Average Case Analysis of a *k*-CNF Learning Algorithm

Daniel S. Hirschberg
dan@ics.uci.edu
Michael J. Pazzani
pazzani@ics.uci.edu

Technical Report 91-50

May 20, 1991

We would like to thank Dennis Kibler for helpful comments on this work and Caroline Ehrlich for reviewing an earlier draft of this manuscript. This research is supported by a National Science Foundation Grant IRI-8908260 and by the University of California, Irvine through an allocation of computer time.

Average Case Analysis of a *k*-CNF Learning Algorithm

Daniel S. Hirschberg
Michael J. Pazzani
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717 USA
(dan@ics.uci.edu)
(pazzani@ics.uci.edu)
(714)856-5888

Abstract

We present an approach to modeling the average case behavior of an algorithm for learning Conjunctive Normal Form (CNF, i.e., conjunctions of disjunctions). Our motivation is to predict the expected error of the learning algorithm as a function of the number of training examples. We evaluate the average case model by comparing the error predicted by the model to the actual error obtained by running the learning algorithm, and show how the analysis can lead to insight into the behavior of the algorithm and the factors that affect the error.

1 Introduction

A goal of research in machine learning is to gain an understanding of the capabilities of learning algorithms. Pazzani & Sarrett (1990) introduced a framework for average case analysis of machine learning algorithms. Here, we show how this framework can be applied to create an average case model of an algorithm for learning monotone *k*-CNF concepts and compare the expected behavior of the algorithm to the observed behavior. We show how the analysis provides insight into factors that affect the behavior of the *k*-CNF algorithm.

The framework attempts to unify the formal mathematical and the experimental approaches to understanding the behavior of machine learning algorithms. In order to achieve this unification, an average case model is needed since experiments lead to findings on the average accuracy of an algorithm.

In contrast, the probably approximately correct (PAC) learning model (e.g., Valiant, 1984; Haussler, 1987) has a much different goal. The PAC model stipulates that a system has learned a concept if the system produces a hypothesis consistent with the training data and can guarantee with high probability that its hypothesis is approximately correct. Approximately correct means that the error (i.e., ratio of misclassified examples to total examples classified by the hypothesis) is less than ε. The learning system is required to produce an approximately correct hypothesis with probability $1-\delta$. For a given class of concepts, the PAC model can be used to determine an upper bound on the number of training examples required to achieve an error of at most ε with probability $1-\delta$. The PAC model has led to important insights about the capabilities of machine learning algorithms. For example, if the hypothesis space searched by a learning algorithm is H and $|H|$ is the size of the hypothesis space, then a bound on the error of the algorithm after N training examples can be given by (Blumer, Ehrenfeucht, Haussler, & Warmuth, 1989):

$$\varepsilon \leq \frac{\ln\left(\frac{1}{\delta}\right) + \ln(|H|)}{N}$$

The PAC model deals with distribution-free, worst-case analyses. As a consequence, the number of examples required to guarantee learning a concept in the worst-case does not accurately reflect the number of examples required to learn an accurate concept in practice. For example, Figure 1 plots the upper bound on the error predicted by the PAC model (with $\delta = 0.05$) for any algorithm that searches a 5-CNF space when the training instances are represented by 5 variables. The mean error for learning the concept $a \wedge (b \vee c) \wedge (b \vee d \vee e)$ with the *k*-CNF algorithm ($k = 5$) proposed in Valiant (1984) is shown as a function of the number of training examples. The error is averaged over 100 random sequences of positive training examples. The error is obtained by testing the result of the learning algorithm on 1000 training examples at each point indicated. In addition, the expected error predicted by the average case model presented here is shown.

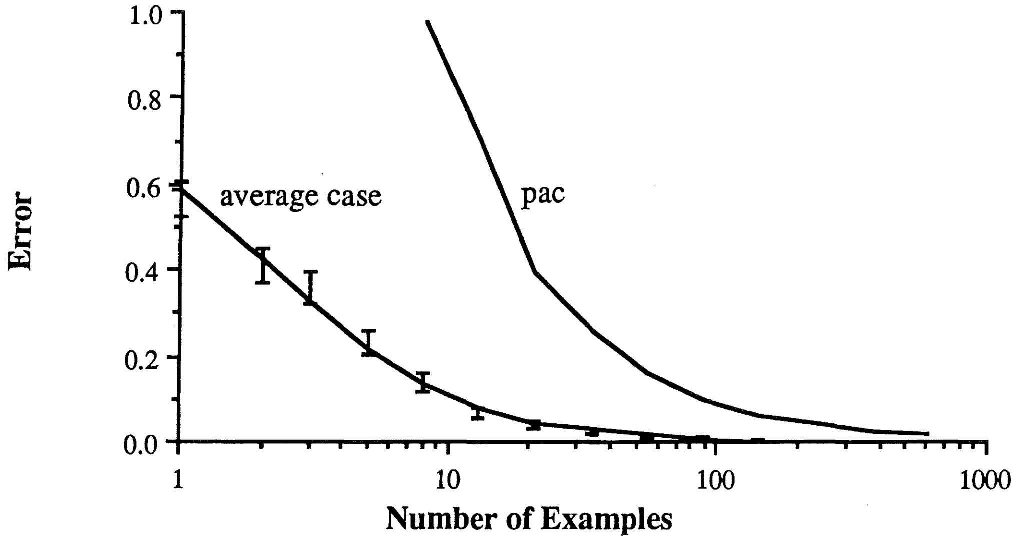

Figure 1: A comparison of the observed error of the *k*-CNF algorithm to the worst-case error predicted by the PAC model and the average case error presented in this paper. The bars are the 95% confidence interval around the mean error. The curves are the bound predicted by the PAC model and expected error predicted by the average case model. Note that to facilitate comparison a log scale is used for the number of training examples.

The experimental result supports the average case model (since the expected error is within a 95% confidence interval around the observed mean error) and the PAC model (since the observed error is always less than the upper bound on the error). There are a variety of reasons for the gap between the observed error and the result on the PAC model. The first, and most important, reason is that the PAC model is not intended to provide any insight into the mean, observed error. Rather, the goal of PAC learning has been to provide an upper bound on the number of examples to learn a concept with high probability to a given accuracy.

Second, the bounds predicted by the PAC model may be tightened somewhat. Buntine (1989) has argued that the Valiant model can produce overly-conservative estimates of error and does not take advantage of information available in actual training sets.

Finally, the PAC model is a distribution-free model. In our simulation, there was a known, fixed distribution of training examples. It may be possible to tighten the bounds by specializing the PAC model for a given distribution. Although there has been some research in this area (e.g., Benedek & Itai, 1987; Kearns, Li, Pitt, & Valiant, 1987; Natarajan, 1987), it has concentrated on showing that certain concepts classes are learnable from a polynomial number of examples given a known or uniform distribution of examples, rather than providing tighter bounds on those concepts that are PAC-learnable.

2 An Average Case Learning Model for *k*-CNF

A restricted version of conjunctive normal form, *k*-CNF, provides a more expressive language of hypotheses than the language of pure conjunctions that we analyzed previously (Pazzani & Sarrett, 1990). Hypothesis in *k-CNF* can be expressed as conjunctions of disjunctions of length at most *k*. Pure conjunctive hypotheses can be viewed as a special form of *k*-CNF (i.e., k=1). For simplicity, we restrict our attentions to monotone *k-CNF* (i.e., *k*-CNF in which no feature is negated).

The goal of the average case model is to predict the expected error as a function of the number of training examples. The framework requires determining:

1. The conditions under which the algorithm changes the hypothesis for a concept.
2. How often these conditions occur.
3. How changing a hypothesis affects the accuracy of a hypothesis.

In order to calculate the error of a 2-CNF algorithm learning D^*, it is necessary to compute the probability that a randomly drawn example is positive and the probability that a randomly drawn positive example contains various combinations of features. We introduce the following notation:

P the probability that a randomly drawn example is a member of S under the assumption that values of each feature are determined independently with probability p_j that feature f_j has a true value.

P_i the probability that a randomly drawn example, X, is a member of S and the i-th feature of X has a true value. Similarly, $P_{\bar{i}}$ is the probability that X is a member of S and the i-th feature of X has a false value. Furthermore, P_{ij} is the probability that X is a member of S and the i-th and the j-th features of X both have true values. This notation, $P_{i\ldots}$, generalizes to any number of subscripts and combinations of negated and unnegated subscripts.

P and $P_{i\ldots}$, are calculated in the following manner. Let X be a randomly drawn positive example. Define the weighted size of a set A of positive examples to be the probability that X is a member of set A. We will use $|A|$ to denote $Pr[X \in A]$. If $A = B \oplus C$, where $\oplus$ denotes union of disjoint sets, then $|A| = |B| + |C|$. We note that if $A = B \cup C$, where B and C are not necessarily disjoint, then $|A| = |B| + |\bar{B} \cap C|$. Define $\{\beta(i,j\ldots,)\}$ to be the set of examples for which the Boolean expression $\beta(i,j\ldots,)$ on the indicated features is true. For example, $\{i \vee \bar{j}\}$ is the set of examples for which either f_i is true or f_j is false. Then, $|\{\beta_1 \vee \beta_2\}| = |\{\beta_1\} \oplus \{\bar{\beta}_1\beta_2\}| = |\{\beta_1\}| + |\{\bar{\beta}_1\beta_2\}|$.

To calculate $P = Pr[X \in \bigwedge_{(i,j) \in D^*} (f_i \vee f_j)]$ we need to evaluate:

$$\left| \bigcap_{(i,j)\in D^*} \{i \vee j\} \right| = \left| \bigcap_{(i,j)\in D^*} \{i\} \oplus \{\bar{i}j\} \right|$$

This intersection can be converted symbolically to the union of a number of disjoint sets. The set is initialized to $\{i_1\} \oplus \{\bar{i}_1 j_1\}$. This set is intersected with $\{i_2\} \oplus \{\bar{i}_2 j_2\}$ to form: $\{i_1 i_2\} \oplus \{\bar{i}_1 j_1 i_2\} \oplus \{i_1 \bar{i}_2 j_2\} \oplus \{\bar{i}_1 j_1 \bar{i}_2 j_2\}$. This process is repeated for each $(i,j) \in D^*$. Although in the worst case, this will result in the union of 2^c disjoint sets (where c is the cardinality of D^*), in practice it is substantially smaller due to cancellation of terms. For example, finding the intersection of all conjunctions of disjunctions of exactly 3 of 5 features requires computing the union of 10 disjoint sets instead of 2^{10} sets:

$i_1 i_2 i_3 \oplus i_1 i_2 \bar{i}_3 i_4 \oplus i_1 i_2 \bar{i}_3 \bar{i}_4 \bar{i}_5 \oplus i_5 \bar{i}_1 i_4 \bar{i}_2 i_3 \oplus \bar{i}_1 \bar{i}_4 i_5 i_2 i_3 \oplus \bar{i}_1 i_4 i_2 i_3 \oplus i_5 \bar{i}_1 i_2 \bar{i}_3 i_4 \oplus i_1 i_5 \bar{i}_2 \bar{i}_3 i_4 \oplus i_1 i_4 \bar{i}_2 i_3 \oplus i_1 i_5 \bar{i}_4 \bar{i}_2 i_3$

Once a CNF has been converted to this form, the probability that the CNF is true for a randomly drawn example can be found by summing the probabilities that it is a member of one of the disjoint sets described by a conjunction of features (or the inverses of features). Since we assume that individual features are independent, the probability that a conjunction of features can be found by multiplying the corresponding values of p_i. Computing P requires converting D^* to this format and then summing the products of the probabilities.

The computation of $P_{i...}$ is similar. For example, $P_{a\bar{b}}$ is the ratio of

$$Pr[X \in \{a\bar{b}\} \;\&\; X \in \bigwedge_{(i,j) \in D^*} (f_i \vee f_j)] \text{ and } Pr[X \in \bigwedge_{(i,j) \in D^*} (f_i \vee f_j)].$$

The numerator can be calculated by symbolically intersecting $a\bar{b}$ with the union of the disjoint sets formed from D^*. For example, $i_1 \bar{i}_6 \cap (i_1 i_2 i_3 \oplus \bar{i}_1 i_2 \bar{i}_3 i_4 \oplus \bar{i}_2 \bar{i}_4 \bar{i}_5) = i_1 i_2 i_3 \bar{i}_6 \oplus i_1 \bar{i}_2 \bar{i}_4 \bar{i}_5 \bar{i}_6$.

The k-CNF algorithm has only one operator to revise a hypothesis. A disjunctive term is dropped from the hypothesis when the term is false in a positive training example. This is modeled by removing pairs from I_{n-1} to form I_n. We will use the notation $r_d(n)$ to indicate the probability that the disjunction corresponding to $d = (i,j)$ remains in I_n. This occurs only if at least one of f_i and f_j has had a true value in all n positive training examples. Similarly, $r_{d_1 d_2}(n)$ indicates the probability that the disjunctions corresponding to d_1 and d_2 both remain in I_n. Note that $r_{d_1}(n) = Pr[X \in \{i_1 \vee j_1\}]$ and $r_{d_1,\ldots,d_h}(n) = Pr[X \in \{(i_1 \vee j_1) \wedge \cdots \wedge (i_h \vee j_h)\}]$. The values can be calculated in the same manner as P.

Note that misclassifying a positive example as a negative example is the only form of error made by the k-CNF algorithm. The hypothesis created by the k-CNF algorithm misclassifies a positive test example if there is at least one pair (i,j) in I_n such that f_i and f_j are both false in the test example. We will use the notation e_n to represent the probability that an example is misclassified after n training examples. $e_n = e_n(1) - e_n(2) + e_n(3) - e_n(4) \ldots e_n(h)$ where h is the cardinality of I_0 and $e_n(a)$ is the probability that an example is misclassified after n examples because it is contra-indicated by at least a of the pairs in I_0.

$$e_n(1) = \sum_{d=(i,j) \in I_0} r_d(n) P_{\bar{i}\bar{j}}$$

$$e_n(a) = \sum_{\substack{d_1 \ldots d_a = (i_1 j_1) \ldots (i_a j_a) \\ \in\ partitions\ (I_o, a)}} r_{d_1 \ldots d_a}(n) P_{\bar{i}_1\bar{j}_1 \ldots \bar{i}_a\bar{j}_a}$$

where *partitions(I,a)* is the set of all subsets of I with length a. In effect, $e_n(1)$ calculates the probability that each pair from I_0 is a pair of I_n weighted by the probability that a randomly drawn training example would be misclassified by that disjunction corresponding to that pair. For larger values of i, $e_n(i)$ corrects $e_n(i-1)$ by taking into consideration the fact that more than $i-1$ of the pairs in I_n result in a misclassification.

Two optimizations simplify the calculation of e_n. First, if an approximation of the error will suffice, it is not necessary to calculate the value of $e_n(j)$ if $e_n(i)$ is less than a threshold and j is greater than i. In Figure 1, we set this threshold to 0.005. Second, many values of $P_{\bar{i}_1\bar{j}_1 \ldots \bar{i}_a\bar{j}_a}$ are 0. This occurs if there is a pair (i,j) in D^* and $\bar{i}$ and $\bar{j}$ are both subscripts of $P_{\bar{i}_1\bar{j}_1 \ldots \bar{i}_a\bar{j}_a}$. Furthermore, if $P_{\bar{i}_1\bar{j}_1 \ldots \bar{i}_{a-1}\bar{j}_{a-1}}$ is 0, then for all i_a and j_a, $P_{\bar{i}_1\bar{j}_1 \ldots \bar{i}_{a-1}\bar{j}_{a-1}\bar{i}_a\bar{j}_a}$ is 0.

2.2 Extending the model for *k*-CNF

The generalization of the model to *k*-CNF is fairly straightforward. First, D^* is now a set of tuples of at most length k. Each tuple corresponds to a set of features corresponding to one of the terms of the correct hypothesis. D_0 (the initial hypothesis) is defined to be the set of all tuples of at most k of the features. From here, the notation of the previous section needs to be extended somewhat, but the technique for calculating P, $r(n)$, and $e(n)$ remains unchanged.

2.3 Extending the model for negative examples

As in Pazzani and Sarrett (1990), this analysis is easily extended to the case where there are both positive and negative examples. Recall that the algorithm does not update its hypothesis when presented with negative examples. Section 3.1 described how to calculate P, the probability that a randomly drawn example is a positive example, from the values of p_i. Once P has been determined, the binomial formula can be used to calculate the error of the *k*-CNF algorithm by calculating a weighted sum of the error times the probability that there are exactly n positive examples for each value of n from 0 to T:

$$\sum_{n}^{T} \binom{T}{n} * e_n P^n (1 - P)^{(T-n)}$$

2.4 Experimentally evaluating the average case model

We experimentally evaluate the average case model by comparing the observed error of the k-CNF algorithm on a given problem to the error predicted by the model for this problem. A problem is specified by providing D^* (the correct answer), m (the number of features used to describe the training examples) and values for p_j (the probability that each feature has a true value in a randomly drawn example). We have tested the model on a wide variety of problems. Figure 1 provides one such comparison. The following section shows how the model can be used to predict the influence of factors that affect the error of the learning algorithm and compares the observed and predicted error.

3 Implications of the Average Case Model for *k*-CNF

In this section, we show how the average case model for *k*-CNF can be used to gain an understanding of some factors that affect the error of the learning algorithm. In particular, we address how the error is increased if an additional irrelevant feature is added to the representation of the training examples.

We consider a relatively simple problem. Consider trying to learn the concept, $i_1 \wedge i_2$ with a 2-CNF learning algorithm. If there are 3 features, then the 2-CNF algorithm will converge on the equivalent (but unsimplified) hypothesis: $i_1 \wedge i_2 \wedge (i_1 \vee i_2) \wedge (i_1 \vee i_3) \wedge (i_2 \vee i_3)$. The set I_0 will contain the singleton $\{(i_3)\}$. With 0 training examples, the initial hypothesis will produce an error on $(1-p_3)$ of the positive examples (since an error is made when i_3 is false and i_3 is false in this proportion of training examples). After n training examples, the hypothesis will produce an error on $(p_3)^n(1-p_3)$ of the positive training examples (since the probability that i_3 is in I_0 is $(p_3)^n$). Figure 2 (lower curve) graphs the observed and expected error under these conditions (with $p_3 = 0.5$).

If there are 4 features, then the 2-CNF algorithm will converge on the equivalent (but unsimplified) hypothesis: $i_1 \wedge i_2 \wedge (i_1 \vee i_2) \wedge (i_1 \vee i_3) \wedge (i_1 \vee i_4) \wedge (i_2 \vee i_3) \wedge (i_2 \vee i_4)$. The set of terms that can cause errors is now $\{(i_3)(i_4)(i_3 \vee i_4)\}$. The initial hypothesis will produce an error on $(1 - p_3p_4)$ of the positive examples (since an error is made when i_3 is false or when i_4 is false). After n training examples, the hypothesis will produce an error on $(p_3)^n(1-p_3) + (p_4)^n(1-p_4) - (p_3p_4)^n(1-p_3p_4)$ of the positive training examples. Figure 2 also graphs the observed and expected error with a total of 4, 5, 6, and 7 features (with all $p_i = 0.5$). Note that for this problem, the analysis is equivalent to learning the always true concept (i.e., $D^* = \{\}$) while ignoring the two features i_1 and i_2. This occurs because both features must appear in every positive example.

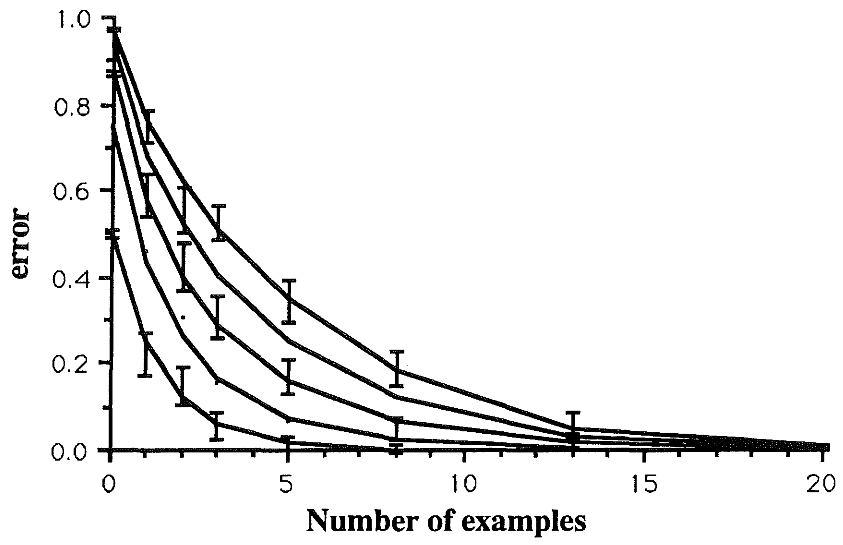

Figure 2. Expected and observed error when learning $i_1 \wedge i_2$ with a 2-CNF algorithm when there are a total of 3 (lowest curve) 4, (next to lowest), 5 (middle), 6 (next to upper) and 7 features (upper curve). The curves represent predicted values and the bars are 95% confidence intervals around the mean values. To avoid clutter, confidence intervals are not shown for 4 and 6 total features.

4 Future Directions for Average Case Modeling

So far, we have provided an analytic method that allows us to predict the average case analysis of Valiant's *k*-CNF learning algorithm. In the previous section, we have shown how the analysis can lead to insight into the effect of increasing the number of features on a given learning learning problem.

In the future, we intend to quantitatively answer questions such as how the error is increased if the value of *k* is larger than necessary. For example, a 2-CNF concept can be learned by a 5-CNF learning algorithm. However, the 5-CNF algorithm searches a larger hypothesis space and one would expect it to be less accurate than a 2-CNF algorithm on the same data. Figure 3 displays predictions of the average case model and results obtained by running the algorithm on sample data sets.

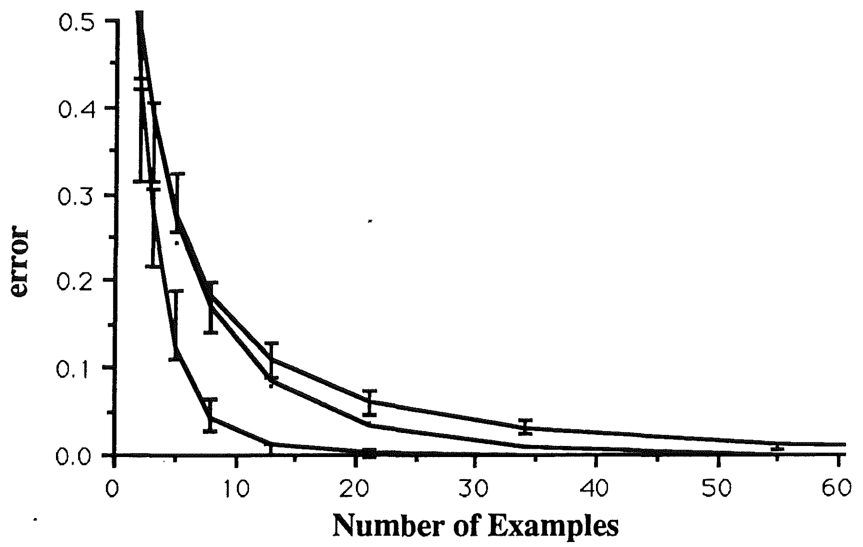

Figure 3. Expected and observed error (with 95% confidence interval) for learning $i_1 \wedge i_2$ when there are a total of 5 features with a 1-CNF algorithm (lower curve), a 2-CNF algorithm (middle curve), and a 5-CNF algorithm (upper curve). To avoid clutter, confidence intervals are not shown for the 2-CNF case

A second future direction consists of comparing the predictions made by the model to results of learning on realistic data sets. We have recently examined the database of congressional voting records stored at the University of California, Irvine and have found that a 3-CNF algorithm can achieve 100% accuracy on distinguishing Democrats from Republicans based upon 16 Boolean features representing votes on certain issues. Such an analysis would provide important feedback on whether the assumptions made by the model are realistic and provide experimental support for the robustness of the model when data sets deviate from these assumptions.

Finally, we intend on creating average case models for additional learning algorithms, such as the *k*-DNF algorithm and decision list algorithms.

5 Conclusion

We have presented an approach to modeling the average case behavior of an algorithm for learning k-CNF. The model predicts the expected error of the algorithm as a function of the number of training examples. We evaluated the average-case model by comparing the error predicted by the model to the actual error obtained by running the learning algorithm. We have shown the analysis can lead to insight into factors that affect the error of the learning algorithm.

The average case model requires much more information about the training examples than the PAC learning model. The information required by the model is exactly the information required to generate artificial data to test learning algorithms. One future research direction would be to relax some of these assumptions. For example, rather than requiring the correct concept definition, it might be possible to perform a similar analysis for a given probability distribution of possible concepts.

Acknowledgements

We would like to thank Dennis Kibler for helpful comments on this work and Caroline Ehrlich for reviewing an earlier draft of this manuscript. This research is supported by a National Science Foundation Grant IRI-8908260 and by the University of California, Irvine through an allocation of computer time.

References

Benedek, G., & Itai, A. (1987) Learnability by fixed distributions. *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp 81-90). Boston: MA: Morgan Kaufmann.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association of Computing Machinery, 36,* 929-965.

Buntine, W. (1989). A Critique of the Valiant model. *Proceedings of the Eleventh Joint Conference on Artificial Intelligence* (pp.837-842). Detroit, MI: Morgan Kaufmann.

Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2,* 139-172.

Haussler, D. (1987). Bias, version spaces and Valiant's learning framework. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 324-335). Irvine, CA: Morgan Kaufmann..

Kearns, M., Li, M., Pitt, L., & Valiant, L. (1987). On the learnability of Boolean formula. *Proceedings of the Nineteenth Annual ACM Symposium on the Theory of Computing* (pp. 285-295). New York City: NY: ACM Press.

Pazzani, M., & Sarrett, W. (1990) Average case analysis of conjunctive learning algorithms. *Proceedings of the Seventh International Workshop on Machine Learning*, Austin, TX: Morgan Kaufmann.

Natarajan, B. (1987). On learning Boolean formula. *Proceedings of the Nineteenth Annual ACM Symposium on the Theory of Computing* (pp. 295-304). New York: ACM Press.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In D. Rumelhart & J. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations* (pp 318-362). MIT Press.

Valiant, L. (1984). A theory of the learnable. *Communications of the Association of Computing Machinery, 27,* 1134-1142.