Expert System Development Using Data Mining

Geoffrey Holmes and Sally Jo Cunningham

Department of Computer Science
University of Waikato
Hamilton
New Zealand

1 INTRODUCTION

In constructing an expert system a knowledge engineer has recourse to two dominant technologies: empirical induction of knowledge from collections of examples, and expertise transfer tools that attempt to extract knowledge from a domain expert in a form suitable for direct inclusion in a knowledge base. Both techniques require domain expertise because the collections of examples must be carefully constructed before they are processed by machine learning programs (Gaines 1991). Machine learning techniques are promising, however, because they tend to produce knowledge in the required form and they are, aside from the pre-processing, autonomous. Current programs such as C4.5 (Quinlan 1993) work well when the collections of examples are structured and where selectivity criteria exist to discriminate examples. They are also relatively robust when presented with noisy, incomplete and irrelevant data.

If we treat a database as a supplier of examples to a machine learning program, we find that the data is often unstructured and unselective. Further, we find that general and common-sense knowledge which we would like to include in rules extracted from the data does not exist in the database. This information must be supplied by an external source, preferably someone who used their knowledge of the domain to construct the database.

There is little doubt that expert systems can benefit from a combination of domain expertise and automated knowledge extraction. Where the knowledge originates from a database we need a domain expert to provide structure, direction and a common-sense understanding of the domain. We need a degree of automation because the amount of data is too great to handle otherwise, and because automation tends to factor out misconceptions and biases introduced by the domain expert .

A new technique, called data mining, has emerged recently (Piatetsky-Shapiro, et al., 1991) for handling these problems. The idea is to explore a rich, relatively unstructured collection of data and retrieve from it unusual patterns, unexpected regularities, implicit information, etc. The mined items might be completely new rules, partial rules, constraints on rules, facts, and so on.

We see this technique as a useful addition to an expert system constructed from or allied with a collection of databases. Such systems have been described as expert database systems but to our knowledge data mining has never been used in their construction. The model we propose is captured in Figure 1 and is an example of what Kerschberg (1992) refers to as a "loosely coupled" architecture. This type of architecture is distinct from an expert deductive database system (Schmidt 1991) in that the components of the loosely coupled system have very little "understanding" of the database. Deductive databases comprise database relations as facts and use deductive rules to express the relationships between the relations. Thus a much greater semantic model of the data is built into the system.

In our model, expert system construction begins with an initial set of data, chosen by the domain expert. As construction progresses, the domain expert may alter the composition of the databases by directing that additional or different data be collected. The domain expert creates the knowledge base both through direct knowledge transfer (or through the construction of example sets for machine learning) and through mining one or more databases. Interactively guided by the expert, the mining tool efficiently organizes the search for findings in the database. These finding are then evaluated for inclusion in the knowledge base.

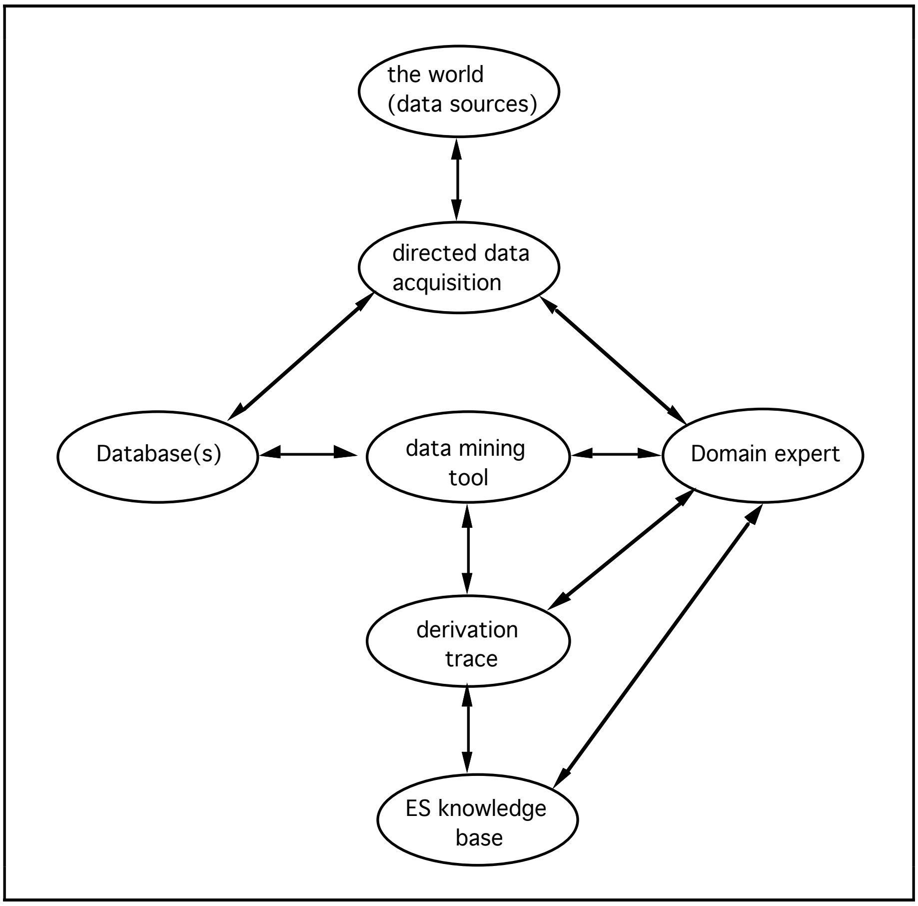

Figure 1. Architecture for using data mining to support expert system construction

The world, however, is not static; new types of data are acquired and stored, and old data is deleted or updated. These changes in the databases used to construct the expert system should be reflected by modifications to the knowledge base (see Section 3.2). Existing rules may be refined by maintaining a link between each rule in the knowledge base and the portion of the database that supports it (if any). The "derivation trace" stores the set of all such links. Significant changes in the data values that support a rule would then signal that the rule should be re-evaluated by the domain expert. It is more difficult to detect the emergence of information that could contribute a new rule to the knowledge base. In this case, we monitor the database for gross changes that indicate that the domain expert should re-mine the altered portion of the database.

We do not attempt to refine the knowledge base automatically. In the deductive database model, changes in the data can and indeed must be automatically reflected in the knowledge base. In our model, one or more relatively unstructured, heterogenous databases are loosely coupled to a knowledge base. Changes in the underlying databases may only *potentially* require a corresponding modification of the knowledge base, since the changes may generate information irrelevant to the knowledge base. In addition, the information extracted from the database may have been augmented or modified by the domain expert, and thus may not be amenable to automated modification in the knowledge base.

We illustrate the potential of this model through a case study utilizing the data mining tool "Explora", developed at the German National Research Center for Computer Science (Klosgen 1992, Klosgen 1993). The next section briefly describes this tool. Section 3 discusses knowledge base construction using it and addresses problems in knowledge base maintenance. Specifically, we discuss the importance of maintaining links between knowledge base rules and raw data, and examine an approach to determining when a "significant" change in the raw data occurs. Section 4 presents our conclusions.

2 THE EXPLORA DATA MINING SYSTEM

Database mining involves the extraction of implicit, "interesting" information from a database (Frawley et. al, 1991). The extraction process should be automated or semi-automated, and should support a high-level presentation of the information content of the data. Search through the space of candidate hypotheses can be made more efficient by the application of domain knowledge and statistical measurements of the database.

The Explora system is a statistically-based mining program described in detail in Klosgen (1992, 1993). Briefly, it identifies patterns of attribute values and value combinations that occur more or less frequently in the database than is expected. These selected patterns describe the dependency between the dependent and independent variables in a mining query. As an example: given a database of employee information, we may wish to determine how the age, sex, educational level, and race (all independent variables) impact an employee's salary (a dependent variable). The system may respond with relatively simple patterns such as "salaries are higher than average for males", or more complex patterns based on a logical conjunction of attributes and ranges, such as "salaries for college-educated employees are higher if the employee is a member of clerical staff and over the age of 40".

Standard statistical packages force the user to specify a hypothesis, select a statistical method to apply, and evaluate the measures produced. In contrast, Explora performs much of this process autonomously. Given a query (e.g., "find factors that tend to lead to high employee salaries"), it constructs an initial search space of hypotheses (candidate factors that may influence salary), selects appropriate tests for verification of these hypotheses, and evaluates the results of these tests.

Explora must search this large initial space of all potential attribute/range combinations for significant factors. The user provides domain knowledge that is used to prune this search space. Two types of domain description aid the pruning process: the aggregation of continuous variable values into meaningful clusters, such as grouping a continuous range of salaries into the categories high, middle, and low income; and the formation of taxonomies of variable values, expressing relationships such as "residents of Auckland are a subset of the residents of the North Island, which are a subset of residents of New Zealand". These clusters and taxonomies permit a partial ordering of patterns: e.g., the group "high-income people in the North Island" is more general than the group "high-income people in Auckland".

Redundancy filters are applied to the partial ordering to eliminate uninteresting portions of the statement space. The selection of a redundancy filter depends on the semantics of the domain, and the appropriate filter (if any) must be specified by the user. There are two types[1]:

specificity filter: If a given factor is found to be significant or interesting, then prune away portions of the search space containing more specific factors.

For example: If employee salaries are higher for the general factor "male employee", then more specific factors (such as "clerical males" or "young males") are discarded as unlikely to prove interesting.

generality filter: If a given factor is found to be significant or interesting, then prune away portions of the search space containing more general factors.

For example: If more than 80% of the people with high salaries are men from Auckland, then this over-representation must hold true for men from the North Island and men from New Zealand. These more general areas of the search space may then be pruned.

There is no doubt that systems such as Explora is useful in analyzing large amounts of data. But while data mining is very powerful, it is also a tool that can be easily abused: it provides its user with a mechanism for potentially making statistical errors easily and extremely quickly.

Although Explora contains some safeguards against the abuse of statistical discoveries, such as maintaining significance levels and minimum sample sizes, it does require the user to know what patterns are appropriate to apply to a given situation. Correlations that are discovered between attributes may prove true for the database as it stands, but not as it evolves. These correlations may need to be proved correct by standard statistical hypothesis testing techniques, and not simply adopted as true. Finally, predictions on the future state of the data could be made from information discovered from the current state of the data . While this is tempting to include, there would be little evidence to support the prediction without a time series analysis of the database. While issues such as these do not require a high degree of statistical sophistication, some training and awareness would be necessary for a knowledge engineer to make effective use of the tool.

---

[1]The specificity and generality filters are referred to by Klosgen [1992, 1993] as probabilistic classification filter and strong necessary filters, respectively.

3 KNOWLEDGE BASE CONSTRUCTION AND MAINTENANCE

The following discussion continues the example introduced in Section 2: we wish to construct an expert system that will predict an employee's expected salary range, given a description of the employee's work history, age, sex, etc. A database of information on current employees is available for mining.

3.1 Knowledge Base Construction

We take as our model an expert system that is constructed (in part) from concepts derived from one or more databases. The knowledge engineer guides the mining of these databases, and incorporates the concepts located into the expert system. This model differs from the machine learning model of expert system construction, in which the expert system is automatically derived from a single set of examples. Our model is advantageous when there is no single database containing all information needed to construct the knowledge base, and when common sense or other unstored information must be added to the system.

We use data mining to suggest information for possible inclusion in rules. The data may yield rules suitable for direct inclusion in the knowledge base. For example, if we wish to determine the employment categories currently receiving the highest salaries, the rule "employees holding the rank of vice-president or above receive the highest salary" can be directly extracted from the database. In some cases the information extracted must be augmented with additional knowledge held by the expert, or combined with information mined from a different database. For example, we may extract the information that "graduates of the University of Waikato receive higher than average salaries". The knowledge engineer may possess additional knowledge that it is the specialized training available at this university that provides a professional advantage, and that other universities are beginning to offer this training. A more general rule would be formed, then, including a test for this training rather than a test for a University of Waikato degree.

Alongside useful information, the data mining technique may also capture irrelevant information. One piece of information that can be mined from an employee database is that no one under the age of 16 receives a salary. A common sense understanding of the world allows a knowledge engineer to discard this fact, since the child labour laws guarantee that the company will never have an employee under 16 years old.

3.2 Knowledge Base Maintenance

How can a system recognize that it is likely that the knowledge base requires modification? After the expert system is constructed, the employee database will continue to be updated, and as a consequence the knowledge base may have to be refined (Figure 2). These changes to the database may be insignificant, or they may indicate major shifts in the high-level information implicit in the data: our existing rule may need to be modified, for example by extending the range of education to encompass both high school and university; if the subpopulation supporting the rule is eliminated (i.e., if the employees forming the basis for the rule leave the company), then the rule may need to be deleted; and shifts in the employee database may indicate that a new, unrelated rule should be created (for example, that clerical workers on the North Island have low salaries).

For each existing rule, a link must be maintained between the portions of the database used to construct the rule, the Explora query that extracted the information relevant to the rule, and the rule itself. In this model the portion of the database supporting the rule is monitored as it is updated (and as insertions and deletions are made), and the appropriate Explora queries are re-executed when it appears likely that "significant" changes have occurred in the data . Results of these queries would then be compared with previous results (stored in a rule derivation trace). If the results differ, then the rule and the derivation trace would be presented to the knowledge engineer to determine whether the knowledge base should be modified.

We do not address the problem of directly detecting the emergence of new rules, except in the general sense of monitoring the database for broad population shifts that indicate that a closer inspection of the data is necessary. When a population change is noted, the knowledge engineer is presented summary information (means, standard deviations, etc.). The knowledge engineer may then choose to mine those portions of the database for information to use in constructing new rules.

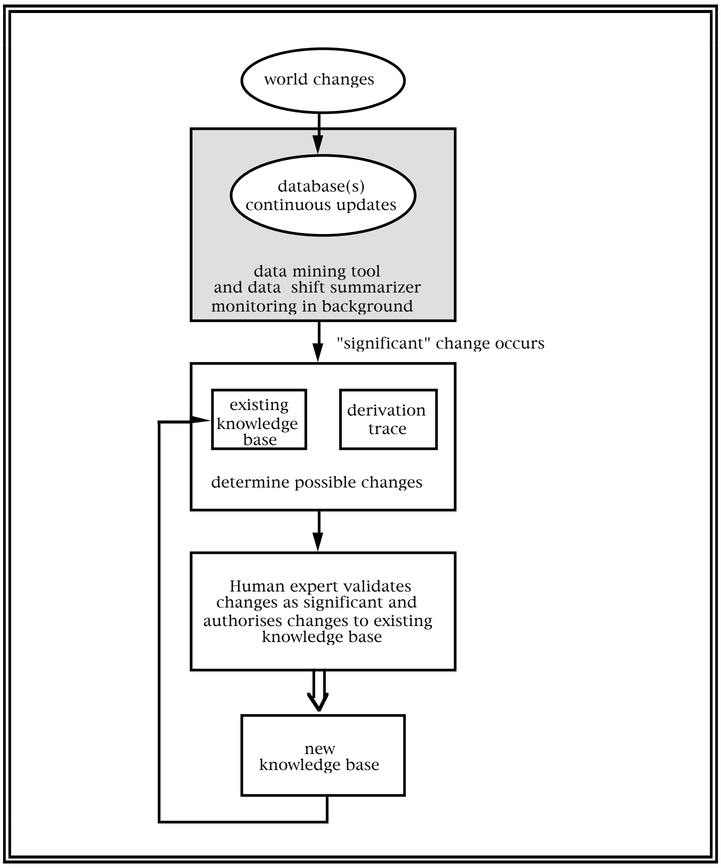

Figure 2. Knowledge base maintenance

The following example illustrates these concepts:

An initial query to Explora is, "what factors indicate that an employee will receive a high salary?", and the system's response is that "males with a high school education receive the highest salaries". We add the rule

(sex = male) and (education level = high school education) --> salary = high
to the knowledge base.

3.2.1 Tracking changes in the support for a rule. Each mined rule in the database can be invalidated in two detectable ways: either the subpopulation associated with the rule can be modified or eliminated so that it no longer supports the rule, or the changes in the global population may invalidate the conclusion of the rule (for example, if everyone but high school-educated males are given large salary increases, then these men will no longer receive "high" salaries even though the contents of their pay-packets remain the same). We consider the latter problem in Section 3.2.1, and concentrate on the former in this Section.

To detect database changes that may invalidate an existing rule, we tag the tuples and their attributes that were extracted by Explora as supporting evidence. We then monitor the updates, deletions, and insertions in this subpopulation:

- *Deletions:* During the initial data mining, the knowledge engineer indicates the level of statistical significance that is appropriate for the Explora query. We keep a count of the tuples in the subpopulation to ensure that it does not drop below the minimum required to maintain significance. If the subpopulation is no longer large enough to support the rule, then the system notifies the knowledge engineer.

- *Updates*: We distinguish between tuple updates which remove support from the rule and updates which do not affect the rule's validity. Updates to attributes not associated with the rule obviously form the latter case, and these are not monitored. Attribute updates that still support the validity of the rule are detected and ignored (for example, if an employee's salary decreases but is still within the "high" range). Updates that violate the rule (for example, an employee gaining a university degree) are treated as a deletion from the supporting subpopulation, and a trace of the update is made. This trace may be useful in helping the knowledge engineer to reformulate the rule; in this case, if the subpopulation's education level rises, an appropriate reformulation might be
  (sex = male) and (education level >= high school education) --> salary = high

- *Insertions*: A new tuple may be added or an existing tuple outside the subpopulation may be modified so it that supports the rule. Optimally, we detect these tuples and tag them as members of the supporting subpopulation. Capturing these additions requires matching each new or updated tuple against the patterns for the entire rule base. While this process is computationally expensive, it does permit additions to offset tuples eliminated from the subpopulation by deletion or updates, and thus can reduce the number of "false alarms" raised by the maintenance system. In a uniprocessor system with a constantly updated database, matching may be unfeasible; in this case, we ignore potential insertions into the subpopulation at the cost of possibly generating erroneous messages about a rule's subpopulation falling below the minimum level for maintaining statistical significance.

To give a flavour of the types of rule refinement that may be necessary, consider the following changes in the database:

- erroneous data in the database is corrected. Suppose that a data entry error caused many female employees to be labelled as male. After the error has been corrected, a new run of the original query determines that the factor determining high salary is “high school education” alone. This indicates that the original rule should be modified as:

  education level = high school --> salary = high

- formerly under-represented subpopulations grow. As an example, members of the subpopulation “age > 40 and a clerical worker” also tend to have high salaries. But because so few members of this population existed in the original database, Explora did not report this group as a statistically significant factor in predicting a high salary. If new tuples are added that are members of this sub-population, then Explora will detect this factor. In this case, the knowledge engineer would add an additional rule to the knowledge base:

  (age > 40) and (occupation = clerical worker) --> salary = high

- data changes reflect “concept drifts” in the world. Concepts implicit in a database will slowly change as the data itself changes, and this change must be reflected back into the knowledge base (Utgoff 1989). As an example, the past few years have seen an inflation in the qualifications required for many jobs. New employees are likely to have undergraduate or graduate degrees. As these new employees gain upper-level positions in the company, a different population description will emerge:

  education level = university --> salary = high

3.2.2 Tracking global changes. As noted above, we use coarse measures rather than close analysis to monitor the database for the possible emergence of new rules. We adopt this policy for pragmatic reasons: it is more efficient for the knowledge engineer to guide the mining process than to force Explora to autonomously create queries, since many queries will be known by the knowledge engineer to be meaningless or irrelevant. Further, the responses to a query must be examined by the knowledge engineer before they can be included in the knowledge base. This approach differs from machine learning algorithms, which create a comprehensive description of a set of examples that in some cases may be automatically updated (see, for example, Utgoff 1989). We argue that our approach is more suitable for working with complex and relatively unstructured databases, and when the information extracted from the database is likely to require augmentation with domain knowledge.

We track global changes in the database by:

- maintaining appropriate ranges for clustered attributes. For example, as salaries change, the ranges for "high", "low", and "average" salaries will also change. If a formula or algorithm has been used to cluster salaries, then these ranges can be automatically re-calculated. When new ranges emerge, the tuples supporting rules are examined to ensure that the rules are still valid. As an illustration: as mentioned in Section 3.2.1, suppose that the salaries of high school-educated males are maintained at the present levels while salaries in general rise. Eventually, the salary ranges will shift so that it is no longer true that these employees receive "high" salaries.

- monitoring the profiles of attributes. We keep the mean and standard deviation of continuous attributes, and allow the knowledge engineer to specify an expected amount of variation for each attribute. If updated values exceed this variation, then a warning message is issued. The knowledge engineer may also choose to track additions to the set of possible values for nominal attributes. For some nominal attributes, such as "employee name", changes in the value set are unimportant; for other attributes, such as "employment category", the addition of a new value could signal a need to refine the knowledge base.

- tracking gross changes in the population. We store the number of insertions to and deletions from the database, which we use to indicate a percentage change in the population. The knowledge engineer specifies a maximum level for this change, and when it is exceeded a warning message is issued.

4 CONCLUSION

It is attractive to loosely couple a set of databases with the knowledge base of an expert system. Information in the databases can support expert system construction by augmenting, confirming, or denying the domain model of the knowledge engineer. Since these databases may be large, complex, and relatively unstructured, data mining tools appear promising as aids to extracting implicit information from the databases. Further, as time passes the implicit information in the databases may be expected to diverge from the explicit knowledge in the expert system: erroneous values may be corrected, patchy ranges of variable values may be filled in, and concepts represented in the database may “drift” to reflect changing conditions in the world. We have described a model for monitoring the database to determine whether the knowledge base should be updated as well.

The most important area for future work is to refine the mechanism for maintaining the database/knowledge base links. While the current model appears adequate, experimentation with thresholds for “significant” concept change, efficient storage of database/knowledge base links, adding a greater semantic content to these links, etc. is indicated.

5 REFERENCES

Frawley, William, Piatetsky-Shapiro, Gregory, and Matheus, Christopher (1991). “Knowledge discovery in databases: an overview”. In Piatetsky-Shapiro and Frawley, pp. 1-25.

Gaines, Brian (1991). “The trade-off between knowledge and data in knowledge acquisition”. In Piatetsky-Shapiro and Frawley, pp. 490-505.

Kerschberg, L. (1992). Foreword section, *Expert Database Systems*. Keith Jeffery editor, Academic Press.

Klosgen, Willi (1992). “Problems for knowledge discovery in databases and their treatment in the statistics interpreter EXPLORA”. *International Journal for Intelligent Systems*, vol 7, no. 7, pp. 649-673.

Klosgen, Willi (1993). “Discovery in databases: A typology of patterns”. Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics, pp. 289-295.

Quinlan, J. Ross (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, Inc.

Piatetsky-Shapiro, Gregory, and Frawley, William J., eds. (1991). *Knowledge Discovery in Databases*. Menlo Park, CA: AAAI Press.

Schmidt, H. (1991). "Meta-Level Control for Deductive Database Systems". *Lecture Notes in Computer Science no. 479*, Springer-Verlag, pp. 33-46.

Utgoff, P.E. (1989). "Incremental induction of decision trees." *Machine Learning*, vol. 4, no. 2, pp. 161-186.