Learning With Many Irrelevant Features

Hussein Almuallim and **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

Abstract

In many domains, an appropriate inductive bias is the MIN-FEATURES bias, which prefers consistent hypotheses definable over as few features as possible. This paper defines and studies this bias. First, it is shown that any learning algorithm implementing the MIN-FEATURES bias requires $\Theta(\frac{1}{\epsilon}\ln\frac{1}{\delta}+\frac{1}{\epsilon}[2^p + p\ln n])$ training examples to guarantee PAC-learning a concept having p relevant features out of n available features. This bound is only logarithmic in the number of irrelevant features. The paper also presents a quasi-polynomial time algorithm, FOCUS, which implements MIN-FEATURES. Experimental studies are presented that compare FOCUS to the ID3 and FRINGE algorithms. These experiments show that—contrary to expectations—these algorithms do not implement good approximations of MIN-FEATURES. The coverage, sample complexity, and generalization performance of FOCUS is substantially better than either ID3 or FRINGE on learning problems where the MIN-FEATURES bias is appropriate. This suggests that, in practical applications, training data should be preprocessed to remove irrelevant features before being given to ID3 or FRINGE.

Introduction

Historically, the development of inductive learning algorithms has been a two-step process: (i) select a representation scheme (e.g., decision trees), (ii) develop an algorithm to find instances of the scheme that are consistent with given collections of training examples. A shortcoming of this approach is that there is no separation between a *specification* of the desired learning behavior of the algorithm and its *implementation*. Specifically, the *bias* of the algorithm is adopted implicitly, particularly as a side-effect of the second step. Often, it is difficult even to state the bias in any simple way. Consequently, it is difficult to tell in advance whether the bias is appropriate for a new learning problem.

Recently, a few authors (Buntine 1990, Wolpert 1990) have advocated a different procedure: (i) adopt a bias over some space of hypotheses (or, equivalently, select a prior probability distribution over the space), (ii) select a scheme for representing hypotheses in the space, and (iii) design an algorithm that implements this bias, at least approximately.

The goal of this paper is to pursue this second procedure. We consider the space of all binary functions defined over n Boolean input features. We adopt the following bias, which we call the MIN-FEATURES bias: if two functions are consistent with the training examples, prefer the function that involves fewer input features (break ties arbitrarily). This is a bias in favor of simplicity—but not mere syntactic simplicity. Functions over fewer variables are *semantically* simpler than functions over more variables.

We begin by adopting a straightforward representation for binary functions defined over n input features. We then analyze the sample complexity of any probably-approximately correct (PAC) learning algorithm that implements the MIN-FEATURES bias. It is proved that

$$\Theta\left(\frac{1}{\epsilon}\ln\frac{1}{\delta}+\frac{1}{\epsilon}\left[2^p + p\ln n\right]\right)$$

training examples are required to PAC-learn a binary concept involving p input features (out of a space of n input features) with accuracy parameter ϵ and confidence parameter δ. Note in this bound that the total number of available features n appears only logarithmically. Hence, if there are k irrelevant features, it only costs us a factor of $\ln k$ training examples to detect and eliminate them from consideration.

Following this analysis, a simple, quasi-polynomial time algorithm that implements the MIN-FEATURES bias is described and analyzed. The algorithm, called FOCUS, first identifies the p features that are needed to define the binary function. It then applies a straightforward learning procedure that focuses on just those p features.

At first glance, it may appear that there are already many algorithms that approximate this bias. For example, ID3 (Quinlan 1986) has a bias in favor of small decision trees, and small trees would seem to test only a subset of the input features. In the final section of the paper, we present experiments comparing FOCUS to ID3 and FRINGE (Pagallo & Haussler 1990). These demonstrate that ID3 and FRINGE are not good implementations of the MIN-FEATURES bias—these algorithms often produce hypotheses as output that are much more complex (in terms of the number of input features used) than the hypotheses found by FOCUS. Indeed, there are some cases in which ID3 and FRINGE miss extremely simple hypotheses.

These results suggest that the FOCUS algorithm will require fewer training examples and generalize more correctly than ID3 in domains where the MIN-FEATURES bias is appropriate. We believe there are many such domains. For example, in many practical applications, it is often not known exactly which input features are relevant or how they should be represented. The natural response of users is to include all features that they believe could possibly be relevant and let the learning algorithm determine which features are in fact worthwhile.

Another situation in which many irrelevant features may be present is when the same body of training data is being used to learn many different binary functions. In such cases, one must ensure that the set of features measured in the data is sufficient to learn all of the target functions. However, when learning each individual function, it is likely that only a small subset of the features will be relevant. This applies, for example, to the task of learning diagnosis rules for several different diseases from the medical records of a large number of patients. These records usually contain more information than is actually required for describing each disease. Another example (given in Littlestone, 1988) involves pattern recognition tasks in which feature detectors automatically extract a large number of features for the learner's consideration, not knowing which might prove useful.

Notation

For each $n \geq 1$, let $\{x_1, x_2, \cdots, x_n\}$ denote a set of n Boolean features and X_n denote the set $\{0,1\}^n$ of all assignments to these features—the set of *instances*. A binary concept c is a subset of X_n. A binary function f represents the concept c if $f(x) = 1$ for all $x \in c$ and $f(x) = 0$ otherwise. Of course, binary functions can be represented as Boolean formulas. A feature x_i, for $1 \leq i \leq n$, is said to be *relevant* to a concept c if x_i appears in every Boolean formula that represents c and *irrelevant* otherwise.

The complexity of a concept, denoted $s(c)$, is defined to be the minimum number of bits needed to encode the concept with respect to some encoding scheme. The encoding scheme we use in this work will be introduced in Section 3. We let $C_{n,s}$ denote the set of all binary concepts of complexity at most s defined on $\{x_1, x_2, \cdots, x_n\}$.

We assume an arbitrary probability distribution D on X_n. For $0 < \epsilon < 1$, a concept h is said to be ϵ-close to a concept c with respect to D if the sum of the probability of all the instances in the symmetric difference of h and c is at most ϵ.

Let f be a function that represents the concept c. Then, for $x \in X_n$, the value $f(x)$ is said to be the *class* of x. A pre-classified *example* of c is a pair of the form $\langle x, f(x) \rangle$. A *sample* of a concept c is a multi-set of examples of c drawn randomly (with replacement) according to D. The *size* of the sample is just the number of instances drawn.

In this work, we adopt the notion of Probably Approximately Correct (PAC) learning as defined by Blumer et al. (1987a). With respect to parameters ϵ and δ, $0 < \epsilon, \delta < 1$, we say that a learning algorithm *PAC learns* (or simply, *learns*) a concept c using a sample of size m if, with probability at least $(1 - \delta)$, this algorithm returns as an hypothesis a concept that is ϵ-close to c when the algorithm is given a sample of c of size m, for all fixed but unknown D.

Formal Analysis

In this section, we first define the MIN-FEATURES bias. We then investigate the sample complexity of any algorithm that implements MIN-FEATURES. Finally, we present the FOCUS algorithm that implements this bias, and we analyze its computational complexity.

The MIN-FEATURES bias can be stated simply. Given a training sample S for some unknown binary function f over X_n, let V be the set of all binary functions over X_n consistent with S. (V is sometimes called the *version space*; Mitchell, 1982.) Let H be the subset of V whose elements have the fewest relevant features. The MIN-FEATURES bias chooses its guess, $\hat{f}$, from H arbitrarily.

Given that we wish to implement and analyze the MIN-FEATURES bias, the first step is to choose a representation for hypotheses. We will represent a concept c by the concatenation of two bit vectors R_c and T_c. R_c is an n-bit vector in which the ith bit is 0 if and only if x_i is irrelevant to c. T_c is the rightmost column of the truth table of a Boolean function f that represents c defined only on those features in $\{x_1, x_2, \cdots, x_n\}$, whose corresponding bits in R_c are set to 1.

With this definition, we can now analyze the sample complexity of MIN-FEATURES—that is, the number of training examples required to ensure PAC learning. We must first define a complexity measure corresponding to our bias. Following Blumer et al. (1987b), we will define the complexity $s(c)$ for concept c to be the number of bits needed to encode c using our bit-vector representation. This measure has the property that $s(c_1) < s(c_2)$ iff the number of relevant features of c_1 is less than the number of relevant features of c_2. Specifically, if c has p relevant features then $s(c) = n + 2^p$. Section 3.1 of Blumer et al. (1987a) gives 3 properties to be satisfied by a reasonable representation of concepts. The reader may verify that these are satisfied by our method of encoding.

Example: Let $n = 5$ and let c be a concept represented by $x_1 \vee x_3$. Then, $R_c = 10100$ and $T_c = 0111$. Hence, the complexity of c is 9. $\square$

The following theorem gives an upper bound on the sample complexity of any algorithm implementing the MIN-FEATURES bias.

Theorem 1 *Let $C_{n,s}$ be the class of concepts defined on n features with complexity at most s. Then, under any probability distribution D, any $n \geq 1$, any ϵ and δ such that $0 < \epsilon, \delta < 1$ and any concept $c \in C_{n,s}$, a sample of size*

$$\frac{1}{\epsilon} \ln \frac{1}{\delta} + \frac{1}{\epsilon} [\log_2(s-n) \ln n + s - n]$$

is sufficient to guarantee that any algorithm implementing the MIN-FEATURES bias will return an hypothesis that is ϵ-close to c with probability at least $1-\delta$.

Proof(Sketch): For any target concept of complexity at most s, the hypothesis space for any algorithm that implements the MIN-FEATURES bias is contained in $C_{n,s}$. We argue that $|C_{n,s}| \leq \binom{n}{\log_2(s-n)} 2^{s-n}$. The result follows immediately by applying the lemma of (Blumer et al. 1987b). $\square$

It is interesting to note that the number of examples sufficient for learning grows only logarithmically in the number of irrelevant features and linearly in the complexity of the concept.

We now show that this bound is tight by exhibiting an identical lower bound using the methods developed by Blumer et al. (1987a) exploiting the Vapnik-Chervonenkis dimension (VC-dimension).

The VC-dimension of a class of concepts C is defined to be the largest integer d such that there exists a set of d instances that can be labelled by the concepts in C in all the 2^d possible ways. It is shown that the number of examples needed for learning any class of concepts strongly depends on the VC-dimension of the class. Specifically, (Ehrenfeucht et al. 1988) prove the following:

Theorem 2 *Let C be a class of concepts and $0 < \epsilon, \delta < 1$. Then, any algorithm that PAC learns all the concepts in C with respect to ϵ, δ and any probability distribution must use a sample of size $\Omega\left(\frac{1}{\epsilon} \ln \frac{1}{\delta} + \frac{VCdim(C)}{\epsilon}\right)$.*

To apply this result, we must first determine the VC-dimension of the $C_{n,s}$, the set of boolean concepts over n features having complexity less than or equal to s.

Lemma 1 *Let $C_{n,s}$ be as in Theorem 1. Then*

$$VCdim(C_{n,s}) \geq \max\left\{\frac{1}{10} \log_2(s-n) \log_2 n,\ s-n\right\}.$$

The proof of this result is lengthy and is omitted for lack of space.

We can now state the lower bound:

Algorithm FOCUS (*sample*)
1. For $i = 0, 1, 2, \cdots$ do:
   1.1 For all $A \subseteq \{x_1, x_2, \cdots, x_n\}$ of size i:
      1.1.1 If there exist no two examples in the sample that agree on all the features in A but do not agree on the class then go to 2.
2. Return *any* concept h consistent with the sample, such that only those features in A are relevant to h.

Figure 1: The FOCUS Learning Algorithm

Theorem 3 *Under the same conditions as Theorem 1, any algorithm that PAC-learns $C_{n,s}$ must use a sample of size*

$$\Omega\left(\frac{1}{\epsilon} \ln \frac{1}{\delta} + \frac{1}{\epsilon} [\ln(s-n) \ln n + s - n]\right).$$

These results show that the presence of many irrelevant features does not make the learning task substantially more difficult, at least in terms of the number of examples needed for learning, since the sample complexity grows only logarithmically in the number of irrelevant features.

Now that we have analyzed the sample complexity of the MIN-FEATURES bias, we exhibit an algorithm that implements this bias. The algorithm given in Figure 1 searches for and returns a consistent hypothesis using a minimal set of attributes, and hence, it implements the desired bias.

To determine the computational complexity of FOCUS, suppose that it is given a sample of size m for a concept of complexity s. The condition in the inner loop can be tested by maintaining an array of length $2^{|A|}$ with an entry for each possible assignment of the features in A. For each example in the sample, we check the values of the features in A as given in the example and label the corresponding entry in the array using the class of the example. If, during this process, a label of any entry has to be reversed, then the result of the test is false. Otherwise, the result is true. This will cost time $O(m \cdot 2^{|A|})$. Since the target concept has complexity s, the value of $|A|$ will reach at most $\log_2(s-n)$. The outer loop is then executed at most $\binom{n}{\log_2(s-n)} = O(n^{\log_2(s-n)})$ times. The computational complexity of this algorithm is dominated by the two nested loops, and, therefore, the algorithm will terminate in time $O((2n)^{\log(s-n)} m)$. This is quasi-polynomial in n and s, but clearly it will be impractical for large values of s.

According to the definition of learnability given in (Blumer et al. 1987a), this says that the class of Boolean concepts, under our complexity measure, is learnable using a polynomial number of examples in quasi-polynomial time. An analogous result is given in (Verbeurgt 1990) where, taking the minimum number of terms needed to encode the concept as a DNF formula as the complexity measure, they obtain a learnability result using a polynomial sample size and quasi-polynomial time. However, their result is only shown for the uniform distribution case, while ours applies to all distributions.

Experimental Work

Several learning algorithms appear to have biases similar to the MIN-FEATURES bias. In particular, algorithms related to ID3 (Quinlan 1986) attempt to construct "small" decision trees. These algorithms construct the decision tree top-down (i.e., starting at the root), and they terminate as soon as they find a tree consistent with the training examples. Features tested at each node are chosen according to their *estimated relevance* to the target concept, measured using the mutual information criterion. In this section, we test these algorithms to see how well they implement the MIN-FEATURES bias.

In particular, we compare three algorithms: (i) **ID3:** As described in (Quinlan 1986), but resolving ties randomly when two or more features look equally good. (ii) **FRINGE:** As given in (Pagallo & Haussler 1990), with the maximum number of iterations set to 10. (iii) **FOCUSed-ID3:** First, a minimum set of features sufficient to produce a consistent hypothesis is obtained as in FOCUS. After finding a minimal-size subset of relevant features, the training examples are filtered to remove all irrelevant features. The filtered examples are then given to ID3 to construct a decision tree.

We consider three evaluation criteria: coverage, sample complexity, and error rate. The *coverage* of a learning algorithm, L, is a measure of the number of distinct concepts that can be learned from a training sample of size m. More precisely, consider the collection of all training samples containing m distinct examples for a concept c, and suppose we give each of these samples to algorithm L. If, for fraction $1-\delta$ of the training samples, L outputs a function that is ϵ-close to the correct concept, then we say that L frequently-approximately correctly (FAC) learns c (Dietterich 1989). The coverage of an algorithm, given m, ϵ, and δ, is the number of concepts that can be FAC-learned by the algorithm.

The *sample complexity* of an algorithm L for a space of concepts C is estimated as the smallest sample size sufficient to enable L to FAC-learn every concept in C. This is equivalent to the sample complexity of PAC-learning, except that it is measured only for the uniform distribution and instances are drawn without replacement.

Finally, the *error rate* for an algorithm on a given concept is measured as the probability that a randomly chosen example would be misclassified by the hypothesis output by the algorithm, assuming the uniform distribution over the space of examples.

Since our objective is to evaluate the learning performance with respect to the MIN-FEATURES bias, we have specialized the above criteria in the following manner. First, concepts with $i+1$ relevant features are not counted in the coverage of an algorithm unless all concepts of i or fewer features are FAC learned as well. If this condition is not included, there exist trivial algorithms that can attain high coverage while learning only very uninteresting concepts. Second, for the sample complexity measurement, we choose C to be a class of concepts with only p or fewer relevant features. Finally, in measuring the error rates of the three algorithms, the target concept is chosen randomly from those concepts having p or fewer features.

One technical problem in performing our experiments is the immense amount of computation involved in the exact measurement of coverage and sample complexity when the number of features is large. Therefore, we employed two techniques to reduce the computational costs of these measurements. First, we exploited the fact that each of the three algorithms is symmetric with respect to permutations and/or negations of input features. More precisely, if an algorithm FAC-learns a concept represented by a Boolean function $f(x_1, x_2, \ldots, x_i, \ldots, x_j, \ldots, x_n)$, then the same algorithm also learns the concepts represented by $f(x_1, x_2, \ldots, x_j, \ldots, x_i, \ldots, x_n)$, $f(x_1, x_2, \ldots, \bar{x}_i, \ldots, x_j, \ldots, x_n)$ and so on for all functions obtained by permuting and/or negating the features in f. These symmetry properties partition the space of concepts into equivalence classes such that it suffices to test one representative concept in each equivalence class to determine FAC-learnability for all concepts in the class.[1] Second, we measured FAC-learning statistically by running each algorithm on a large number of randomly-chosen samples (10,000 or 100,000 depending on the experiment). This number was observed to be large enough to reliably determine the FAC-learnability of concepts.

Experimental Results

EXPERIMENT 1: Coverage. In this experiment, we measured the MIN-FEATURES-based coverage of each of the three algorithms. For each algorithm, we counted the number of concepts learned *in order* as a function of the size m of the sample and the total number of features n. The learning parameters were $n = 5, 6, 7$, and 8, $\epsilon = 0.1$, $\delta = 0.1$ and m varying. The number of samples tested per concept was 10,000. Figure 2 shows the result for $n = 8$. The results for $n = 5, 6, 7$ were similar.

EXPERIMENT 2: Sample Complexity. In this experiment, we estimated the sample size needed to learn all concepts having 3 or fewer relevant features out of a total of 8, 10, or 12 available features. As

[1] For counting techniques that can be employed to find the number of equivalence classes and the number of concepts in a given equivalence class see Harrison (1965) and Slepian (1953).

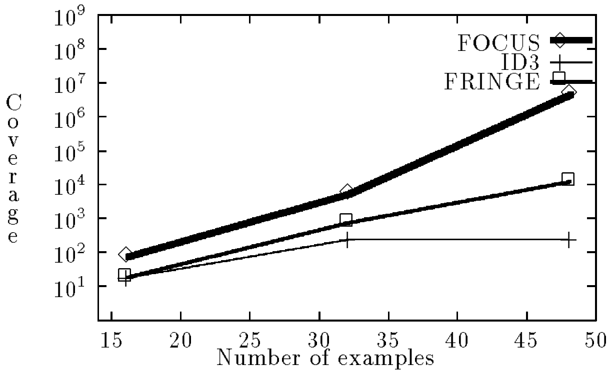

Figure 2: Coverage of the three algorithms for $n = 8$.

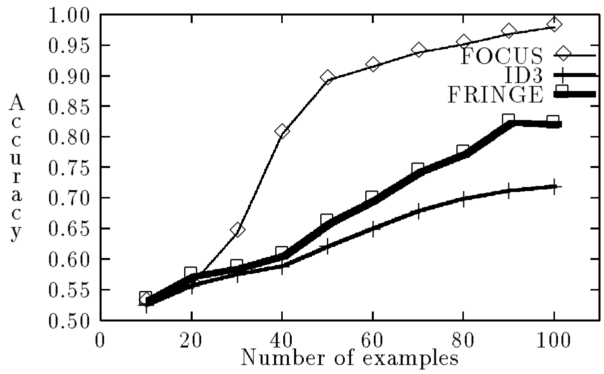

Figure 3: Learning curve for the randomly chosen concept $f(x_1, \ldots, x_{16}) = x_1 x_2 x_3 \bar{x_4} \vee x_1 x_2 x_3 x_4 \bar{x_5} \vee x_1 x_2 \bar{x_3} x_4 x_5 \vee x_1 \bar{x_2} x_3 \vee x_1 \bar{x_2} \bar{x_3} \bar{x_4} \vee \bar{x_1} x_2 x_3 x_4 x_5 \vee \bar{x_1} \bar{x_2} x_3 x_4 x_5 \vee \bar{x_1} \bar{x_2} x_3 \bar{x_4} \vee \bar{x_1} \bar{x_3} x_4 x_5 \vee \bar{x_1} \bar{x_3} \bar{x_4} \bar{x_5}$ which has 5 relevant features out of 16.

before, ϵ and δ were 0.1. The number of samples tested per concept was 100,000. The results are given in the following table:

No. features	ID3	FRINGE	FOCUS
8	194	52	34
10	648	72	40
12	2236	94	42

EXPERIMENT 3: Error rates. In the previous experiments we were looking at the "worst case" performance of the learning algorithms. That is, given a reasonable sample size, an algorithm may learn all the concepts under consideration with the exception of few that require a substantial increment in the sample size. Such an algorithm could exhibit poor performance in the previous two experiments. The purpose of this experiment is to perform a kind of "average case" comparison between the three algorithms. The procedure is to plot the learning curve for randomly chosen concepts with few relevant features.

We randomly selected 50 concepts each having at most 5 (out of n) relevant features. For each of these concepts, we measured the accuracy of the hypotheses returned by the three algorithms while successively increasing the sample size. For each value of m, the accuracy rate is averaged over 100 randomly chosen samples. This experiment was performed for $n = 8$, 12, and 16. 50 randomly chosen concepts of no more than 5 relevant features were tested for each value of n.

Figure 3 shows a pattern typical of all learning curves that we observed. Over all 50 concepts, after 60 examples, the mean difference in accuracy rate between FOCUS and ID3 was 0.24 (variance 0.0022). The mean difference between FOCUS and FRINGE was 0.21 (variance 0.0020).

EXPERIMENT 4: Irrelevant Features. The goal of this experiment was to see how the three algorithms are influenced by the introduction of additional (irrelevant) features whose values are assigned at random.

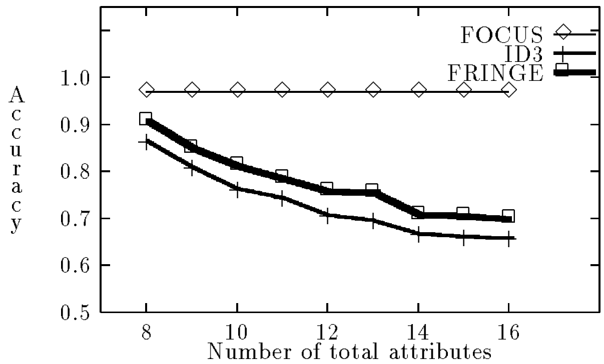

Figure 4: Accuracy of the three algorithms on a randomly chosen concept-sample pair as more irrelevant random features are introduced. The sample size was 64.

For this purpose, we choose a concept-sample pair at random, and measure the accuracy of the hypothesis returned by each algorithm while adding more and more irrelevant features to the sample. The concepts chosen have 5 relevant features out of 8. The sample size was chosen such that all the three algorithms are reasonably accurate when tested using only the 8 starting features. A sample of such a size is chosen randomly and then augmented by successively adding random features to bring the total number of features up to $n = 16$. For each value of n, the accuracy is averaged over 100 runs.

This experiment was repeated on more than 50 concept-sample pairs. A typical result of these runs is shown in Figure 4.

Discussion

These experiments show conclusively that the biases implemented by ID3 and FRINGE, though they may be interesting and appropriate in many domains, are *not* good approximations of the MIN-FEATURES bias. The final experiment shows this most directly. Using the MIN-FEATURES bias, FOCUS maintains a constant, high level of performance as the number of irrelevant features is increased. In contrast, the performance of ID3 and FRINGE steadily degrades. This occurs because ID3 and FRINGE are proposing hypotheses that involve many extra features (or perhaps different features) than those identified by FOCUS.

This also explains the results of Experiments 1, 2, and 3. In Experiment 2, we see that many more training examples are required for ID3 and FRINGE to find good hypotheses. These extra training examples are needed to force the algorithms to discard the irrelevant features. This also means that, for a fixed sample size, ID3 and FRINGE can learn many fewer concepts (with respect to the MIN-FEATURES bias), as shown in Experiment 1. Experiment 3 shows that if the MIN-FEATURES bias is appropriate, then FOCUS will give much better generalization performance than either ID3 or FRINGE.

Conclusion

This paper defined and studied the MIN-FEATURES bias. Section 3 presented a tight bound on the number of examples needed to guarantee PAC-learning for any algorithm that implements MIN-FEATURES. It also introduced the FOCUS algorithm, which implements MIN-FEATURES, and calculated its computational complexity. Finally, Section 4 demonstrated empirically that the ID3 and FRINGE algorithms do not provide good implementations of the MIN-FEATURES bias. As a consequence, ID3 and FRINGE do not perform nearly as well as FOCUS in problems where the MIN-FEATURES bias is appropriate. These results suggest that one should not rely on ID3 or FRINGE to filter out irrelevant features. Instead, some technique should be employed to eliminate irrelevant features and focus ID3 and FRINGE on the relevant ones.

There are many problems for future research. First, we need to develop and test efficient heuristics for finding the set of relevant features in a learning problem. Analysis must be performed to ensure that the heuristics still have near-optimal sample complexity. Second, we need to address the problem of determining relevant features when the training data are noisy. Third, some efficient variant of FOCUS should be tested in real-world learning problems where the MIN-FEATURES bias is believed to be appropriate.

Acknowledgements

The authors gratefully acknowledge the support of the NSF under grant number IRI-86-57316. Hussein Almuallim was supported by a scholarship from the University of Petroleum and Minerals, Saudi Arabia. Thanks also to Nick Flann for helpful comments.

References

Buntine, W. L. 1990. Myths and Legends in Learning Classification Rules. In Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90), 736–742. Boston, MA: Morgan Kaufmann.

Blumer, A.; Ehrenfeucht, A.; Haussler, D.; and Warmuth, M. 1987a. 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.

Blumer, A.; Ehrenfeucht, A.; Haussler, D.; and Warmuth, M. 1987b. Occam's Razor. *Information Processing Letters,* 24:377–380.

Dietterich, T. G. 1989. Limitations on Inductive Learning. In Proceedings of the Sixth International Workshop on Machine Learning, 124–128. Ithaca, NY: Morgan Kaufmann.

Ehrenfeucht, A.; Haussler, D.; Kearns, M.; and Valiant, L.G. 1988. A General Lower Bound on the Number of Examples Needed for Learning. In Proceedings of the First Workshop on Computational Learning Theory, 139–154. Boston, MA: Morgan Kaufmann.

Harrison, M. 1965. *Introduction to Switching and Automata Theory.* McGraw Hill, Inc.

Littlestone, N. 1988. Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine Learning,* 2:285–318.

Mitchell, T. M. 1982. Generalization as Search. *Artificial Intelligence*, 18:203-226.

Pagallo, G.; and Haussler, D. 1990. Boolean Feature Discovery in Empirical Learning. *Machine Learning,* 5(1):71–100.

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

Slepian, D. 1953. On the Number of Symmetry Types of Boolean Functions of n Variables. *Can. J. Math.*, 5(2):185–193.

Verbeurgt, K. 1990. Learning DNF Under the Uniform Distribution in Quasi-polynomial Time. In Proceedings of the Third Workshop on Computational Learning Theory, 314–326. Rochester, NY: Morgan Kaufmann.

Wolpert, D. 1990. A Mathematical Theory of Generalization: Parts I and II. *Complex Systems, 4* (2):151–249.