Pre-publication draft of a paper which appeared in the *Proceedings of the First Australian Workshop on Knowledge Acquisition for Knowledge-Based Systems (AKAW '91)*, pages 44-52

Data driven inductive refinement of production rules

Geoffrey I Webb
Department of Computing and Mathematics, Deakin University, Geelong 3217, Australia

Abstract
This paper presents algorithms for inductive refinement of production rules based on the DLG data-driven machine learning algorithm. These algorithms modify the input production rules with reference to a set of examples so as to ensure that all positive examples are covered and no negative examples are covered. The input production rules may either have been previously learnt by a machine learning system or be extracted from an existing expert system.

1 Introduction

In many circumstances, the output of an autonomous data driven machine learning system will require further refinement before it is suitable for use as a reliable classification system. One of the motivations of the research presented herein is to provide automated tools to assist in this refinement process.

To date, the primary role of data driven machine learning has been the initial development of a classification system. This paper examines data driven machine learning techniques for revising existing classification systems. The techniques to be examined operate on production rules.

The success of these techniques enables the development of a new type of knowledge acquisition system, one that combines the power of data-driven machine learning (Michalski, 1984) with that of interactive knowledge elicitation (Bareiss, 1989; Boose & Bradshaw, 1987; Gruber, 1989).

The form of inductive refinement discussed herein is an interactive process. Inductive refinement is employed along side computer driven elicitation of knowledge from a human expert.

2 The DLG algorithm

The techniques to be discussed are based on the DLG data driven machine learning algorithm (Webb, 1991a, 1991b). DLG is a computationally efficient variant of the Aq (Michalski, 1984) and least generalisation (Plotkin, 1970, 1971) algorithms that can develop class descriptions from both discrete and continuous attribute value data. There are several reasons why DLG is particularly suited to the purpose of inductive refinement.

1. It can be applied to production rules which are a convenient format for communicating knowledge between a computer system and a human expert.
2. Its computational efficiency results in rapid induction which is conducive to successful interactive operation.
3. It can be revised readily to inductively refine existing production rules.

The DLG algorithm generates a class description in disjunctive normal form. One representation for such class descriptions is a set of production rules.

Applied to production rules, DLG operates by generating a sequence of rules each of which covers a subset of the examples of the class. Taken together, these rules form a ruleset that covers all examples of the class. Each rule is formed by selecting one example, forming a maximally specific rule that covers it and then generalising this rule to also cover other examples. It can be described as Algorithm 1.

It should be noted that this algorithm is most successful when disjunction is not permitted in the antecedents of the production rules. This in no way limits the expressive power of the production rule set, as disjunction can be obtained by forming separate rules with each disjunct as the

antecedent of a different rule, For the purposes of this paper, it will be assumed henceforth that disjunction is not permitted in the antecedents of production rules.

2.1 Simple DLG revision

A simple alteration to the DLG algorithm allows it to accept as input a set of production rules and then revise these against the available examples. This is presented as Algorithm 2.

This algorithm performs three types of modification to a production rule set. Rules may be added, deleted or generalised. Rules are generalised if it is possible to do so in such a manner as to cover positive instances not covered by rules already examined, while not covering any negative instances. Rules are deleted if they cover instances in *NEG* or cannot be generalised to cover any instance in *POS* Rules are added if the ruleset, after generalisation and deletion of initial rules, does not cover all instances in *POS*.

```
Input:      POS (a training set of instances belonging to the class of interest)
            NEG(a training set of instances not belonging to the class of interest)
Output:     R (a set of production rules for the class.)
      initialise R to ?
      while POS is not empty
            randomly select and remove an instance i from POS.
            initialise c to the most specific rule that covers i.
            for x is set to each successive instance in P0S
                  set l to a least generalisation of c that covers x.
                  if l does not cover any instance in NEG
                        set c to l
            remove from POS all instances covered by c.
            add c to R.
end while.
```

Algorithm 1: DLG

An example will serve to illustrate the process. The examples to be presented will relate to tile data set presented in Table 1. For the purposes of illustration, we will assume that the conditions of rules take the form of a simple conjunction of clauses, each of which relates to a single attribute and each of which takes the form of a test for range or set membership or equality. It should be noted, however, that the technique is in no way restricted to such simple types of rules.

Let us suppose that C, the initial ruleset for A, is

IF Colour = black & Age ≥ 43 & 11 ≤ Price ≤ 23 THEN A; and
IF Size ≥ 10 & Weight ≤ 3 THEN A.

First, we initialise R to ? . *POS* is not empty and *C* is not empty, so we set *c* to

IF Colour = black & Age ≥ 43 & 11 ≤ Price ≤ 23 THEN A

and remove this rule from C. This rule, c, does not cover any case in NEG, so for each successive case, x, in *POS* we obtain a least generalisation of c that that covers x. The least generalisation of c that covers the first case is

IF Colour = black & Age ≥ 37 & 11 ≤ Price ≤ 23 THEN A.

This does not cover any case in *NEG* and so is accepted. This continues with the successive generalisations.

Size	Colour	Weight	Age	Price	Class
16	black	3	37	11	A
20	black	1	43	23	A
32	black	2	48	15	A
21	black	2	29	19	A
31	black	5	23	2	A
10	black	5	21	1	A
19	black	4	19	9	B
12	black	6	42	0	B
16	black	5	12	4	B
9	red	0	43	26	B
1	blue	5	38	3	B
8	blue	2	39	17	B

Table 1: Example Data

IF Colour = black & Age ≥ 37 & 11 ≤ Price ≤ 23 THEN A,
IF Colour = black & Age ≥ 37 & 11 ≤ Price ≤ 23 THEN A,
IF Colour = black & Age ≥ 29 & 11 ≤ Price ≤ 23 THEN A,
IF Colour = black & Age ≥ 23 & 2 ≤ Price ≤ 23 THEN A, and
IF Colour = black & Age ≥ 21 & 1 ≤ Price ≤ 23 THEN A.

None of these covers a case in *NEG* and thus all are accepted. The clause c now equals the last of these values. It covers all cases in *POS*, so all cases are removed from *POS* and *R* is set to *R v c* which equals

IF Colour = black & Age ≥ 21 & 1 ≤ Price ≤ 23 THEN A.

POS is now empty, so the algorithm terminates with this single rule as the result.

This first simple example demonstrates the generalisation and deletion of rules from the initial ruleset. Rules are added to the ruleset when it is not possible to generalise the initial rules to cover all cases in *POS*. For example, if the initial ruleset contained only the rule

IF Size ≥ 20 & Age ≥ 10 & Price ≥ 5 THEN A,

this would be generalised to

IF Size ≥ 20 & Age ≥ 10 & Price ≥ 2 THEN A

which does not cover the first or last case in *POS.* Consequently, a second rule would be added - **IF1O≤Size<16 & Colour = black & 3 ≤Weight ≤ 5 & 21 ≤ Age ≥ 37 & 1 ≤ Price ≥ 11 THEN A.**

```
Input:      POS (a training set of instances belonging to the class of interest)
            NEG (a training set of instances not belonging to the class of interest)
            C (a set of production rules)
Output:     R (a set of production rules for the class.)
        initialise R to ? .
        while POS is not empty and C is not empty
            randomly select and remove one of the rules r from C.
            if r does not cover any instance in NEG
                for x is set to each successive instance in POS
                    set l to the least generalisation of r that covers x.
                    if l does not cover any instance in NEG
                        set r to l.
                if r covers one or more instances from POS
                    remove from POS all instances covered by r.
                    add c to R.
end while.
while POS is not empty
            randomly select and remove an instance i from POS.
            initialise r to the most specific rule that covers i.
            for x is set to each successive instance in POS
                set l to the least generalisation of r that covers x.
                if l does not cover any instance in NEG
                    set r to l
            remove from POS all instances covered by r.
            add c to R.
end while.
```

Algorithm 2: Simple DLO revision

```
Input:      POS (a training set of instances belonging to the class of interest)
            NEG (a training set of instances not belonging to the class of interest)
            C (a class description in disjunctive normal form)
Output:     R (a class description for the class.)
    initialise R to ? .
    while POS is not empty and C is not empty
        randomly select and remove one of the rules c from C.
        if c covers an instance in POS
            randomly select an instance x from POS that is covered by c
            for every attribute not mentioned in c specialise c to require that the value
            of that attribute be the value of that attribute for x
        if c does not cover any instance in NEG
            for x is set to each successive instance in POS
                set I to the non-disjunctive least generalisation of c that covers x.
                if I does not cover any instance in NEG
                    set c to I.
            if c covers one or more instances from POS
                remove from POS all instances covered by c.
                set R to c v R.
end while.
while POS is not empty
        randomly select and remove an instance i from POS.
        initialise c to the most specific non-disjunctive class description that covers i.
        for x is set to each successive instance in POS
            set I to the non-disjunctive least generalisation of c that covers x.
            if I does not cover any instance in NEG
                set c to I
        remove from POS all instances covered by c.
        set R to c v R.
end while.
```

Algorithm 3: DLG revision II

2.2 The DLG Revision II algorithm

This deficiency is remedied by the DLG revision II algorithm, presented as Algorithm 3. This algorithm differs from the simple DLG revision algorithm only in that it performs an initial specialisation on each clause of each rule based on a randomly chosen positive instance. This enables the algorithm to add conjuncts to a rule, based on observed values from positive cases. The output of this final algorithm covers all positive cases and no negative case from the training set.

This revised algorithm can add, delete, specialise and/or generalise rules. Rules are only deleted if they cover negative examples.

To demonstrate this algorithm, with reference to Table 1, consider the case where *C* is

IF 20 ≤ Age ≤ 50 & Price ≥ 1 THEN A.

First, we initialise R to ?. *POS* is not empty and *C* is not empty, so we set *c* to the only rule in the ruleset,

IF 20 ≤ Age ≤ 50 & Price ≥ 1 THEN A.

This covers instances in *POS*, so we select one such case, *x*, in this example, the first case. For every attribute not mentioned in *c*, *c* is specialised to require that the value for the attribute be the value of that attribute for *x*. The rule *c* is set to the result,

IF Size = 16 & Colour = black & Weight = 3 & 20 ≤ Age ≤ 50 & Price ≥ 1 THEN A.

This does not cover any case in *NEG* and so is generalised against all of the remaining cases in *POS*, resulting in the rule *c* becoming

IF 10 ≤ Size = 32 & Colour = black & 1 ≤ Weight = 5 & 20 ≤ Ages ≤ 50 & Price ≥ 1 THEN A.

This correctly covers all cases in *POS* while covering no case in *NEG*.

Note that the DLG revision II algorithm, unlike the DLG and simple DLG revision algorithms, is restricted in application to attribute value machine learning (Quinlan, 1986.) If such an algorithm were desired for machine learning in another context, such as structural description machine learning (Dietterich & Michalski, 1981), the initial specialisation step would need to be modified accordingly.

2.3 Rule simplification

A remaining deficiency of this algorithm is that all the resulting rules will refer to all attributes. In consequence, they will almost certainly be over specialised and unnecessarily complex. One manner in which to rectify this problem is to apply conjunct deletion strategies that have been developed for use with the DLG algorithm (Webb, 1991c.) There are a number of these strategies, all of which are based on search techniques that delete conjuncts from rules without making the rule cover any cases in *NEG*.

The most simple of these strategies involves a sequential scan through the conjuncts in each rule. Each successive conjunct is deleted and the resulting rule is evaluated to determine if it covers any negative cases. If it does cover negative cases, then the conjunct is reinstated. Applying this strategy in a left to right scan to the rule

IF 10 ≤ Size = 32 & Colour = black & 1 ≤ Weight = 5 & 20 ≤ Age ≤ 50 & Price ≥ 1 THEN A,

which was the result of the most recent example, above, produces

IF Colour = black & 20 ≤ Age ≤ 50 & Price ≥ 1 THEN A.

After both inductive revision and conjunct deletion, the end result is that one additional conjunct, Colour = black, has been added to the rule that was input to the refinement process.

A potential problem associated with the application of a conjunct deletion strategy is that conjuncts that are in some manner key to rules in the input ruleset may be deleted. In this case, it is possible that the output of the refinement process will bear little or no obvious surface relationship to the input. This can be prevented by disallowing the deletion of conjuncts that appeared in the input, or are direct generalisations of conjuncts that appeared in the input. Alternatively, the user can be permitted to identify key conjuncts, which may be generalised but not deleted from the result. This latter option introduces greater flexibility at the cost of greater demands upon the user.

3 Conclusion

Previous research into data-driven induction has concentrated on the induction of new knowledge. The algorithms presented above support a different application of induction, the refinement of existing knowledge. These algorithms enable the development of interactive knowledge acquisition environments, of which Einstein (Webb, 1991c) is an example, where both the human expert and the computer system can provide input to every stage of the knowledge acquisition and refinement process.

References

[1] Bareiss, R. (1989) *Exemplar Based Knowledge Acquisition: A Unified Approach to Concept Learning.* Academic Press, San Diego.

[2] Boose, J. H. & Bradshaw, J. M. (1987) **Expertise transfer and complex problems: Using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems**. *Int. J. Man-Machine Studies*, 26: 3-28.

[3] Dietterich, T. G. & Michalski, R. S. (1981) **Inductive learning of structural descriptions: Evaluation criteria and comparative review of selected methods**. *Artificial Intelligence*, 16: 257-294.

[4] Gruber, T. (1989) *The Acquisition of Strategic Knowledge*. Academic Press, San Diego.

[5] Michalski, R. S. (1984). **A theory and methodology of inductive learning**. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Springer-Verlag, Berlin pp. 83-129.

[6] Plotkin, G. D. (1970). **A note on inductive generalisation**. In B. Meltzer & D. Mitchie (Eds.) *Machine intelligence 5*, Edinburgh University Press, pp. 153-163.

[7] Plotkin, G. D. (1971). **A further note on inductive generalisation**. In B. Meltzer & D. Mitchie (Eds.) *Machine Intelligence 6*, Edinburgh University Press, pp. 101-124.

[8] Quinlan, J. R. (1986) **Induction of decision trees**. *Machine Learning*, 1: 81-106

[9] Quinlan, J. R. (1987). **Simplifying decision trees**. *International Journal of Man-Machine Studies*, 27:221-234.

[10] Webb G. I. (1991a) **Rule optimisation and theory optimisation: Heuristic search strategies for data-driven machine learning**. In H. Motada, R. Mizoguchi, J. Boose & B. Gaines (Eds) *Knowledge Acquisition for Knowledge-Based Systems*. IOS Press, pp. 219-232.

[11] Webb, G. I. (1991b) **Learning disjunctive characteristic descriptions by least generalisation**. *Technical Report 2/91*, Department of Computing and Mathematics, Deakin University, Geelong.

[12] Webb, G.I. (1991c) **Einstein-An interactive inductive knowledge acquisition tool**. *Technical Report 3/91*, Department of Computing and Mathematics, Deakin University, Geelong.