Using Standard APIs for Data Mining in Prediction

Jaroslav Zendulka
Brno University of Technology, Czech Republic

INTRODUCTION

There are three standardization initiatives concerning application programming interfaces (API) for data mining — OLE DB for Data Mining (OLE DB for DM), SQL/MM Data Mining (SQL/MM DM), and Java Data Mining (JDM) (Schwenkreis, 2001; Grossman et al., 2002; Grossman, 2004). Their goal is to make it possible for different data mining algorithm providers from various software vendors to be easily plugged into applications. Although the goal is the same for all the APIs, the approach applied in each of them is different. OLE DB for DM is a language-based interface developed by Microsoft, SQL/MM DM is an ISO/IEC standard based on user-defined data types of SQL:1999, and JDM, which is being developed under SUN's Java Community Process, contains packages of data mining oriented Java interfaces and classes.

This short paper presents a simple example that shows how the APIs can be used in an application for prediction based on classification. The objective is to demonstrate basic steps that the application must include if we decided to use a given API. The example also helps to understand better different approaches the APIs are based on.

BACKGROUND

A brief characterization of all three APIs is presented in another article in this book (Zendulka, 2005). There are several introductory publications that deal with OLE DB for DM (Han & Kamber, 2001; Netz et al., 2001) and its referential implementation in Microsoft SQL Server 2000 (de Ville, 2001; Whitney, 2000; Tang & Kim, 2001). Keeping (2002) presented a scenario for using OLE DB for DM. The full specification was published by Microsoft in 2000 (OLE DB, 2000). Melton & Eisenberg (2001) presented an introductory paper to the SQL Multimedia and Application Package (SQL/MM), part of which the specification of data mining support known as SQL/MM DM is. The standard was accepted as ISO/IEC standard in 2002 (SQL 2002). JDM is developed as a Java Specification Request JSR-73. At the time of writing this paper, it was in the stage of final draft public review (Hornick et al., 2004).

Classification is a two-step process. In the first step, a model is built describing training data. The model can have a form of, for example, a decision tree. In the second step, the model is used for prediction. First, the predictive accuracy of the model is estimated using testing data. If the accuracy is considered acceptable, the model can be used to classify future (previously unseen) data (Han & Kamber, 2001).

These two steps are refined in the application according to the API used for implementation.

MAIN THRUST

Consider that based on a debt level, income level and employment type we want to predict the credit risk of a customer. A set of data is stored in a Customers table with columns CustomerID, DebtLevel, Income, EmploymentType, and CreditRisk. The CreditRisk column will be a target for prediction.

OLE DB for Data Mining

OLE DB for DM uses SQL CREATE, INSERT and SELECT statements with extended syntax and semantics in some cases to provide a language-based API for data mining services provided by a data mining provider that implements the required data mining technique. There are four basic steps that must be performed by our prediction application:

1. Define a data mining model.
2. Populate the data mining model.
3. Test the data mining model.
4. Apply the data mining model.

First, it is necessary to *define* a data mining model. OLE DB for DM provides a CREATE statement for this (the square brackets are name delimiters by convention for Microsoft SQL server):

```
CREATE MINING MODEL [RiskPrediction]          %Model name
(
[CustomerID]          LONG KEY,               %Source columns
[DebtLevel]           TEXT DISCRETE,
[Income]            TEXT DISCRETE,
[EmploymentType]    TEXT DISCRETE,
[CreditRisk]        TEXT DISCRETE PREDICT,     %Target
)
USING [Decision_Tree]                          %Algorithm
```

The statement specifies the name of the model, columns used for prediction, and a mining algorithm. Each column definition can contain a lot of specialized information. In our example, the CustomerID column is specified as the identifier of cases, the other columns are attributes of cases with discrete values, and the CreditRisk is a target for prediction. OLE DB for DM also supports cases that represent hierarchical data (nested tables). In addition, it is possible to introduce attributes with a predefined meaning, for example, the probability of an associated value, support, etc. Finally, a data mining algorithm is specified.

Once a data mining model is defined, it can be *populated* with training data stored in the Customers table:

```
INSERT INTO [RiskPrediction]
 ([CustomerId], [DebtLevel], [Income], [EmploymentType],
  [CreditRisk])
SELECT [CustomerID], [DebtLevel], [Income], [EmploymentType],
  [CreditRisk]
FROM Customers
```

The data mining algorithm analyzes the input cases and populates the patterns it has discovered to the data mining model. That is, the data mining model is built in this step.

The model can be *tested* or *applied* using a SELECT statement with a PREDICTION JOIN operator:

```
SELECT Customers.[CustomerID], [RiskPrediction].[CreditRisk],
FROM [RiskPrediction] PREDICTION JOIN Customers
 ON [RiskPrediction].[DebtLevel] = Customers.[DebtLevel] AND
  [RiskPrediction].[Income] = Customers.[Income] AND
  [RiskPrediction].[EmploymentType] = Customers.[EmploymentType]
```

For each case from the input (a row of the Customers table with new data), PREDICTION JOIN, using the conditions in the ON clause, will find the best prediction.

To validate the accuracy of the trained model, we can use a similar statement for testing data with both predicted and known values, and to process these values.

For descriptive mining techniques, a populated data mining model is only browsed using the ordinary SQL SELECT statement.

SQL/MM Data Mining

SQL/MM DM is based on SQL:1999 and its structured user-defined data types (UDT). The structured UDT is the fundamental facility in SQL:1999 (Melton & Simon, 2001). It allows specifying types of objects that encapsulate both attributes and methods. The SQL/MM DM standard specifies a set of UDTs supporting data mining. If the UDTs are implemented, any SQL-based application can employ them.

Consider our example and assume that we want to make it possible to re-compute the classification model from time to time. The following steps must be performed to specify a data mining task that defines the model (all names starting with the DM_ prefix are UDTs or methods specified by the standard):

1. Create a DM_MiningData value using a static method DM_defMiningData. The value will contain metadata describing *physical data* - the Customers table.
2. Create a DM_LogicalDataSpec value using a DM_genDataSpec method of the DM_MiningData value. The value will represent the input of a data mining task as a set of data mining fields - *logical data*. A DM_genDataSpec method can generate this representation from the DM_MiningData value.
3. Create a DM_ClasSettings value using a default constructor, assign the input of the model (the DM_LogDataSpec value) using a method DM_useClasDataSpec, and declare the CreditRisk column as a target field using a DM_setClasTarget method. The DM_ClasSettings value will contain *settings* that are used to generate the model, for example, cost rate, an input of the model, and a target field.
4. Create a DM_ClasBldTask value using a static method DM_defClasBldTask. The method takes two DM_MiningData values for training and testing data, and a DM_ClasSetting value as arguments. In our example, we assume that the model will only be trained during building. The DM_ClasBldTask value represents the *task* that builds the model.
5. Store the newly created DM_ClasBldTask value in a table; assume MyTasks, with Id and Task columns so that the classification model can be re-computed. Let us insert the value to the table with ID = 1.

```
All the steps can be expressed as a single SQL statement:
WITH MyData AS (—Physical data—(Step 1)
DM_MiningData::DM_defMiningData(‘Customers’)
)
INSERT INTO MyTasks (Id, Task) (Step 5)
VALUES (1,—ID of the model
  DM_ClasBldTask::DM_defClasBldTask((Step 4)
   MyData, NULL,—Training and testing data
   (
     (NEW DM_ClasSettings()). (Step 3)
      DM_useClasDataSpec(MyData.DM_genDataSpec())

                                                    (Step 2)
        ).DM_setclasSetTarget(‘CreditRisk’)  —Target column
  )
)
```