A data mining approach to support the development of new fuels and technology

G. S. Terra¹, ², C. L. Curotto³ & N. F. F. Ebecken¹
¹COPPE/ UFRJ, Civil Engineering Department, Cidade Universitária, Ilha do Fundão, Caixa Postal 68.506, Rio de Janeiro, RJ, Brasil
²CEFET-Campos/UNED-Macaé, Rodovia Amaral Peixoto, km 164, Imboassica, Macaé, RJ, Brasil, CEP 27.973-030
³DCC/UFPR, PPGEC, CP 19011, 81531-990 Curitiba, PR, Brasil

Abstract

In the present work, data mining techniques are used to model the non trivial relationships between properties that characterize the fuels, engine technologies and car emissions. Using models to predict car emissions from fuel properties and technologies engines can improve their development process. To support models of relational data, using the Object Linking and Embedding Database for Data Mining technology, a Simple Naive Bayes Incremental classifier was implemented in Microsoft® SQL Server™, supporting numeric input attributes, multiple prediction attributes and incremental update of data. Computational experiments using real word data sets were made to evaluate the results obtained by this classifier.

1 Introduction

The automobile industries are stimulated to develop an efficient car, with new technologies, that improves the autonomy and reduce the pollutant emission. The downstream oil company also contributes when develops new fuels that provides a positive interaction with those new car technologies. The fuel characteristics are modified when additives are incorporated in their formulation or changes are made in the proportion of their basic composition. During the fuel development phase different combinations are tested in cars where theirs emissions are measured in controlled tests. The results were stored in a database and a classification model was built to improve this research area.
To achieve the tight coupling of Data Mining (DM) techniques in Database Management Systems (DBMS) technology, a number of approaches have been developed in the last years. These approaches include solutions provided by both research groups of companies and academic researches.

Toward this objective the Microsoft® (MS) OLE DB for DM technology [1] provides an industry standard for developing DM algorithms. This technology was included in the last release of the MS SQL Server™ [2]. The Analysis Services component of this software includes a DM provider supporting two algorithms: one for classification by decision trees [3] and another for clustering [4]. The DM Aggregator feature of this component and the OLE DB for DM Sample Provider [5] made possible for developers and researchers to implement new DM algorithms.

On the other hand, relational data impose some additional effort to deal with them. While conventional classifiers assume that data sets are recorded in single flat files or tables, a relational classifier have to face with more complex data structures. For example, consider the simple Customer relational data showed by Figure 1. In order to predict customer behavior one needs customer personal data provided by table Customer and customer purchases provided by tables
Customer 1 Purchages and Customer 2 Purchages. This simple example shows clearly how complex can be relational data.

One approach to solve this problem is to use a single flat table constructed by performing a relational join operation on the tables. This approach produces an extremely large, and impractical to handle, table with lots of repeated data. In consequence the multi-relational approach has been receiving considerable attention in the literature [6]. This second approach relies on developing specific algorithms to deal with the relational feature of the data.

OLE DB for DM technology supports nested tables (also known as table columns). As showed by Figure 1, The Data Mining Model (DMM) customer_sn1 has three columns: Id, the key column that identifies the customer; Gender, the column of customer personal data; and Product Purchases, the table column that provides customers behavior information. IdP is an alias of Id, the key column that identifies the customer in the Product Purchases table. By using this rich feature complex relational data mining models can be analysed.

We propose a third approach: to use a traditional enhanced Bayesian classifier, the SNB1 incremental classifier [7] using the OLE DB for DM technology. The formulation of this classifier is showed and the implementation is briefly described. Some early experimental results carried out by using this classifier with some known relational data sets are showed.

2 The tools

The implementation and the experiments was made by using an IBM PC compatible microcomputer, Intel Pentium V 2.40 GHz processor inside, 512 MB of RAM memory, 768 MB initial and 1024 MB maximum virtual memory, 40 MB hard disk. The operating system was the MS Windows® 2000 Advanced Server Service Pack 4 (SP4) with MS SQL Server™ 2000 Enterprise SP3A installed. The development tools were the MS Visual Studio 6.0 SP5 with Visual C++®, Visual J++® and Visual Basic® compilers; MS Platform SDK February 2003 Edition and Sandstone Visual++ Parse 4.00. The template for developing the DM provider was the source code of OLE DB for DM Sample Provider [5].

3 The SNB1 classifier

The Naive Bayes classifier is well known and has been described in many papers and books [8]. The main objective of a classifier is to predict the class attribute of a case based on a set of cases of input attributes. Succinctly the SNB1 classifier uses counts of discrete and continuous attributes occurrences and means and standard deviations of continuous attributes to do this task. The support for incremental update of data sets is achieved storing the sum and the square sum of continuous attribute values, computing means and deviations as necessary. Multiple prediction attributes are supported by an adequate data structure [7].
3.1 Formulation of SNB\(^t\) classifier

Suppose a training data set formed by:

- \(v\) total number of training cases;
- \(X^t = x_{i^t}^t, i = 1, n\) vector of a training case for prediction attribute \(t\), formed by \(n\) attribute values of vector \(A^t\);
- \(A^t = a_{i^t}^t, i = 1, n\) vector of attribute values of training data set for prediction attribute \(t\), formed by \(n\) values;
- \(O^t = o_{i^t}^t, i = 1, n\) vector of number of possible attribute values for each attribute \(a_{i^t}^t\), for prediction attribute \(t\), formed by \(n\) values. For a continuous attribute the vector element has a value equal to 2 (one for existing values and another for missing values). For a discrete attribute the vector element has a value equal to the number of discrete values plus one (for missing values);
- \(C^t = c_{j^t}^t, j = 1, m^t\) vector of classes of each case may belong, for prediction attribute \(t\);
- \(p\) total number of prediction attributes;
- \(t = 1, p\) prediction attribute index.

Suppose an unknown data set represented by:

- \(u\) number of unknown cases;
- \(Y^t = y_{i^t}^t, i = 1, n\) vector of an unknown case, for prediction attribute \(t\), formed by \(n\) attribute values of vector \(A^t\).

For an unknown case of this data set, represented by vector \(Y^t\), for prediction attribute \(t\), the SNB\(^t\) classifier will select the class with the greater a posteriori probability using the following equation:

\[
P(c_j^t | Y^t) > P(c_k^t | Y^t) \text{ for } 1 \leq k \leq m^t, k \neq j
\]

Being \(P(c_j^t | Y^t)\) (probability of event of class \(c_j^t\) for a case \(Y^t\)) for the prediction attribute \(t\), computed by the following equation:
\[ P(c_j^t | Y^t) = \frac{s_j^t}{v} \prod_{i=1}^{n} P(y_i^t | c_j^t) \quad (3-2) \]

where:

\( s_j^t \)  
number of training cases of class \( c_j^t \);  

\( v \)  
total number of training cases;  

\( P(y_i^t | c_j^t) \)  
probability of event of value \( y_i^t \) for an attribute \( a_i^t \) of an unknown case \( Y^t \) for a class \( c_j^t \).

The probabilities \( P(y_i^t | c_j^t) \), for prediction attribute \( t \), are computed from the training data set by the following equations, depending on the type of attribute \( a_i^t \). If \( a_i^t \) is discrete, then:

\[ P(y_i^t | c_j^t) = \frac{r_{ji}^{th}}{s_j^t} \quad (3-3) \]

where:

\( r_{ji}^{th} \)  
number of training cases of class \( c_j^t \) with the value \( y_i^t \), order \( h \), for attribute \( a_i^t \);  

\( s_j^t \)  
number of training cases of class \( c_j^t \).

On the other hand, if \( a_i^t \) is continuous, then the following equation will be used:

\[ P(y_i^t | c_j^t) = \frac{1}{\sqrt{2\pi\sigma_{ji}^t}} e^{-\frac{1}{2} \left( \frac{y_i^t - \mu_{ji}^t}{\sigma_{ji}^t} \right)^2} \quad (3-4) \]
where:

\[ y_t^i \] value \( y_t^i \) for attribute \( a_t^i \) of unknown case \( Y_t \);

\[ e \] Neper's number;

\[ \mu_{ji}^t \] and \[ \sigma_{ji}^t \] respectively, mean and standard deviation for attribute values \( x_t^i \) for attribute \( a_t^i \) of training cases of class \( c_j^t \), computed by the following equations.

The mean, for prediction attribute \( t \), is computed by:

\[
\mu_{ji}^t = \frac{1}{r_{ji}^{t1}} \sum_{k=1}^{r_{ji}^{t1}} x_{ij}^{tk} \tag{3-5}
\]

where:

\[ z_{ji}^t \] sum of values \( x_t^i \) for attribute \( a_t^i \) for training cases of class \( c_j^t \);

\[ r_{ji}^{t1} \] number of training cases of class \( c_j^t \) of any value \( x_t^i \), order \( h = 1 \) (\( h = 1 \) is used for existing values of continuous attributes and \( h = 0 \) is used for missing values of continuous and discrete attributes), for attribute \( a_t^i \);

\[ x_{ij}^{tk} \] values \( x_t^i \) for attribute \( a_t^i \) for training cases of class \( c_j^t \).

The standard deviation, for prediction attribute \( t \), is computed by:

\[
\sigma_{ji}^t = \sqrt{\frac{1}{r_{ji}^{t1}} \left( \frac{z_{ji}^{t2}}{r_{ji}^{t1}} - 1 \right)} \tag{3-6}
\]
where:

\[ q_{ji}^{t} \]  
sum of the squares of the values \( x_{ij}^{t} \) for attribute \( a_{i}^{t} \) 
for training cases of class \( c_{j}^{t} \); 

\[ z_{ji}^{t} \]  
sum of the values \( x_{ij}^{t} \) for attribute \( a_{i}^{t} \) for training 
cases of class \( c_{j}^{t} \); 

\[ r_{ji}^{t1} \]  
number of training cases of class \( c_{j}^{t} \) of any value \( x_{ij}^{t} \), 
order \( h = 1 \) (\( h = 1 \) is used for existing values of 
continuous attributes and \( h = 0 \) is used for missing 
values of continuous and discrete attributes), for 
attribute \( a_{i}^{t} \).

### 3.2 Training and prediction algorithms of SNB\(_i\) classifier

To deal with incremental update of training data, the Bayesian classifier 
described in the previous item needs to store some data, which will be retrieved 
when new training data is inserted.

### 3.3 Implementation of SNB\(_i\) classifier

The implementation was made starting from the source code of the OLE DB for 
DM Sample Provider [5].

As stated by Netz et al [9][10] the key operations that must be supported by a 
DM provider algorithm on DM models are reproduced as follows:

- Define a mining model, identifying the set of prediction attributes, the 
  set of input attributes, and the algorithm used to populate the mining 
  model;
- Populate a mining model from training data using the algorithm 
  specified;
- Predict attributes for new data using a mining model that has been 
  populated;
- Browse a mining model for reporting and visualization applications.

These key operations will be the key steps of the DM provider implementation 
and the well known AllEletronics customer database [8] will be used as the 
training data set to illustrate these steps activated by DM SQL statements.

The syntax of DM SQL commands of the new algorithm is defined using a 
parse analyzer and only a few modifications in the source code are necessary to 
support new algorithms. The Relational Mining Model Editor of Analysis 
Services Manager can be used to create DM SQL commands [1]. The Sample 
Provider source code includes all necessary steps to create the mining model.
The DM SQL command showed below is generated automatically by Analysis Services Manager and is used to populate the mining model using data stored on MS SQL Server. All support for this task must be developed for new algorithms. The model data structure is defined and all functions related to training the data set, assembling the model tree, saving and loading this model are developed.

The DM SQL command to predict attributes can be manually assembled or generated by the Data Transformation Services of MS SQL Server. All support for this task must be developed for new algorithms.

The model of DM is exposed for visualization and report generator applications by means of rowsets. The content of these rowsets is retrieved by DM provider client applications by DM SQL Select queries. All support for this task must be developed for new algorithms.

4 Computational experiments

The classifiers used in the experiments are the SNB\textsuperscript{i} incremental classifier [10] and the MSDT classifier [3] included in MS Analysis Services component of MSSQL.

4.1 Data

The prepared data set owns 130,143 cases with 63 input attributes and one prediction attribute with two classes. This data set was split in two: training and test data sets. This split was achieved using a uniform random distribution maintaining the same class distribution. The training data set are composed by 89,543 cases (68.80% of the whole data) and the test data set is composed by 40,600 cases (31.20% of the whole data). The Tables 1 and 2 show the results of elapsed processing times and accuracy.

Table 1: Elapsed times(s).

<table>
<thead>
<tr>
<th>Step</th>
<th>SNB\textsuperscript{i}</th>
<th>MSDT\textsuperscript{1}</th>
<th>MSDT\textsuperscript{2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>127</td>
<td>838</td>
<td>27,396</td>
</tr>
<tr>
<td>Train Prediction</td>
<td>632</td>
<td>471</td>
<td>580</td>
</tr>
<tr>
<td>Test Prediction</td>
<td>290</td>
<td>215</td>
<td>244</td>
</tr>
<tr>
<td>Total</td>
<td>1,049</td>
<td>1,524</td>
<td>28,220</td>
</tr>
</tbody>
</table>

Table 2: Accuracy (%).

<table>
<thead>
<tr>
<th>Data set</th>
<th>SNB\textsuperscript{i}</th>
<th>MSDT\textsuperscript{1}</th>
<th>MSDT\textsuperscript{2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>70.88</td>
<td>89.50</td>
<td>93.87</td>
</tr>
<tr>
<td>Test</td>
<td>70.88</td>
<td>87.96</td>
<td>90.24</td>
</tr>
</tbody>
</table>
In spite of showing the smallest training time, the SNB\textsuperscript{i} classifier presented again the worst result for accuracy. It should be noted the high elapsed processing time spend by the MSDT\textsuperscript{2} classifier. The lower accuracy presented by the SNB\textsuperscript{i} classifier is probably due to the fact that this problem presents dependence among the input attributes, not considered by this classifier.

5 Conclusion

The OLE DB for DM technology proved to be a very useful tool to implement DM algorithms, achieving the complete database querying and mining integration.

It was demonstrated that the use of microcomputers is possible for the solution of medium level DM tasks.

As predicted, due to its statistical formulation, as well as the data structure of the implementation, the SNB\textsuperscript{i} classifier showed high scalability inside the limits of the equipment used in the experiments.

Since doesn't exist an optimum algorithm for all kinds of problems, but only an optimum algorithm for each problem, the good results showed by the first experiment and the excellent results of elapsed processing times considering large data sets puts the SNB\textsuperscript{i} classifier as a mandatory option in the study of these sorts of problems. Using the incremental training associated to the prediction by parts, virtually problems with any size can be solved.

The non linear behavior of the classifier for large data sets suggests additional research using equipments with larger capacity, including the use of federations of databases servers, as well as multiprocessors computers.

References


