# Efficient Algorithms for Identifying Relevant Features

Hussein Almuallim
Thomas G. Dietterich
Department of Computer Science
Oregon State University
Corvallis, OR 97331-3202

92-30-03

# Efficient Algorithms for Identifying Relevant Features

Hussein Almuallim
Thomas G. Dietterich
303 Dearborn Hall
Department of Computer Science
Oregon State University
Corvallis, OR 97331-3202
almualh@cs.orst.edu
tgd@cs.orst.edu
Phone: 503-737-5566
FAX: 503-737-3014

## Abstract

This paper describes efficient methods for exact and approximate implementation of the MIN-FEATURES bias, which prefers consistent hypotheses definable over as few features as possible. This bias is useful for learning domains where many irrelevant features are present in the training data.

We first introduce FOCUS-2, a new algorithm that exactly implements the MIN-FEATURES bias. This algorithm is empirically shown to be substantially faster than the FOCUS algorithm previously given in [Almuallim and Dietterich 91]. We then introduce the Mutual-Information-Greedy, Simple-Greedy and Weighted-Greedy algorithms, which apply efficient heuristics for approximating the MIN-FEATURES bias. These algorithms employ greedy heuristics that trade optimality for computational efficiency. Experimental studies show that the learning performance of ID3 is greatly improved when these algorithms are used to preprocess the training data by eliminating the irrelevant features from ID3's consideration. In particular, the Weighted-Greedy algorithm provides an excellent and efficient approximation of the MIN-FEATURES bias.

# 1 Introduction

In many inductive learning applications, one has to deal with training data that contain many features that are irrelevant to the target concept being learned. In these domains, an appropriate bias is the MIN-FEATURES bias, which prefers any consistent hypothesis definable over as few features as possible. Previous work [Almuallim and Dietterich 91] showed that this simple bias is strong enough to yield polynomial sample complexity. The same study showed that—contrary to expectations—the performance of conventional inductive learning algorithms such as ID3 [Quinlan 86] and FRINGE [Pagallo and Haussler 90] is seriously reduced by the presence of irrelevant features. These results suggested that one should not rely on these algorithms to filter out irrelevant features. Instead, some technique should be employed to eliminate irrelevant features and focus the learning algorithm on the relevant ones.

Given a set of training examples of an unknown target concept, the task for any algorithm implementing the MIN-FEATURES bias is to find the smallest subset of the given features that permits a consistent hypothesis to be defined. In previous work [Almuallim and Dietterich 91], an algorithm called FOCUS was presented that exactly implements the MIN-FEATURES bias. However, the worst-case running time of this algorithm is exponential in the number of relevant features. The goal of this paper is to describe more efficient algorithms for exact and approximate implementation of the MIN-FEATURES bias.

Specifically, the paper first introduces FOCUS-2, a new algorithm that exactly implements the MIN-FEATURES bias. This algorithm is empirically shown to be substantially faster than FOCUS. We then introduce the Mutual-Information-Greedy, Simple-Greedy and Weighted-Greedy algorithms, which apply efficient heuristics for approximating the MIN-FEATURES bias. Unlike FOCUS-2, these algorithms employ *greedy* heuristics that trade optimality for computational efficiency. Experimental studies comparing these heuristics to FOCUS, ID3, and FRINGE show that the Weighted-Greedy algorithm provides an excellent and efficient approximation of the MIN-FEATURES bias.

The task of selecting a subset of the available features that meets a given criterion has long been known in the field of pattern recognition as the problem of "feature selection" or "dimensionality reduction." However, most of the work in this area seeks to enhance the computational efficiency of particular classifiers while leaving their accuracy unaffected, whereas the goal of this paper is to improve the accuracy of the classifier by selecting the minimal number of features.

The typical case studied in pattern recognition involves a classifier that is capable of performing quite well without feature selection (i.e., using all of the available features). However, for ease of hardware implementation and speed of processing, it is necessary to reduce the number of features considered by the classifier. Generally, the classifiers studied in pattern recognition have the so-called monotonicity property ([Narendra and Fukunaga 77]) that as the number of features is reduced, the accuracy decreases. The goal of feature selection is to eliminate as many features as possible without significantly degrading performance.

Most feature selection criteria in pattern recognition are defined with respect to a specific classifier or group of classifiers. For example, [Kittler 80] show methods for selecting a small subset of features that optimizes the expected error of the nearest neighbor classifier. Similar work has addressed feature selection for the Box classifier [Ichino and Sklansky 84a], the

linear classifier [Ichino and Sklansky 84b] and the Bayes classifier [Queiros and Gelsma 84]. Other work (aimed at removing feature redundancy when features are highly correlated) is based on performing a principal components analysis to find a reduced set of new uncorrelated features defined by *combining* the original features using the eigenvectors [Morgera 86, Mucciardi and Gose 71]. To our knowledge, the problem of finding the smallest subset of Boolean features that is sufficient to construct a consistent hypothesis (regardless of the *form* of the hypothesis)—which is the topic of this paper—has not been addressed.

# 2 Background

Let $\{x_1, x_2, \cdots x_n\}$ be a set of $n$ Boolean features, and let $U_n$ denote the set of all possible assignments to these features. A *concept* $c$ is a subset of $U_n$ (i.e., all positive instances of $c$). An *example* for a concept $c$ is a pair $\langle X, class \rangle$, where $X \in U_n$ and *class* is $+$ if $X \in c$ and $-$ otherwise. A *sample* is a set of examples drawn at random from $U_n$.

Given a training sample and a set of features $Q$, a *sufficiency test* is a procedure for checking whether $Q$ is sufficient to form a consistent hypothesis. The sufficiency test can be implemented simply by checking whether the sample contains a pair $\langle X_1, + \rangle$ and $\langle X_2, - \rangle$ of positive and negative examples such that $X_1$ and $X_2$ have the same values for all the features in $Q$. If such a pair appears, then $Q$ cannot discriminate all of the positive examples from all of the negative examples. In general, $Q$ is a sufficient set if and only if no such a pair appears in the training sample.

For a pair of examples $\langle X_1, + \rangle$ and $\langle X_2, - \rangle$, we define a *conflict* generated from this pair as an $n$-bit vector $a = \langle a_1 a_2 \cdots a_n \rangle$ where $a_i = 1$ if $X_1$ and $X_2$ have different values for the feature $x_i$ and 0 otherwise. We will say that $a$ is *explained* by $x_i$ if and only if $a_i = 1$. Using this terminology, a set $Q$ of features is sufficient to construct a hypothesis consistent with a given training sample if and only if every conflict generated from the sample is explained by some feature in $Q$.

**Example:** Let the training sample be $\{\langle 010100, + \rangle, \langle 110010, + \rangle, \langle 101111, + \rangle, \langle 011000, - \rangle, \langle 101001, - \rangle, \langle 100101, - \rangle\}$. Then, the set of all conflicts generated from this sample is: $a_1 = \langle 001100 \rangle$, $a_2 = \langle 111101 \rangle$, $a_3 = \langle 110001 \rangle$, $a_4 = \langle 101010 \rangle$, $a_5 = \langle 011011 \rangle$, $a_6 = \langle 010111 \rangle$, $a_7 = \langle 110111 \rangle$, $a_8 = \langle 000110 \rangle$ and $a_9 = \langle 001010 \rangle$.

The subset $\{x_1, x_3, x_4\}$ is sufficient to form a consistent hypothesis (e.g., $\bar{x}_1 \bar{x}_3 \vee (\bar{x}_3 \oplus x_4)$), and that all subsets of cardinality less than 3 are insufficient. □.

Given a sufficient subset of features, it is easy to construct a consistent hypothesis. For example, the algorithm ID3 [Quinlan 86] can be applied to the training sample but restricted to consider only the features in the given subset. Hence, in the rest of this paper, finding a solution will be taken to mean identifying a subset of features sufficient to form a consistent hypothesis.

# 3 Improving the FOCUS Algorithm

The FOCUS algorithm given in [Almuallim and Dietterich 91] works by trying all the subsets of features of increasing size until a sufficient set is encountered. In the example of the

**Algorithm** FOCUS-2(*Sample*)
1. If all the examples in *Sample* have the same class, return $\phi$.
2. Let $G$ be the set of all conflicts generated from *Sample*.
3. *Queue*= $\{M_{\phi,\phi}\}$. /* This is a first-in-first-out data structure. */
4. Repeat
   4.1. Pop the first element in *Queue*. Call it $M_{A,B}$.
   4.2. Let $OUT = A$.
   4.3. Let $a$ be the conflict in $G$ not explained by any of the features in $A$, such that $|Z_a - B|$ is minimized, where $Z_a$ is the set of features explaining $a$.
   4.4. For each $x \in Z_a - B$
      4.4.1. If Sufficient($A \bigcup \{x\}$), return ($A \bigcup \{x\}$).
      4.4.2. Insert $M_{A \bigcup \{x\},OUT}$ at the tail of *Queue*.
      4.4.3. $OUT = OUT \bigcup \{x\}$.

**end** FOCUS-2.

Figure 1: The FOCUS-2 learning algorithm.

previous section, FOCUS tests the $\binom{6}{0} + \binom{6}{1} + \binom{6}{2} = 22$ subsets of features of size 0, 1 and 2, and some of the $\binom{6}{3} = 20$ subsets of size 3 before returning a solution. By doing so, FOCUS is not exploiting all the information given in the training sample. Consider, for instance, the conflict $a_1 = \langle 001100 \rangle$. This conflict tells us that any sufficient set of features must contain $x_3$ or $x_4$ in order to explain the conflict. Hence, none of the sets $\{x_1\}, \{x_2\}, \{x_5\}, \{x_6\}, \{x_1, x_2\}, \{x_1, x_5\}, \{x_1, x_6\}, \{x_2, x_5\}, \{x_2, x_6\}, \{x_5, x_6\}$ can be solutions. Therefore, all of these sets can immediately be ruled out of the algorithm's consideration. Many other subsets can be similarly ruled out based on the other conflicts.

Figure 1 shows the FOCUS-2 algorithm, which takes advantage of this observation. In this algorithm, we use a first-in-first-out queue in which each element denotes a subspace of the space of all feature subsets. Each element has the form $M_{A,B}$, which denotes the space of all feature subsets that include all of the features in the set $A$ and none of the features in the set $B$. Formally,

$$M_{A,B} = \{T \mid T \supseteq A, T \bigcap B = \phi, T \subseteq \{x_1, x_2, \cdots, x_n\}\}.$$

For example, the set $M_{\phi,\phi}$ denotes all possible feature subsets, the set $M_{A,\phi}$ denotes all feature subsets that contain at least the features in $A$, and the set $M_{\phi,B}$ denotes all feature subsets that do not contain any features in $B$.

The main idea of FOCUS-2 is to keep in the queue only the *promising* portions of the space of feature subsets—i.e. those that *may* contain a solution. Initially, the queue contains only the element $M_{\phi,\phi}$ which represents the whole power set. In each iteration in Step 4, the space represented by the head of the queue is partitioned into disjoint subspaces, and those subspaces that cannot contain solutions are pruned from the search.

Consider again the conflict $a_1 = \langle 001100 \rangle$. Suppose the current space of possible feature subsets is $M_{\phi,\phi}$. We know that any sufficient feature subset must contain either $x_3$ or $x_4$.

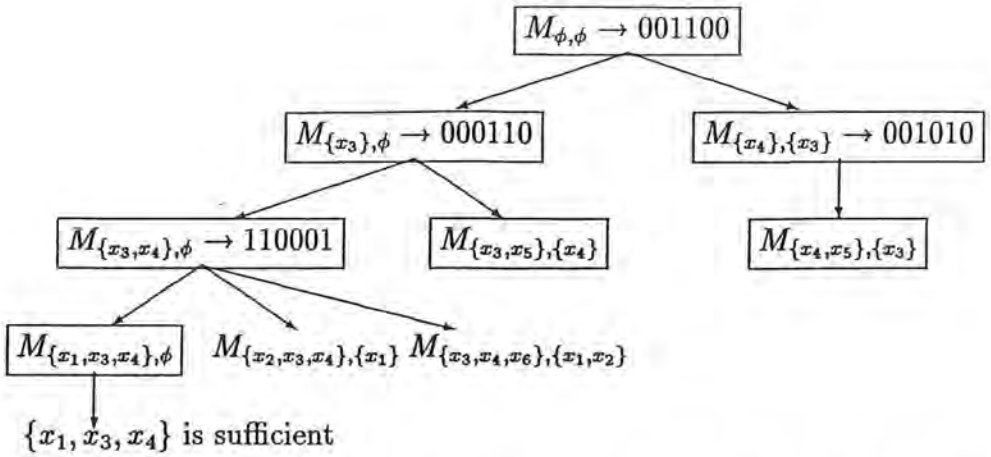


Figure 2: An example of FOCUS-2. Rectangles indicate where the sufficiency tests occurred.

We can incorporate this knowledge into the search by refining $M_{\phi,\phi}$ into the two subspaces $M_{\{x_3\},\phi}$ (all feature subsets that contain $x_3$) and $M_{\{x_4\},\{x_3\}}$ (all feature subsets that contain $x_4$ and do not contain $x_3$). Note that the second subscript of $M$ is used to keep the various subspaces disjoint. Clearly, conflicts with fewer 1's in them provide more constraint for the search than conflicts with more 1's. Hence, the algorithm (Step 4.3) searches for the conflict with the smallest number of unexplained 1's and incorporates it into the search.

In detail, here is how FOCUS-2 behaves on the example given in the previous section. As shown in Figure 2, the algorithm starts by processing $M_{\phi,\phi}$. The conflict $a_1 = \langle 001100 \rangle$ is selected in Step 4.3 and $M_{\phi,\phi}$ is replaced by $M_{\{x_3\},\phi}$ and $M_{\{x_4\},\{x_3\}}$. Next, for $M_{\{x_3\},\phi}$, the conflict $a_8 = \langle 000110 \rangle$ is selected, and $M_{\{x_3,x_4\},\phi}$ and $M_{\{x_3,x_5\},\{x_4\}}$ are added to the queue. $M_{\{x_4\},\{x_3\}}$ is then processed with $a_9 = \langle 001010 \rangle$ and $M_{\{x_4,x_5\},\{x_3\}}$ is inserted. Finally, when $M_{\{x_3,x_4\},\phi}$ is processed with $a_3 = 110001$, the algorithm terminates in Step 4.4.1 before adding $M_{\{x_1,x_3,x_4\},\phi}$ to the queue, since $\{x_1, x_3, x_4\}$ is a solution.

Using FOCUS-2, the number of sufficiency tests is only 7. By comparison, FOCUS must perform at least 23 sufficiency tests (to test each of the 22 subsets of size up to 2, and at least one of the 20 subsets of size 3). Because FOCUS-2 only prunes subspaces that cannot possibly explain all of the conflicts, it is sound and complete—it will not miss any sufficient feature subsets. Furthermore, because it considers the subspaces $M_{A,B}$ in order of increasing size of $A$, it is guaranteed to find a sufficient subset with the smallest possible size. Finally, of course, the number of sufficiency tests performed by FOCUS-2 will typically be much less (and certainly never more) than the number of tests performed by FOCUS.

# 4 Heuristics for the MIN-FEATURES bias

Exact implementation of the MIN-FEATURES bias in domains with large numbers of features can be computationally infeasible[1]. In such cases, one may be willing to employ efficient

[1]The reader may have already noticed the connection between the MIN-FEATURES bias and the Minimum-Set-Cover problem, which is known to be NP-hard [Garey and Johnson 79]. However, note that we assume here the existence of a *small* set of features that forms a solution. This corresponds to restricting the Minimum-Set-Cover problem to instances that have small covers.

heuristics that provide good but not necessarily optimal solutions. In this section, we describe three such algorithms. Each of these algorithms implements an iterative procedure where in each iteration the feature that seems most promising is added to the partial solution. This continues until a sufficient set of features is found. The only difference between the three algorithms is the criterion used in selecting the best feature in each iteration.

The algorithms are listed in Figure 3. In the following, we describe the selection criteria implemented by each algorithm.

**The Mutual-Information-Greedy (MIG) Algorithm:** For a given set of features $Q$, imagine that the training sample is partitioned into $2^{|Q|}$ groups such that the examples in each group have the same truth assignment to the features in $Q$. (One can think of this as a completely balanced decision tree with $2^{|Q|}$ leaves.) Let $p_i$ and $n_i$ denote the number of positive and negative examples in the $i$-th group, respectively. The *entropy* of $Q$ is defined as

$$Entropy(Q) = -\sum_{i=0}^{2^{|Q|}-1} \frac{p_i + n_i}{|Sample|} \left[ \frac{p_i}{p_i + n_i} \log_2 \frac{p_i}{p_i + n_i} + \frac{n_i}{p_i + n_i} \log_2 \frac{n_i}{p_i + n_i} \right]$$

with the convention that $a \log_2 a = 0$ when $a = 0$.

In the Mutual-Information-Greedy algorithm, the feature that leads to the minimum entropy when added to the current partial solution is selected as the best feature.

**The Simple-Greedy (SG) Algorithm:** This algorithm chooses each time the feature that explains the largest number of conflicts that are not yet explained. The conflicts that are explained by this feature are then removed from the set of conflicts. The process is repeated until all conflicts are removed.

**The Weighted-Greedy (WG) Algorithm:** In the Simple-Greedy algorithm, every conflict contributes a unit increment to the score of each feature that explains it. In the Weighted-Greedy algorithm, the increment instead depends on the total number of features that explain the conflict. The intuition is that if a feature uniquely explains a conflict, then that feature *must* be part of the solution set of features. If $A_{x_i}$ is the set of conflicts explained by a feature $x_i$, then the score of $x_i$ is computed as:

$$score_{x_i} = \sum_{a \in A_{x_i}} \frac{1}{\#\text{ of features explaining } a \; - 1}$$

Under this heuristic, when a feature $x_i$ explains a conflict $a$, the contribution of $a$ to the score of $x_i$ is inversely proportional to the number of *other* features that explain $a$. If only a few other conflicts explain $a$ then $x_i$ receives high credit for explaining $a$. In the extreme case where $a$ is exclusively explained by $x_i$, the score of $x_i$ becomes $\infty$. This causes the feature to be included in the solution with certainty.

# 5 Experimental Results

## 5.1 Sample Complexity and Accuracy

In this subsection, we test the value of each of the heuristics of Section 4 for learning tasks where many irrelevant features are present. The performance of each of these algorithms

**Algorithm:** Mutual-Information-Greedy(*Sample*)
1. $Q = \phi$.
2. Repeat until *Entropy*$(Q) = 0$:
   2.1. For each feature $x_i$, let $score_{x_i}$ =*Entropy*$(Q \bigcup \{x_i\})$.
   2.2. Let *best* be the feature with the lowest score.
   2.3. $Q = Q \bigcup \{$ *best* $\}$.

**end** Mutual-Information-Greedy

**Algorithm:** Simple-Greedy(*Sample*)
1. $Q = \phi$.
2. Let $A$ be the set of all conflicts generated from *Sample*.
3. Repeat until $A$ is empty:
   3.1. For each feature $x_i$, let $score_{x_i}$ = the number of conflicts explained by $x_i$.
   3.2. Let *best* be the feature with the highest score.
   3.3. $Q = Q \bigcup \{$ *best* $\}$.
   3.4. Remove from $A$ all the conflicts explained by *best*.

**end** Simple-Greedy.

**Algorithm:** Weighted-Greedy(*Sample*)
1. $Q = \phi$.
2. Let $A$ be the set of all conflicts generated from *Sample*.
3. Repeat until $A$ is empty:
   3.1. For each feature $x_i$:
      3.1.1. Let $A_{x_i}$ be the set of conflicts explained by $x_i$.
      3.1.2 Let $score_{x_i} = \sum_{a \in A_{x_i}} \frac{1}{\#\text{ of features explaining } a - 1}$.
   3.2. Let *best* be the feature with the highest score.
   3.3. $Q = Q \bigcup \{$ *best* $\}$.
   3.4. Remove from $A$ all the conflicts explained by *best*.

**end** Weighted-Greedy.

Figure 3: Three heuristics for approximating the MIN-FEATURES bias.

is compared to that of FOCUS, ID3 and FRINGE through experiments similar to those reported in [Almuallim and Dietterich 91]. Note that MIG, SG, WG, and FOCUS are not complete learning algorithms—rather they are preprocessors that provide us only with a set of features sufficient to construct a consistent hypothesis. To construct an actual hypothesis, we first filter the training examples to remove all features not selected during preprocessing. Then we give the filtered examples to ID3 to construct a decision tree. Our version of ID3 performs no windowing or forward pruning and employs the information gain (mutual information) criterion to select features.

We report here two kinds of experiments. In the first experiment, we are interested in the *worst-case* performance over a class of concepts, where each concept is definable over at most $p$ out of $n$ features (and hence, the MIN-FEATURES bias is appropriate). As our measure of performance, we employ the sample complexity—the minimum number of training examples needed to ensure that every concept in the class can be learned in the PAC sense [Blumer et.al. 87]. We estimate the sample complexity with respect to fixed learning parameters $p, n, \epsilon$, and $\delta$ and with training samples drawn according to the uniform distribution.

In the second experiment, we are interested in the *average-case* performance of the algorithms. We randomly generated a collection of concepts that involve only a few features among many available ones. We then measured the accuracy rate of each algorithm while progressively increasing the size of the training sample—i.e. by plotting the learning curve for each of the concepts under consideration.

**EXPERIMENT 1: Sample Complexity.** The goal of this experiment is to estimate the minimum number of examples that enables each algorithm to PAC learn all the concepts of at most 3 relevant features out of $n$ available features for $n = 8, 10$ and 12. To decide whether an algorithm $L$ learns a concept $c$ for sample size $m$, we generate 100,000 random samples of $c$ of size $m$. We conclude that $c$ is learned by $L$ if and only if for at least 90% of these samples $L$ returns a hypothesis that is at least 90% correct. Thus, the quantity measured here can be viewed as an empirical estimate of the *sample complexity* of each algorithm [Blumer et.al. 87] for $\epsilon = \delta = 0.1$. To reduce the computational costs involved in this experiment, we exploited the fact that the algorithms are symmetric with respect to the permutation and negation of any subset of the features of the target concept [Almuallim 91].

The results of this experiment for $n = 8, 10$ and 12 are shown in Figure 4.

**EXPERIMENT 2: Learning Curve.** The purpose of this experiment is to perform a kind of "average-case" comparison between the algorithms. The experiment is conducted as follows. First, we randomly choose a concept such that it has only a few relevant features among many available ones. We then run each of the algorithms on randomly-drawn training samples of this concept and plot the accuracy of the hypothesis (i.e., the percentage of the examples correctly classified by the hypothesis) returned by each algorithm against the training sample size. This is repeated for various sample sizes for the same concept.

The above procedure was applied on 100 randomly selected concepts each having at most 5 relevant features out of 16 available features. For each of these concepts, the sample size $m$ was varied from 20 to 120 examples. For each value of $m$, the accuracy rate was averaged over 100 randomly drawn training samples.

Figure 5 shows a pattern typical of all learning curves that we observed. As a way to

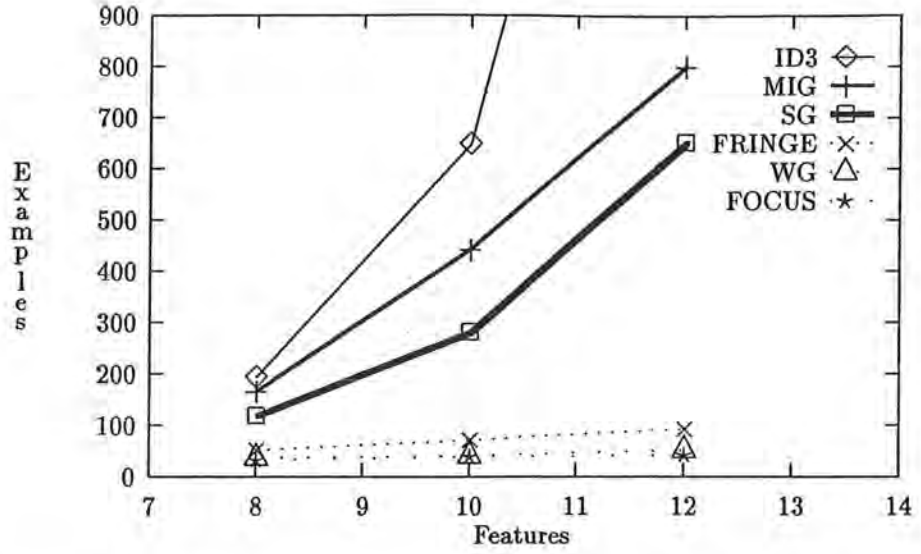


Figure 4: The number of examples needed for learning all the concepts with 3 relevant features out of 8, 10, and 12 available features. ID3 requires 2236 examples when the total number of features is 12.

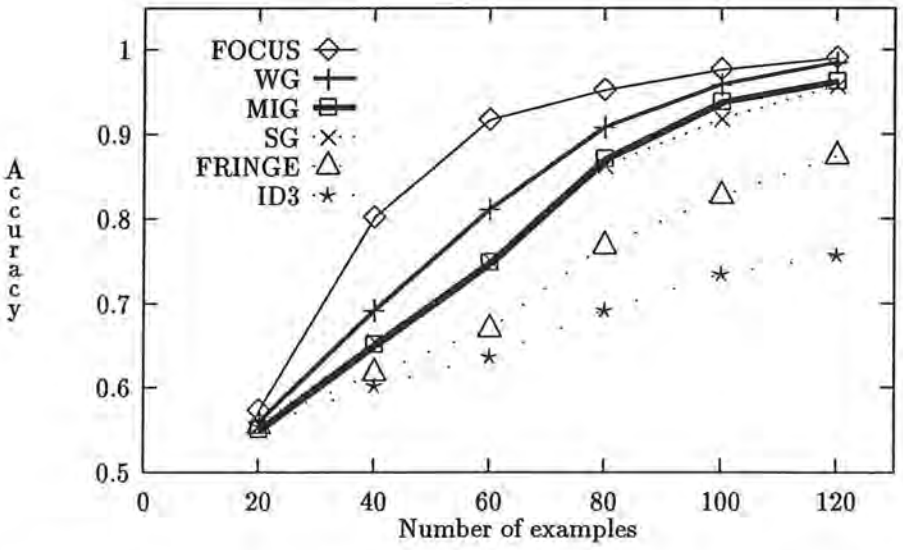


Figure 5: Learning curve for the randomly chosen concept $f(x_1, \ldots, x_{16}) = x_1x_2x_3\bar{x}_4 \vee x_1x_2x_3x_4\bar{x}_5 \vee x_1x_2\bar{x}_3x_4x_5 \vee x_1\bar{x}_2x_3 \vee x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2x_3x_4x_5 \vee \bar{x}_1\bar{x}_2x_3x_4x_5 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4 \vee \bar{x}_1\bar{x}_3x_4x_5 \vee \bar{x}_1\bar{x}_3\bar{x}_4\bar{x}_5$ which has 5 relevant features out of 16.

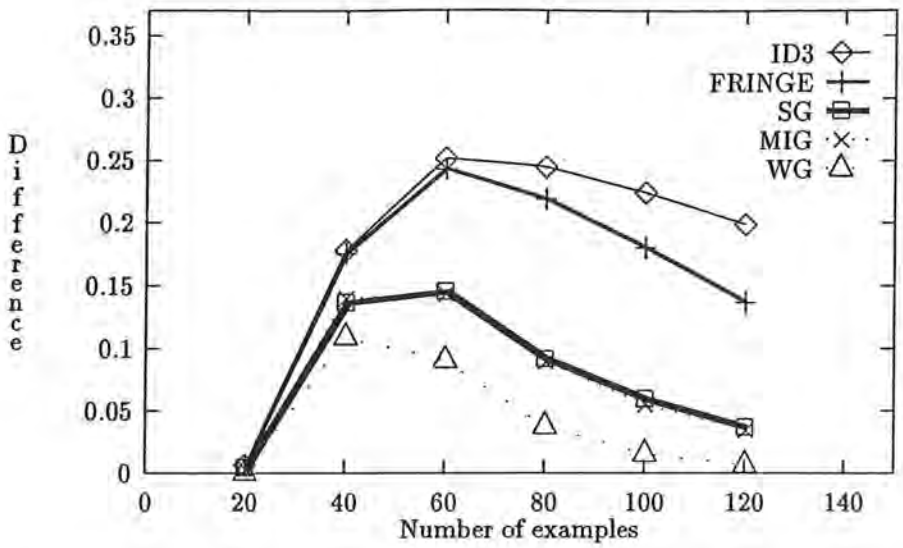


Figure 6: The difference between the accuracy of FOCUS and the accuracy of the other algorithms averaged over all the randomly chosen 100 target concepts.

combine the results of the 100 concepts, we have measured the difference in accuracy between FOCUS and each of the other algorithms for each sample size, and averaged that over all the 100 target concepts. The result is shown in Figure 6.

**DISCUSSION:** The performance of the algorithms tested in the above experiments can be summarized as follows. (i) Each of the three heuristics improved the performance of ID3 when learning in the presence of irrelevant features. (ii) Weighted-Greedy gave the overall best approximation to the MIN-FEATURES bias. The performance of this algorithm was quite close to that of FOCUS both in the worst and average cases. (iii) Mutual-Information-Greedy and Simple-Greedy are very much alike. These algorithms maintained a reasonable average-case performance, but exhibited a rather bad worst-case performance. (iv) Finally, FRINGE showed poor average-case performance, but its worst-case performance is almost as good as Weighted-Greedy and substantially better than Mutual-Information-Greedy and Simple-Greedy. In terms of the computational costs, however, FRINGE is much more expensive than any of the three heuristics considered here.

## 5.2 Execution Time Comparisons

In this subsection, we compare the computational costs of FOCUS, FOCUS-2 and WG measured as the the number of sufficiency tests and the amount of CPU-time required by each algorithm to return a solution. The three algorithms were implemented in C. Special attention was given to optimizing the implementation of FOCUS.

The experiments were conducted using target concepts that have only a few relevant features out of many available. The relative performance of the algorithms was greatly affected by the problem size measured as the number of the available features and the ratio of the relevant features to that number. However, when the problem size was reasonably large,

Table 1: The number of sufficiency tests and CPU-time of FOCUS, FOCUS-2 and WG for a target concept with 9 relevant features out of 25 available features.

| Algorithm | | Training Set Size | | | | |
|---|---|---|---|---|---|---|
| | | 100 | 200 | 300 | 400 | 500 |
| FOCUS | Suff.Tests | 442189.4 | 1329330.5 | 2908068.9 | 2665664.0 | 2695393.0 |
| | Time (seconds) | 9.74 | 55.1 | 211.1 | 191.7 | 186.1 |
| FOCUS-2 | Suff.Tests | 10593.1 | 9785.3 | 11339.5 | 8768.9 | 5582.3 |
| | Time (seconds) | 2.0 | 7.1 | 20.5 | 31.9 | 32.5 |
| WG | Suff.Tests | 8.5 | 10.1 | 10.7 | 10.5 | 11.0 |
| | Time (seconds) | 0.16 | 0.86 | 2.31 | 4.3 | 7.0 |

the relative performance followed a consistent trend. This trend is illustrated by Table 1 where we give the result for a target concept with 9 relevant features out of 25 available features. Training examples were drawn with replacement under the uniform distribution and the training sample size was varied from 100 to 500. The numbers in this table are averaged over 10 runs for each training sample size.

Overall, we found that FOCUS-2 was several times faster than FOCUS and that Weighted-Greedy was further many times faster than FOCUS-2. It is interesting to note that the number of sufficiency tests done by FOCUS remains steady as the training sample size grows, since it blindly follows the same steps for any training sample. FOCUS-2, on the other hand, does a progressively smaller number of sufficiency tests as the number of training examples increases. This is because with a larger sample there is a greater chance of getting conflicts that are explained by only few features, and consequently, a better chance for significant reduction in the number of sufficiency tests needed by FOCUS-2.

# 6 Conclusions and Future Research

This paper dealt with the problem of reducing the computational costs involved in implementing the MIN-FEATURES bias. Section 3 introduced the FOCUS-2 algorithm, which provides an implementation of this bias that is substantially faster than the FOCUS algorithm previously given in [Almuallim and Dietterich 91]. Section 4 introduced three efficient heuristics for approximating the MIN-FEATURES bias. Experimental studies were reported in which each of these algorithms was used to preprocess the training data to remove the irrelevant features. All of these algorithms were found to be helpful in improving the performance of ID3 in learning tasks where many irrelevant features are present. In particular, the Weighted-Greedy algorithm exhibited excellent performance that closely matches what is obtained by the exact MIN-FEATURES bias. We recommend that, in applications where the MIN-FEATURES bias is appropriate, the Weighted-Greedy algorithm should be applied to preprocess the training sample before invoking a decision-tree algorithm, such as ID3.

All of the approximation algorithms we give can be shown to be polynomial time algorithms. A challenging goal for future research is to prove formal results on the sample

complexity of these and similar approximation algorithms.

The work reported in this paper assumes noise-free training data. A direct way to deal with classification noise is to modify the given algorithms by relaxing the requirement of explaining *all* the conflicts generated from the training data. That is, we search for a small set of features that may leave a certain percentage of the conflicts unexplained, where such percentage can be determined through cross-validation. Studying this and more sophisticated approaches to dealing with noise and applying the resulting techniques to real-world problems are two important topics for future work.

# 7 Acknowledgments

The authors gratefully acknowledge the support of the NSF under grant number IRI-86-57316.

# References

[Almuallim and Dietterich 91] Almuallim, H. and Dietterich, T. G. 1991. Learning With Many Irrelevant Features. Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91), 547–552.

[Almuallim 91] Almuallim, H. 1991. Exploiting Symmetry Properties in the Evaluation of Inductive Learning Algorithms: An Empirical Domain-Independent Comparative Study. Technical Report, 91-30-09, Dept. of Computer Science, Oregon State University, Corvallis, OR 97331-3202.

[Blumer et.al. 87] Blumer, A.; Ehrenfeucht, A.; Haussler, D.; and Warmuth, M. 1987. Learnability and the Vapnik-Chervonenkis Dimension, Technical Report UCSC-CRL-87-20, Department of Computer and Information Sciences, University of California, Santa Cruz, Nov. 1987. Also in *Journal of ACM,* 36(4):929–965.

[Ichino and Sklansky 84a] Ichino, M. and Sklansky, J. Optimum Feature Selection by Zero-One Integer Programming. IEEE Trans. Sys. Man & Cyb., Vol. SMC-14, No. 5, 737–746, Sep 1984.

[Ichino and Sklansky 84b] Ichino, M. and Sklansky, J. Feature Selection for Linear Classifiers, The Seventh International Conference on Pattern Recognition, 124–127, 1984.

[Garey and Johnson 79] Garey, M. R. and Johnson D. S. 1979. Computers and Intractability. W.H. Freeman and Company.

[Kittler 80] Kittler, J. 1980. Computational Problems of Feature Selection Pertaining to Large Data Sets. Pattern Recognition in Practice. Gelsma, E.S. and Kanal, L.N. (eds.) North-Holland Publishing Company, 405–414.

[Morgera 86] Morgera, S.D. Computational Complexity and VLSI Implementation of an Optimal Feature Selection Strategy, Pattern Recognition In Practice II, Gelsma, E.S. and Kanal, L.N. (eds.) Elsevier Science Publishers B.V. (North-Holland), 389–400, 1986.

[Mucciardi and Gose 71] Mucciardi, A.N. and Gose, E.E. A Comparison of Seven Techniques for Choosing Subsets of Pattern Recognition Properties, IEEE Trans. Computers, Vol. C-20, No. 9, 1023–1031, Sep 1971.

[Narendra and Fukunaga 77] Narendra, P.M. and Fukunaga, K. A Branch and Bound Algorithm for Feature Subset Selection, IEEE Trans. Computers, Vol. C-26, No. 9, 917–922, 1977.

[Pagallo and Haussler 90] Pagallo, G.; and Haussler, D. 1990. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–100.

[Queiros and Gelsma 84] Queiros, C.E. and Gelsma, E.S. On Feature Selection, The Seventh International Conference on Pattern Recognition, 128–130, 1984.

[Quinlan 86] Quinlan, J. R. 1986. Induction of Decision Trees, *Machine Learning*, 1(1):81–106.