# Knowledge-rich induction of classification rules

**Brent Martin**
Department of Computer Science, University of Waikato, Hamilton, New Zealand
Email: bim@waikato.ac.nz

## Introduction

The purpose of this research was to produce a machine learning system that can take advantage of many forms of background knowledge to guide the induction of classification rules. This system will be used for knowledge discovery in databases (also known as "database mining"), as the use of background knowledge can considerably reduce the search space of a database knowledge search. A new system, MARVIN++, is introduced, that attempts to satisfy this aim.

The following sections describe some of the problems encountered while database mining and suggest forms of background knowledge that can help to overcome them.

## Irrelevant data

For most machine learning algorithms that produce classification rules based on attribute or attribute-value testing the size of the search space will be proportional to (among other things) the number of attributes available, and the number of examples in the data set. Any irrelevant attributes or examples provided will therefore increase the size of the search space unnecessarily, and so the time taken to perform the search is needlessly increased. Determining which examples constitute a critical set is very difficult. It is usually possible, however, to make intelligent decisions about the relevance of attributes. In addition to knowing which attributes are irrelevant to the current search, some idea of the level of importance of the remaining attributes may be known. Attributes can then be biased by weighting each attribute so that unimportant attributes are not considered, and the search space is thus reduced.

## Noise

The quality of data collected for classification may vary over, say, time or location. If it is known that a subset of the data is less reliable than average, then those tuples can be weighted accordingly so that they have less impact on the final rules. This helps to minimise the chance of noise appearing in the rules, while not totally eliminating any genuine patterns that may be strong in this subset of data. Additionally only some of the attributes may be less reliable than usual for these examples, so instead of weighting the whole tuple, each attribute value can be given a weighting, indicating that "for tuple m the value of attribute n is not very reliable". This method gives very precise information about the quality of the data.

## Non-ideal attributes

In typical machine learning problems the attributes presented to the algorithm have been carefully chosen to represent the data in the clearest, most informative way. If, for example, it is known that a person's country of birth is important to the classification, but not the street number, then it is the country that is recorded. In an existing database one may not have this luxury. In addition, simple input differences may obscure the data further (eg. ALLIN1, ALL-IN-1, ALLINONE, ALL-IN-ONE, etc are all examples of the same object). By presenting known generalisations for a given attribute as an attribute-value hierarchy, the search algorithm may then climb this hierarchy as necessary to test whether a more general representation of the attribute value may result in a better classification. The search algorithm begins by considering only the raw attributes. If the rules produced are too specialised (ie. too many rules result), then each attribute may in turn be

generalised by climbing its attribute hierarchy. The new value (eg. “science”) is then substituted for each of the attribute values that generalise to this value (eg. “physics”, “maths”, “chemistry” and “computer science”), and the search for rules is repeated. In this way, a data set containing only actual class names may result in a rule based on the school of study, which has been implicitly derived from the data set.

Humans often detect patterns in problems even when they are not looking for those patterns. If for example, we are classifying the cost of motor vehicles by (amongst other things) brand name, we might discover that the formula is similar for cars of brands Jaguar, Aston Martin and MG. The regularity noted for these brands would seem to indicate some sort of significance in this grouping. This idea can be applied to machine learning.

Discovering important functions of attributes involves two basic operations: grouping and conjunction. Grouping of attribute-values in a decision tree is applied when more than one attribute value for a particular attribute test has the same subtree, suggesting that there may be some significance to this group of values. By grouping these attribute-values together into a single attribute-value test the size and complexity of the tree is reduced without sacrificing accuracy. For non-numeric fields, this grouping suggests some underlying commonality in these attribute values of which we were not previously aware (ie. it was not explicitly recorded in the data). For numeric fields, grouping attribute values may produce ranges of values. Grouping may thus be used to find “natural” ranges in the data, overcoming the problems of artificially ranging the data, which can obscure knowledge when concepts span more than one range through incorrect selection of the range boundaries.

Another type of repetition that may occur in a tree is more than one occurrence of a particular subtree, where the root of the subtree may be any attribute-value in the entire tree. By joining together the rules leading to this subtree, however, the repetition may be avoided, and the size of the tree reduced substantially as a result. The discovery and use of functions is a post-processing step. After performing a search and inducing a decision tree (or rule set), the output from the search is analysed for redundancy. Any redundant subtrees (or rule subsets) are then used to induce new functions, which are then applied to the current search result. The resulting functions are also stored for later use.

**Weak data sets**

Most classification problems assume that each example falls into a single unique class. When searching for rules in databases, however, the data set may be incomplete, and so an example may fall into more than one class. If a particular example falls into multiple classes, then one of two approaches is usually taken; either all candidate classes are presented or the most likely class is chosen. The most likely class is usually decided based on either which of the candidate classes contains the most occurrences of the example, or (if the number of occurrences in each candidate class is the same) the most common overall of the candidate classes. Presenting all classes can be confusing if an example belongs to a large number of classes. Additionally the result from the classification may necessarily need to be a single value (eg. to be used to fill in some other database field). Selecting the most popular class gives a single answer that may be incorrect, with no way of knowing whether or not the answer is correct. A solution to this problem is to introduce some more classes that are generalisations of the main classes. An example that cannot be classified into one of the main classes may be able to be classified into a more general class, with the resulting class being correct, but less specific than the main class. Class hierarchies need not be specially constructed by the user. Since the class of a given example is simply an attribute value, the attribute hierarchy for this attribute may serve as a reasonable class hierarchy. As this attribute hierarchy may

have been induced while inferring groupings, the class hierarchy may therefore be inferred from the data.

## MARVIN++

Having described some interesting methods for incorporating knowledge into a search, the question was asked "Can these methods be integrated into a single system?" MARVIN++ attempts do this, by adapting MARVIN (Sammut & Banerji, 1986), an algorithm that provides a very general way of learning knowledge concepts, to perform induction of classification rules. MARVIN is based around the idea of using previously learned knowledge (concepts) to describe new ones; MARVIN++ uses previously learned concepts to induce classification rules for new problems, leading to the learning of new concepts.

MARVIN++ is presented with all examples (both positive and negative) and it estimates the best one to use at each iteration. Instead of generating a crucial example, MARVIN++ simply compares the concept itself to the negative examples. If a negative example can be found that satisfies the concept, then the concept is not valid, and either requires further specialisation or must be discarded.

In the absence of any background knowledge, the concepts presented to MARVIN++ are of the form

ANY_A1 <– a11.
ANY_A1 <– a12.
…
ANY_A1 <– a1n.

In other words, MARVIN++ is able to specify either an attribute value or "don't care" for each attribute, with each rule containing a clause for each attribute.

As with most machine learning algorithms, MARVIN++ cannot find the globally optimum solution, as this would require that MARVIN try all permutations of concepts over the set of examples, which is computationally infeasible, and so a local optimum is attempted, by trying to choose the best generalisation to apply first, and then continuing to generalise the example (by applying further concepts) in sequential fashion until the concept can be generalised no further. The CHOOSER algorithm selects the best attribute to generalise by determining the probability that an example is in the positive set given that attribute A has value V. This is the same formula used by PRISM (Cendrowska, 1987). To determine the best example to use, the probabilities of each attribute value are summed to give the "predictive power" of the example. The example with the highest score is the next example used. Concepts are tried in the order of most to least important attribute(s) replaced, to maximise the number of attributes generalised.

Background knowledge may be added by incoporating the appropriate concepts into the initial concept set. Attribute (and class) hierarchies are included, for example, by adding further concepts that subset the attribute values into important groups. Further concepts then group these concepts into higher-level groupings, until eventually the top concept (ANY_An) is reached.

## Conclusions

The system described above, in the absence of background knowledge, performs classification in a satisfactory manner, producing results similar to PRISM, on which the classification algorithm is loosely based. Further research is being carried out to determine how effective MARVIN++ is at utilising background knowledge to guide the induction of classification rules, and hence to discover knowledge in databases.

## References


Cendrowska, J. (1987) An algorithm for inducing modular rules. *Int J Man-Machine Studies 27*(4): 349-370

Sammut, C. and Banerji, R.B. [1986] Learning Concepts by Asking Questions. *Machine Learning: an Artificial Intelligence Approach.* Michalski et al, Morgan Kaufmann.