Average Case Analysis of k-CNF and k-DNF learning algorithms

Daniel S. Hirschberg Michael J. Pazzani Kamal M. Ali

July 1, 1994

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717, USA
dan@ics.uci.edu pazzani@ics.uci.edu ali@ics.uci.edu
(714) 856-5888

Abstract

We present average case models of algorithms for learning Conjunctive Normal Form (CNF, i.e., conjunctions of disjunctions) and Disjunctive Normal Form (DNF, i.e., disjunctions of conjunctions). Our goal is to predict the expected error of the learning algorithm as a function of the number n of training examples, averaging over all sequences of n training examples. We show that our average case models accurately predict the expected error and demonstrate that 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 average case models of an algorithm for learning monotone k-CNF concepts and an algorithm for learning monotone k-DNF concepts.

1.1 The PAC Framework

The framework presented in this paper attempts to unify the formal mathematical and the experimental approaches to understanding machine learning algorithms. To achieve this unification, an average case model is needed to predict the expected error[1] of a learning algorithm as a function of the number of training examples. In contrast, the probably approximately correct (PAC) learning model (e.g., Blumer, Ehrenfeucht, Haussler, & Warmuth, 1989; Haussler, 1987; Valiant, 1984) has a different goal. The goal of the PAC model is to determine whether a given algorithm can learn an arbitrary concept from a given concept class, given a sufficient number of examples. Following the definition given in Haussler (1987), an algorithm $\mathcal{A}$ is a *polynomial learning algorithm* for a class of hypothesis spaces $\mathbf{H}$ if for all ϵ and δ such that $0 < \epsilon, \delta < 1$ and $n, s \geq 1$ there exists a sample size $S_A(\epsilon, \delta, n, s)$ polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, n and s, such that for all domains X of complexity n, all probability distributions P on X and all target concepts $c \in H_X$ of complexity at most s, given a random sample[2] of c of size $S_A(\epsilon, \delta, n, s)$ drawn independently according to P, $\mathcal{A}$ produces a hypothesis in H_X that, with probability at least $1 - \delta$, has error at most ϵ.

Note that the distribution from which the training examples are drawn is not specified. In particular, the system is deemed able to learn the concept only if it can learn from any distribution. Furthermore, the PAC model requires that the sample size required to learn a concept grow polynomially with respect to the following four quantities: $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, a measure of the complexity of the instance space, and a measure of the complexity of the target concept.

[1] The expected or mean error $E_{\mathcal{A},C,\mathcal{D},n}$ of an algorithm $\mathcal{A}$ learning a concept C after training on n examples independently drawn according to some distribution $\mathcal{D}$ is the expectation of making a classification error on one (test) example independently drawn from the instance space according to $\mathcal{D}$.

[2] Sample refers to a set of training examples and sample size refers to some measure of that set, usually the cardinality.

Although the PAC model has been used to predict the accuracy of an algorithm as a function of the number of training examples (Haussler, 1990), it is primarily used to show that certain classes of concepts are not learnable and concerns itself with learnability in the limit.

A model related to the PAC model has been proposed to deal more directly with analyzing the worst case error of learning algorithms (Haussler, Littlestone, & Warmuth, 1990; Hembold, Sloan, & Warmuth, 1990). This model is designed to predict the probability of a classification error on the N+1st example after being trained on N training examples.

1.2 A framework for predicting the mean error rate

In this paper we present a framework for analyzing learning algorithms that will yield the expectation of making a classification error, given the learning algorithm, the target concept, a distribution on the instance space and the number of training examples.

The goal of the analysis here is to predict the mean error of a given algorithm on a given problem from a given distribution. The average case analysis[3] can be used to compare the expected error rate of different algorithms on the same problem. As a consequence, the average case model can be used to show that one learning algorithm will, on average, converge to the correct solution more rapidly than another. In addition, it can be used to prove that the asymptotic accuracy of one algorithm is greater than that of another.

Our framework for average case analysis requires us to determine:

1. The conditions under which the algorithm changes its hypothesis of a concept.

2. How often these conditions occur.

3. How changing a hypothesis affects the accuracy of that hypothesis.

These criteria are instantiated for the learning algorithms under consideration in sections 2.1 and 2.2.

[3]In our analysis, we average over all training sequences of N examples, not over different target concepts or distributions.

1.3 A Comparison of the frameworks

The PAC model and the model used to predict the probability of a classification error on the N+1st example are worst case, distribution-free models. It may be possible to tighten the bounds on the error rate by specializing the models for a given distribution. Although there has been some distribution-specific research (e.g., Benedek & Itai, 1987; Kearns, Li, Pitt, & Valiant, 1987; Natarajan, 1987), it has concentrated on showing that certain concepts are PAC-learnable from a polynomial number of examples given a known or uniform distribution of examples, rather than providing tighter bounds on the rate of classification error made by those algorithms. Blumer *et al.* (1989), for example, give the following upper bound on the mean error

$$\frac{VCdim(h)}{N}$$

where h is the hypothesis space, N is the number of training examples, and $VCdim(h)$ is the Vapnik-Chervonenkis dimension of that hypothesis space (Vapnik & Chervonenkis, 1971). Other distribution-specific work on bounding the amount of error has been done by Haussler *et al.* (1990). Theorem 3.1 in Haussler *et al.* (1990) yields the following lower bound on the mean error that can be achieved by any algorithm on the worst case distribution:

$$\frac{max(1, VCdim(h) - 1)}{2eN}$$

given that $VCdim(h)$ is finite and $N > VCdim(h)$.

However, this analysis pertains to the worst case distribution of training examples. One would expect the mean error rate curve to be lower for other distributions. As Figures 1a and 1b illustrate, a model such as ours, that is able to accept the the concept and the distribution as parameters, can make mean error predictions that are much closer to the empirically observed error rate. Theorem 3.1 makes mean error rate predictions assuming the worst case distribution, so its predictions are further than ours from the empirically observed average error rate. Furthermore, as Figure 1b shows, the functional form of the error curve predicted by Theorem 3.1 is qualitatively different from the error curve that is empirically observed. A common way used to describe how well data fits a model is to

use the coefficient of determination[4] (Winer, 1971). For the curves in Figure 1a, r^2 has a value of 0.9493 for Theorem 3.1 and a value of 0.9996 for our average case model, indicating that the average case model fits the data more closely than does Theorem 3.1. Although this comparison is limited to a specific distribution and concept, we believe our model would make mean error rate predictions that are closer than that of the PAC model to the empirically observed values for a wide range of concepts and distributions because it is able to exploit knowledge of the target concept and distribution.

Figure 1a graphs mean error rate predictions for learning the 2-CNF expression $i_1 \wedge i_2$ over the instance space of 5 Boolean features, given that each feature is independent and takes on the value TRUE with probability 0.5. Note that Theorem 3.1 only applies when $N > VCdim(h)$, which is 10 for this example, and that the error rates predicted by Theorem 3.1 do not depend on the concept being learned; they are solely determined by the hypothesis space and N. In particular, Theorem 3.1 would predict the same error rate curve for any two concepts in that hypothesis space whereas our model would predict different error rates.

INSERT FIGS 1A AND 1B ABOUT HERE

2 Average Case Learning Models for k-CNF and k-DNF

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). Hypotheses in k-CNF can be expressed as conjunctions of disjunctions,

[4]The coefficient of determination is defined as follows (where $\overline{O}$ is the average value of O_i):

$$r^2 = \frac{\sum_i (O_i - P_i)^2}{\sum_i (O_i - \overline{O})^2}$$

The use of this coefficient is not limited to linear regression, it can be also used to describe the goodness of fit of non-linear models to data.

where the disjunctions are of length at most k. Similarly, hypotheses in k-DNF can be expressed as disjunctions of conjunctions of length at most k. For simplicity, we restrict our attentions to monotone k-CNF and k-DNF (i.e., forms in which no feature is negated). We will restrict our attention to the algorithms proposed in Valiant (1984) (see Table 1 and Table 2).

The k-CNF algorithm initializes the hypothesis to the most specific k-CNF concept and gradually makes the hypothesis more general by deleting conjuncts that are not consistent with positive training examples. For example, if there are 3 features (a, b and c) then the initial hypothesis for a monotone 3-CNF algorithm is:

$$a \wedge b \wedge c \wedge (a \vee b) \wedge (a \vee c) \wedge (b \vee c) \wedge (a \vee b \vee c).$$

If the first positive training example is $a \wedge \overline{b} \wedge \overline{c}$ then the hypothesis will be revised to

$$a \wedge (a \vee b) \wedge (a \vee c) \wedge (a \vee b \vee c).$$

Similarly, the k-DNF algorithm initializes the hypothesis to the most general k-DNF concept over the given instance language, and gradually makes the hypothesis more specific by deleting conjunctive terms that are consistent with negative training examples.

In the following two sections, we apply our framework to analyzing these algorithms for learning 2-CNF and 2-DNF concepts from a distribution in which feature values (all features are Boolean) of the training examples are selected independently. Thus the instance space is $\{0,1\}^C$ where C is the number of Boolean features. We use our framework to predict the mean error rate of these algorithms and compare the predictions to empirically derived average error rates.

2.1 A model for 2-CNF

PUT TABLE 1 AROUND HERE.

We will first analyze the algorithm in Table 1 for $k = 2$, using only positive, independently drawn examples. The following notation describes the problem, independent of the

algorithm.

C^* The CNF expression for the target concept.

f_j The j-th feature of an example.

p_j The probability that the j-th feature has a true value.

m The total number of features used to describe an example.

We will use the following notation to describe the analysis of the algorithm:

D^* The set of pairs (i, j) that are in the target concept. For 2-CNF, (i, j) would be in D^* if the disjunction $f_i \vee f_j$ was a conjunct of the target concept.

D_n The set of pairs $\{(i, j) | i < j\}$ representing the hypothesis after n positive training examples. Note that $D_0 \supseteq \ldots D_n \supseteq D^*$. See Table 1 for initialization of D_0.

I_n The subset of pairs from D_0 that are in the learned hypothesis after n examples but are not in the target concept.

C_n The evolving hypothesis (which becomes increasingly general for the CNF algorithm).

S The set of all positive examples.

Finally, we introduce notation that is specific to the 2-CNF algorithm learning D^*.

P The probability that a randomly drawn example is a member of S. Let $\{i_1 \wedge \overline{i_2}\}$ denote the set of examples for which $i_1 \wedge \overline{i_2}$ is true. (This notation generalizes to any number of negated and unnegated subscripts.) Then $P = Pr[X \in \{(i_1 \vee j_1) \wedge \ldots \wedge (i_h \vee j_h)\}]$ where $(i_1 \vee j_1) \wedge \cdots \wedge (i_h \vee j_h)$ is the target concept.

$P_{i,\ldots,j}$ The probability that the features f_i through f_j are true for the example X, given that X is a member of S. We need this term to calculate the probability that the learned hypothesis will make an error on a single positive test example. Note that some of the indices may be negated.

$$P_{i,\ldots j} = Pr[X \in \{i \wedge \ldots \wedge j\} \mid X \in S]$$

$r_{d_1 \cdots d_h}(n)$ The probability that each of the extraneous conjuncts $d_1 ... d_h$ have been true for n positive examples. We need this term to determine the probability of getting to some incorrect approximation to the target concept.

$$r_{d_1 \cdots d_h}(1) = Pr[X \in \{(i_1 \vee j_1) \wedge \cdots \wedge (i_h \vee j_h)\} \mid X \in S]$$

where $r_{d_1 \cdots d_h}(n)$ is the n-th power of $r_{d_1 \cdots d_h}(1)$ because the examples are independently drawn with replacement.

Note that misclassifying a positive example is the only form of error made by this 2-CNF algorithm. The learned hypothesis 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 a test example is misclassified after n training examples. So e_n represents the mean error after learning on n examples, and it is the overall quantity that our average case analysis aims to estimate. Note that e_n is summing across all possible incorrect learned hypotheses. One way to calculate e_n is to use inclusion-exclusion:

$$e_n = e_n(1) - e_n(2) + e_n(3) - e_n(4) ... e_n(|D_0| - |D^*|)$$

where $|D_0| - |D^*|$ is an upper bound on the number of possible erroneous disjunctive terms that could survive after n examples and $e_n(\mathrm{a})$ is calculated by the following:

$$e_n(a) = \sum_{d_1 .. d_a = (i_1, j_1) .. (i_a, j_a) \in partitions(I_0, a)} r_{d_1 \cdots d_h}(n) \, P_{\overline{i}_1 \overline{j}_1 \cdots \overline{i}_a \overline{j}_a}$$

where $partitions(I, a)$ is the set of all subsets of I with length a. $e_n(1)$ calculates the probability of getting an error from some learned hypothesis containing exactly one erroneous disjunction. $r_d(n)$ is the probability that a learned hypothesis containing the disjunction d survives n examples and $P_{\overline{ij}}$ is the probability that the learned hypothesis containing $f_i \vee f_j$ makes an error. Thus $e_n(1)$ 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 the disjunction corresponding to that pair. For larger values of i, $e_n(i)$ corrects $e_n(i-1)$

by taking into consideration the possibility when more than $i-1$ of the pairs in I_n result in a misclassification.

An optimization simplifies the calculation of e_n. For many subscript combinations, $P_{\overline{i}_1\overline{j}_1\cdots\overline{i}_a\overline{j}_a}$ is 0. This occurs if there is a pair (i_k, j_k) in D^* and $\overline{i_k}$ and $\overline{j_k}$ are both in the set of subscripts of $P_{\overline{i}_1\overline{j}_1\cdots\overline{i}_a\overline{j}_a}$. Furthermore, if $P_{\overline{i}_1\overline{j}_1\cdots\overline{i}_{a-1}\overline{j}_{a-1}}$ is 0, then for all i_a and j_a, $P_{\overline{i}_1\overline{j}_1\cdots\overline{i}_a\overline{j}_a}$ is 0.

2.2 An Average Case Model for 2-DNF

PUT TABLE 2 AROUND HERE

Our model for the 2-DNF algorithm shown in Table 2 follows straightforwardly from the 2-CNF model. As is true for CNF, $D_0 \supseteq \ldots D_n \supseteq D^*$, but now the evolving hypothesis is becoming increasingly specific. Note that only negative examples are needed for learning and that the only kinds of errors possible are errors of commission.

We revise the notation of section 2.1 with D^* being reinterpreted to mean the set of pairs (i, j) such that the conjunction $f_i \wedge f_j$ is a disjunct of the target concept:

$\overline{P}$ The target concept C^* is now in DNF form but we want $Pr[X \in S]$ which can be calculated by inverting C^* into non-monotone CNF form and then calculating the probability using the techniques in section 2.1.

$P_{i,\ldots,j}$ This is now the conditional probability that the i-th feature (and so on) are true given that the example X is **not** a member of S. The definition of $P_{i,\ldots,j}$ for k-DNF is

$$Pr[X \in \{i..\}|\overline{S}] = \frac{Pr[X \in \{i..\} \wedge \overline{S}]}{\overline{P}}$$

$r_{d_1\cdots d_h}(n)$ The probability that each of the extraneous disjuncts d_i were false for n negative examples. This is the n-th power of $r_{d_1\cdots d_h}(1)$ which (for DNF) is

$$r_{d_1\cdots d_h}(1) = Pr[\overline{i_1 j_1 \ldots i_h j_h} \mid \overline{S}]$$

Once again, inclusion-exclusion is used to calculate the mean error for the 2-DNF algorithm. The only difference is that now an error is made when an extraneous disjunct is true, so the indices to P are unnegated:

$$e_n(a) = \sum_{d_1..d_a=(i_1,j_1)..(i_a,j_a)\in partitions(I_0,a)} r_{d_1\cdots d_h}(n)\, P_{i_1 j_1..i_a j_a}$$

2.3 Extending the models to k-CNF and k-DNF

The generalization to k-CNF and k-DNF is made by allowing D^* to contain k-tuples. D_0 is initialized to all subsets of size at most k over the m features. The techniques for calculating P, $r(n)$ and $e(n)$ remain the same. In addition, as in Pazzani and Sarrett(1990), the model can be extended to handle negative training examples (in the k-CNF case) and positive training examples (in the k-DNF case). These were omitted from the initial analysis since the learning algorithms ignore these examples. The binomial distribution is used to calculate the probability of getting exactly p positive and n negative examples from a total set of T training examples. Hirschberg and Pazzani (1992) extend the k-CNF model to deal with noise in the training data.

3 Implications of the Average Case Model learning for k-CNF and k-DNF concepts

In this section, we show how the average case models can be used to gain an understanding of some factors that affect the classification error of the learning algorithm. If the value of k is larger than necessary (learning a 2-CNF concept with a 5-CNF algorithm, for example) the mean error rate curve decreases more gradually (as N increases) for $k = 5$ than it does for $k = 2$. The 5-CNF algorithm searches a larger hypothesis space and one would expect it to be less accurate than a 2-CNF algorithm given the same concept and instance space.

INSERT FIGURE 2 AROUND HERE

Here, 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 3 features (i_1, i_2, i_3) are used to represent the examples, then the 2-CNF algorithm will eventually converge on the following hypothesis: $i_1 \wedge i_2 \wedge (i_1 \vee i_2) \wedge (i_1 \vee i_3) \wedge (i_2 \vee i_3)$ which is truth-equivalent to the target concept $i_1 \wedge i_2$. 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 the fraction $1 - p_3$ of the positive training examples). After n training examples, the expected fraction of independently drawn positive examples on which the hypothesis will make an error will be $(p_3)^n(1 - p_3)$ (since the probability that i_3 has been true for n positive examples even when i_3 is not a part of the target concept is $(p_3)^n$). The lower curve in Figure 2 graphs the observed and expected error under these conditions (with $p_i = 0.5$ for all i.)

If there are 4 features in the input representation, and the concept to be learned is still $i_1 \wedge i_2$ then the 2-CNF algorithm will converge on the hypothesis $i_1 \wedge i_2 \wedge (i_1 \vee i_2) \wedge (i_1 \vee i_3) \wedge (i_1 \vee i_4) \wedge (i_2 \vee i_3) \wedge (i_2 \vee i_4)$ which is truth-equivalent to the target concept. The set of terms from D_0 (when the instances are represented by 4 features) that can cause errors is $\{i_3, i_4, i_3 \vee i_4\}$. The initial hypothesis will produce an error on $(1 - p_3 p_4)$ 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 expected fraction of independently drawn positive examples on which the hypothesis will make an error will be $(p_3)^n(1 - p_3) + (p_4)^n(1 - p_4) + (p_3 p_4)^n(1 - p_3 p_4)$. Comparing this expression to that for one irrelevant feature, $(p_3)^n(1 - p_3)$, one can see that the expected error for any n will increase as the number of irrelevant features increases. Note that the analysis for this concept is equivalent to the analysis of the "always true" concept (i.e., $D^* = \{\}$) over the instance space of $\{i_3, i_4\}$, ignoring the two features i_1 and i_2. This holds because both features i_1 and i_2 must appear in every positive example even when the examples are drawn from a space of 4 Boolean features. Figure 2 graphs the effect of varying the number of features in the input representation from 4 to 7 on the observed and expected error (all with

$p_i = 0.5$). Figure 3 illustrates a similar effect for the 2-DNF algorithm learning $i_1 \vee i_2$.

INSERT FIGURE 3 AROUND HERE

4 Conclusions

Previous research has used knowledge of distributions to show certain classes of concepts are learnable in the limit using a polynomial number of examples. The research that has concentrated on predicting mean error estimates has done so using a worst case distribution. We present a model that uses knowledge of a specifiable distribution to predict the mean error rates of a k-CNF algorithm and a k-DNF algorithm. We have illustrated how knowledge of the distribution allows error rate estimates that come closer to the empirically measured values than the error estimates made using a worst case assumption. The model also predicts that the average error will increase as irrelevant features are added to the instance space. This is true for both k-CNF and k-DNF concepts. Empirical measurements also bear this out and were used to validate the model. The model is able to make better error estimates because it exploits knowledge of the distribution and applies to a specific, although commonly used algorithm.

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

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

Haussler, D. 1987. *Applying Valiant's Learning Framework to AI Concept Learning Problems.* Technical Report UCSC-CRL-87-11, University of California, Santa Cruz.

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

Haussler, D. 1990. Probably Approximately Correct Learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, (pp. 1101-1108) Boston: AAAI Press.

Haussler, D. 1986. *Quantifying inductive bias in concept learning.* Technical Report UCSC-CRL-86-25, University of California, Santa Cruz.

Haussler, D., Littlestone, N. and Warmuth, M. 1990. *Predicting 0,1-functions on randomly drawn points.* Technical Reports USCS-CRL-90-54, University of California, Santa Cruz.

Hembold, D., Sloan, R., and Warmuth, M. 1990. Learning nested differences of intersection-closed concept classes, *Machine Learning*, 5, 165-196.

Hirschberg D., Pazzani M. 1992. Average Case Analysis of Learning k-CNF concepts. In *Proceedings of the Ninth International Workshop on Machine Learning*, (pp. 206-211) Aberdeen, UK: Morgan Kaufmann.

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

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

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

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

Valiant, L. 1985. Learning disjunctions of conjunctions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp 560-566). Los Angeles, CA: Morgan Kaufmann.

Vapnik, V. and Chervonenkis, A. 1971. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability with Applications*, 16, 264-280.

1. Initialize the hypothesis to the conjunction of all disjunctions of length at most k of the features that describe training examples.

2. If the new example is a positive example, and the hypothesis misclassifies the new example, then remove all disjunctions from the hypothesis that are false in the example.

TABLE 1

1. Initialize the hypothesis to the disjunction of all conjunctions of length at most k of the features that describe training examples.

2. If the new example is a negative example, and the hypothesis misclassifies the new example, then remove all conjunctions from the hypothesis that are true in the example.

TABLE 2

Table 1. The k-CNF learning algorithm.

Table 2. The k-DNF learning algorithm.

Figures 1a and 1b. Comparison of mean error predictions from Haussler's Theorem 3.1 to our average case model for the concept $i_1 \wedge i_2$. The instance space consisted of five Boolean features. Figure 1b plots the log of the error to illustrate that the functional form of the error curve as predicted by Theorem 3.1 is different from the functional form observed from empirical testing.

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 mean error rate values as predicted by our average case model. The circles represent the sample average error rate as determined by empirical tests. The bars are 95 percent confidence intervals around the empirically determined average error values. To avoid clutter, confidence intervals are not shown for the 4 feature and 6 feature cases.

Figure 3. Expected and observed error when learning $i_1 \vee i_2$ with a 2-DNF algorithm when there are a total of 3 (lowest curve), 4 (middle), and 6 features (upper curve). The curves represent mean error rates as predicted by our average case model. The circles represent sample average error rates as determined by empirical tests and the bars are 95 percent confidence intervals around the empirically determined error values. To avoid clutter, some confidence intervals are not shown.