Implementing data mining algorithms with Microsoft SQL Server

C. L. Curotto¹ & N. F. F. Ebecken²
¹CESEC/UFPR – Civil Engineering Graduate Program, Brazil
²COPPE/UFRJ – Civil Engineering Graduate Program, Brazil

Abstract

The OLE DB for DM (Microsoft's object-based technology for sharing information and services across process and machine boundaries focused on database mining applications) specification provides an industry standard for implementation of data mining algorithms aggregated with Microsoft SQL Server 2000. The Simple Naïve Bayes classifier is implemented using the OLE DB for DM Resource Kit. Numeric input attributes, multiple prediction trees and incremental classification are considered. All necessary steps to implement this algorithm are explained and discussed. Some results are shown to illustrate the capabilities of the implementation.

1 Introduction

Nowadays database system managers like MS (Microsoft) SQL (Standard Query Language) Server [1] are available, with resources for manipulation of terabytes of data with parallel processing of queries (with multiprocessor servers) using microcomputers [2]. This situation suggests the integration of DM technology by using database managers to enlarge the scope of this technology at a low cost.

This approach of integration, achieved by tightly coupling DM and OLAP (On-Line Analytical Processing) techniques in database application development environments, is matter of current interest. It has been discussed in conferences such as ICDM'98 (First International Conference on DM), happened on September 1998 in Rio de Janeiro – Brazil, ICDM'00 (Second International Conference on DM), happened on July 2000, Cambridge – UK and more recent ones.

Agrawal [3] presented a methodology for tightly coupling of DM application to relational database system - IBM DB2/CS – based on utilization of user

In this way, the MS OLE DB for DM specification [12] was released in July 2000. This specification provides an industry standard for DM so that different DM algorithms from various DM developers can be easily plugged into user applications. OLE DB for DM specifies the API between DM consumers (applications that use DM features) and DM providers (software packages that provide DM algorithms). In September 2000, the MS SQL Server 2000 software was released with an important component: the Analysis Services. Included, the first OLE DB for DM provider supporting two algorithms: one for decision trees classification [13,14] and other for clustering [15]. In March 2001, the DM Aggregator feature for MS SQL Server 2000 Analysis Services was included in the Service Pack 1 [16]. To complete the set of tools the OLE DB for DM Resource Kit [17] was released in June 2001. This kit includes a DM Sample Provider that implements the Naive Bayes algorithm. Netz [18,19] presents an useful overview that describes the OLE DB for DM technology. Siedman brings out an excellent reference to SQL for DM [20].

However, the task of creating a provider and implementing an algorithm remains complex due to the lack of discussion about experiences in doing this work. The main objective of this paper is to describe an experience in this field, making this task affordable for DM algorithm developers. In this direction an enhanced version of the Simple Naive Bayes classifier is implemented considering numeric input attributes, multiple prediction trees and incremental classification.

2 The tools

The implementation was made with an IBM PC compatible microcomputer, Intel Pentium III 500 MHz processor inside, 512 MB of RAM memory, 30 MB hard disk. The operating system is the MS Windows 2000 Advanced Server SP2 (Service Pack 2) with MS SQL Server 2000 Enterprise SP2 installed. The development tools are MS Visual Studio 6.0 SP5 with Visual C++, Visual Java and Visual Basic compilers [21]; MS Platform SDK February 2001 Edition [22] and Sandstone Visual++ Parse 4.00 [23]. The template for developing the DM provider is the Sample Provider of OLE DB for DM Resource Kit [17]. Also Kim’s [24] utility DMSamp is used.
3 The implementation

3.1 The start point

The source code included with Sample Provider of OLE DB for DM Resource Kit [17] includes the complete implementation of an aggregated provider as well as the following:

a. All required OLE DB objects, such as session, command, and rowset;
b. The OLE DB for DM Syntax Parser;
c. Tokenization of input data;
d. Query processing engine;
e. A sample Naive Bayes algorithm;
f. Model archiving in XML and binary formats.

The Sample Provider must be prepared to receive the implementation of a new algorithm. The first step is to build the Sample Provider and then handle with a few installations problems (Kim [25]), the source code will be ready to be modified. The next step is to make an extensive correction of the main file of the syntax parser because this file is not clean and when any modification is made to change or insert a new algorithm, Visual Parse crashes and produces a corrupted file. The Sample Provider source code implements an aggregated DM provider. To implement a standalone provider that also can be used in aggregated mode some modifications must be made. This feature is useful for debugging the new algorithm and for using the provider with standalone applications without MS SQL Server. All of these tasks will be described in detail in [26].

After this task is completed, only modifications directly related to the new algorithm should be carried out.

3.2 The Simple Naive Bayes algorithm

This algorithm will not be described in details because it is well known through many papers and books: Han & Kamber book [27] is an excellent and didactic example. Also, many implementations of this algorithm exist, including the one carried out by Witten & Frank [28] in Weka project, used to compare the results of the DM provider implemented.

The main objective of a DM algorithm is to predict attributes based on a set of cases of input attributes. Succinctly the Simple Naive Bayes Classifier uses counts of occurrences of categorical and numeric attributes and means and standard deviations of numeric attributes to do this task. For supporting incremental update of the case set, it is enough to store the sum and the square sum of numeric attributes values, computing means and deviations as necessary. Multiple trees of prediction are supported by an adequate data structure.

3.3 OLE DB for DM

A complete specification of this technology is found in [12]. Netz [18,19] describes the basic philosophy and design decisions leading to the present specifi-
cation of OLE DB for DM. He stated precisely the key operations to be supported by a DM provider algorithm on DM models, reproduced as follows:

a. Define a mining model, identifying the set of attributes of data to be predicted, the set of attributes to be used for prediction, and the algorithm used to build the mining model.

b. Populate a mining model from training data using the algorithm specified.

c. Predict attributes for new data using a mining model that has been populated.

d. Browse a mining model for reporting and visualization applications.

These key operations will be described as follows in context of the DM provider implementation. Table 1 presents the results taken from AllElectronics customer database [27] that will be used as a training data set to illustrate the implementation steps. Table 2 presents the marginal model statistics.

### Table 1. AllElet training data set.

<table>
<thead>
<tr>
<th>Case</th>
<th>Age</th>
<th>Income</th>
<th>Student</th>
<th>Credit Rating</th>
<th>BuysComputer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>31-40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>31-40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>9</td>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>10</td>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>11</td>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>12</td>
<td>31-40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>13</td>
<td>31-40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>14</td>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

### Table 2. AllElet training data set - marginal model statistics

<table>
<thead>
<tr>
<th>Attribute states</th>
<th>Age</th>
<th>Income</th>
<th>Student</th>
<th>Credit Rating</th>
<th>BuysComputer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>5</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>31-40</td>
<td>medium</td>
<td>6</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>Missing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
3.4 Creating the mining model

The syntax of the new algorithm is defined using Visual Parse and a few modifications must be made in Sample Provider projects to insert the support for this new algorithm.

The Relational Mining Model Editor of Analysis Services Manager can be used to create DM SQL commands. Details about this language are found in MS SQL Server Books on Line [1], OLE DB for DM specification [12] and Siedman [20]. Concerning the Sample Provider, the OLE DB for DM Syntax Parser does all necessary steps to create the mining model.

The syntax of SQL for DM command to create a model with two predictive attributes of data set of the Table 1 is shown below:

```
```

3.5 Populating the mining model

The next lines show the syntax of SQL for DM command to populate the mining model using data from MS SQL Server:

```
INSERT INTO [AllElet_SNB'S] (SKIP, [Age], [Income], [Student], [CreditRating], [BuysComputer]) OPENROWSET ('SQLOLEDB.1', 'Provider=SQLOLEDB;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=AllElet;Data Source=CLC', 'SELECT "dbo''.AllEletTrNumReg" AS "NumReg", "dbo''.AllEletTrAge" AS "Age", "dbo''.AllEletTrIncome" AS "Income", "dbo''.AllEletTrStudent" AS "Student", "dbo''.AllEletTrCreditRating" AS "CreditRating", "dbo''.AllEletTrBuysComputer" AS "BuysComputer" FROM "dbo''.AllEletTrain"
```

This SQL is generated automatically by Analysis Services Manager. Support for inserting cases in the mining model must be developed for new algorithms. The developer must aim special attention when doing this task. The data structure that represents the model of the algorithm is defined and all functions related to training the data set, assembling the model tree, saving and loading this model are developed. This data structure must support the processing of the two following operations.

3.6 Predicting attributes

The syntax of SQL for DM command to predict attributes must be made by the user using an application such as Kim's [24] DMSamp or using the Data Transformation Services of MS SQL Server. Support for this task must be developed for new algorithms. An example of this command is shown below:
3.7 Browsing a mining model

Figure 1 shows the Relational Mining Model Editor browsing the two prediction model trees of the test sample and Figure 2 shows the complete tree of BuysComputer prediction attribute. It can be observed that the prediction tree list...
box is empty. It happens because Microsoft, against its own specification, does not permit this kind of model to be shown normally in your browser. The prediction tree list box is hard coded to look for the Microsoft Decision Trees algorithm and other algorithms can’t get their content to appear there. Tree node types can’t be used without crashing the browser, but using a simple trick this problem is avoided: the root node will be the model node and their children will be the roots of the prediction trees.

Figure 2. AllElet BuysComputer prediction tree.
For debugging purposes and for use in client applications DM SQL queries can be made to retrieve the rowset contents of the DM model. Figure 3 brings out Kim’s [24] utility DMSamp showing part of the results of the query that retrieves the entire contents of the test model:

```
SELECT * FROM [AllElet_SNB].CONTENT
```

Figure 3.

4 Conclusion

The experience of implementing a simple algorithm in OLE DB for DM technology was very useful as a start point for doing more complex tasks. The sharing of this experience will help other developers and researchers doing similar work. This technology seems to be an excellent tool to do efficient implementation of data mining algorithms to achieve the complete database querying and mining integration.

Acknowledgments

To CAPES and UFPR – Federal University of Paraná for its financial support. To Kris Ganjam, Claude Seidman, Raman Iyer, Peter Kim and Jamie MacLennan for their valuable help.

References


