Pre-publication draft of a paper which appeared in the Proceedings of the Fifth Australian Joint Conference on Artificial Intelligence (AI 92), pages 329-334.

MAN-MACHINE COLLABORATION FOR KNOWLEDGE ACQUISITION

GEOFFREY I WEBB
Department of Computing and Mathematics
Deakin University, Geelong, Victoria 3217, Australia

ABSTRACT

Both machine learning and knowledge elicitation from human experts have unique strengths and weaknesses. Man-machine collaboration for knowledge acquisition allows both knowledge acquisition techniques to be employed hand-in-hand. The strengths of each can alleviate the other's weaknesses. This has the potential to both reduce the time taken to develop an expert system while increasing the quality of the finished product. This paper discusses techniques for man-machine collaboration for knowledge acquisition and describes Einstein, a computer system that implements those techniques.

1. Introduction

The traditional approach to knowledge acquisition is knowledge elicitation. In essence, a human expert is required to articulate a decision making procedure which is then encoded as a knowledge base. Knowledge elicitation relies both upon the reliability of the expert's knowledge and upon the ability of the expert to articulate the decision procedures implicit in that knowledge. In practice, expert's knowledge is not infallible and experts prove to be extremely poor at articulating decision procedures. Indeed, it has been claimed that *the more competent domain experts become, the less able they are to describe the knowledge they use to solve problems*[1].

Data-driven machine learning (DDML) provides an alternative approach to knowledge acquisition. A DDML system analyses a set of examples of decision making (called the training set) in order to develop a decision procedure. DDML has demonstrated the capacity to produce expert systems of outstanding accuracy in suitable domains[2]. However, it is also subject to a number of deficiencies, specifically, that a set of examples is unlikely to be complete (cover all relevant possibilities) in any complex domain and, current systems do not have access to substantial common sense and domain specific knowledge.

Man-machine collaboration for knowledge acquisition (MMCKA) combines both knowledge acquisition methodologies enabling the expert to contribute domain specific and common sense knowledge without the need to articulate complete decision procedures and the computer to contribute a capacity for exhaustive formal analysis. This paper describes techniques for MMCKA and Einstein, an implemented system that embodies those techniques.

2. DLGref

The DDML algorithm at the heart of the Einstein system is DLGref[3]. DLGref can develop new rules as well as specialising and/or generalising existing rules. It uses heuristic search while developing an expert system that seeks to minimise changes to the initial expert system while producing a system that maximises an evaluation criteria with respect to the training set of examples. One such criteria ensures completeness and consistency with respect to the training set. Alternative criteria allow for noise and related phenomena by employing a trade-off between maximising the number of positive cases that a rule covers and minimising the number of counter-examples that are covered.

DLGref differs from the two major previous DDML refinement algorithms, SEEK2[5] and GEM[6] in a number of important respects. Unlike SEEK2, DLGref is able to add new rules to a rule set

Published in A. Adams & L. Sterling (Eds) *AI '92*. World Scientific, Singapore, pp. 329–334.

and utilises a progressive search in a single direction (from specialised rules to more general rules). SEEK2 alternates between specialising and generalising rules, a process that carries a risk of successive steps undermining each others effect. Unlike GEM, DLGref minimises the degree of change to the initial expert system. GEM uses the initial expert system to seed a search process, but then allows unbounded change to be wrought. Further, DLG, and hence DLGref, is computationally efficient, making it suitable for interactive use. DLG and DLGref are described and evaluated in detail elsewhere[3,7,8,9,10]. Due to their complexity these descriptions will not be repeated herein.

3. Einstein

Einstein is a MMCKA system based on the DLGref algorithm. Two versions have been implemented, an interactive version that runs on Macintosh computers and an autonomous version that runs on other platforms. This paper describes the interactive Macintosh system.

Einstein differs from previous interactive induction systems[12] in that it operates at all stages upon production rules rather than decision trees and that the induction process can be used to refine existing rules rather than being restricted to the development of new knowledge bases. In consequence, unlike previous systems, Einstein can be used in an iterative refinement process during which both the human expert and the machine learning sub-system contribute successive modifications to a knowledge-base.

Einstein provides DDML, knowledge-base editing, and case based analysis facilities. In order to use the DDML and case-based analysis facilities it is necessary to provide a set of example cases. These may be partitioned into a training set and an evaluation set. The former is used by the DDML sub-system and the latter is used in evaluating system performance. Initiative within the system is left in the hands of the user.

Einstein currently supports a simple rule based expert system language in which conditions are conjunctions of attribute value tests and conclusions are a simple classification. All conclusions assign one only from a set of mutually exclusive classifications. Thus, all rule sets are flat. That is, every rule directly relates the raw attributes to an ultimate conclusion. There are no intermediate reasoning steps. However, this restriction is not inherent in the methodologies employed. Future versions of the system will support intermediate conclusions and more powerful paradigms than attribute-value tests.

Two types of attributes are distinguished. Ordinal attributes have values for which it is possible to develop tests of the form $min \leq attribute \leq max$. For nominal attributes, tests are restricted to set membership criteria. The system supports up to thirty-two binary (two valued) attributes and an additional thirty-two non-binary attributes. Non binary nominal attributes may have up to sixty-four values. Nominal attributes with more than sixty-four attributes may be simulated through the use of multiple attributes. Ordinal attributes may be either integer or real valued.

Once an expert system has been developed, it can be output as a working stand-alone Macintosh program, as Clips source code or in a generic expert system format. There are plans to support other expert system formats as demand dictates.

An initial knowledge base might be formed by importing an existing expert system developed outside the system; application of the DDML sub-system; or the user creating a set of rules. Rules created by the user may reflect solely his prior knowledge, intuitions and experience, or may be developed within the system through analysis of example cases. The initial rules need not be exhaustive or complete. They may simply embody general intuitions about the domain which are believed to be helpful even if known to be inaccurate.

Once a set of rules exist, six primary types of user action are available—case based analysis, example verification and refinement, rule execution step through, model refinement, case based critique, DDML rule refinement, configuration of the DDML sub-system and rule editing.

3.1. Case based analysis

Einstein provides facilities for the user to examine both the performance of the rule set as a whole on the training set and the performance of individual rules on individual examples. Rule set evaluation causes the current rule set to be applied to every case in the training set. A detailed breakdown of performance is presented on a decision by decision basis.

Individual rule evaluation is automatically performed whenever a set of rules exists. One rule is always identified as the current rule. The user may select at will the rule that is to be the current rule. A number of windows present example cases that lie in specific relationships to the current rule.

The positive examples window displays all examples for which the rule fires and for which the rule's conclusion is correct. This is useful for the user in evaluating why a successful rule succeeds. By examining the positive examples it is possible to see how the conditions identified in the rule relate to its conclusion. This is also useful when considering how a rule should be changed as it will generally be desirable to avoid changes that reduce the number of positive examples. Examination of the positive examples can suggest types of changes that will not cause a rule to fail for cases in which it currently correctly succeeds.

The counter examples window displays all examples for which the rule fires and for which the rule's conclusion is incorrect. This information has a number of useful applications. It can be used to guide a consideration of possible changes to the rule to make it more specific. In particular, when changing a rule it is generally desirable to reduce the number of counter examples. Comparing the features of positive examples with the features of counter examples provides a powerful source of guidance when considering potential rule refinements.

Counter examples can also be used to explore why particular clauses in a rule are necessary. It is frequently the case that an expert will examine the rules developed by Einstein's DDML sub-system and wonder why a particular clause is included in a rule. The DDML sub-system will usually include a clause in a rule only if it is needed to prevent the rule from incorrectly firing for cases to which its conclusion does not apply. By deleting the clause, the user will receive powerful evidence of the utility of the clause in the form of a display of all the cases for which the rule fires incorrectly in the clause's absence. Examination of these cases provides a context for evaluating the potential for alternative forms of the rule that do not employ the clause in question.

The uncovered examples window displays all examples for which the rule does not fire even though the conclusion for the rule applies. This provides useful guidance to the user as to the manner in which a rule might usefully be made more general.

The insufficient evidence window displays all example cases for which, due to missing information, it is not possible to determine whether or not the rule should fire. This information is useful when examining the effect of incomplete information on the rule set and may lead to the development of auxiliary rules or the alteration of existing rules in order to accommodate the possibility of missing data.

3.2. Example verification and refinement

The use of case based analysis also leads to an on-going process of example verification and refinement. For example, if the user modifies a rule developed by the DDML sub-system and one or more counter examples are displayed, an examination of the counter examples may vindicate the user's modification by revealing errors in those example cases. Similarly, an examination of the positive examples for a rule that appears anomalous to the user may reveal that it is based on defective examples.

A further facility for example verification is a window that displays all indistinguishable examples. These are example cases that have identical descriptions except that they are assigned different decisions. The existence of indistinguishable examples indicates that

either the examples are incorrect or that the domain model is not sufficiently powerful to support definitive decision making.

The user is free to examine and modify the examples at all times.

3.3. Rule execution step through

The user is also able to evaluate the rule set by examining its operation in a normal operating environment. In one window the user interacts with the expert system. Meanwhile, the rules being utilised are displayed in the rule editing window. This enables the user to examine how the rules operate in practice, and to quickly identify and correct deficiencies.

3.4 Model refinement

The user is free at any stage to add or delete attributes from the domain model used by the system. This enables him to change the language that is available for use in expressing rules.

A decision to modify the domain model may be prompted by identification of indistinguishable examples, by an inability to specify a satisfactory rule that excludes certain counter examples or includes certain uncovered examples or by a realisation as to why the induction sub-system is failing to produce certain types of rule.

No previous DDML system has incorporated the ability to transform the domain model in an interactive manner and to apply that revised model to refine rules developed under the original model.

3.5. Case based critique

People are used to conveying information in the form of examples. It will frequently be the case that a user will know that a rule or set of rules are incorrect, but be at a loss as to how they might be improved. In such a circumstance, the user will often be able to convey the deficiency in the form of examples. These examples may be either cases for which the current rule base would reach the wrong conclusion, or cases for which the current rule base would fail to reach a conclusion.

Einstein allows the user to specify additional example cases at any stage, thus supporting case based rule critique. The new cases are taken into account during subsequent rule evaluation and DDML refinement.

3.6. DDML rule refinement

The DDML rule refinement facilities can be called upon at any stage to refine either the rules relating to a single conclusion or the entire rule set. The user may further constrain the automatic refinement facilities by specifying for individual rules whether they may be refined or not. DLGref refines rules in context of a rule set. As a result, the rules that are specified as not to be refined are taken into account when other rules are modified. DLGref adds rules to the rule set as necessary to make the rules for a single decision or the entire rule set, as appropriate, complete with regard to the training set.

DLGref is computationally efficient allowing interactive application of rule refinement with large sets of example cases even on low end Macintosh systems.

3.7. Configuration of the DDML sub-system

The user is provided with a large range of options for managing the DDML sub-system. These range from means of providing domain specific guidance to specification of the induction techniques that should be employed. As already mentioned, the user can convey domain knowledge to the DDML sub-system by specifying pertinent examples and counter-examples, by specifying rules and by controlling which rules the system may modify.

A further facility for conveying domain specific guidance is available in the form of the ability to identify key attributes. These are factors of which the DDML sub-system should

always take account when developing or refining rules for a specific decision unless there are compelling reasons to the contrary. This facility is particularly useful when there are insufficient examples for the system to be able to determine that a particular factor is important for a particular decision or when formal analysis is able to develop several distinct decision methods for a decision and the user wishes to control which is utilised.

The user is also provided with more general control over the form of expert system to be developed and the induction techniques to employ.

The user may specify that rules are to be absolute or relative. Absolute rules are interpretable in isolation. It is possible for more than one absolute rule to fire for a single case. In this circumstance a conflict resolution strategy must be employed. By contrast, relative rules are considered in order. Only one relative rule may fire for any given case, the first rule whose condition is satisfied. The induction of relative rules is faster than the induction of absolute rules as when developing a relative rule it is only necessary to consider those cases that are not covered by rules previously developed. However, absolute rules are easier for an expert to understand and maintain as each such rule can be considered in isolation, whereas the meaning of a relative rule is dependent upon all rules in a rule set that precede it.

Two settings allow the user to extend the basic DLG search strategy. The number of concurrent hypotheses allows the user to specify that instead of using the standard DLG search which maintains and considers only a single alternative at a time, multiple candidates should be maintained and considered. At the conclusion of the search the best candidate is accepted. The number of induction cycles allows the user to specify that multiple complete expert systems should be developed. Of those developed, the one containing the least rules is selected. These two extensions to the DLG algorithm are described by Webb[7,10]. Both have demonstrated the induction of more accurate expert systems than the basic DLG algorithm at a cost of decreased computational efficiency.

The final type of option controls further refinement of the rules developed by the DLG induction stage. The basic DLG algorithm, when applied to attribute value data, develops rules whose conditions refer to all attributes and whose conditions are bounded by the most extreme observed cases. The further refinement stage examines these rules and generalises them by deleting superfluous conditions and extending the boundaries of range tests beyond the most extreme observed values. Five strategies are available. No rule refinement leaves the output of the DLG stage unaltered. Conservative rule refinement applies the conservative conjunct deletion strategy[9] with two scans, as a result of which clauses are only deleted if there is strong evidence that they are redundant. Radical rule refinement applies the radical conjunct deletion strategy[9] with two scans, as a result of which as many clauses as possible are deleted. Radical rule refinement also extends range boundaries beyond the most extreme observed values to a position between the most extreme observed positive example and the most extreme observed counter-example. Finally, fast conservative and fast radical rule refinement provide single scan, and thus faster, but less conservative or radical, refinements.

3.8. Rule editing

A comprehensive rule editor allows the user to modify the rule set at any stage. Thus, the user may perform modifications as a result of detecting deficiencies by reviewing the current set of rules, due to interactions with the case based analysis tools or during a rule execution step through.

4. Future directions

As mentioned above, Einstein is limited to rules that employ attribute-value tests and does not allow rules that derive intermediate conclusions. Both of these limitations will be addressed in future versions of the system. Intermediate conclusions can be supported by allowing attributes to be identified that may both be the target for rules (appear in a conclusion) and be

used in rule conditions. A hierarchy of such attributes would need to be defined in order to prevent circular references between rules. When the induction sub-system was applied, it would consider each such attribute as the target classification attribute in turn.

In addition, new attributes could be defined in terms of arithmetic and logical operations upon existing attributes. Research into automated methods for inducing useful attributes has demonstrated considerable success[12,13]. In addition, the user could supply such attributes as appropriate.

In order to develop a more powerful paradigm than the current use of attribute-value tests, consideration is being given to the application of DLGref to the refinement of logic programs. In theory, this can be achieved simply by replacing the use of least generalisation by relative least general generalisation[14].

5. Summary

Einstein is a fully implemented knowledge acquisition environment that allows a user to collaborate with a machine learning system in a completely integrated manner during all stages of the knowledge acquisition cycle. This allows both the user and the computer to contribute their unique capabilities to the knowledge acquisition task. Such collaboration has the potential to reduce the time taken to develop an expert system while improving the quality of the finished product.

6. Acknowledgments

This research has been supported by the Australian Research Council.

7. References

1. D. A. Waterman, *A Guide to Expert Systems*. (Addison-Wesley, Reading, Mass., 1986) .

2. J. R. Quinlan, *International Journal of Man-Machine Studies*, 27 (1987) 221.

3. G. I. Webb, *Data-driven inductive knowledge-base refinement*. Submitted for publication.

4. R. S. Michalski, in *Machine learning: An artificial intelligence approach*, ed. R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Springer-Verlag, Berlin, 1984), p.83.

5. A. Ginsberg, *Automatic Refinement of Expert System Knowledge Bases*. (Pitman, London, 1988).

6. R. E. Reinke and R. S. Michalski, in J. E. Hayes, D. Michie and J. Richards, *Machine Intelligence 11*, (Clarendon Press, Oxford, 1988) p. 263.

7. G. I. Webb, in *Proceedings of the First Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop*, (Tokyo, 1990) p.219.

8. G.I. Webb, in *Proceedings of the First Australian Knowledge Acquisition for Knowledge-Based Systems Workshop*, (Pokolbin, 1991) p. 44.

9. G.I. Webb, *Learning disjunctive class descriptions by least generalization*. Submitted for publication.

10. G. I. Webb, *Heuristic search for induction by generalization*, submitted for publication.

11. Attar Software. *Structured Decision Tasks Methodology for Developing and Integrating Knowledge Base Systems*, (Attar Software, Leigh, Lancashire).

12. S. P. Yip and G. I. Webb, to be published in Proceedings of the 1992 Pacific Rim International Conference on Artificial Intelligence (Soul, Korea, 1992).

13. S. P. Yip and G. I. Webb, in these proceedings.

14. S. Muggleton and S. Feng, in *Proceedings of the First Conference on Algorithmic Learning Theory* (Tokyo, 1990).