

Machine learning from agricultural databases: practice and experience

Stephen R. Garner, Geoffrey Holmes, Robert J. McQueen and Ian H. Witten

Department of Computer Science
University of Waikato
Hamilton, New Zealand
srg1@cs.waikato.ac.nz

ABSTRACT

Per capita, New Zealand is reputed to be one of the world's leading collectors of information in databases. The country, quite rightly, places a substantial investment in stored knowledge. As databases grow, however, interest tends to shift from techniques for retrieving individual items to large-scale analysis of the data as a whole. Data can be analysed to view trends, pick out anomalies, discover relationships, check that policy is turned into practice, and so on. With large databases, such analyses can be costly.

At Waikato University we are examining these issues in the context of the agricultural sector of the economy. Our project centers around techniques of "data mining" or "machine learning," which automatically analyse large bodies of data to discover hidden dependencies. Other methods of exploratory data analysis—many of them interactive—are being investigated as an adjunct to the machine learning algorithms. This paper reviews the software that has been developed to support the application of machine learning techniques, and comments on our success with particular agricultural databases. We discuss those aspects of the databases that hinder the analysis, to assist with future data collection activities.

1. Introduction

More and more agricultural data—both in New Zealand and overseas—is being collected and stored in databases. As the volume increases, the gap between generating and collecting the data and actually being able to understand it is widening. In order to bridge this knowledge gap a variety of techniques known as "data mining" or "knowledge discovery in databases" (KDD) are being developed. KDD can be defined as the extraction of implicit, previously unknown, and potentially useful information from data (Piatetsky-Shapiro and Frawley, 1991), and can be built upon a variety of technologies of which machine learning is one of the most important. Typical characteristics of knowledge discovery systems include representing the knowledge in a high-level language, some measure of the accuracy of the knowledge, extracting knowledge that is interesting, with predictable and acceptable run-time efficiency.

Machine learning is a fairly new technology that can facilitate discovery of rules and patterns in sets of data. Typical uses include identifying clusters in data, and inferring sets of rules that describe each category or class in a data set. A wide variety of different machine learning systems are available. Generally, each implementation only accepts data in its own individual format, and has its own ways of specifying parameters and output. The machine learning group at the University of Waikato has developed a software system called WEKA (for Waikato Environment for Knowledge Analysis) that is designed to unify a variety of machine learning schemes within a common interface for database format, scheme configuration, and output handling. This paper describes the WEKA system, its continuing use with several different agricultural databases, and issues that have arisen from its application.

2. The WEKA workbench

WEKA is not so much a single program as a collection of interdependent modules bound together by a common user interface. The modules fall into three categories: data set processing, machine learning schemes, and output processing. The first category involves extracting information about a data set for the user, splitting data into test and training sets, filtering out features in the data not required by the user, generating new features based on domain knowledge, and translating the information into a form suitable for a machine learning scheme to work with. Machine learning schemes implement standard learning algorithms: they take a data set and produce some output, generally in the form of a set of rules. Output modules process the output from a machine learning scheme, perhaps evaluating a rule set against a test file or displaying a decision tree graphically in a window.

WEKA differs from other machine learning environments in that its target user is a domain expert—in our case an agricultural scientist—who wants to apply machine learning to real-world data and is more interested in the knowledge extracted than the method used to produce it. Other systems, such as the MLC++ project at Stanford University (Kohavi *et al.*, 1994) and the European Machine Learning Toolbox (Kodratoff *et al.*, 1992), are intended for use by machine learning researchers and programmers developing and evaluating machine learning schemes, while the Emerald system (Kaufman *et al.*, 1993) is designed purely as an educational tool.

Although its target audience is different, WEKA is flexible enough to provide a basis for machine learning research, and it has been deployed successfully in undergraduate courses on machine learning. It is not a multi-paradigm learner, that is, it does not combine machine learning techniques to produce new hybrid schemes. Instead, it concentrates on simplifying access to standard schemes so that their performance can be evaluated and compared.

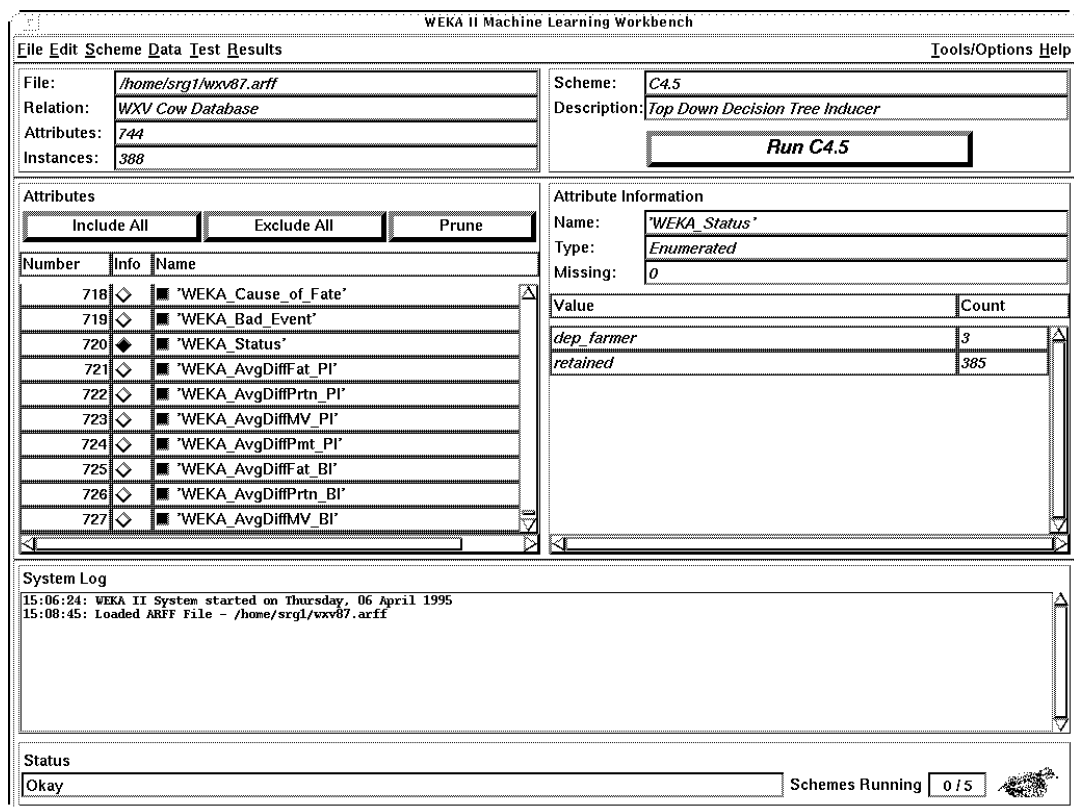


Figure 1. Main screen of the WEKA workbench

The WEKA workbench, illustrated in Figure 1, is written using the TCL/TK scripting language (Ousterhout, 1994) and X-window tool kit, and runs on Sun UNIX systems under Solaris 2. The individual modules are mostly written in C or in pattern-matching languages such as AWK. They can be invoked outside WEKA if desired, for example in a shell script for multiple data set feature filtering. The machine learning schemes are written in a variety of programming languages. Currently, WEKA incorporates schemes written in C, C++ and LISP. A simple rule evaluator has been implemented in C, while a more complete one, PREval, has been developed in PROLOG.

The following sections describe the common data set format, the machine learning schemes that have been incorporated, the rule format, and the rule evaluator.

2.1 Data set format

WEKA stores data in a common file format called ARFF (attribute-relation file format), presenting users with a consistent view of the data regardless of the machine learning scheme being used. ARFF defines a data set in terms of a relation (i.e. table) made up of attributes (i.e. columns) of data. Information about the names of the relation and the types of the attributes are stored in a header, with the examples (i.e. records or instances) being represented as rows of data in the body of the file. This format provides the basic two-dimensional data set that machine learning schemes typically require, where each row in the table represents a example described by a vector of attribute-value pairs.

Tools for processing ARFF files perform such tasks as splitting data into test and training sets, filtering out attributes, providing summary information such as the frequency of unspecified or “missing” values, and converting ARFF files to the input format assumed by each machine learning scheme.

The Machine Learning Database Repository at the University of California, Irvine, contains a large number of data sets, most of which have been converted into ARFF so that WEKA users can test machine learning schemes on standard data (Murphy and Aha, 1994).

2.2 Machine learning schemes

WEKA incorporates machine learning schemes for both supervised and unsupervised learning. The former is when a desired class is assigned to each example in the data set, and the aim is to induce rules that classify unseen examples. The latter is when there is no desired class, and the aim is to induce rules that split the examples into natural groupings.

The unsupervised learning schemes in WEKA include CLASSWEB (Gennari, 1989), a variant of the COBWEB (Fisher, 1987) conceptual clustering system, and AUTOCLASS (Cheeseman *et al.*, 1988), a Bayesian classifier. These schemes aim to cluster the data sets into natural groupings. CLASSWEB produces a hierarchical description called the “concept hierarchy,” with the most general concept or class at the top and the most specific ones at the bottom. AUTOCLASS induces a number of classes and assigns each instance a probability of membership in them.

C4.5 (Quinlan, 1993), FOIL (Quinlan, 1990; Quinlan and Cameron-Jones, 1993), INDUCT (Gaines, 1991), IBL (Aha, 1991), κ^* (Cleary and Trigg, 1995) and 1R (Holte, 1993) make up the set of supervised learning schemes. They all work with data in which a classification has already been defined for each instance: this classification may have been assigned by a human expert or by an unsupervised scheme such as AUTOCLASS. In the case of C4.5 the output is a decision tree in which internal nodes represent tests on an attribute and leaf nodes specify the classification of the examples that reach that node. INDUCT, FOIL and 1R produce rules that describe a classification

based on combinations of attribute tests. K^* and IBL are instance-based learning schemes that use similarity measures (Kolmogorov complexity in K^* , Euclidean distance in IBL) to match a new instance to the most similar ones already seen.

2.3 Output processing

All output from a machine learning scheme is passed back to WEKA in the form of text, and can be displayed in a scrollable viewer. Text can be copied to other applications, printed, or saved to a file. A novel feature of the workbench is that it supplies a facility for “external” evaluation that can perform tests on the output from any machine learning scheme in a uniform way. If the user selects external evaluation, the output, as well as being displayed as text, is converted into an internal WEKA rule format and evaluated. The rule format is PROLOG-based, and rules can be executed using an evaluator called PREval. The precise details of output translation vary for the individual schemes. For FOIL, which produces its output as rules anyway, it is a simple matter to convert the rule format to PREval. INDUCT, which produces rules in a special “ripple-down” structure, provides an option to produce its output directly in the PREval format. For C4.5, both decision trees and rules have to be converted into PREval rules.

PREval takes a set of rules and an ARFF file, and evaluates how well the rules cover the classifications. It provides figures for classification accuracy, including the percentage correctly classified, incorrectly classified, classified by multiple rules, and not classified at all, as well as confusion matrices to show the distribution of misclassifications, and statistics on how well each rule does. PREval also incorporates an entropy measure for calculating both the complexity of rule sets, and the complexity of data sets with respect to a rule set.

3. Agricultural Applications of WEKA

We examine four projects in which WEKA has been used to assist data analysis: a search for rules describing culling decisions in dairy herds, the isolation of factors governing apple bruising, the detection of cows “in heat” based on data collected during milking, and the analysis of a survey of microcomputer use in dairy farming. A more detailed account of the cow culling is given first, followed by brief discussions of the other studies.

3.1 Cow culling study

The Livestock Improvement Corporation in Hamilton operates a relational database system to track genetic history and production records of twelve million dairy cows and sires, of which three million are currently alive. Production data are recorded for each cow from four to twelve times per year, and additional information is recorded as events occur. Farmers receive information from the Livestock Improvement Corporation in the form of reports from which comparisons within the herd can be made. Two types of information that are produced are the *production* and *breeding indexes* (PI and BI respectively), which both indicate the merit of the animal. The former reflects the quality of the milk produced, with respect to measures such as fat, protein and volume, and indicates its worth as a production animal. The latter reflects the likely merit of a cow’s progeny, indicating its worth as a breeding animal. In a well-managed herd, the average value of these indexes will increase every year, as superior animals enter the herd and low-performance ones are removed.

One major decision that farmers must make each year is whether to retain a cow in the herd or remove it, usually to an abattoir. About 20% of the cows in a typical New Zealand dairy herd are culled each year, usually near the end of the milking season as feed reserves run short. The cows’ breeding and production indexes influence this

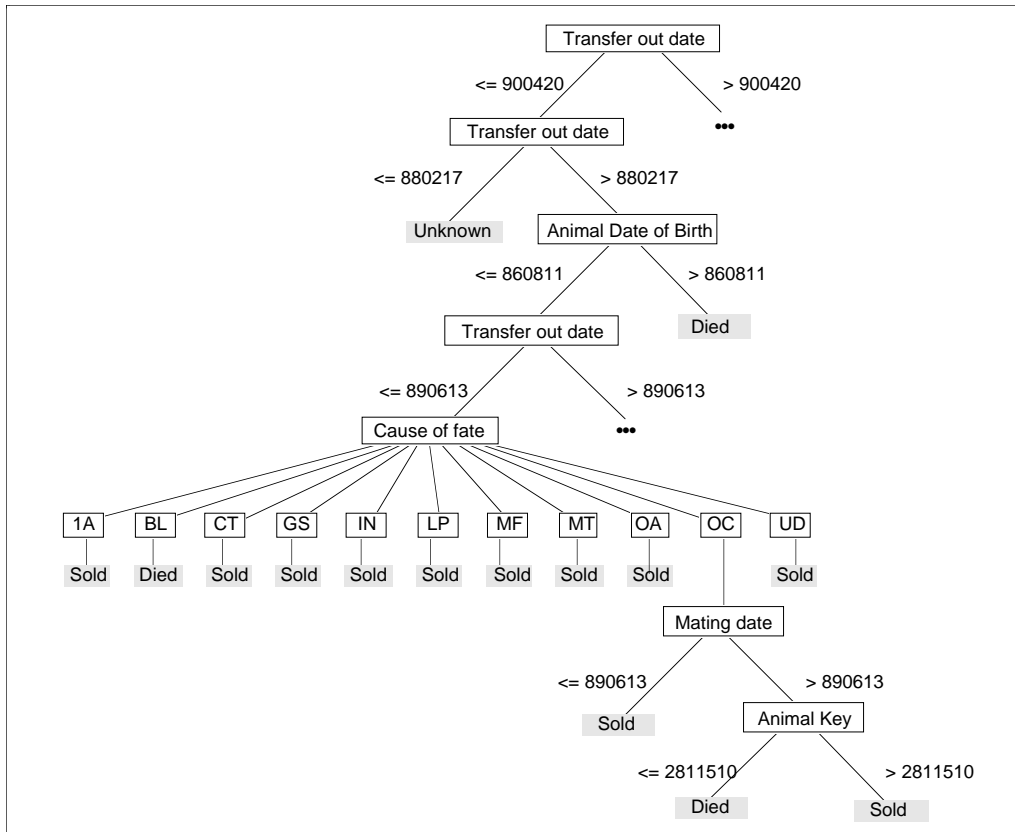


Figure 2. Decision tree induced from raw herd data

decision, particularly when compared with the other animals in the herd. Other factors that may influence the decision are:

- age: a cow is nearing the end of its productive life at 8–10 years;
- health problems;
- history of difficult calving;
- undesirable temperament traits (kicking, jumping fences);
- not being in calf for the following season.

The Livestock Improvement Corporation hoped that machine learning tools would provide insight into the rules that particular farmers actually use to make their culling decisions, enabling better information to be provided to farmers in the future. They provided data from ten herds over six years, representing 19 000 records each containing 705 attributes.

The principal tools used for analysis were C4.5 and FOIL. The initial raw data set was run through C4.5 on the workbench. Cows were classified on their *fate code* attribute, which can take the values *sold*, *dead*, *lost* and *unknown*. The resulting tree, shown in Figure 2, proved disappointing.

At its root is the *transfer out date* attribute. This implies that the culling decision for a particular cow is based mainly on the date on which it is culled, rather than on any attributes of the cow. Next, the *date of birth* is used, but as the culling decisions take place in different years, an absolute date is not meaningful. The cow's age would be useful, but is not explicitly present in the data set. The *cause of fate* attribute is strongly associated with the *fate code*; it contains a coded explanation of the reason for culling. This attribute is assigned a value *after* the culling decision is made, so it is not available to the farmer who makes the culling decision. Furthermore, we would

like to be able to predict this attribute—in particular the *low production* value—rather than include it in the tree as a decision indicator. Its presence artificially elevated the classification accuracy, predicting the culling decision correctly 95% of the time on test data. *Mating date* is another absolute date attribute, and animal key is simply a 7-digit identifier.

In discussions with staff from the Livestock Improvement Corporation, it was suggested that the culling decision may not be based on a cow's absolute performance, but on its performance relative to the rest of the herd. To test this, attributes representing the difference in production from the average production over the cow's herd were added to the database. In order not to bias the learning process, the original attributes were retained in the data set, and the new attributes were not distinguished in any way—it was left to the machine learning schemes to decide if they were more helpful for classification than the original attributes. This process involved close cooperation with Livestock Improvement Corporation. Discussions often ended up proposing more derived attributes, and clarifying the meaning of particular attributes. Staff were also able to evaluate the plausibility of rules, which was helpful in the early stages when recreating existing knowledge was a useful indication of the correctness of our approach.

After normalizing the data and adding derived attributes, C4.5 produced the tree in Figure 3. The *fate code*, *cause of fate* and *transfer out date* have been combined into a *status code* which takes the values *culled* or *retained*. For each year, the records for cows that have previously been culled or have not yet been born are removed. Cows that were alive and were not transferred in that year are marked as *retained*, otherwise they are marked *culled*. If, however, a cow died of disease or some other factor outside the farmer's control, the record is removed. After all, the aim of this exercise is to discover the farmer's culling rules rather than the incidence of disease and injury.

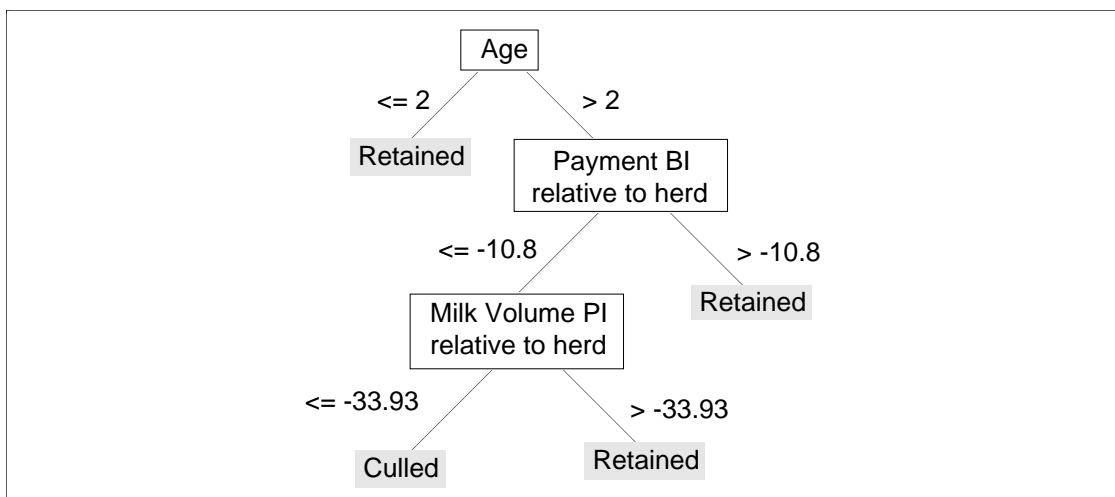


Figure 3. Cow culling decision tree

The tree in Figure 3 is much more compact than the one shown in Figure 2. It was produced with 30% of the instances, and correctly classifies 95% of the remaining ones. The unconditional retention of cows two years or younger is due to the fact that they have not yet begun lactation, and so there is no indication of their productive potential. The next decision is based on the cow's worth as a breeding animal, which is calculated from the earnings of its offspring. The volume of milk that the cow produces is used for the final decision. This tree's decisions are plausible from a

farming perspective, and its compactness indicates that it is a good explanation of the culling process. It is interesting to note that it consists entirely of derived attributes, further emphasising the importance of the preprocessing step.

3.2 Apple bruising

Several data sets were supplied by Lincoln Technology in order to investigate factors involved in the bruising of apples during storage. Collected over a period of five years, the data amounted to over 17 000 bruise measurements. It included information such as the variety of apple, the temperature at which it was stored, and its diameter and mass. Experiments had been carried out which involved dropping the apple from different heights on to each side (an apple was assumed to have six sides), recording the energy of the impact and the depth and diameter of each bruise, from which the bruise volume was calculated.

To analyse this data it was decided to classify the apples based on bruise volume, and a new attribute was created with values “tiny,” “small,” “medium” and “large.” Each instance was assigned to one of those classes, and the data set was converted to ARFF format and processed using WEKA.

The most obvious result, and the one we expected, was that the greater the impact energy, the larger the bruise. Another result was that apples stored in a cool environment tend to bruise more readily than ones stored at room temperature. This demonstrates one of the uses of WEKA for scientists, in that having “discovered” these hypotheses the data can be analysed with a statistical package to prove or deny the hypotheses.

3.3 Cows in heat

The Dairying Research Corporation has developed an advanced cow milking system that is used to gather basic production and health-related data about each cow, at every milking. This system allows a cow to be tracked across the whole of a milking season with the aim of reporting significant events to the farmer as they occur. These events include the detection of mastitis, faults in the milking machinery and whether a cow is “in heat.” The latter is important because most dairy herd reproduction in New Zealand is carried out by artificial insemination and the farmer needs to know as soon as possible when the cow is at her most fertile, to improve the chances of a successful insemination.

The detection of whether a cow was in heat was chosen for the initial analysis, and a data set of about 38 000 records made up from a 1993 data set of 140 cows with two milkings a day over 137 days. The initial work was carried out using the similarity based learning schemes C4.5 and FOIL after the time series data had been converted into a form suitable for WEKA. The results of these analyses were not promising due to the skewed nature of the data (about 98% of the examples were negative.) This problem, known as the “small disjunct” problem, will be discussed further in Section 4, and analysis of this data is now being pursued using hidden Markov modelling techniques (Rabiner, 1991) similar to those used in speech recognition.

3.4 Microcomputers in farming

This project analysed the results of a survey of computer use by dairy farmers in the Waikato region in an effort to identify interesting and useful features. The data recorded the results of 308 interviews. The project sought those characteristics that determine whether or not a farmer uses a microcomputer.

Having been translated into ARFF with each instance being assigned to one of the classes “user” or “non user,” the data was processed by several machine learning schemes.

The survey noted whether a farmer owned a computer system, as well as whether it was used for the farm. WEKA extracted the following pieces of interesting knowledge.

- If the farmer’s spouse and children use a computer, the farmer probably does too.
- If the farmer has a tertiary education, he or she will probably be a computer user.
- Farmers who do the farm accounts by themselves, and are relatively new farmers on small farms, are probably computer users.

4. Issues raised through the analysis of databases

When using the WEKA workbench for knowledge discovery, the vast bulk of the work is in data preparation and preprocessing. This section will discuss issues and caveats that have arisen from the previously mentioned projects. First we discuss the preparation of the database, and then the application of the machine learning technology.

4.1 Preparation of the database

The first step in applying knowledge discovery techniques is to massage the data into a form suitable to be processed by the automated tools. In the case of the WEKA system, the data must be extracted and translated into ARFF format. Typically this involves taking the physical extract of some database and then processing the data through a series of steps to transform it into an ARFF data set.

A common form of data extraction is the retrieval of data from a relational database using some form of query language (e.g. SQL). This will often involve taking data from a variety of sources or tables in the database, and combining them into a single table. This process, called denormalization, is the reverse of the process that was used to structure the database in the first place in order to impose integrity constraints, reduce update anomalies and eliminate data redundancy. The problems arising are two-fold.

First, the data set that is extracted can be very large, in terms of both the number of columns or attributes in the data set and the number of rows or examples. The machine learning software and the computer system it is running on both must have sufficient resources to cope with the data. The second problem is that denormalization may reintroduce relationships, such as functional dependencies, between attributes that were the reason for structuring the database in the first place. If these relationships are identified as patterns by the knowledge discovery system, they will not be considered “interesting” by the user.

Most machine learning tools only work with a single relation and will introduce these two problems, though there are newer techniques that use first-order logic (e.g. FOIL) and work with multiple relations.

Once the database is in a single relation, each attribute must be examined in order to determine its data type—for example whether an attribute’s contents are numeric or symbolic values. Numeric values may include measurements, such as the diameter of an apple, while symbolic values could be a day of the week or whether a cow has had a calf this year. This information may be ascertained from the original database’s data dictionary (if there is one), but there are a variety of possible pitfalls.

Databases that have been designed for custom software may contain certain numeric values that are actually not to be treated as numbers but rather as codes describing a particular condition. For example, a field “production index” may have the value -1 to indicate that no measurement was recorded. In such cases, one may replace the -1 with a “missing value” token, or if that is not possible then simply remove those records. Another example is a field in the database that is of type integer but whose contents are not used arithmetically. This may arise in the case of an “identification number” field, in which case certain operations—such as taking the average of the field’s values—are meaningless. Changing this field to one where the numbers are treated as nominal values will eliminate the possibility of the KDD system creating inappropriate rules.

Conversely, symbolic values may have an associated ordering—for example the days of the week—in which case it may be better to restructure the attribute into one that allows a machine learning scheme to produce rules that express situations such as

“working_day \leq Tuesday”

rather than

“working_day is Monday” or “working_day is Tuesday”.

Mapping the symbolic values to integers will allow such operations, though one must be careful not to permit arithmetic or statistical operations to be carried out on the integers.

The user preparing the data set needs to be aware of co-dependent and implied attributes. The former occur when one attribute contains a measurement and another indicates how accurate that measurement is. The second attribute might be a necessary qualification on the first one—for example only use the measurement if its accuracy exceeds 95%. The latter occur when the absence of a value in one attribute implies the existence of another attribute’s value. For example, in the cow culling data set the absence of a value in the “transfer in date 3” field means to check the “transfer in date 2” field to determine when the cow was transferred into the herd.

Missing values in the data set can create a variety of problems. Sometimes missing values merely indicate the absence of information; other times they actually convey information. In the first case all missing values should be replaced by a token recognised by the KDD system. In ARFF, missing values are represented by “?”. The presence of the missing value token means that the KDD system will avoid creating rules for which the absence of a value determines the classification. On the other hand, a missing value may convey information—for example, in an attribute that contains disease information the absence of a value could signal “No disease”. Sometimes, both cases described above occur together in a single attribute, and it may not be possible to distinguish a missing value from a default one.

Some learning schemes, such as hidden Markov models, enable time series data in which each instance is a set of measurements at a particular time to be modelled—such as the “cows in heat” data. Most machine learning schemes assume that each instance is independent of each other, and so the data needs to be restructured into a suitable form. Typically this involves merging a fixed number of records together using a sliding window approach. The main drawbacks are the increased number of attributes and the problem of determining the size of the window. Table 1 shows an original data set, and Table 2 illustrates the converted one.

Time	A	B	Class
1	A ₁	B ₁	Class ₁
2	A ₂	B ₂	Class ₂
3	A ₃	B ₃	Class ₃

Table 1. Original time series data set

A _{n-2}	B _{n-2}	A _{n-1}	B _{n-1}	A _n	B _n	Class _n
...
A ₁	B ₁	A ₂	B ₂	A ₃	B ₃	Class ₃
A ₂	B ₂	A ₃	B ₃	A ₄	B ₄	Class ₄

Table 2. Converted time series data set

New attributes may also need to be created, and existing ones removed. In the cow culling study it was necessary to create about forty new attributes, including a status code showing whether a cow is retained or culled and a variety of production and breeding indexes relative to the herd in a specific year. The most common operations include quantising a numeric attribute into intervals or classes; calculating a new attribute using a formulae; and creating a new class attribute based on a series of logical conditions.

Finally, one should make sure that the data is consistent within each attribute. In the apple bruising data set, new data collection methods meant that the units used to measure each apple changed, and midway through the data set one attribute altered from measuring the drop height to measuring the impact energy. Also, the apples were modelled as being completely spherical, and when apple diameters were calculated from other attributes in the data, diameters ranging from 0.5 mm to over half a meter were encountered!

4.2 Machine learning

Having discussed the conversion process from an existing database into an ARFF file, let us turn to the application of the machine learning technology. As with any data analysis technique, the higher the data quality, the better the model will be. Thus if many attribute values are missing or corrupt, the schemes may not have enough data to build a complete model. Two problems have been encountered while working with agricultural data sets.

The first is that classes may overlap when different classes have very similar attribute values. Table 3 shows the results of evaluating a model describing bruise size created using C4.5 on a data set of 1440 instances. The model misclassified 404 apples in the data set, an error rate of 28.1%. The apples with tiny bruises seem to form a well-defined class, but the distinction between apples with medium bruises and those with larger bruises is not readily predictable using the attributes in the data set.

	<i>Classified as</i>			<i>Actual Class</i>
	<i>small</i>	<i>medium</i>	<i>large</i>	
<i>tiny</i>	11	2		<i>tiny</i>
302	41	59	1	<i>small</i>
10	4	663	26	<i>medium</i>
		291	30	<i>large</i>

Table 3. Classification accuracy matrix for apple bruising

A second problem occurs when one of several classes is much larger than the others. Known as the “small disjunct” problem, this may cause the machine learning scheme to describe the data as a single class and treat the instances in the smaller classes as anomalies or errors. For example, in the data set describing cows in heat, 98% of the examples are not in heat and the remaining 2% are not clearly distinguishable using the attributes in the data set. Thus the schemes come up with the general rule that a cow is not in heat, and that rule is correct 98% of the time! Work is continuing in this area. Hybrid schemes might augment C4.5 with an instance-based learner (Ting, 1994). Moreover, the number of instances of the larger class may be reduced in the training set while maintaining the numbers of instances in the smaller classes.

5. Summary

Here are some pointers for the successful application of knowledge discovery tools to real-world data sets.

First, it is necessary to spend a significant amount of time on preparing the data for analysis. This involves identifying the features in that data that will be needed, extracting the data, and converting it into a format that the KDD tools can analyse.

Second, it is important to get to know the data. This may involve the machine learning researcher becoming an expert on the data, or developing a close relationship with the experts in the domain. Knowledge of the domain will allow—new derived attributes to be developed, judgements to be made as to how to treat missing values and integer codes, and the ability to guide the discovery process and evaluate its outputs.

Third, the discovery process is an iterative one. After taking the data set and analysing it, the results are used to augment the data set with new attributes or to configure a scheme’s parameters. Then the process is repeated. The WEKA workbench is a good tool for this as it provides the user with uniform facilities for manipulating data sets, configuring learning schemes and handling output.

Factors such as computing resources and flexibility in tool selection also play an important role in the KDD process. The more disk space is available the more data sets can be kept online; and the more CPUs the more discovery processes can be run at once. In the case of tool selection, if the tools available do not suit the data set then one should be prepared to seek for new tools to incorporate into the workbench, and to complement the KDD tools with others, such as a statistical package, in order to evaluate and test the output.

The technology involved in extracting knowledge from databases is still relatively new and is continuing to improve. These points may help to successfully apply this technology today.

References

- Aha, D.W., Kibler, D. and Albert, M.K. (1991) "Instance-based learning algorithms." *Machine Learning* 6, pp. 37–66.
- Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W. and Freeman, D. (1988) "AUTOCLASS: A Bayesian classification system." *Proc Int Conf on Machine Learning*, pp 54–64, Ann Arbor, MI: Morgan Kaufmann.
- Cleary, J.G. and Trigg, L. (1995) "K*: an instance-based learner using an entropic distance measure." *Proc 12th Int Conf on Machine Learning*, Morgan Kaufmann.
- Fisher, D. (1987) "Knowledge Acquisition Via Incremental Conceptual Clustering." *Machine Learning*, 2, pp. 139–172.
- Gaines, B.R. (1991) "The tradeoff between knowledge and data in knowledge acquisition in knowledge discovery in databases," *AAAI Press*, pp 491–505.
- Gennari, J.H. (1989) "A survey of clustering methods," *Technical Report 89-38*, Irvine: University of California, Dept. of Information and Computer Science.
- Holte, R.C. (1993) "Very simple classification rules perform well on most commonly-used datasets." *Machine Learning* 11, pp. 63–91.
- Kaufman, K.A. and Michalski, R.S. (1993) "EMERALD 2: An Integrated System of Machine Learning and Discovery Programs to Support Education and Experimental Research," Technical Report MLI 93-10, George Mason University.
- Kodratoff, Y., Sleeman, D., Uszynski, M., Causse, K., and Craw, S. (1992) "Building a machine learning toolbox," In: Steels, L. and Lepape, B. (Editors), *Enhancing the Knowledge Engineering Process*, Elsevier Science Publishers B.V.: pp 81-108.
- Kohavi, R., John, G., Long, R., Manley, D. and Pflieger, K. (1994) "MLC++: A Machine Learning Library in C++," *Tech Report*, Computer Science Dept, Stanford University.
- Murphy, P.M. and Aha, D.W. (1994) UCI repository of machine learning databases. For information contact ml-repository@ics.uci.edu
- Ousterhout, J.K. (1994) *Tcl and Tk toolkit*. Addison Wesley.
- Piatetsky-Shapiro, G. and Frawley, W.J. (Editors) (1991) *Knowledge discovery in databases*. AAAI Press, Menlo Park, CA.
- Quinlan, J.R. (1990) "Learning logical definitions from relations." *Machine Learning* 5, pp 239–266.
- Quinlan, J.R. (1992) *C4.5: Programs for Machine Learning*. Morgan Kaufmann.
- Quinlan, J.R. and Cameron-Jones, R.M. (1993) "FOIL: a midterm report," *Proc European Conf on Machine Learning*, pp 3–20. Springer Verlag.
- Rabiner, L.R. (1990) "A tutorial on hidden Markov models and selected applications in speech recognition," In: Waibel, A. and Lee, Kai-Fu. (Editors), *Readings in Speech Recognition*, Morgan Kaufmann Publishers, Inc., San Mateo, California: pp 267–296.
- Ting, K.M. (1994) "The problem of small disjuncts: its remedy in decision trees." In: Elio, R. (Editor), *Proc. Tenth Canadian Conference on Artificial Intelligence*; pp 91–97. Canadian Society for Computational Studies of Intelligence.