

# 2 Scaling to Domains with Irrelevant Features

Pat Langley and Stephanie Sage

## 2.1 Introduction and Background

One of the central problems in machine induction involves discriminating between features that are relevant to the target concept and ones that are irrelevant. Presumably, many real-world learning tasks contain large numbers of irrelevant terms, and for such tasks, one would prefer to use algorithms that scale well along this dimension. More specifically, one would like the number of training instances required to reach a given level of accuracy to grow slowly with increasing numbers of irrelevant features.

Some theoretical results of this sort are available for particular induction algorithms. For instance, Littlestone (1987) has shown that the number of classification errors made by his WINNOW method grows only logarithmically with the number of irrelevant features. However, his results were worst case and focused on linearly separable concepts. Pazzani and Sarrett (1992) have presented an average-case analysis for the WHOLIST algorithm, and Langley and Iba (1993) have reported similar analyses for simple Bayesian classifiers and nearest neighbor algorithms. Taken together, these analyses suggest that, when many irrelevant attributes are present, methods for inducing logical descriptions like WHOLIST scale the best, probabilistic methods fare next best, and nearest neighbor methods scale the worst.

However, both Pazzani and Sarrett's and Langley and Iba's analyses held only for conjunctive concepts, and most of the algorithms they analyzed make strong assumptions about the nature of the target concept. To see if their conclusions carried over to more robust algorithms and to more challenging target concepts, we designed an experimental study to extend the previous results. The next section reviews two induction algorithms and motivates their selection for this comparative study. The subsequent section describes our experimental design before we go on to report and discuss the results we obtained. After this, we describe a third algorithm designed to overcome weaknesses revealed in these studies, followed by additional experiments that evaluate its behavior. Finally, we discuss related work on irrelevant attributes and some directions for future research.

## 2.2 Review of Two Induction Algorithms

We decided to examine the behavior of two well-known induction algorithms – Quinlan’s (1993) C4.5 and the simple nearest neighbor method (Cover & Hart, 1967; Aha, Kibler, & Albert, 1991) – in domains involving increasing numbers of irrelevant features. Let us briefly review these algorithms.

Quinlan’s C4.5 represents instances as attribute-value pairs. Classification consists of sorting a test instance through a *decision tree*. Each node in the tree specifies one attribute, with the branches from that node corresponding to the values of the attribute. Starting at the root node, the test instance is sorted down the branch for its value on the given attribute. The leaves of the tree are labeled with class names, so that once an instance is sorted to a leaf, the label is used to predict its class.

A decision tree is constructed for a set of training data by first choosing the attribute that best discriminates among the classes evident in the data. In particular, C4.5 uses a measure of information gain to make the selection, though other metrics are possible. The system uses the selected attribute as the root of the decision tree and creates a branch for each value. C4.5 then sorts the observed instances down the branches according to their values on the root attribute. Finally, the algorithm calls on itself recursively to build decision trees rooted at each of the children, terminating when all instances at a given node have the same class. This class becomes the label for the leaf.

The nearest neighbor algorithm also represents instances as attribute-value pairs. In this case, classification begins by measuring the distance of a test instance from each of a set of stored instances. In principle, one can use any distance measure; in a Boolean domain, a natural measure is the number of attribute values that differ between the test instance and the stored instance (sometimes called the *city-block* metric). The class of the nearest stored instance is used to predict the class of the test instance. Learning consists of storing each training instance in memory.

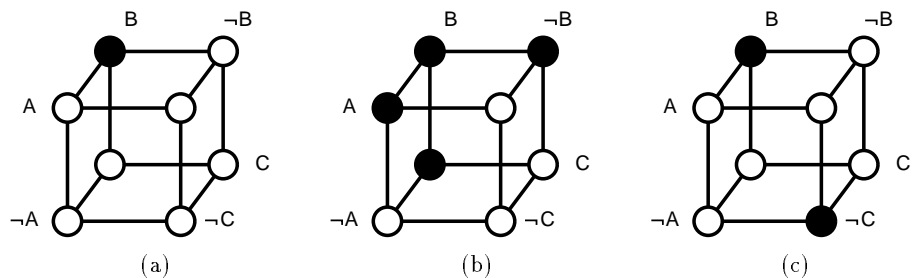
This choice of algorithms let us contrast a system that uses a subset of (presumably) relevant features to classify data (C4.5) with a system that uses information about all features, including irrelevant ones (nearest neighbor). We predicted that classification accuracy for the two methods would be comparable when the number of irrelevant features was small. We believed that C4.5 would scale well with the addition of irrelevant attributes,

as the evaluation function would tend to select relevant attributes and ignore irrelevant ones. However, we believed that performance of the nearest neighbor method would degrade as the number of irrelevant dimensions increased. Far from ignoring irrelevant, this method makes equal use of information about all attributes.

### 2.3 Experimental Design

One of the primary goals of machine learning research is to match real-world induction problems with appropriate techniques for solving them. In order to make such matches, one must understand how the characteristics of domains and algorithms interact to produce good (or poor) behavior. Artificial domains are an invaluable tool for helping us to achieve this end. By systematically varying domain characteristics, such as the number of irrelevant attributes, we can measure the effect of these variables on behavior. Thus, for our purposes the domains of choice were artificial. We focused on two-class domains involving three Boolean attributes. From the  $2^3 = 256$  possible concepts in this instance space, we selected three concepts to represent a range of expected difficulty. At first glance, this may seem to be a tiny sample of the possible concepts in the space. However, the 256 concepts can be collapsed into only 19 equivalence classes based on attribute and value name substitution (Schlimmer, 1987). By class name substitution we can further reduce this number to 12. Since we wanted to hold the number of relevant dimensions constant as we added irrelevant, we retained only those concept types with exactly three relevant features. This left us with eight concept types from which to choose our representative sample. Below we describe the three concept types we selected.

The first of the concepts we chose was the simple three-feature conjunct ( $A \wedge B \wedge C$ ). Another concept, known as an “m of n” concept, can be stated as  $((A \wedge B) \vee (A \wedge C) \vee (B \wedge C))$ . Although linearly separable, this concept is not conjunctive and we expected it to represent a somewhat more difficult problem than the first. As our third task, we chose  $((A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C))$ . This case resembles a parity concept, in that none of the relevant features ( $A$ ,  $B$ , and  $C$ ) in isolation can be distinguished from irrelevant ones. Figure 2.1 displays the extension of these concepts in the absence of irrelevant attributes.

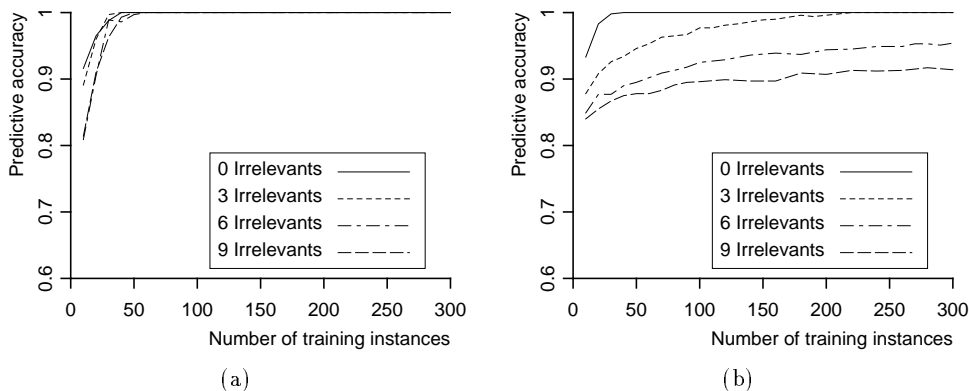
**Figure 2.1**

Extensional definitions of the three target concepts used in our experiments, after removing irrelevant features, corresponding to the Boolean combinations (a)  $((A \wedge B \wedge C))$ , (b)  $((A \wedge B) \vee (A \wedge C) \vee (B \wedge C))$ , and (c)  $((A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C))$ . A black sphere represents a positive instance, whereas a white sphere indicates a negative one.

We examined the behavior of C4.5 and the nearest neighbor method in the presence of zero through nine irrelevant dimensions for each of the three concept types. Thus, we tested each algorithm in 30 different experimental conditions. For each condition, we trained the two algorithms until they reached at least 95% accuracy on a test set of 200 instances. We required that this level of accuracy be maintained for 50 subsequent instances before declaring that the accuracy criterion was met (at the first of those 50 instances).

We ran C4.5 without pruning and with the  $m$  parameter set to one; default settings were used for all other parameters. These choices were based on pilot studies indicating that these conditions produced the best learning curves. The difference between the pruned and unpruned versions of C4.5 was small. The nearest neighbor algorithm was applied using the city-block distance metric to determine the nearest stored instance.

Each algorithm was tested on the same training and test data for a given experimental condition. For each condition, we created twenty training sets and one test set using random sampling with replacement from the instance space. Thus, the data were uniformly distributed and the relative frequencies of positive and negative instances reflected their distribution in the instance space. An irrelevant feature took on a value of 0 or 1 with equal likelihood. Each data point reported in the next section is an average over twenty runs.

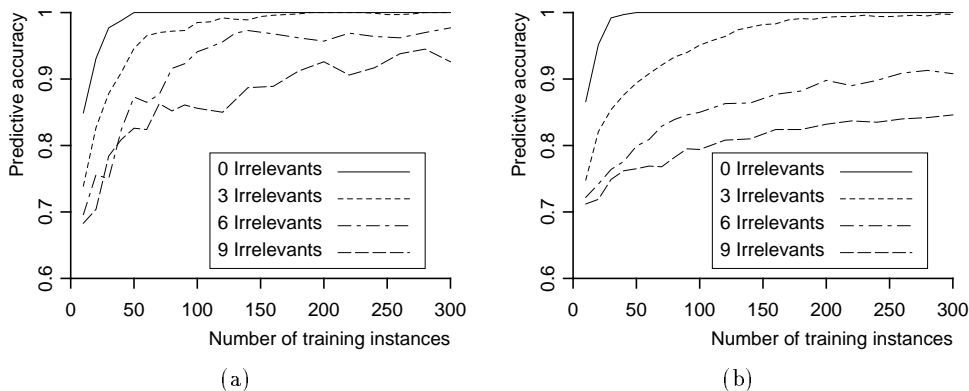
**Figure 2.2**

Learning curves for different numbers of irrelevant attributes, on the Boolean target concept  $((A \wedge B \wedge C))$ , for (a) decision-tree induction using C4.5 and (b) the simple nearest neighbor algorithm.

## 2.4 Experimental Results

We first examined the learning curves generated by the two induction algorithms. Since the results for the two concepts  $((A \wedge B \wedge C))$  and  $((A \wedge B) \vee (A \wedge C) \vee (B \wedge C))$  were virtually identical, we have omitted graphs for the latter. Learning curves for the two other concepts appear in figure 2.2 and figure 2.3. The four curves in each graph depict behavior when the target concept contains zero, three, six, and nine irrelevant features. The results for one, two, four, five, seven, and eight irrelevants are not presented for the sake of readability, but they are consistent with the curves shown. Although no error bars are displayed, we did compute the 95% confidence intervals for predictive accuracy. These ranged from 0.022 to 0.061 after ten instances and decreased, roughly monotonically, toward zero with increasing numbers of instances.

Figure 2.2 shows the learning curves for each algorithm on the concept  $((A \wedge B \wedge C))$ . As expected, the two methods perform comparably when no irrelevant features are present, but as irrelevant dimensions are added, the performance of the nearest neighbor method degrades significantly. These results are consistent with our understanding of the C4.5 algorithm, which prefers more discriminating attributes for use in its decision tree. For this target concept, each of the three relevant features is more discriminating than the irrelevant features, and their combination is sufficient to discrim-



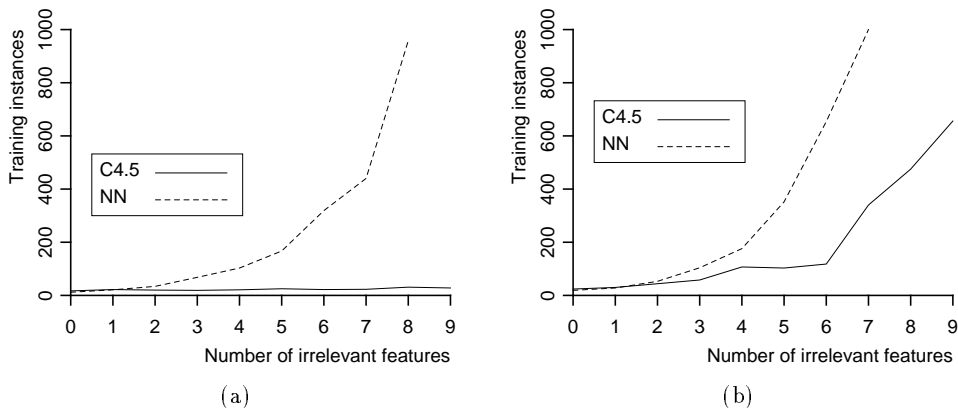
**Figure 2.3**

Learning curves for different numbers of irrelevant attributes, on the Boolean target concept  $((A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C))$ , for (a) decision-tree induction using C4.5 and (b) the simple nearest neighbor algorithm.

inate the classes completely. As a result, the irrelevant dimensions are not included in the decision tree and are ignored in the classification process.

Figure 2.3 presents the learning curves for the parity-like concept. In this case, C4.5 degrades almost as severely as nearest neighbor. Apparently, the nature of the target concept interacts with the number of irrelevant variables. An examination of this target concept reveals that only the conjunction of the three relevant features is sufficient to discriminate the classes. Starting at the root of the tree, there is no information gain no matter which single feature is tested. As a result, a truly irrelevant feature is as likely to be included in the decision tree as a relevant one, effectively reducing the amount of training data lower in the tree. This can lead to the omission of relevant features in the final tree. Indeed, as more irrelevant dimensions are added to such a target concept, the probability of adding an irrelevant test to the decision tree increases.

Although the learning curves described above reveal the effect of irrelevant attributes on the two algorithms' behavior, they do not explicitly relate a performance measure to this factor. However, we can use them to generate such a relationship in a straightforward manner. First, one selects a desired level of classification accuracy. Next, one finds the number of training instances at which each learning curve first exceeds that level and maintains it for a specified number of instances. Finally, one plots this number as a function of the number of irrelevant attributes, producing a *scaling* curve.

**Figure 2.4**

The number of training instances required for decision-tree induction and nearest neighbor to reach 95% accuracy on a separate test set, as a function of the number of irrelevant features, for the target concepts (a)  $(A \wedge B \wedge C)$  and (b)  $((A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C))$ .

The shape of this curve (e.g. linear or exponential) corresponds roughly to the notion of *sample complexity* in computational learning theory.

Figure 2.4(a) reports the scaling curve for the target concept  $(A \wedge B \wedge C)$  at the 95% level of accuracy. We required that this accuracy be maintained for at least 50 instances after the criterion level is first reached. The resulting curve for C4.5 appears constant or at most logarithmic, whereas that for nearest neighbor seems to grow exponentially with the number of irrelevant features. Similar results hold for the “m of n” target concept  $((A \wedge B) \vee (A \wedge C) \vee (B \wedge C))$ , which we have not shown.

These results contrast sharply with those shown in figure 2.4(b) for the target concept  $((A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C))$ . Here the number of training cases needed to reach a given level of accuracy appears to grow exponentially with the number of irrelevants for both algorithms, though the constant for C4.5 is smaller. This suggested that extending the C4.5 curves for the other concept might also reveal exponential growth, but additional runs on the conjunctive concept with 16 irrelevant features showed no detectable increase over the results for nine irrelevants.

In summary, the experiment revealed a clear difference in the effect of irrelevant attributes on the behavior of the two induction algorithms. The rate of learning for the nearest neighbor method decreases with the number of irrelevant dimensions, regardless of the target concept. In contrast, the

effect of irrelevant attributes on C4.5 depends on the nature of the target concept. As long as single relevant features discriminate among the classes, C4.5's learning rate is unaffected by the introduction of irrelevant dimensions. However, when features interact, that is, when none of the relevant attributes in isolation has any power to discriminate the classes, the rate of learning slows significantly.

## 2.5 Induction of Oblivious Decision Trees

The reason for C4.5's difficulty with the parity-like concept is that it builds its decision tree in a top-down manner, adding the best single feature at each node. Due to the interaction of the relevant features in this concept, none of them distinguishes the classes any better than an irrelevant feature. As a result, this method is likely to add irrelevant features to the tree.

One way to overcome the problem of interacting features is to allow tests of more than one feature at a node. However, this results in an exponential increase in the number of tests to consider. Moreover, when many of the features are irrelevant, this leads to much wasted effort. An alternative approach starts with a tree containing all of the features and removes the irrelevant ones. In contrast to the top-down approach, it is easy to identify relevant features since removing a single one, even if it interacts with other features, reduces the predictive accuracy of the tree, but removing an irrelevant feature does not.

The latter insight provides the basis for our algorithm. In order to implement this method, we must first specify how to construct the initial tree. One could apply a variant of C4.5 to build a decision tree containing all the features. However, pruning each subtree separately would be computationally expensive. Also, it is awkward to remove irrelevant features tested in the middle of the tree. Although this could be achieved by translating the tree into rules and then dropping tests from the rules, there is a much simpler method.

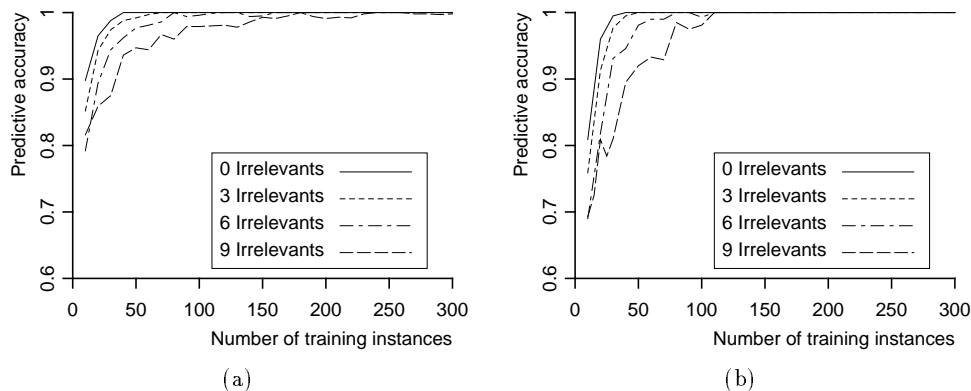
We describe our approach in terms of an abstract data structure called an *oblivious* decision tree (Kohavi, 1994), which tests the same feature across an entire level of the tree. For example, we might test feature  $a$  at the top node, feature  $b$  at each of its children, feature  $c$  at each of its grandchildren, and so on. The beauty of this data structure is that it allows one to remove an entire level of the tree at one time. In addition, the order of the tests

is arbitrary, so irrelevant features that are embedded in the tree can be “sliced” out without affecting tests of relevant features lower in the tree. For purposes of prediction, a probability distribution over the classes may be associated with each leaf.

For our purposes, classification in an oblivious decision tree consists of sorting an instance to a leaf and predicting the most likely class associated with that leaf. If no class distribution exists at a given leaf, then the nearest leaf with a distribution is used. In case of ties, the distributions from the nearest leaves are combined. Note that if all of the features are included in the tree, this classification scheme is functionally equivalent to the simple nearest neighbor method. In fact, the algorithm we are about to describe is implemented as a variant of nearest neighbor in which irrelevant features are ignored in computing the distance metric. However, we will continue to use the oblivious decision tree analogy throughout this paper, since we find it more helpful for descriptive purposes.

OBLIVION carries out greedy backward elimination of features from oblivious decision trees. The method begins with a full oblivious tree (containing all features) and obtains a conservative estimate of its classification accuracy using  $n$ -way cross validation on the training set. The system then removes each attribute in turn, estimates the accuracy of the resulting tree in each case, and selects the most accurate. If this best tree makes no more errors than the current one, OBLIVION replaces the current tree with the best one and continues the process. On each step the algorithm tentatively prunes each of the remaining features, selects the best, and generates a new tree with one fewer attribute. This continues until the accuracy of the best pruned tree is less than the accuracy of the current one. OBLIVION’s time complexity is polynomial in the number of features, growing with the square of this factor.

We expected that OBLIVION would perform comparably to C4.5 on the conjunctive concept  $((A \wedge B \wedge C))$ , but that it would learn the parity-like concept  $((A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C))$  more quickly. Since the system starts with all the relevant features and only prunes ones that are irrelevant, it should not suffer in the presence of interacting relevant features. In the next section, we examine and compare the behavior of OBLIVION with that of C4.5 and nearest neighbor.



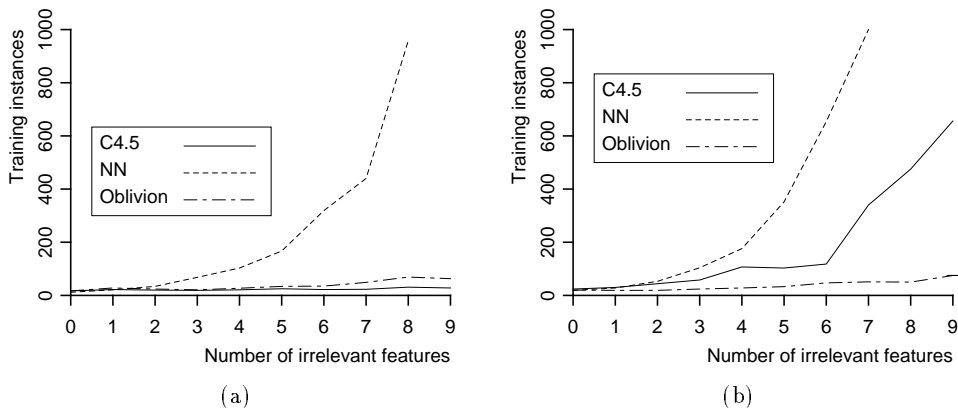
**Figure 2.5**

Learning curves for different numbers of irrelevant attributes using oblivious decision tree induction with OBLIVION, on the Boolean target concepts (a)  $(A \wedge B \wedge C)$  and (b)  $((A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C))$ .

## 2.6 Experimental Results with OBLIVION

We obtained learning curves and scaling curves for OBLIVION in much the same manner as for the other induction methods. In this case, we tested the algorithm only on the conjunctive and parity-like concepts, omitting the “m of n” concept. Thus, we tested OBLIVION in twenty experimental conditions generated by adding zero through nine irrelevants to the two basic target concepts.

Figure 2.5 presents the resulting learning curves for zero, three, six, and nine irrelevants, based on the average of twenty runs. As in the previous section, error bars are omitted for readability, but the 95% confidence intervals ranged from 0.020 to 0.051 at the outset of learning, decreasing roughly monotonically to zero in each case. As expected, OBLIVION improves upon the performance of C4.5 on the parity-like concept. Although the effect of irrelevant features is still visible, all four curves reach the criterion level more quickly. Surprisingly, C4.5 outperforms OBLIVION on the conjunctive concept. Since the pattern of results is different on this concept than on the parity-like concept, we believe the poor performance is not directly due to the presence of irrelevants, but the explanation remains unclear. In any case, OBLIVION still reaches high levels of accuracy within 50 instances, even with nine irrelevants, so this discrepancy is not of great concern.

**Figure 2.6**

The number of training instances required for decision-tree induction, nearest neighbor, and OBLIVION to reach 95% accuracy on a separate test set, as a function of the number of irrelevant features, for the target concepts (a)  $(A \wedge B \wedge C)$  and (b)  $((A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C))$ .

Figure 2.6 includes scaling curves for OBLIVION along with those for C4.5 and nearest neighbor. Again, we present the number of instances required to reach 95% accuracy on a separate test set as a function of the number of irrelevant features. These results demonstrate the superiority of OBLIVION over C4.5 in scaling to increasing numbers of irrelevant features on the parity-like concept. The difference between C4.5 and OBLIVION on the conjunctive concept is comparatively small. Indeed, the sample complexity for OBLIVION appears nearly linear on this concept. In contrast, C4.5's sample complexity on the parity-like concept is clearly exponential.

In the absence of knowledge about the prevalence of irrelevant or interacting features in a given domain, OBLIVION appears to be the algorithm of choice. Moreover, it seems likely that some natural domains will have interaction among their features. OBLIVION's improved behavior over C4.5 in the presence of interacting features outweighs its slightly poorer showing when no interactions are present. The algorithm's ability to scale well to increasing numbers of irrelevant features, regardless of the presence of interaction, is a clear benefit.

## 2.7 Related Work

Other researchers have addressed the issue of induction in the presence of irrelevant features. Early approaches to this problem emphasized *filtering* methods. Under this model, feature selection is distinct from the process of constructing an intensional description (e.g. a decision tree). The relevant features are first selected and then passed to an induction algorithm. Methods taking this approach include Almuallim and Dietterich's (1990) FOCUS algorithm and a related technique by Schlimmer (1993), as well as Kira and Rendell's (1992) RELIEF. Both FOCUS and RELIEF pass the selected features to a decision-tree algorithm, but one can use any induction technique at this stage. Cardie (1993) described an approach that selects a set of features for nearest neighbor retrieval, and Kubat, Flotzinger, and Pfurtscheller (1993) used a similar filtering scheme with a naive Bayesian classifier.

In contrast, much of the recent work has applied a *wrapper* model to the problem of feature selection. First described as a general framework by John, Kohavi, and Pfleger (1994), this approach uses the same algorithm to select features as it does to construct the intensional description. Methods combining the wrapper model with decision-tree induction include those of John et al (1994) and Caruana and Freitag (1994), but again, one can use any induction algorithm. Moore and Lee (1994), Skalak (1994), Townsend-Weber and Kibler (1994), and Aha and Bankert (1994) have implemented the wrapper model with nearest neighbor methods. OBLIVION is another example of a wrapper system, but it combines characteristics of both decision tree and nearest neighbor techniques.

Another broad class of methods retains all attributes, but places weights on them, so that some influence the classification process more than others. Examples of this approach include Rumelhart's (1986) backpropagation algorithm for determining weights in a multi-layer neural network, Aha's (1990) method for weighting features in the nearest neighbor framework, and Gennari, Langley, and Fisher's (1989) use of conditional probabilities in a concept hierarchy. Such techniques have more representational power than feature selection methods, but they also have greater potential for overfitting the training data, and it remains unclear which scheme gives the best scaling to domains with many irrelevant features.

## 2.8 Directions for Future Work

Clearly, the experimental results we have presented are preliminary and must be treated with caution. In future work, we hope to replicate our results in the presence of noise and on a broader range of target concepts, including ones that incorporate more relevant features, alternative Boolean combinations, and numeric attributes. In addition, we would like to explore alternative schemes for searching the space of oblivious decision trees that may further improve OBLIVION's ability to scale to domains with many irrelevant features. We also hope to supplement our studies of artificial domains by applying OBLIVION to some natural domains.

Another direction for future work involves comparing our approach with other methods for feature selection. In particular, the current proliferation of wrapper models, for both decision-tree and nearest neighbor methods, deserves our attention. We should also study the behavior of attribute-weighting schemes. Examining the relations among these approaches and our own may lead to further development and improvement of our algorithm.

Despite the work that remains, we believe our initial studies provide a foundation for evaluating methods for handling irrelevant features. Our methodology for comparing these algorithms, specifically our use of scaling curves, allows us to directly measure their performance as a function of the number of irrelevants. We have applied this methodology to compare two standard approaches to induction along with a new algorithm designed to overcome their limitations, thus providing a context for future comparative studies along the same lines.

**Acknowledgments** Thanks to David Aha, Jeff Schlimmer, Russ Greiner, and Bharat Rao for discussions that improved the work reported in this paper, and to Siemens Corporate Research for providing space and facilities during a visit by the authors. This research was supported in part by ONR Grant No. N00014-94-1-0505, and in part by AFOSR Grant No. F49620-94-1-0118.

┌

└