Systematic search for categorical attribute-value data-driven machine learning.

Geoffrey I. Webb
School of Computing and Mathematics, Deakin University
Geelong, Vic., 3217, Australia.

Abstract

Optimal Pruning for Unordered Search is a search algorithm that enables complete search through the space of possible disjuncts at the inner level of a covering algorithm. This algorithm takes as inputs an evaluation function, *e*, a training set, *t*, and a set of specialisation operators, *o*. It outputs a set of operators from *o* that creates a classifier that maximises *e* with respect to *t*. While OPUS has exponential worst case time complexity, the algorithm is demonstrated to reach solutions for complex real world domains within reasonable time frames. Indeed, for some domains, the algorithm exhibits greater computational efficiency than common heuristic search algorithms.

Introduction

A *class description* (hereafter referred to as simply *description*) is an expression that describes a class of objects. Covering algorithms[1,2,3,4] use two levels of search to infer a disjunctive description. Although there is considerable minor variation, at the outer level they all operate approximately as follows:

```
description = false
while the training set is not empty
    form disjunct, the best non-disjunctive description with respect to the training set
    remove all cases covered by disjunct from the training set
    description ← description ∨ disjunct
end while
```

Some algorithms develop ordered descriptions by allowing disjuncts to be developed for any class at any stage[2]. Other algorithms[1,3,4] develop unordered descriptions by developing all non-disjunctive descriptions for a single class at one time.

Each disjunct is formed by search. Typically, the search space from which a disjunct is selected is extremely large. To enable search of such large search spaces, previous algorithms have employed heuristic search.

In recent years, there has been considerable interest in systematic search algorithms for use within the inner level of a covering algorithm. The first of these, LEI[5], was restricted to inferring classifiers that were complete and consistent with regard to the training set and used the strict AQ technique of searching only the generalisations of the most specialised description that covered a randomly selected positive case. Rymon's algorithm[6] constrains the size of the search space by exploring only conjunctions of clauses where each clause allows a single value only for an attribute. Schlimmer[7] performs systematic search through the same space as is explored in this research. To achieve this, its search is bounded by specifying a maximum depth to be considered.

This paper presents an algorithm for systematic search through the space of all possible non-disjunctive descriptions.

To appear in N. Foo & C. Rowles (Eds) *AI '93*. World Scientific, Singapore.

An expanded version of this paper is available as Deakin University School of Computing and Mathematics Technical Report C93/17.

Optimal Pruning for Unordered Search: A systematic search algorithm

The Optimal Pruning for Unordered Search (OPUS) algorithm is applicable to any language for expressing descriptions in which:

- a non-disjunctive description can be constructed from the most general description ANYTHING by the application of a sequence of specialisation operators; and
- the order in which those specialisation operators are applied does not affect the cover of the resulting description.

Although OPUS is applicable to inferring ordered or unordered rules[8], this paper considers only the latter case.

Inductive bias is specified to the algorithm via a value function. *value*() is a function that provides a numeric evaluation of a description such that the higher the evaluation the higher the preference for the description. This research uses the Laplace function[8],

$$value(D) = \frac{\mathrm{pos\_cover}(D) + 1}{\mathrm{total\_cover}(D) + no\_of\_classes}$$

where *pos_cover*(*D*) is the number of cases covered by *D* that belong to the class with which *D* is associated, *total_cover*(*D*) is the total number of cases covered by *D* and *no_of_classes* is the number of classes in the domain.

To reduce time complexity OPUS prunes the search space through optimistic pruning and other pruning techniques. The efficiency of OPUS arises from the manner in which it ensures that the size of the region pruned by each pruning action is maximised.

Two functions complement the value function to support pruning. An optimistic estimate (value that is not lower than the actual maximum) of the maximum value of any specialisation of *description* obtained by application of any combination of operators in the set *operators* is provided by *potential_value*(*description*, *operators*). One definition for use with the Laplace value function when developing unordered rules is

$$potential\_value(D,O) = \frac{pos\_cover(D) + 1}{pos\_cover(D) + neg\_cover(D \& O) + no\_of\_classes}$$

where *neg_cover*(*D* & *O*) is the total number of cases not belonging to the class that are covered by the description formed by applying all specialisation operators in *O* to *D*.

Another function, *cannot_improve*(*D*, *E*, *O*) is true only if

- no specialisation obtained by application of any combination of operators in *O* to *E* can have a higher value than the highest valued specialisation obtained by application of any combination of operators in *O* to *D* given that *O* does not contain the specialisation operator used to derive *S* from *D*; and
- if *cannot_improve* holds between a node and all of its children, then no specialisation of a child can improve upon the parent.

Within OPUS, three items are stored within the data structure associated with a description, the actual description (*x*→*desc*), the last specialisation operator applied in order to obtain the description (*x*→*most_recent_operator*) and a set of operators that remain to be applied in the search space descending from the description (*x*→*search_map*). The left arrow '←' denotes assignment.

```
Algorithm: OPUS
    $best \rightarrow desc \leftarrow$ ANYTHING
    $best \rightarrow search\_map \leftarrow \{operator_1, operator_2, \ldots operator_n)$
    $open \leftarrow \{ best \}$
    while $open$ is not empty
        remove $current$, the node that maximises $potential\_value(current \rightarrow desc,$
        $current \rightarrow search\_map)$, from $open$
        $new\_descs \leftarrow \{\}$.
        for o ← each operator in current→search_map in turn
            new→desc ← current →desc & o
            new→most_recent_operator ← o
            if value(new→desc) > value(best→desc) then
                best ← new
            end if
            if potential_value(new→desc, current→search_map) > value(best→desc) and
            not cannot_improve(current→desc, new→desc, current→search_map )
                add new to new_descs
            else
                current→search_map ← current→search_map – o
            end if
        end for
        for every description d in new_descs
            if for some x in new_descs, cannot_improve(x→desc, d→desc,
            current→search_map )
                remove d from new_descs
                current→search_map ← current→search_map – d→most_recent_operator
            end if
        end for
        for every description d in new_descs ordered from lowest to highest value on
        potential_value(d→desc, current→search_map)
            current→search_map ← current→search_map – d→ most_recent_operator
            d→search_map ← current→search_map
            if potential_value(d→desc, d→search_map) > value(best→desc)
                add d to open
            end if
        end for
    end while
```

OPUS maximises the effect of any pruning that can be performed. However, in the worst case, OPUS will not be able to perform any pruning resulting in the examination of all 2^n points in the search space. Thus, the worst case complexity is exponential. The important issue with regard to the algorithm's utility is to what extent is it successful in decreasing computation time on real world induction problems. This is a matter for experimental evaluation.

Experimental evaluation

OPUS has been implemented in 'C' on a Solbourne 5/602 computer. This implementation of OPUS handles categorical attribute-value data only. It uses the Laplace evaluation function and the associated *potential_value* and *cannot_improve* functions, described above. Thus, it is applicable only to develop unordered rules.

The OPUS algorithm is embodied within the following covering algorithm (Class).

```
    for class ← each class in turn
        while the training set contains objects belonging to class class
            divide the training set into POS and NEG where POS contains all objects in the
            training set belonging to class and NEG contains all other objects
            disjunct ← OPUS(POS, NEG)
            remove all objects of class class covered by disjunct from the training set
            add to the ruleset the rule IF disjunct THEN class
        end while
        restore to the training set all objects removed above
    end for
```

Table 1: Summary of experimental data sets.

Domain	Description	#Attribs	#Attrib Vals	#Objects	#Classes
Breast Cancer	Medical prognosis.	9	57	286	2
HouseVotes 84	Predict political affiliation of US Senators from voting record.	16	48	435	2
Lymphography	Medical diagnosis.	18	60	148	4
F11 Multiplexer	Artificial data.	11	22	500	2
Primary Tumor	Medical diagnosis.	17	42	339	22

This is equivalent to the ordered rules version of the CN2 algorithm with OPUS employed in the place of the CN2 *find_best_complex* heuristic search function.

When the rules so developed are applied to previously unsighted objects, three classes of outcome are possible.

1. One rule fires. In this case the conclusion for the rule is applied to the object.
2. Multiple rules fire. In this case the conclusion for the rule with the highest value, as determined by the evaluation function during induction, is applied to the object.
3. No rule fires. In this case the object is deemed to belong to the class represented by the most objects in the training set.

For evaluation, Class was applied to a number of widely studied data sets from the UCI machine learning repository[9]. These data sets are described in Table 1. The number of attribute values (presented in column 5) treats missing values as distinct values. The number of class descriptions OPUS considers for each domain is 2^n, where n is the number of attribute values.

Each data set was randomly divided into training (80% of data) and evaluation (remaining 20% of data) sets. The Class, Einstein[10] and CN2[2] induction systems were applied to each training set. Each classifier so developed was evaluated against the corresponding evaluation set. This process was repeated 100 times for each data set.

Einstein employs a variant of the AQ learning algorithm that performs search from specific to general classifiers. It was used with most extreme start, most extreme ordering, multiple start rule optimisation with a beam width of 20 and conservative conjunct deletion[10,11]. CN2 is an re-implementation of the unordered rules version of the CN2 algorithm[8]. It was used with a beam width of 20 and no significance testing. Class, Einstein and CN2 all used the same covering algorithm and Laplace evaluation function, differing only in that Einstein and CN2 used different heuristic search techniques to try to maximise the evaluation function when developing each disjunct whereas Class employed systematic search. The rule sets developed by the systems were applied with the same conflict resolution strategies (resolution of multiple or no matches), those outlined above.

Table 2 presents a summary of the experimental results. For each data set it presents

- The mean accuracy obtained when applying the classifier to the evaluation set.
- The mean number of CPU seconds taken to execute the program.
- The mean value of the first rule inferred for each class.
- The mean value of all rules inferred.
- The mean number of rules developed.

All means, other than those for Class, are accompanied by the p value for a matched pairs two-tailed t-test comparing the value with the corresponding value for Class. A p of 0.05 or lower indicates that the difference is statistically significant at the 0.05 level.

For all data sets other than F11 multiplexer (for which the most extreme ordering heuristic, employed in this study, produces a poor result for Einstein[11]), and Lymphography, the accuracy of both the classifiers produced by heuristic search

Table 2: Summary of experimental results.

Data set	Algorithm	Accuracy	CPU secs	First val	All vals	#Rules
Breast Cancer	Class	70.7	553.8	0.96	0.82	29.8
	CN2	74.1 (p 0.00)	51.3 (p 0.00)	0.95 (p 0.00)	0.86 (p 0.00)	21.0 (p 0.00)
	Einstein	71.0 (p 0.58)	22.3 (p 0.00)	0.93 (p 0.00)	0.89 (p 0.00)	27.4 (p 0.00)
HouseVotes 84	Class	93.9	2.3	0.99	0.82	14.3
	CN2	94.7 (p 0.00)	27.0 (p 0.00)	0.99 (p 1.00)	0.84 (p 0.00)	12.8 (p 0.00)
	Einstein	94.7 (p 0.00)	22.9 (p 0.00)	0.99 (p 1.00)	0.97 (p 0.00)	11.7 (p 0.00)
Lymphography	Class	78.1	2.9	0.71	0.66	11.4
	CN2	76.4 (p 0.03)	14.7 (p 0.00)	0.70 (p 0.00)	0.63 (p 0.00)	12.7 (p 0.00)
	Einstein	83.0 (p 0.00)	2.2 (p 0.00)	0.71 (p 0.00)	0.80 (p 0.00)	9.9 (p 0.00)
F11 Multiplexer	Class	99.1	2.8	0.97	0.93	22.3
	CN2	99.9 (p 0.00)	12.6 (p 0.00)	0.97 (p 1.00)	0.93 (p 1.00)	21.6 (p 0.00)
	Einstein	96.4 (p 0.00)	33.1 (p 0.00)	0.96 (p 0.00)	0.96 (p 0.00)	21.9 (p 0.18)
Primary Tumor	Class	34.2	21.1	0.21	0.17	57.6
	CN2	35.4 (p 0.01)	141.6 (p 0.00)	0.20 (p 0.00)	0.16 (p 0.00)	61.1 (p 0.00)
	Einstein	42.6 (p 0.00)	7.2 (p 0.00)	0.20 (p 0.00)	0.20 (p 0.00)	45.7 (p 0.00)

(Einstein and CN2) was greater than that of the classifiers produced by systematic search (Class). In all but one of these cases, the difference in accuracy was statistically significant. This suggests that the Laplace value function is not an optimal selection criteria for a covering algorithm and that it interacts with the heuristics employed by Einstein and CN2 to create a superior, implicit, evaluation function.

Caution is required in comparing computation time, due to variation in implementation efficiency between programs. Nonetheless, it is surprising that the mean computation time of Class was only in one case (breast cancer) significantly greater than those of both heuristic search techniques. For two of the five data sets the mean computation time for Class was actually significantly less than that for either of the heuristic search covering algorithms. That is, Class, performing repeated search, each time finding a solution that maximised the evaluation function, did so in less time than the heuristic search techniques took to find a solution.

The highest computation time for Class is recorded for the breast cancer data. One possible contributing factor to this effect is that this data has the highest mean number of values per attribute. In consequence, each specialisation step will, on average, alter the proportion of objects covered by the least amount. To illustrate this effect, consider specialisation on an attribute with two values. Specialisation to exclude one value will, on average, reduce the cover of a description by half. By contrast, if an attribute has ten mutually exclusive values, then specialisation by one value will, on average, reduce the cover by only one tenth. With a smaller effect of specialisation, it is likely to be less apparent whether or not one such specialisation leads to an optimum solution.

The mean value of the first rule developed for each class provides a measure of the relative performance of the covering algorithms on the same search task. (Once the first rule has been developed, cases are deleted from the training set, altering the potential value that can be achieved.)While the values of the classifiers found by the heuristic search algorithms are in many cases significantly lower than the actual maxima (found by Class), the magnitude of these differences is small. The mean value of all rules shows that maximising the value of the first rule will often lead to the development of subsequent rules with lower value. There appears to be a correspondence between the number of rules developed and the mean value of the rules. In general, a larger number of rules co-occurs with in a lower mean value. There appears to be little correlation between the mean value of the rules and the mean accuracy.

There is no apparent pattern to the relative complexities of the rule sets produced by the three covering algorithms.

Conclusion

Class is a data driven induction algorithm that performs systematic search to find a class description that maximises a classifier evaluation function. Although the worst case complexity of the algorithm is exponential, the performance of an implementation of this algorithm on real world data exhibits quite adequate performance, and, indeed, in many cases outperforms common heuristic search algorithms.

Comparative evaluation of the classifiers developed using systematic and heuristic search with the Laplace classifier evaluation function, demonstrates that, in most cases, maximisation of Laplace actually leads to a decrease in classification performance. This strongly suggests that the Laplace classifier evaluation function interacts with heuristic search algorithms to form implicit evaluation functions that outperform the explicit Laplace function when evaluating disjuncts within a covering algorithm. Explicit representation of such implicit evaluation functions presents a challenge for future machine learning research.

Acknowledgments

This research was supported by the Australian Research Council. Fred Brkich, Doug Newlands and Ross Quinlan made many helpful comments on drafts of this paper.

References

1. Michalski, R. S. (1980) Pattern recognition as rule-guided inductive inference. IEEE Transactions on Pattern Recognition and Machine Intelligence, 2, 349-361.

2. Clark, P. & Niblett, T. (1989) The CN2 induction algorithm. Machine Learning, 3, 261-284.

3. Muggleton, S. & Feng, S. (1990). Efficient induction of logic programs In Proceedings of the First Conference on Algorithmic Learning Theory, Tokyo.

4. Quinlan, J. R. (1991) Determinate Literals in Inductive Logic Programming. Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, Morgan Kauffman, Los Altos, pp. 746-750.

5. Webb, G. (1990) Techniques for efficient empirical induction. In C. J. Barter & M. J. Brooks (Eds) *AI '88*. Springer-Verlag, Berlin, pp. 225–239.

6. Rymon, R. An SE-tree based characterization of the induction problem. *Proceedings of the 1993 International Conference on Machine Learning*.

7. Schlimmer, J. C. (1983) Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. *Proceedings of the 1993 International Conference on Machine Learning*.

8. Clark, P. & Boswell, R. (1991) Rule induction with CN2: some recent improvements. In *Proceedings of the Fifth European Working Session on Learning*, pp. 151-163.

9. Murphy, P. & Aha, D. (1993). *UCI Repository of machine learning databases*. [Machine-readable data repository]. University of California, Department of Information and Computer Science, Irvine, CA.

10. Webb, G. (1992) Man-machine collaboration for knowledge acquisition. In A. Adams & L. Sterling (Eds) *AI '92*. World Scientific, Singapore, pp. 329–334.

11. Webb, G. (1993) *Search strategies for induction by generalization.* Technical Report TR C93/03 Deakin University School of Computing and Mathematics, 13pp.