the data tuples in the data set.

In [4], an attribute-oriented induction method was proposed, which substitutes the primitive data by high-level concepts in the concept hierarchies, assuming that the concept hierarchies (which permit the learned rules to be represented in a simple and explicit form) are provided by experts. Nonetheless, there are several drawbacks with this method: first, its generalization relies on the concept hierarchies. Recall that, if the attribute is of a symbolic type, generalization may be easy since the distinct number of values for symbolic attributes is limited. For numeric attributes, it is very difficult to generalize because of the wide distribution of the values or because of the many distinct different values. Second, this method uses a threshold $T_N$ (the number of desirable tuples) as the stopping criterion for the generalization. There is no definite rule to choose $T_N$. Normally we do not know the desirable value for $T_N$ before the generalization. The choice of $T_N$ has a great effect on the generalized relation. A larger $T_N$ will undergeneralize the data, while a small $T_N$ will overgeneralize the data and introduce many inconsistencies which did not exist before. Thus, this action would change the characteristics of the data.

Numeric attributes in the data set also create a problem for the rule induction algorithm. Normally numeric attributes are discretized into a few intervals prior to running the rule induction algorithms. There are many different discretization algorithms in the literature [5]. It is generally understood that no discretization algorithm is the best across all application domains. All of these algorithms generally discretize the numeric attribute into some intervals based on some criterion measure without checking whether the numeric attribute is actually relevant to the learning concept.

It is desirable that a discretization algorithm can automatically discretize the numeric attributes as well as remove those irrelevant numeric ones. Based on this philosophy, we develop a novel discretization and elimination algorithm DBChiMerge: DBChiMerge inherits the advantages of ChiMerge [6] and Chi2 [7]. Similar to Chi2, the significant level $\alpha$ which is used for merging values of the attributes is automatically determined according to the stopping criterion. However, DBChiMerge cannot handle symbolic features. A natural solution is to integrate symbolic attribute generalization and numeric attribute discretization. Based on this consideration, we propose a hybrid algorithm DB-HReduction, which integrates DBChiMerge and the attribute-oriented generalization method. Our algorithm DB-HReduction can process both numeric and symbolic attributes efficiently and effectively.

2. DBCHIMERGE: DISCRETIZATION/ELIMINATION OF NUMERIC ATTRIBUTE

DBChiMerge uses the $\chi^2$ statistics to determine if the relative class frequencies of adjacent intervals are distinctly different or if they are similar enough to justify merging them into a single interval. $\chi^2$ is a statistical measure used to test the hypothesis that two distinct attributes are statistically independent. Applied to the discretization problem, it tests the hypothesis that the class attributes are statistically independent of to which two adjacent intervals an example belongs. If the conclusion of the $\chi^2$ test is that the class is independent of the interval, then the interval should be merged. On the other hand, if the $\chi^2$ test concludes that they are not independent, it indicates that the difference in irrelevant class frequencies is statistically significant, and therefore, the intervals should remain separate. If a numeric attribute is discretized into one interval only without generating more inconsistencies than allowed, it simply means that this attribute is not relevant to determine classes according to the $\chi^2$ statistics, and hence, this
attribute is removed. This discretization procedure can automatically discretize attributes as well as remove the insignificant ones.

In DBChiMerge, we need to construct a contingency table first in order to compute the $\chi^2$. A sample contingency table for numeric attribute $C_i$ and the decision attribute $D$ is shown in Table 1. The DBChiMerge algorithm consists of an initialization step and a bottom-up merging process, where intervals are continuously merged until a termination condition is satisfied. Here is the algorithm.

**Algorithm 1. DBChiMerge: Numeric Attribute Discretization/Elimination.**

**Input:** A relation table $T$, numeric attribute $C_i$, and decision attribute $D$, $\chi^2$ threshold value $\beta$.

**Output:** Intv—a set of intervals for attribute $C_i$.

**Method:**

1. Construct the contingency table for the attributes $C_i$ and $D$
2. Compute the $\chi^2$ value for each adjacent interval
3. While exist some pair of intervals with $\chi^2$ less than $\beta$ Do {
   3.1 Combine the pair of adjacent interval with the lowest value and add up their corresponding frequency counts in the contingency table
   3.2 Recalculate the $\chi^2$ value for each pair of adjacent intervals
4. If only one interval left, Then remove $C_i$ from $T$
   Else replace the numerical values by the corresponding intervals.

A contingency table is done by

1. selecting the numeric attribute $C_i$ and decision attribute $D$ from the original table,
2. sorting the dataset based on the numerical attribute $C_i$, and
3. calculating the count values.

The three steps are done by the following SQL statements.

```sql
Select $C_i$, $D$, Count(*)
From $T$
Group by $C_i$, $D$
Order by $C_i$
```

To compute the last row of $c_{ti}$ ($i = 1, \ldots, m$) in the contingency table, execute the following SQL statements:

```sql
Select count(*) from $T$
Where $D = d_i$
```

And $c_{total}$ can be computed by the SQL statement

```sql
Select count(*) from $T$
```
The formula \( \chi^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} (a_{ij} - e_{ij})^2 / e_{ij} \) is used to calculate the \( \chi^2 \) value for each pair of adjacent intervals, where \( e_{ij} = \text{expected frequency of } a_{ij} = r_i * c_{tj} / c_{\text{total}} \). If either \( r_i \) or \( c_{tj} \) is 0, \( e_{ij} \) is set to 0. (Obtaining the \( \chi^2 \) value also requires specifying the number of degrees of freedom, which will be one less than the number of classes.) For example, when there are three classes (thus, two degrees of freedom), the \( \chi^2 \) value at 95% level is 5.99. The meaning of this threshold is that among cases where the class and attributes are independent, there is a probability that the computed \( \chi^2 \) value will be less than 5.99, and thus, \( \chi^2 \) values in excess of the threshold imply that the attribute and class are not independent.

As a result, choosing high values for \( \chi^2 \) threshold causes the merging process to continue, resulting in discretization with fewer and larger intervals. The recommended procedure for using the DBChi-Merge would be to set the \( \chi^2 \) threshold \( \beta \) at the 95% significant level.

### 3. THE HORIZONTAL REDUCTION ALGORITHM: DB-HREDUCTION

In order to control the data reduction to avoid introducing too much inconsistency to distort the data property, inconsistency checking is used. The inconsistency criterion specifies to what extent the dimensionally reduced data can be accepted. Following the method proposed in [7], the inconsistency rate of a dataset is calculated as follows:

1. two instances are considered inconsistent if they match except for their class labels;
2. for all the matching instances (without considering their class labels), the inconsistency count is the number of the instances minus the number of most frequent class labels seen; for example, there are \( n \) matching examples among them, \( s_1 \) tuples belongs to class 1, \( s_2 \) to class 2, and \( s_3 \) to class 3, where \( s_1 + s_2 + s_3 = n \) (if \( s_3 \) is the largest among the three, the inconsistency count is \( n - s_3 \)); and
3. the inconsistency rate is the sum of all of the inconsistency counts divided by the total number of instances.

Horizontal reduction is performed on the data set by examining attributes one by one. In the algorithm presented below, for a numeric attribute, based on the inconsistency threshold \( \delta \), DBChi-Merge can discretize the attribute into a few intervals. If a numeric attribute is discretized into one interval without generating more inconsistencies than \( \delta \), it simply means that this attribute is not relevant to determine classes according to the \( \chi^2 \) statistics, and hence, this attribute is removed. This discretization procedure can automatically discretize the numeric attribute as well as removing those irrelevant ones. For symbolic attributes, the substitution continues as long as it satisfies the inconsistency threshold. For an attribute with no concept hierarchy and many distinct values, although this attribute can distinguish the class uniquely, it has no predictive power, and thus, is removed also. For example, in a telephone billing database, a set of features includes the customer’s ID (CID) and features which describe a customer’s behavior. This CID is unique for each customer, but it is not useful to derive any general patterns for customers and has no predictive capability, so CID should be dropped. This attribute value substitution corresponds to climbing generalization tree and attribute removal corresponds to dropping conditions in machine learning. As a result of generalization and discretization, different tuples may become identical tuples, and the redundant tuples are merged. This procedure greatly reduces the number of tuples horizontally.

**Algorithm 2. DB-HReduction.** (DataBases Horizontal Reduction.)

Input:

(i) a relational table \( T(C_1, C_2, \ldots, C_n, D) \),
(ii) allowable inconsistency rate \( \delta \),
(iii) \( \chi^2 \) threshold value \( \beta \),
(iv) noise filter threshold \( \gamma \).
Output: a dataset satisfying the inconsistency criterion after discretization and generalization.
Method:
1. For each attribute $C_i$ in the data set {
2. While (Inconsistency($T$) < $\delta$) Do {
3. If $C_i$ is a numeric attribute
   Then apply DBChiMerge to discretize or eliminate the attribute;
   If $C_i$ is a symbolic attribute
   Then If there is a concept hierarchy for it
   Then replace the primitive values by high level concepts
   Elseif the number of distinct values for the attribute is greater than some predefined
   threshold value and there is no concept hierarchy for it,
   Then remove it from $T$)
4. Merge redundant tuples and records the number of identical tuples in vote
5. Compute the frequency ration of each tuple
6. Filter out those tuples with frequency ration less than noise filter threshold $\gamma$.

Example 1. Suppose we have a collection of Japanese and American cars with the attributes plate, make_model, color, width, height of the car, number of cylinders, weight, power, and mileage depicted in Table 2 (mileage is the decision attribute) and the concept hierarchy table for the car relation is shown in Figure 1. The task-specific concept hierarchies (shown in Figure 1) are constructed by both domain experts and knowledge discovery tools based on the statistics of data distribution in the database. The most general concept is the null description (described by a reserved word “ANY”), and the most specific concepts correspond to the specific values of attributes in the database.

Table 2. Car relation.

<table>
<thead>
<tr>
<th>Plate#</th>
<th>Make_Model</th>
<th>Color</th>
<th>Width</th>
<th>Height</th>
<th>Cylinder</th>
<th>Door</th>
<th>Power</th>
<th>Weight</th>
<th>Mileage</th>
</tr>
</thead>
<tbody>
<tr>
<td>LKIPO8</td>
<td>Ford_Escort</td>
<td>green</td>
<td>63.4</td>
<td>52.4</td>
<td>6</td>
<td>2</td>
<td>high</td>
<td>1020</td>
<td>low</td>
</tr>
<tr>
<td>IUTY89</td>
<td>Honda_Civic</td>
<td>red</td>
<td>65.9</td>
<td>53.7</td>
<td>4</td>
<td>4</td>
<td>high</td>
<td>890</td>
<td>high</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PO87YH</td>
<td>Mazda_626</td>
<td>black</td>
<td>65.7</td>
<td>54.0</td>
<td>4</td>
<td>2</td>
<td>low</td>
<td>850</td>
<td>medium</td>
</tr>
<tr>
<td>KMN98O</td>
<td>Dodge_Daytona</td>
<td>navy</td>
<td>63.5</td>
<td>55.3</td>
<td>6</td>
<td>4</td>
<td>high</td>
<td>1500</td>
<td>high</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TYUR78</td>
<td>Toyota_Tercel</td>
<td>brown</td>
<td>66.6</td>
<td>53.7</td>
<td>4</td>
<td>4</td>
<td>low</td>
<td>780</td>
<td>high</td>
</tr>
</tbody>
</table>

\{(Honda_Civic, Honda_Acura, ..., Honda_Accord) \subset Honda\}
\{(Toyota_Tercel, ..., Toyota_Camry) \subset Toyota\}
\{(Mazda_323, Mazda_626, ..., Mazda_939) \subset Mazda\}
\{(Toyota, Honda, ..., Mazda) \subset Japan(Car)\}
\{(Ford_Escort, Ford_Probe, ..., Ford_Taurus) \subset Ford\}
\{(Chevrolet_Corvette, Chevrolet_Camaro, ..., Chevrolet_Corsica) \subset Chevrolet\}
\{(Dodge_Stealth, Dodge_Daytona, ..., Dodge_Dynasty) \subset Dodge\}
\{(Ford, Dodge, ..., Chevrolet) \subset USA(Car)\}
\{(Japan(Car), ..., USA(Car)) \subset Any(Make_model)\}

Figure 1. Concept hierarchy table.

The first attribute, “Plate#”, is the key attribute of the relation. The key value is distinct for each tuple in the relation. If there is no higher-level concept provided for such an attribute in the concept tree, the values of the attribute cannot be generalized, and they should be removed in the generalization. Also, other candidate key attributes or nonkey attributes (like the color of the car) can be eliminated under a similar condition. We then examine the remaining attributes and perform generalization for symbolic attributes and discretization for numeric attributes. For symbolic attributes, the primitive value in the attribute is generalized to a more abstract level,
e.g., from Mazda 323 to Mazda, and then to Japan for attribute Make_model. For a numeric attribute, we apply DBChi-Merge and discretize the attribute into a few intervals based on the inconsistency rate. For example, the weight of the car is discretized into three intervals [500–800), [801–1050], and [1051–1500]. Some numeric attributes, for example, width are discretized into one interval [63.4–66.6], height into [52.4–55.3], which means that these two attributes are not relevant to the decision attribute mileage based on the \( \chi^2 \) value, so they are removed. After discretization and generalization, Table 3 is obtained.

<table>
<thead>
<tr>
<th>Make</th>
<th>Cylinder</th>
<th>Door</th>
<th>Power</th>
<th>Weight</th>
<th>Mileage</th>
<th>Vote</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>6</td>
<td>2</td>
<td>high</td>
<td>[1051–1500]</td>
<td>low</td>
<td>20</td>
</tr>
<tr>
<td>Japan</td>
<td>4</td>
<td>4</td>
<td>high</td>
<td>[801–1050]</td>
<td>high</td>
<td>35</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Japan</td>
<td>4</td>
<td>2</td>
<td>low</td>
<td>[500–800]</td>
<td>medium</td>
<td>40</td>
</tr>
<tr>
<td>USA</td>
<td>6</td>
<td>4</td>
<td>high</td>
<td>[1051–1500]</td>
<td>high</td>
<td>60</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Japan</td>
<td>4</td>
<td>4</td>
<td>low</td>
<td>[500–800]</td>
<td>high</td>
<td>60</td>
</tr>
</tbody>
</table>

A lot of computational intensive operations in our algorithms (DB-Hreduction, DBChiMerge) are performed using the database Count, Update operations. For example, for the above example, DBChiMerge identifies the weight to be discretized into three intervals, and then the following SQL statements are automatically based on the discretization results and update the table to the desired intervals.

- Update Car Set Weight = [1050–1500]
- Update Car Set Weight = [1050–1500] Where Weight <= 1050
- Update Car Set Weight = [1050–1500] Where Weight <= 800

The above SQL code example achieves the following if-then-else block logic.

- If Weight <= 800 then Weight = [500–800]
- Else if Weight <= 1050 then Weight = [801–1050]
- Else Weight = [1051–1500]

For attribute Make_model, the algorithm finds out that the primitive values of the Make_model should be generalized to their manufacture country in the hierarchy tree, the following SQL statement is generated and executed.

- Update Car Set Make_model = Japan Where Make_model In (Honda_Civic, Honda_Acura, Toyota_Tercel, ..., Mazda_939)
- Update Car Set Make_model = USA Where Make_model In (Ford_Escort, ..., Chevrolet_Corvette, ..., Dodge_Dynasty)

Suppose there are \( N \) tuples in the databases, \( A_s \) symbolic and \( A_n \) numeric attributes for each tuple, and \( H \) levels for each concept tree; then time complexity in the worst case is analyzed as follows: checking the inconsistency rate of data set is \( O(N) \), the time for substituting the lower level concepts by the higher-level concepts is \( N \), and the time for checking redundant tuples is \( O(N \log N) \). Since the height of concept tree is \( H \), the time spent on each symbolic attribute is at most \( H \ast (N + N \log N) \). For each numeric attribute, discretization is \( O(N \log N) \). Obviously, the upper bound of the total time for processing is \( A_s \ast H \ast (N + N \log N) + A_n \ast (N + N \log N) \). In general, \( A_s \), \( A_n \), and \( H \) are much smaller than \( N \) in a large database, and therefore, the time complexity of our approach is \( O(N \log N) \) in the worst case.

4. CONCLUSION

To reduce the search space, our algorithm reduces both the attributes and tuples of the data set, which reduces the search spaces to maximum extent without losing essential information. The
algorithm DB-HReduction is implemented as a preprocessing step in the DBClass algorithm [8]. In our method, attribute generalization, discretization, and elimination are integrated. Numeric attributes are discretized into a few intervals. If discretization results in one interval, then the attribute is removed. The primitive values of symbolic attributes are replaced by high-level concepts, and some obvious superfluous attributes or irrelevant symbolic attributes are also eliminated. The data reduction is done by merging identical tuples after substituting an attribute value by its higher value in a predefined concept hierarchy for symbolic attributes, or the discretization of continuous (or numeric) attributes, or the removal of insignificant or irrelevant numeric and symbolic attributes. This algorithm greatly decreases the number of tuples for the later data mining algorithm. The benefits of our algorithm are two-fold:

1. Increase the accuracy of the mining algorithm, since these superfluous or irrelevant attributes tend to fool the data mining algorithms, generate spurious or bogus “pattern”.
2. Reduce the running time of the mining algorithm, thus speeding up the mining procedure.

REFERENCES

4. X. Hu and N. Cercone, Data mining via discretization and generalization, Knowledge and Information Systems, (1999).
10. J. Han, Y. Cai and N. Cercone, Knowledge discovery in databases: An attribute-oriented approach, In 18th VLDB Conference.