Inclusive pruning: A new class of pruning rule for unordered search and its application to classification learning.

Geoffrey I Webb
School of Computing and Mathematics
Deakin University
Geelong Vic. 3217 Australia
webb@deakin.edu.au

Abstract

This paper presents a new class of pruning rule for unordered search. Previous pruning rules for unordered search identify operators that should not be applied in order to prune nodes reached via those operators. In contrast, the new pruning rules identify operators that should be applied and prune nodes that are not reached via those operators. Specific pruning rules employing both these approaches are identified for classification learning. Experimental results demonstrate that application of the new pruning rules can reduce by more than 60% the number of states from the search space that are considered during classification learning.

Keywords search, unordered search, pruning rules, machine learning

1 Introduction

Unordered search is search in which the order in which the search operators are applied does not affect the outcome. Subset selection, a special case of unordered search where each search operator can be applied once only, has received extensive study [7, 9, 14, 19]. Pruning is critical to successful search in such spaces. Previous research has examined pruning rules that identify search operators that cannot lead to solutions and exclude those operators from consideration. These are referred to as *exclusive pruning rules.* This paper presents a further class of admissible pruning rules: *inclusive pruning rules.* These rules identify search operators that must be applied to reach a solution and prune paths that do not include those operators.

Unordered search is applicable to many classification learning tasks, notably, search for conjunctions of constraints that identify a class or part of a class. For this search problem, search usually starts with a general classifier and each search operator corresponds to the conjunction of a constraint to the classifier under construction [5, 11]. Such search is unordered as the order in which constraints are added to a classifier is not significant.

In recent years there has been an increase in interest in the development of admissible search strategies for such classification learning tasks [6, 14, 15, 16, 18]. A search algorithm is admissible if and only if it is guaranteed to find the nominated search goal. The use of admissible search has much to offer classification learning as a scientific discipline. Most notably, the use of admissible search makes it possible to unequivocally manipulate and study learning biases. Results from experiments that seek to study alternative learning biases but employ heuristic search will always be equivocal as it will never be possible to precisely identify and quantify the learning biases introduced through the use of the search heuristic. Indeed, results of previous experimentation [18] suggests that at least one heuristic search algorithm (CN2 [2]) introduces an unidentified bias that has positive effect on classification accuracy. Previous work [18] has also demonstrated that admissible search can be more efficient than common heuristic search techniques.

Crucial to the efficiency of admissible search is the use of appropriate pruning rules. Only by pruning the majority of the search space is it possible to systematically search the large search spaces involved in many machine learning problems. Appropriate pruning can also greatly increase the efficiency of heuristic search in classification learning.

The new class of admissible pruning rules, inclusive pruning, is evaluated with respect to search in classification learning. A taxonomy of admissible pruning rules for unordered search is developed and seven admissible pruning rules for search in classification learning are described. Experimental evaluation of the relative effectiveness of selected rules is provided for appropriate classification learning tasks from the UCI machine learning repository [8].

Proceedings of the 19th Australasian Computer Science Conference, Melbourne, Australia, January 31–February 2 1996, pp. 1-10.

2 Formal framework

This paper restricts itself to consideration of search in the context of propositional classification learning. While many of the principles examined have wider application, the implications are not made explicit.

It is assumed that the search seeks to uncover an expression in the form of a conjunction of propositions that maximizes a preference measure with respect to a training set. An AQ-like [5] covering algorithm can be used to form disjunctive propositional classifiers through repeated search for such conjunctive expressions [18].

Each state in the search space corresponds to a conjunction of propositions. Each search operator has the effect of creating a new state by conjoining a specific proposition to the expression for the original state. The state resulting from application of operator o to state s is denoted by $s \wedge o$.

Search is guided by consideration of a set of training examples. These examples are divided into a set of positive examples P and a set of negative examples N.

For any state s, $pos(s)$ denotes the set of positive examples and $neg(s)$ denotes the set of negative examples covered by the expression for s. The initial state is denoted by IS. IS is the most general possible expression. $pos(IS) = P$. $neg(IS) = N$. The state at a node n in the search tree is denoted by $state(n)$.

For any operator o, $pos(o)$ and $neg(o)$ are used as abbreviations for $pos(IS \wedge o)$ and $neg(IS \wedge o)$, respectively.

For a set x, $|x|$ will denote the cardinality of x.

The term *search space below node n* will denote all states that can be reached from n by application of any combination of operators active at node n.

For any state s and operator o , $pos(s \wedge o) = pos(s) \cap pos(o)$ and $neg(s \wedge o) = neg(s) \cap neg(o)$. It follows that multiple applicatons of a single operator have the same effect as a single application of that operator ($s \wedge o \wedge o = s \wedge o$), at least with regard to the positive and negative cover of the resulting states. As the positive and negative cover are the only factors considered in the preference functions that will be employed, multiple operator applications are not considered during the search as they cannot improve the value of a state.

It also follows that the order in which operators are applied does not alter the value of the resulting state. That is, for any state s and operators i and o, $s \wedge i \wedge o = s \wedge o \wedge i$, at least with respect to value. In consequence, each combination of operator applications need be considered once only. A search task for which the order in which operators are applied is not significant is called an unordered search task.

As each operator can be applied once only and order is not significant, this search task is equivalent to a subset selection problem—select the subset x of the available operators for which the application of the operators in x to IS results in a state with maximal value.

The search problem requires optimization search. A solution is any state that maximizes a value function called the preference function.

Two preference functions will be examined within this paper. The maximum consistent preference function allows only expressions that do not cover any negative cases. Of these expressions it favors the expression that covers the most positive cases. It can be expressed as

if $|neg(s)| > 0$ then $value(s) = -|neg(s)|$

else $value(s) = |pos(s)|$.

where s is the state being evaluated.

The Laplace preference function [1] trades-off accuracy against generality. It is defined as

$$value = \frac{|pos(s)| + 1}{|pos(s)| + |neg(s)| + no\_of\_classes}$$

The exact pruning rules that may be employed for a given search problem will depend upon the available means of identifying solutions and of identifying whether the application of a given operator below a given node can lead to a solution.

The current discussion assumes a context of optimization search. All states have values. The value of state s is denoted by $value(s)$. For convenience, the value of the state at node n will also be denoted by $value(n)$. All states with maximum values from the search space are acceptable solutions.

We also assume the ability to identify an optimistic value for any node. An optimistic value $optimistic(n)$ of node n is a value that is no higher than the maximum value of a state in the search space below n.

3 Admissible search strategies for unordered search

The OPUS^o search algorithm [18] has demonstrated the capacity to efficiently search many common classification learning search spaces.

Prior to OPUS^o, fixed-order search was used exclusively for admissible search in classification learning [14, 15, 16, 17]. Fixed-order search orders the search space in advance so that the proportion of the search space under each node is predetermined. In contrast, OPUS^o can reorder the search space to optimize the proportion of the search space under each node. OPUS^o outperforms fixed-order search by optimizing the proportion of the search space pruned by each pruning action without any significant computational or storage overhead [19].

OPUS^o operates by maintaining and manipulating at each node in the search tree a set of operators that may be applied below that node. These operators are called the active operators at the node.

At any time during the search, there will be a set of states $EXAMINED$ that have been encountered. At any time t, $BEST$, denotes the first state encountered during the search to time t that satisfies the constraint that for no state s in $EXAMINED$, $value(s) > value(BEST)$.

The Cover machine learning system utilizes the OPUS^o search algorithm within an AQ-like covering algorithm. In the default mode that is employed in the experimentation below, rule sets for multiple classes are learnt by considering each class in turn. As each class is considered it is treated as the positive class and the union of all other classes forms the negative class. The propositions considered take the form $A \neq v$, where A is an attribute and v is one of its allowable values. Unknown is treated as a distinct attribute value. Conjunctions of such propositions have equivalent expressive power to internal disjunctive expressions. The system currently supports only induction from categorical attribute-value data.

4 Inclusive and exclusive pruning rules

Previous unordered search algorithms [9, 12, 14, 15, 16, 18] have pruned the search tree by considering at each node the effect of applying each of the available operators. If one of the operators cannot lead to a solution, all potential solutions below the current node that include that operator can be pruned from the search tree (although OPUS^o and Feature Subset Selection are the only algorithms that actually prune all these nodes). This style of pruning shall be called exclusive pruning. Operators that can be excluded from consideration are pruned.

In the context of unordered search with each operator applied at most once, the exclusion of each operator from the search tree below a node halves that search tree as exactly half of the original search tree is reached via application of each operator.

This paper introduces another type of pruning, inclusive pruning. This approach identifies all search operators o such that if there is a solution below the current node then at least one such solution can be reached by application of o. It is possible to prune from the search tree all states that are not reached via application of any such operator o.

For unordered search with each operator applied at most once, each inclusive pruning action halves the search space below a node in the same manner as exclusive pruning, as exactly half of the search space is not reached via application of each operator.

Inclusive pruning is not restricted to unordered search during which each operator can be applied at most once. It is applicable to any unordered search.

Inclusive pruning cannot be trivially recast in terms of exclusive pruning because excluding states that are not reached by an operator o will not in general be equivalent to excluding all states that are reached by any identifiable set of other operators.

Exclusive pruning has the effect of decreasing the set of operators active at the current search node through deletion of the pruned operator.

Inclusive pruning with operator o to be included has the effect of replacing the state s for the current search node n with the state $s \wedge o$. Only o is removed from the set of the operators active at n.

The next four sections present a taxonomy of four basic categories of exclusive and inclusive pruning rule. Example rules of each category are defined for classification learning.

5 Obstructive operator pruning

An operator o is obstructive at node n if it must be the case that no solution can be reached through application of o below n. The search space reached through application of an obstructive operator can be pruned as doing so cannot prune any solutions from the search tree. Obstructive operator pruning is the form of exclusive pruning that can be applied to any unordered search task.

A version of optimistic pruning [10] can be defined as an obstructive operator pruning rule. This version of optimistic pruning enables the pruning from the search space below node n of all states reached via application of operator o if $optimistic(state(n) \wedge o) < value(BEST)$. If $optimistic(state(n) \wedge o) < value(BEST)$, then it follows that no state that can be reached via application of o from node n can have higher value than the best value encountered so far. It follows that no state in this area of the search tree can be a solution.

This version of optimistic pruning can be defined as follows:

> **Obstructive optimistic pruning**
> For any node n and operator o if $optimistic(state(n) \wedge o) < value(BEST)$ then prune all nodes reached via application of o from the search tree below n.

The effectiveness of this rule will depend on the precision of the optimistic value function.

For the preference functions defined above, the value of a node depends solely on how many positive and negative cases from the training set the expression at the node covers. Further, decreasing negative cover while holding positive cover constant increases value as does increasing positive cover while holding negative cover constant. As each search operator makes the expression at a node more specific, negative and positive cover can only decrease in the search space below a node. In this context, a simple optimistic evaluation function is obtained by determining the value of a node with the same positive cover as the current node but with a negative cover of zero. No node in the search tree below the current node can exceed this value.

For OPUS^o search, a more precise optimistic evaluation can be obtained by determining the minimum negative cover of any state in the search tree below node n. The state a reached by applying to $state(n)$ all operators active at n must be the most specific state in the search tree below n and so must have the minimum negative cover. It follows that no state in the search tree below n can exceed the value of a state that covers all of the positive cases covered by $state(n)$ and all of the negative cases covered by a.

The evaluation of optimistic pruning, below, employs this latter optimistic evaluation function.

6 Ineffectual operator pruning

An operator o is ineffectual at node n if it must be the case that if a solution can be reached through application of o below node n then there must also exist another solution below n that can be reached without application of o. Portions of the search space reached through application of an ineffectual operator can be pruned so long as only one solution is sought, as ineffectual operator pruning cannot prune all solutions from the search tree. It cannot, however, be applied if all solutions are sought, as it may result in some solutions being pruned from the search tree. Ineffectual operator pruning is a form of exclusive pruning.

For the preference functions employed in this research, any operator that does not reduce the negative cover of a state is ineffective [18] This can be justified as follows. Consider the application of operator o at node n. If $neg(state(n)) = neg(state(n) \wedge o)$, then for any set of operators a such that $state(n) \wedge o \wedge a$ (the result of applying all operators in a to $state(n) \wedge o$) is a solution, there must be a state $state(n) \wedge a$ (the result of applying all operators in a to $state(n)$) that covers the same cases as $state(n) \wedge o \wedge a$. It must cover the same negative cases because operator o cannot reduce the negative cover of a specialization of $state(n)$. It must cover the same positive cases. This is because, being a generalization of $state(n) \wedge o \wedge a$ it cannot cover fewer cases and, given the preference functions and that it covers the same negative cases, $state(n) \wedge o \wedge a$ could not be a solution if $state(n) \wedge a$ covered more positive cases.

From these considerations it is possible to derive the negative cover neutral pruning rule.

> **Negative cover neutral pruning**
> For any node n and operator o, if $neg(state(n)) = neg(state(n) \wedge o)$ then prune all potential solutions reached via application of o from the search tree below n.

For the maximum consistent and Laplace preference functions an operator o is also ineffective at node n if there is another operator a active at n such that $neg(state(n) \wedge a) \subseteq neg(state(n) \wedge o)$ and $pos(state(n) \wedge a) \supseteq pos(state(n) \wedge o)$. In this case, if there is a solution obtained via application of o and a below n then o must be redundant in the solution as its inclusion cannot decrease the negative cover of the resulting state as it cannot exclude any negative cases not also excluded by a. For any solution reached via o but not a there must be another solution obtained by substituting a for o, as the substitution of a for o will always result in a state with equal or lower negative cover and equal or higher positive cover.

From these considerations it is possible to derive the relative cover pruning rule.

> **Relative cover pruning**
> For any node n and operator o if there exists another operator a active at n such that $neg(state(n) \wedge a) \subseteq neg(state(n) \wedge o)$ and $pos(state(n) \wedge a) \supseteq pos(state(n) \wedge o)$ then prune all potential solutions reached via application of o from the search tree below n.

Another form of ineffectual operator pruning can be defined as a variant of optimistic pruning. If the optimistic value of the state reached via application of operator o at node n is equal to the value of b, the highest valued state encountered so far, then it follows that if a solution lies below $n \wedge o$ then b must also be a solution. Consequently, it is possible to prune the tree below n that can be reached via application of o without risk of pruning the only solution in the search tree.

The combination of this ineffectual operator pruning variant with the obstructive operator pruning variant of optimistic pruning defined above results in the following exclusive pruning rule:

> **Optimistic pruning**
> For any node n and operator o if $O(state(n) \wedge o) \leq V(b)$ then prune all potential solutions reached via application of o from the search tree below n.

7 Required operator inclusion

An operator o is required at node n if it must be the case that all solutions that lie in the search tree below node n can only be reached through application of o. The search space reached without application of a required operator can be pruned as doing so cannot prune any solutions from the search space. Required operator pruning is a form of inclusive pruning that may be applied to any unordered search task.

With respect to the maximum consistent preference function, an operator is required at node n if it is the only operator active at n that can exclude a negative case covered by $state(n)$. This is because, under the maximum consistent preference function, a state can only be a solution if it covers no negative cases. It follows that if there is a negative case x that is covered by $state(n)$, not covered by $state(n) \wedge o$ and is covered by $state(n) \wedge a$ for every operator a available at n other than o, then all solutions that lie below n must be reached via operator o. Consequently, all states below n that are not reached via o can be pruned from the search tree.

These considerations lead to the following pruning rule for the maximum consistent preference function.

> **Required operator inclusion**
> For any node n and operator o, if there exists a negative case x such that $state(n)$ covers x and $state(n) \wedge o$ does not cover x and there is no other operator $a \neq o$ such that $state(n) \wedge a$ does not cover x then it is possible to prune from the search tree all states below n that are not reached via o.

A less powerful version of this rule is applicable to the Laplace preference function. This version of the rule requires the additional constraint that the decrease in negative cover is not accompanied by any decrease in positive cover. In this case, for any state below n that is not reached via o, the application of o must decrease the negative cover but not the positive cover and so must increase the value of the state. It follows that o must be included in any solution that lies below n.

> **Required operator inclusion for the Laplace preference function**
> For any node n and operator o, if there exists a negative case x such that $state(n)$ covers x and $state(n) \wedge o$ does not cover x and there is no other operator $a \neq o$ such that $state(n) \wedge a$ does not cover x and $pos(state(n)) = pos(state(n) \wedge o)$ then it is possible to prune from the search tree all states below n that are not reached via o.

8 Non-obstructive operator inclusion

An operator o is non-obstructive at node n if it must be the case that if a solution can be reached without application of o below node n then there must also exist another solution below n that can be reached through application of o. The regions of the search tree reached without application of an non-obstructive operator can be pruned if a single solution is sought as doing so cannot prune all solutions from the search space. Non-obstructive operator pruning is a form of inclusive pruning.

Non-obstructive operator pruning is only applicable in search problems for which a single solution is sought. A search for all solutions cannot employ non-obstructive operator pruning as it may result in some solutions being pruned from the search space.

As already discussed, for the preference functions defined above, the value of a state depends solely upon how many positive and negative cases an expression covers. Further, decreasing negative cover while holding positive cover constant increases value as does increasing positive cover while holding negative cover constant. In this context, for any node n and operator o, if $pos(state(n)) = pos(state(n) \wedge o)$ then (as the operators cannot increase negative cover) it follows that for any solution x below n, $x \wedge o$ must also be a solution as $pos(x \wedge o) = pos(x)$ and $neg(x \wedge o) \leq neg(x)$. It follows that all states below n that are not reached via application of o can be pruned from the search tree without pruning a sole solution.

From these considerations, the following pruning rule can be derived.

> **Positive cover neutral inclusion**
> For any node n and operator o, if $pos(state(n)) = pos(state(n) \wedge o)$ then all states below n that are not reached via application of o can be pruned from the search tree.

As this rule will always apply whenever required operator inclusion for the Laplace preference function rule applies, the latter is not examined in the experimentation below.

9 Preventing redundant operators in solutions

The OPUS^o algorithm is capable of finding solutions that contain redundant operators. An operator o is redundant with respect to a solution s if there exists another solution x such that x is reached via a path identical to that of s except that it does not contain operator o.

The principle of Occam's Razor is frequently invoked to justify the selection of classifiers that correspond to non-redundant solutions in the search space under investigation. A non-redundant solution in this search space will be a conjunction of propositions c such that no expression produced through deleting a single conjunct from c that has the same value as c.

If non-redundant solutions are sought, the solution s found by OPUS^o can be simplified by the simple process of considering the deletion of each operator from s in turn. If the result r of deleting an operator is also a solution then s should be replaced by r.

It should be noted that the use of non-obstructive operator inclusion increases the probability of solutions containing redundant operators. Operators that are not necessary to reach a solution may be included within a partial solution on the basis that they cannot prevent the discovery of a solution. If solutions without redundant operators are sought then the above solution simplification process should be applied.

The cost of solution simplification is trivial compared with the potential savings from non-obstructive operator inclusion. Each application of non-obstructive operator inclusion halves the search space below the current state. If applied at the root of the search space, this reduces the search space by $2^{(n-1)}$ states, where n is the number of operators. By contrast, solution simplification only ever requires the evaluation of a number of potential solutions no greater than n.

10 Computational complexity

The worst case complexity for unordered search will always be exponential. Assuming that each operator can only be applied once, for n search operators there will be 2. states to explore. The worst case will occur when no pruning can be performed and all states must be evaluated. Thus the worst case the computational complexity is $O(2^n)$. It is only possible to analyze average case complexity with respect to a specific search task as it is only with respect to a specific task that it is possible to determine the frequency and distribution of the pruning actions that can be performed. The most straight forward manner in which to evaluate average case computational complexity with regard to a search task is to perform the task and to record the number of operations required. This is done below.

11 Experimental evaluation

The relative utility was evaluated of each of the following five pruning rules—optimistic pruning, negative cover neutral pruning, relative cover pruning, required operator inclusion and positive cover neutral inclusion. OPUS^o was employed to find classifiers that maximized each of the preference functions using first all the pruning rules and then each set of pruning rules formed by leaving one rule out. This enabled the relative effectiveness of each rule to be evaluated.

All fourteen categorical classification problems from the UCI machine learning repository [8] held at Deakin University due to previous experimental work, were considered. These are described in Table 1. Note that some of the data sets are not truly categorical in nature, in that some or all of the values are ordinal. However, these discrete ordinal values have been treated as categorical for the sake of this experimentation.

The following search algorithm was employed. This description is based upon that of Webb [5] with the addition of explicit pruning actions as appropriate.

Each node, n, in the search tree has associated with it three items of information:
n.state: the state that is associated with the node;
n.active: the set of operators to be explored in the sub-tree descending from the node; and
n.op: the operator that was applied to the parent node's state to create the current node's state.

A constant *min_val* sets a lower bound on the value of an acceptable solution. For the current application, *min_val* was set to the value of a state that covered no cases. Optimistic pruning was used to prune states the search space below which could not exceed *min_val*.

Algorithm: OPUS^o

```
1. Put the start node $n$ on a list called OPEN of unexpanded nodes. Set $s.active$ to the set of all operators, $o1, o2, ...on$. Set $n.state$ to the start state.
2. Set $BEST$, the best node examined so far, to $n$.
3. If $OPEN$ is empty, exit successfully with the solution represented by $BEST$.
4. Remove from $OPEN$ the node $n$, that maximizes $optimistic(n, n.active)$.
5. Initialize to $n.active$ a set containing those operators that are still under consideration, called $CUR$.
6. Initialize to {} a set of nodes, called $NEW$, that will contain the descendants of $n$ that are not pruned.
7. For every operator $o$ in $n.active$
   (a) Generate $n'$, a node for which $n'.state$ is set to the expression formed by application of $o$ to $n.state$.
   (b) If $value(n') > value(BEST)$
       i. Set $BEST$ to $n'$.
       ii. Remove from $OPEN$ all nodes $x$ such that $optimistic(x, x.active) \leq value(BEST)$ (optimistic pruning).
   (c) If $optimistic(n', CUR) > min\_val$ and $optimistic(n', CUR) > value(BEST)$ (optimistic pruning) and $neg(n') \subseteq neg(n)$ (negative cover neutral pruning)
       i. Add $n'$ to $NEW$.
       ii. Set $n'.op$ to $o$.
       else
       i. Remove $n'.op$ from $CUR$.
8. (positive cover neutral inclusion) For every node $n'$ in $NEW$
   (a) If $pos(n')$ equals $pos(n)$
       i. Remove $n'$ from $NEW$
       ii. Apply $n'.op$ to every state in $NEW$
       iii. Remove $n'.op$ from $CUR$.
       iv. While there is a state $x$ in $NEW$ such that $value(x) > value(BEST)$
       i. Set $BEST$ to $x$.
       ii. Remove from $OPEN$ all nodes $z$ such that $optimistic(z, z.active)$ is less than $value(BEST)$.
9. (required operator inclusion) For every node $n'$ in $NEW$
```

Table 1: Summary of experimental data sets.

Domain	Description	Values	Cases	Classes
Audiology	Medical diagnosis	162	226	24
Slovenian Breast Cancer (SBC)	Medical prognosis	57	286	2
House Votes 84	Predict political affiliation from US Senate voting record	48	435	2
Lenses	Lense prescription	12	24	3
Lymphography	Medical diagnosis	60	148	4
Monk 1	Artificial data	17	556	2
Monk 2	Artificial data	17	601	2
Monk 3	Artificial data	17	554	2
Multiplexer	Artificial data	22	500	2
Mushroom	Identify poisonous mushrooms	126	8124	2
Primary Tumor	Medical diagnosis	42	339	22
Soybean Large	Botanical diagnosis	135	307	19
Tic Tac Toe	Identify won or lost positions	27	958	2
Wisconsin Breast Cancer (WBC)	Medical diagnosis	91	699	2

```
    (a) If there is a case that is in $neg(x)$ of any other state $x$ in $NEW$ that is not in $neg(n')$
        i. Remove $n'$ from $NEW$
        ii. Apply $n'.op$ to every node in $NEW$
        iii. Remove $n'.op$ from $CUR$.
        iv. While there is a node $x$ in $NEW$ such that $value(x) > value(BEST)$
            A. Set $BEST$ to $x$.
            B. Remove from $OPEN$ all nodes $z$ such that $optimistic(z, z.active) \leq value(BEST)$.
10. (relative cover pruning) For every node $n'$ in $NEW$
    (a) If there is another node $x$ in $NEW$ such that $neg(x) \subseteq neg(n')$ and $pos(x) \supseteq pos(n')$
        i. Remove $n'$ from $NEW$
        ii. Remove $n'.op$ from $CUR$.
11. For every node $n'$ in $NEW$, selecting each time the node that minimizes $optimistic(n', CUR)$,
    (a) Remove $n'.op$ from $CUR$.
    (b) If $optimistic(n', CUR) > value(BEST)$,
        i. Set $n'.active$ to $CUR$.
        ii. Add $n'$ to $OPEN$.
12. Go to step 3.
```

This algorithm was employed to optimize each of the preference functions for each class. For each search, the size of the search space was $2^{|values|}$. This search space was systematically searched once for each class.

Table 2 presents the results for the maximum consistent preference function and Table 3 presents the results for the Laplace preference function. The column titled none indicates the number of states examined when all pruning rules are applied. For each of the pruning rules a column presents the number of states searched when all rules other than the named rule are employed. Each such tally is followed by an indication of the percentage by which the number of states examined is reduced by the addition of the rule to the other rules. This equals $(a*100)/p$, where a is the number of states explored using all rules and p is the number of states explored using all rules other than the one in question. A state is considered to have been examined if it is generated at step 7a of the algorithm. Note that required operator inclusion is not applicable with the Laplace preference function. Note also that results could not be obtained in a number of cases as the size of the list of open nodes, $OPEN$, exceeded the capacity of the SPARCstation 5 computer on which experimentation was conducted (the system's virtual memory grew to exceed that available: approximately 450Mb). These cases are indicated by a dash.

A number of results are notable. First, optimistic pruning is clearly the most important of the pruning rules. For every data set, a failure to employ optimistic pruning results in a substantial increase in the number of states examined. The remaining pruning rules appear to be ordered from most to least important as follows—required operator inclusion, positive cover neutral inclusion, negative cover neutral pruning and relative cover pruning. While application of the latter results in some minor decreases in the number of states examined, this in no case amounts to as much as a 10% decrease. It is also notable that required operator inclusion has greatest effect for the two data sets with the largest search spaces. This rule has little effect for many of the simple search problems, but greatly reduces the search space for complex search problems.

These results provide a slightly biased account of the importance of optimistic pruning, however, as OPUS^o is structured toward maximizing the effect of optimistic pruning actions. Specifically, OPUS^o

Table 2: Results for the maximum consistent preference function.

Data	**None**	**positive cover neutral inclusion**		**required operator inclusion**		**negative cover neutral pruning**		**optimistic pruning**		**relative cover pruning**	
	n	n	Δ	n	Δ	n	Δ	n	Δ	n	Δ
Audiology	3968	5703	30	3990	1	4097	3	4502	22	3987	0
House Votes 84	286	287	0	521	45	286	0	439921	100	286	0
Lenses	30	36	17	30	0	30	0	132	77	30	0
Lymphography	748	982	24	871	14	842	11	19808	96	772	3
Monk 1	289	312	7	330	12	303	5	10079	97	289	0
Monk 2	2636	2639	0	4320	39	2645	0	7209	63	2636	0
Monk 3	280	281	0	280	0	280	0	12688	98	280	0
Multiplexer	2559	2559	0	2761	7	2559	0	56943	96	2559	0
Mushroom	258	341	24	258	0	313	18	–		258	0
Primary Tumor	3957	6270	37	4552	13	4047	2	16988	77	3980	1
SBC	7198	8430	15	14296	50	8254	13	26675	73	7255	1
Soybean Large	3094	6561	53	3172	2	3190	3	12641	76	3140	1
Tic Tac Toe	2649	2649	0	2894	8	2653	0	859966	100	2649	0
WBC	165737	171182	3	421171	61	362338	54	–		165750	0
Mean			15		18		8		81		0.5

Table 3: Results for the Laplace preference function

Data	**None**	**positive cover neutral inclusion**		**negative cover neutral pruning**		**optimistic pruning**		**relative cover pruning**	
	n	n	Δ	n	Δ	n	Δ	n	Δ
Audiology	4093	6401	36	4620	11	6024	68	4097	0
House Votes 84	521	533	2	529	2	907459	100	527	1
Lenses	30	38	21	30	0	99	30	30	0
Lymphography	871	1130	23	1048	17	5137	83	871	0
Monk 1	330	357	8	344	4	18317	98	330	0
Monk 2	4320	4326	0	4329	0	6587	34	4320	0
Monk 3	280	281	0	280	0	19377	99	280	0
Multiplexer	2769	2769	0	2769	0	78613	96	2769	0
Mushrooms	258	386	33	309	17	–		258	0
Primary Tumor	6078	10745	43	6276	3	26895	77	6091	0
SBC	14315	17418	18	17586	19	147209	90	14413	1
Soybean Large	3027	7475	60	3258	7	35521	91	3075	2
Tic Tac Toe	2894	2894	0	2902	0	–		2894	0
WBC	421171	446992	6	1035790	59	–		421991	0
Mean			15		7.8		79		0.5

maximizes the proportion of the search space placed under states with low optimistic values on the basis that this maximizes the probability of pruning large sections of the search space. The relative effectiveness of each of the other pruning rules could also be emphasized by alternative strategies for organizing the search space. For example, the effectiveness of required inclusion could be increased by maximizing the proportion of the search space placed under states with high negative cover, as this would increase the scope for pruning large numbers of states through the application of this rule.

Nonetheless, it seems safe to conclude that where suitable optimistic evaluation functions are available, optimistic pruning is the single most important weapon in the fight against the exponential search space explosion in unordered search for classification learning.

It is clear, however, that the new inclusive pruning rules can play important supporting roles. There are some search tasks for which neither inclusive pruning rule results in a reduction in the amount of the search space explored. However, there are also some tasks for which each rule reduces the amount of the search space explored by over 50%. While inclusive pruning cannot replace exclusive pruning, it has an important role to playing in augmenting it.

12 Conclusion

Inclusive pruning is a new form of pruning for unordered search for which each pruning action delivers the same reduction in the search space (halving the search space below a state) as does exclusive pruning. While inclusive pruning has been demonstrated in the context of the OPUS^o search algorithm, it could also be applied to fixed operator ranking algorithms for unordered search.

Inclusive pruning rules for search in classification learning have been defined and demonstrated to deliver substantial savings in the proportion of the search space traversed (in the case of the Wisconsin Breast Cancer data, a reduction by 61%).

Inclusive pruning is by no means, however, restricted to application in machine learning. It is applicable to any unordered search problem. Previous applications of unordered search include subset selection [9, 12, 20], feature selection [4], truth maintenance [3], hitting sets and diagnosis [13].

The successful application of admissible search to every categorical classification task in the UCI repository demonstrates that admissible search is a viable tool for classification learning. This is not to argue that all classification learning should employ admissible search. Rather, it suggests that admissible search should be applied where it may be of value, for example, when admissible evaluation of alternative learning biases is desired. When specific learning biases are to be applied for particular purposes, it may, however, be desirable to apply heuristic algorithms that closely approximate the desired bias.

Even within such heuristic algorithms, however, there may still be room for the application of pruning heuristics. The potential importance of this is emphasized by previous experimentation that has demonstrated that admissible search with appropriate pruning rules is frequently more efficient than common heuristic search algorithms that do not employ these pruning rules [19].

This paper has presented a new form pruning for unordered search and demonstrated that its use can greatly reduce the proportion of the search space examined in search for classification learning.

References

[1] Peter Clark and T. Niblett. Induction in noisy domains. In I. Bratko and N. Lavrač (editors), *Progress in Machine Learning*, pages 11–30. Sigma Press, Wilmslow, 1987.

[2] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, Volume 3, pages 261–284, 1989.

[3] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, Volume 28, pages 127–162, 1986.

[4] M. Ichini and J. Sklansky. Optimum feature selection by zero-one integer programming. *IEEE Transactions on Systems, Man, and Cybernetics*, Volume SMC-14, No 5, pages 737–746, 1984 (September/October).

[5] Ryszard S. Michalski. A theory and methodology of inductive learning. In Ryszard S. Michalski, Jaime G. Carbonell and Tom M. Mitchell (editors), *Machine Learning: An Artificial Intelligence Approach*, pages 83–129. Springer-Verlag, Berlin, 1983.

[6] Steven Minton and Ian Underwood. Small is beautiful: A brute-force approach to learning first-order formulas. In *AAAI-94*, 1994.

[7] Bernard M. E. Moret and Henry D. Shapiro. On minimizing a set of tests. *SIAM Journal on Scientific and Statistical Computing*, Volume 6, Number 4, pages 983–1003, 1985.

[8] Murphy, P. and D. Aha. UCI Repository of Machine Learning Datasets. 1994.

[9] P. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, Volume C-26, pages 917–922, 1977.

[10] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, Reading, Mass., 1984.

[11] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, Volume 5, pages 239–266, 1990.

[12] M.S. Ridout. An improved branch and bound algorithm for feature subset selection. *Applied Statistics*, pages 139–147, 1987.

[13] R. Rymon. Search through systematic set enumeration. In *Proceedings KR-92*, pages 268–275, Cambridge, MA, 1992.

[14] R. Rymon. An SE-tree based characterization of the induction problem. In *Proceedings of the 1993 International Conference on Machine Learning*, San Mateo, CA, 1993. Morgan Kaufmann.

[15] J. C. Schlimmer. Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. In *Proceedings of the 1993 International Conference on Machine Learning*, pages 284–290, San Mateo, CA, 1993. Morgan Kaufmann.

[16] Richard Segal and Oren Etzioni. Learning decision lists using homogeneous rules. In *AAAI-94*, Seattle, WA, 1994. AAAI press.

[17] Geoffrey I. Webb. Techniques for efficient empirical induction. In C. J. Barter and M. J. Brooks (editors), *AI'88 – Proceedings of the Third Australian Joint Conference on Artificial Intelligence*, pages 225–239, Adelaide, 1990. Springer-Verlag.

[18] Geoffrey I. Webb. Systematic search for categorical attribute-value data-driven machine learning. In Chris Rowles, Huan Liu and Norman Foo (editors), *AI'93 – Proceedings of the Sixth Australian Joint Conference on Artificial Intelligence*, pages 342–347, Melbourne, 1993. World Scientific.

[19] Geoffrey I. Webb. OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, Volume 3, pages 431–465, 1995.

[20] B Yu and Yuan Baozong. A dynamic selection algorithm for globally optimal subsets. *Engineering Application Artificial Intelligence*, Volume 5. No. 5, pages 457–462, 1992.