# Discovering Knowledge Hidden in a Chemical Database Using a Commercially Available Data Mining Tool

**C.H.Bryant[♣◇], A.E.Adam[♣◇], G.V.Conroy[♣◇], D.R.Taylor[♣♡] and R.C.Rowe[♠]**

[♣]University of Manchester Institute of Science and Technology, PO Box 88, Manchester, M60 1QD.
([◇]Computation Department [♡]Chemistry Department)
[♠]Zeneca Pharmaceuticals, Alderley Park, Macclesfield, Cheshire, SK10 2NA.

## 1 Introduction

This paper describes a commercially available tool that is designed to facilitate the discovery of knowledge hidden in databases. The potential of the tool for scientific applications is illustrated via a case-study. As far as the authors are aware this is both the first application of the particular tool to a scientific domain[1] and the first project to describe the application of data mining techniques to the domain of the case-study (see Section 3).

## 2 DataMariner, a Tool for Data Mining

The tool that was used in this project is called DataMariner (Release 2.3.1) [Logica, 1993] [Nabney & Grasl, 1991] which is a data mining package developed by Logica. It incorporates both cluster analysis and rule induction algorithms. Cluster analysis can be used to discover new classes of related data. Once such classes are identified, rule induction can be used to generate rules for membership of these classes. The classes are disjunctive and non-hierarchical.

DataMariner induces rules with the following syntax.
classname_rule_no
IF clause_1 clause_2 ... THEN conclusion_1 (probability_1) conclusion_2 (probability_2) ...

The rule consequent is an implicit disjunction of clauses, where each clause is a conclusion about class membership and has a probability associated with it. The rule antecedent is an implicit conjunction of clauses. Each one of these clauses can only involve one attribute. Thus rules in which there is a disjunction involving two or more attributes are not allowed. A clause of the rule antecedent can specify the value(s) of an attribute as one of the following:–

**discrete value** (eg. `colour = red`)

**disjunction of discrete values** (eg. `colour = red OR blue`)

**negation of a discrete value** (eg. `colour != red`)

**closed numerical range** (eg. `temperature >= 18 AND <= 65`)

**open numerical range** (eg. `temperature >= 21`)

DataMariner comprises a number of tools. A description of each is given below.

**Classify** This uses cluster analysis to find classes for examples on the basis of the attribute values contained in the example set.

[1]The developers of the tool stated this to be the case too.

**New Attributes** This can create new attributes that provide good discrimination between classes by combining numerical attributes.

**Simplify** This uses cluster analysis on the attributes to reduce their number whilst losing as little information as possible.

**Merge** This can be used to merge values of attributes. Three types of merging are possible:–

1. a number of values of a discrete attribute can be merged into another discrete value.
2. the values of a discrete attribute can be mapped onto numeric values thus making a numeric attribute.
3. the values of a numeric attribute can be mapped onto a number of discrete values thus making a discrete attribute.

**Divide** This can be used for two purposes.

Firstly it can be used to divide the data between two files, one to be used for training, that is to be used by DataMariner to induce rules, and the other for testing, that is to be used by DataMariner to evaluate the rules induced. Divide uses a pseudo-random number generator to ensure that the examples selected for the training file are taken randomly. Divide can be instructed to put a specified proportion of examples in the training file.

Secondly Divide can be used to split the data into several training and test files so that cross-validation can be performed. Cross-validation is described in Section 2.1

**Induce** This produces a set of rules describing each class in turn, where the classes are sorted by the number of examples belonging to each class in descending order. The induction process continues for each class until all the examples that belong to that class are covered by the induced rules. Induce uses an algorithm[2] developed from the PRISM algorithm [Cendrowska, 1987].

The order of the induced rules describing each class is important. Once the first rule has been induced for a class, then all the examples which are covered by that rule are ignored when inducing the next rule. Thus an example obeys a second induced rule only if it does *not* obey the first rule and does obey the second rule.

**Prune** This can be used to prune rules. It examines each clause in each rule, starting with the last clause in a rule, to test whether a clause significantly improves the proportion of examples correctly allocated to the class. If a clause fails this test then it is removed and the preceding clause is tested. If it does not fail then the preceding rule is tested. If all the clauses of a rule are found to make an insignificant contribution then the whole rule is removed.

Prune uses the Fisher one-tailed statistic to decide whether a clause significantly improves the proportion of examples correctly allocated to the class; no domain knowledge is used to support its actions.

The level of pruning can be controlled using a parameter known as the prune-level. The level can be regarded as a filter, where a high figure implies that more should be retained. Pruning with the prune-level set to 0% would remove all of the rules. Pruning with the prune-level set to 100% would not remove any clauses or rules, although this would remove redundant conditions.

[2]Details of the specific algorithm used by Induce are not given because they could not be released by Logica.

**Evaluate** The rules induced by DataMariner can be tested using Evaluate. Evaluate uses the induced rule-set to classify some examples and compares the results with the actual classifications, that is those classifications which are known before the rules are induced.

Evaluate assigns each example to the class with the highest probability associated with it amongst all the rules that fire.

Evaluate generates a variety of information that guides the data analyst in identifying any problems or omissions in the rules. This information may include for example suggestions on how the values of attributes could be merged.

The data used by DataMariner has to be in a flat ASCII text file. This file, known as the example file, must be in a rectangular format. Each example must have a single value for each of a number of attributes; the same attributes must be used for each example. One, and only one, of the attributes is used as a classifier by Induce.

The way in which DataMariner interprets the data given to it can be controlled in a number of ways. Some examples of these are described below. DataMariner can be instructed to:–

- ignore one or more attributes, and their values.
- treat one or more discrete attributes as ordinal types and prevent the generation of disjunctive clauses containing non-contiguous values of these attributes. DataMariner treats a discrete variable as nominal unless it is given this instruction.
- use a specified attribute as the classifier.
- only generate rules for a number of specified classes.

## 2.1 Cross-Validation

Cross-validation allows the accuracy of a rule-set to be estimated when all the available examples have been used to induce that rule-set. This section explains when and why cross-validation should be used, how it is performed and explains and justifies the statistical formulae used to determine the result of a cross-validation.

If the number of examples in the example-file is not small then the examples can be divided between those to be used by Induce for training and those to be used by Evaluate for testing. However when the number of examples is small all the examples have to be used for training to ensure that the accuracy of the induced rules will be acceptable; consequently none of the examples can be used exclusively for testing. Cross-validation overcomes this problem; it is performed as follows:–

1. The set of examples is divided into $x$ equal and disjoint segments.
2. $x$ partitions are formed; for each partition one segment is used as the test set and the remainder of the examples are used for training. A different segment is used as the test set for each partition. Thus every example appears in one test set and $(x-1)$ training sets.
3. $x$ rule-sets, one for each partition, are induced and tested.
4. An estimate of the accuracy of the original rule-set is obtained by calculating the mean and standard error of the accuracies for the rule-sets induced during cross-validation. The formulae used to calculate these statistics are explained and justified in the next section.

### 2.1.1 Statistics used in Cross-validation

An experiment that comprises a fixed number of independent trials each with two possible outcomes, success and failure, and the same probability of success is known as a binomial experiment. The probability of a given number of successes is described by the binomial distribution.

The tests performed during cross-validation are binomial experiments: the evaluation of each partition can be considered to be a number of independent trials in which the rule-set induced either succeeds or fails to classify each example correctly. The number of trials is the same for all the partitions and is equal to the number of examples in the file.

A theorem of mathematical statistics states that if $n$ trials in a binomial distribution are observed, the number of successes obtained will have a theoretical population mean of $n\pi$ and a standard error[3] of $\sqrt{n\pi(1-\pi)}$, where $\pi$ is the probability of success. In other words, if $n$ trials are repeatedly observed the distribution of the number of successes would be characterised by that mean and standard deviation. [Pollard, 1972]

Since DIVIDE splits the example file into disjoint segments of equal size, and since only one accuracy figure is calculated for each of these, the probability of success, $\pi$, is taken as the mean of the accuracies calculated by EVALUATE for each partition divided by 100.

$$\pi = \sum_{i=1}^{t} a_i \frac{1}{100t}$$

where $t$ is the number of partitions and $a$ is an accuracy expressed as a percentage.

Given that the number of successes has a theoretical population mean of $n\pi$ and a standard error of $\sqrt{n\pi(1-\pi)}$ the percentage of successes, $p$, has the following theoretical population mean, $\mu_p$, and standard error, $\sigma_p$.

$$\mu_p = n\pi\frac{100}{n} = 100\pi$$

$$\boxed{\mu_p = n\pi\tfrac{100}{n} = 100\pi}$$

$$\sigma_p = \sqrt{\left(\frac{100}{n}\right)^2 n\pi(1-\pi)} = \sqrt{\frac{100\pi(100-100\pi)}{n}} = \sqrt{\frac{\mu_p(100-\mu_p)}{n}}$$

# 3 A Case-Study

The database used in the case-study contained data that had been extracted from reports in the chemical literature describing analytical separations of a particular type of chemical mixture, known as an enantiomer pair, using a special type of equipment referred to as Pirkle-type chiral stationary phases (CSPs) [Taylor & Maher, 1992]. Such separations are performed in many disciplines, such as agrochemistry, medicine and pharmacology, but few guidelines exist on the choice of CSP and they are difficult to access. Consequently a rule-set that recommended a CSP chiral selector for a given enantiomer pair could save time and money. Thus an attempt was made to develop such a rule-set using DATAMARINER.

[3] A standard error is an estimated standard deviation of a parameter, the value of which is not known exactly.

### 3.1 How DataMariner was Used

DataMariner was used in an interactive manner. A series of experiments was performed on the database using seven of the eight tools; each experiment was designed after considering the results of previous experiments. Domain knowledge was used when interpreting results and deciding what experiments should be performed. Knowledge of the domain was used to select the attribute to be used as the classifier, to select which attributes should be ignored by DataMariner, to justify the ordinal types that were used and to confirm that the merges suggested by DataMariner made chemical sense.

The way in which DataMariner interpreted the data was successfully controlled: values were ignored, merged and ordered in various ways as required. Rules were successfully induced using Induce from the data in an example-file which itself had been created from data in the INGRES database. The rule-sets induced were evaluated using Divide and Evaluate. All of the tools of DataMariner proved to be useful for this domain except for the following: Classify and New Attributes were not needed, Simplify did not produce any results that made chemical sense, and Prune did not appear to have a beneficial affect and may have had a harmful one.

### 3.2 Results Obtained Using DataMariner

Rules were induced that recommended particular CSP chiral selectors based on the structural features of an enantiomer pair. Although it was not easy to provide a chemical justification for the rules induced by looking at the rules themselves the cross-validation performed on the optimal rule-set induced suggested that this rule-set would recommend as its first choice a correct CSP chiral selector for 63.1% $\pm$3.0% of enantiomer-pairs that can be separated on Pirkle-type CSPs. This was more than ten times greater than the accuracy that would have resulted from choosing one of the selectors at random.

## 4 Conclusions

DataMariner was successfully used to develop and validate rules for the domain of the case-study. Although it is not easy to provide a justification for the rules by looking at them the results suggest that they have a high degree of accuracy.

The funding was provided by EPSRC, under the remit of the Total Technology programme, and by Zeneca Pharmaceuticals. R. Dallaway and I.T.Nabney of Logica Cambridge Ltd. provided helpful advice on the use of DataMariner. C. Bryant is greatful for this and the hospitality shown to him by all at Logica Cambridge Ltd.

## References

[Cendrowska, 1987] Cendrowska, J. (1987) PRISM: An Algorithm for Inducing Modular Rules. *International Journal of Man-Machine Studies*, 27, pp349-370

[Logica, 1993] Logica UK Limited (1993) DataMariner. User Manual. Version B.

[Taylor & Maher, 1992] Taylor, D.R.; Maher, K. (1992) Chiral Separations by High-Performance Liquid Chromatography. *Journal of Chromatographic Science*, 30, pp67-85

[Nabney & Grasl, 1991] Nabney, I.; Grasl, O (1991) Rule Induction for Data Exploration. *Proceedings of Avignon 91: Expert systems and their applications*, 1, pp329-341.

[Pollard, 1972] Pollard, A.H. (1972) Introductory Statistics: A Service Course, Pergamon Press, Australia.