Average Case Analysis of Empirical and Explanation-based Learning Algorithms

Wendy E. Sarrett (sarrett@ics.uci.edu)
Michael J. Pazzani (pazzani@ics.uci.edu)
Department of Information & Computer Science
University of California, Irvine, CA 92717

Technical Report 89–35

6 October 1989

This research is supported by a Faculty Research Grant from the Academic Senate Committee on Research of the University of California, Irvine.

We would like to thank Dennis Volper for his helpful comments and suggestions, Doug Schmidt for programming support, and Caroline Ehrlich for reviewing a previous copy of this paper.

Average Case Analysis of Empirical and Explanation-based Learning Algorithms

Wendy E. Sarrett and Michael J. Pazzani
Department of Information and Computer Science
University of California, Irvine
Irvine, Ca. 92717

October 6, 1989

Abstract

We present an approach to modeling the average case behavior of learning algorithms. Our motivation is to mathematically model the performance of learning algorithms in order to better understand the nature of their empirical behavior. We are interested in how differences in learning algorithms influence the expected accuracy of the concepts learned.

We present the *Average Case Learning Model* and apply the model to three learning algorithms: a purely empirical algorithm (Bruner's Wholist), an algorithm which prefers analytical (explanation-based) learning over empirical learning (EBL-FIRST-TM) and an algorithm integrating both analytical and empirical learning (IOSC-TM). The Average Case Learning Model is unique in that it is able to accurately predict the expected behavior of learning algorithms. We compare average case analysis to Valiant's *Probably Approximately Correct* (PAC) learning model.

KEYWORDS: Machine Learning, Analysis of Algorithms, Empirical Learning, Explanation-based Learning, Average Case Analysis

1 Introduction

In this paper, we introduce the *Average Case Learning Model* and apply the model to several learning algorithms. Average case analysis measures an algorithm's expected performance. The model predicts the expected accuracy of a learning algorithm as a function of the number of training examples. Accuracy is defined as the percentage of the population that is classified correctly. This is measured experimentally by taking a sample of the population and calculating the percentage of the sample that is correctly classified. We will compare theoretical and experimental results to demonstrate that applying the model does indeed lead to a correct model of the algorithms' average case behavior.

We have three goals in performing this work:

1. Model the behavior of various machine learning algorithms. This consists of:
   - Estimating empirically how accurate the hypothesis created by a algorithm is after a number of training examples.
   - Deriving a model to predict the expected accuracy of the hypothesis created by an algorithm as a function of the number of training examples.
   - Comparing the estimates with the value predicted by the model to verify that the formula is correct.
2. Compare the accuracy of various machine learning algorithms and find the conditions under which one learning algorithm is expected to be more accurate than another.
3. Gain insight into how the algorithms' differences influence their performance.

Combining average case analysis and experimentation is an important aspect of our work. Experimentation serves two purposes. First, it allows the accuracy of different algorithms to be compared and yields further insight on how the algorithms' differences influence their performance. Second, it provides evidence further that our theories about the algorithms' performance are correct. Both average case analysis and experimentation are beneficial because the results from the two techniques support each other. The experimental results provide evidence that the theoretical results are correct, while the theoretical results provide an explanation of the experimental results.

We concern ourselves with average case analysis rather than worst case analysis for two reasons. First, we want to gain an understanding of the algorithms' behavior in practice. Second, we prefer average case analysis because the results obtained can be tested experimentally. Currently, in machine learning, some researchers validate learning algorithms by running experiments on sets of standardized problems. Others perform complexity analyses of algorithms to gain an understanding of the class of concepts that are learnable (or approximately learning). We have unified these approaches by performing an average case analysis that permits a direct comparison of experimental and theoretical approaches.

In the remainder of this paper. we describe the learning framework and the Average Case Learning Model. Next, we apply the Average Case Learning Model to several learning algorithms.

2 An Overview of Our Approach

In presenting the framework for concept learning we define three components. First, we define the concept description language, and thereby define the class of learnable concepts. Second, we define the class of learning algorithms we analyze. Finally, we define the performance task that describes how the concept definitions are used.

2.1 Framework for Learning

The concept representation language defines the class of concepts that the algorithms can learn. Currently, we restrict our attention to conjunction of Boolean features. Learned concepts are represented as conjunctions of surface features. For example, a piece of crystal might be represented as $(glass \wedge expensive)$, where "glass" and "expensive" are surface features. [1]

[1] This is a fairly restrictive concept definition language. However, our intention is to begin with the simplest algorithms and representations and extend our results to more powerful algorithms and more expressive representation languages.

A training example consists of a set of surface features. These surface features are a subset of the surface features in the concept description language. Positive examples must include those surface features required by correct definition. However, they also may include irrelevant features. For example, if the concept being learned is $A \wedge B$, any positive example must include at least A and B. Negative training examples must not include the conjunction of the surface required by the concept definition.

We consider both empirical and analytical learning techniques. Empirical learning algorithms create hypotheses for concept definitions by finding correlations among several training examples. Analytical learning algorithms use existing knowledge to create new concepts that are implications of the existing knowledge [1]. We consider "one-sided" learning algorithms that maintain a single hypothesis about a concept definition and make the hypothesis more general in response to classification errors. A hypothesis is generalized by dropping features from the conjunctive expression describing the hypothesis. As long as the concept being learned is representable as a pure conjunctive expression, the hypothesis will never be more general than the correct definition.

The hypothesis produced by a learning algorithm is used to classify test examples. Since the algorithms are one-sided, negative examples will always be classified correctly. A positive example will be incorrectly classified if the hypothesis contains an irrelevant feature that is not present in the example. For example, suppose the current hypothesis is $A \wedge B \wedge C$ and the true concept definition is $A \wedge B$. If a positive example A, B, E is drawn it will be classified, incorrectly, as negative since the current hypothesis requires C to be present in any positive example. The accuracy of a hypothesis is determined by computing how many correct classifications it makes on a set of test examples.

2.2 The Average Case Learning Model

Our model for analyzing the behavior of learning algorithms consists of determining 1) under what conditions a feature is dropped from a hypothesis, 2) how often do these conditions occur, and 3) how does dropping a feature improve accuracy.

In the algorithms we study, there are two conditions that could result in dropping a feature from a hypothesis. Empirical algorithms drop an irrelevant feature that does not appear in a positive example. Suppose a hypothesis requires that a particular feature be present in positive examples. If a positive example does not contain the feature, then clearly the feature's presence is not required in positive examples of the concept. Thus, it should be dropped from the hypothesis. Analytical algorithms drop a feature that is not supported by the domain theory. A concept being learned by an explanation-based algorithm is defined as the conjunction of preconditions of rules in the domain theory used to explain why the concept belongs to a particular class. Therefore, if the domain theory does not require the presence of a given feature, then the feature is not included in the hypothesis.

Once we know the conditions under which a feature will be dropped, the next question to answer is how often do these conditions occur. This is a function of the distribution of training examples and the probability that an irrelevant feature appears in positive training examples. An irrelevant feature that has a high probability of appearing in a positive example will require more training examples in order to be dropped.

The extent to which dropping a feature improves accuracy is a function of the probability of the feature appearing in a positive test example. If a hypothesis contains an irrelevant feature that rarely appears in a test example, then the hypothesis will be fairly accurate.

2.3 Assumptions

In performing our analysis we make three assumptions:

- Training examples are drawn from a known uniform distribution.
- The probabilities of features appearing in positive examples are independent.
- A true hypothesis can be represented as a conjunction of the given features.

3 The Algorithms

The purpose of this section is three-fold:

- Present the empirical, analytical and integrated algorithms.

- Demonstrate how the model predicts the accuracy of an empirical learning algorithm.
- Discuss how the model extends to learning the domain theory for analytical learning.

3.1 Wholist

Wholist [2] is the simplest algorithm to which we apply the Average Case Learning Model. It is a strictly empirical algorithm. A more recent variation of this algorithm is known as the one-sided algorithm for pure conjunctive concepts [3].

Bruner's Wholist:

```
Input: h -- Hypothesized definition of the concept being learned.
       t-- A training example consisting of a set of features.
       Member?-Boolean, indicates if t is a training example
Output: h' -- Updated hypothesized definition of the concept
        being learned.

Wholist(h, t, Member?)
  begin
   C = Classify(h,t);
   If (C =0 and Member? = True)
      Then  Let h' = Drop_Features(h,t)
   endif;
  end.
```

The hypothesis, h, is a conjunction of base-level features. h is initialized to the conjunction of all surface features. Each training example, t, is a set of surface features. h' is the new hypothesis created by removing features that are not present in the training example from the hypothesis. Both h and t may be represented as Boolean vectors of length $|F|$ where F is the set of features in the concept definition language. A value of 1 means the feature is present and 0 means the feature is absent. *Drop_Features(h,t)* returns the

Boolean AND of h and t, resulting in the deletion of features from h that are not in t. Since this algorithm only drops features from h, h' will always be identical to or more general than h. h' can now be used to classify test examples as positive or negative. The procedure *Classify* returns 0 if t is a negative example of h and 1 if t is a positive example of h. t is classified as positive example by h if t has a value of 1 for every feature that h has a value of 1.

3.2 Wholist Model

The purpose of this average-case analysis is to estimate the accuracy of the hypothesis maintained after a given learning algorithm sees N training examples. Positive training examples are drawn with a fixed probability from the set of all training examples. Since the hypothesis never misclassifies negative examples, our analysis predicts the probability that a positive example will be correctly classified.

Before discussing the model we must define some notation:

- Let S = the set of positive examples.
- Let F = the set of features in the concept definition language.
- Let $|F|$ = the cardinality of F.
- Let H = the set of possible hypotheses.
- Let h = the current hypothesis in H. h will be represented as a Boolean vector of length $|F|$ where 0 means the feature is irrelevant, 1 means relevant. This Boolean vector represents a conjunction of the relevant features.
- Let X = the concept being learned.
- Let T^+ = a positive example, i.e., $T^+ \in S$. T^+ is represented as a Boolean vector of length $|F|$ where 0 means the feature is not present in a training example and, 1 means that the feature is present.

- Let $\Pr(S)$ = the probability that a positive training example is selected.[2]

- Let R = the set of relevant features in F (i.e., those features with value $= 1$ in all $T^+ \in S$.

- Let I = the set of irrelevant features in the definition of the concept before any training examples are seen. $I \subset F$. Note that $I \cup R = F$.

- Let f_j = an element of I. In a given T^+ or h, $f_j = 1$ and $f_j = 0$ denotes the presence or absence of a feature, respectively.

- Let $|I|$ = the cardinality of I.

- Let $\Pr(f_j)$ = the probability that irrelevant feature $f_j = 1$ in T^+. These probabilities are assumed to be independent. In addition, note that $\Pr(S)$ is independent of $\Pr(f_j)$.

- Let N = the total number of training examples seen.

- Let $b(i, N, \Pr(S))$ = the probability, as given by the binomial formula,[3] of drawing i positive training examples if there are a total of N training examples and the probability of a drawing a positive example is $\Pr(S)$.

Lemma 1 *After N training examples, the probability that an irrelevant feature $f_j = 1$ in h is* $\Pr(f_j)^i$ *where i is the number of positive examples in that set of N.*

Proof:

$f_j = 1$ in h only if $f_j = 1$ for all i positive examples. Since $\Pr(f_j)$ is independent of whether $f_j = 1$ in previous training examples, the probability that f_j has appeared in all i positive examples is $\Pr(f_j)^i$. □

Lemma 2 $\forall h \in H, T^+ \in S$ *and* $f_j \in I$, *(f_j causes T^+ to be misclassified by h) $\Leftrightarrow$ ($f_j = 1$ in h and $f_j \neq 1$ in T^+).*

[2] $\Pr(S)$ does not equal 1 for two reasons: 1) negative training examples are allowed, and 2) the algorithm may incrementally learn more than one concept at a time. $\Pr(S)$ represents the probability that a given example is a positive example of a given concept.

[3] $\binom{N}{i} * p^i(1-p)^{(N-i)}$ where p is the probability of "success"; in this case $\Pr(S)$.

Proof:

$\Leftarrow$

This follows directly from the Wholist algorithm. We will prove (f_j causes T^+ to be misclassified) $\rightarrow$ $((f_j = 1 \text{ in } h) \wedge (f_j \neq 1 \text{ in } T^+))$ by proving the contrapositive: $\neg((f_j = 1 \text{ in } h) \wedge (f_j \neq 1 \text{ in } T^+)) \rightarrow \neg(f_j$ causes T^+ to be misclassified). There are two cases.

Case 1: $f_j = 0$ in h indicates that the hypothesis considers f_j to be irrelevant. Therefore, the value of f_j in T^+ cannot cause a positive example to be falsely classified as a negative example.

Case 2: $f_j = 1$ in h and $f_j = 1$ in T^+. The feature f_j in T^+ has the correct value. Therefore, this feature can not cause T^+ to be misclassified.

Since these are the only possible cases and h, f_j and T^+ were chosen without any special properties, we can conclude that $\forall h \in H, f_j \in I$ and $T^+ \in S$: $\neg((f_j = 1 \text{ in } h) \wedge (f_j \neq 1 \text{ in } T^+)) \rightarrow \neg(f_j$ causes T^+ to be misclassified). Thus by contrapositive we can conclude:

$\{\forall h, f_j$ and $T^+ | (f_j$ causes T^+ to be misclassified) $\rightarrow (f_j = 1 \text{ in } h \wedge f_j \neq 1 \text{ in } T^+)\}$.

$\Rightarrow$

Suppose $f_j = 1$ in h but $f_j = 0$ in T^+. h requires that $f_j = 1$ in T^+ in order for T^+ to be correctly classified as positive. Since $f_j = 0$ in T^+ it will be incorrectly misclassified as negative.

Since h, f_j and T^+ were chosen without any special properties, we can conclude that $\forall H, f_j$ and T^+: if $f_j = 1$ in h and $f_j = 0$ in T^+ then f_j will cause T^+ to be misclassified. □

The major result of this section is the following theorem:

Theorem 1 *After N training examples, the expected accuracy of the hypothesis maintained by Wholist is:*

$$\sum_{i=0}^{N} \left[b(i, N, \Pr(S)) * \prod_{j=1}^{|I|} (1 - \Pr(f_j)^i (1 - \Pr(f_j))) \right]$$

Proof: Suppose Wholist has seen N training examples including i elements of S where $0 \leq i \leq N$. By Lemma 2 the probability that a given f_j will cause a hypothesis to misclassify a given T^+ is the probability that $f_j = 1$ in h and does not appear in T^+. By lemma 1 we know that the probability that the feature $f_j = 1$ in h is $\mathrm{Pr}(f_j)^i$. Furthermore, it is assumed that $f_j = 1$ in T^+ with probability $\mathrm{Pr}(f_j)$ (and the probability that $f_j = 0$ is $(1 - \mathrm{Pr}(f_j)))$. Since the probability that $f_j = 1$ in h after i positive examples and the probability that $f_j = 0$ in T^+ are independent, the probability that they both occur is $\mathrm{Pr}(f_j)^i(1 - \mathrm{Pr}(f_j))$. Thus the probability that feature f_j does not cause the hypothesis to misclassify T^+ is $1 - \mathrm{Pr}(f_j)^i(1 - \mathrm{Pr}(f_j))$. For T^+ to be classified correctly, none of these features can cause the hypothesis to misclassify T^+. Since these probabilities are independent, the probability that no feature causes the hypothesis to misclassify T^+ is the product over all j of the probability that feature f_j does not cause the hypothesis to misclassify T^+: $\prod_{j=1}^{|I|}(1 - \mathrm{Pr}(f_j)^i(1 - \mathrm{Pr}(f_j)))$. Note that since the only constraint on i is that it must lie from 0 to N inclusive we can conclude that this result holds $\{\forall i | 0 \leq i \leq N\}$. The probability that a test example will be misclassified after i positive training examples is weighted from $i = 0$ to N according to the probability that i of the N training examples are positive examples. This probability is given by the binomial formula, $b(i, N, \mathrm{Pr}(S))$. Thus the expected accuracy of the hypothesis is $\sum_{i=0}^{N}\left[b(i, N, \mathrm{Pr}(S)) * \prod_{j=1}^{|I|}(1 - \mathrm{Pr}(f_j)^i(1 - \mathrm{Pr}(f_j)))\right]$. □

To further verify the Average Case Learning Model for Wholist, we ran a simulation with an artificially constructed training and test set. The concept to be learned was constructed from a set of 10 features, 5 of which were relevant, 5 of which were irrelevant. The probability of a given irrelevant feature appearing ranged from 5% to 35%. In this simulation, the probability of a positive example being selected is 40%.

[Figure 1 goes here]

All simulations presented in this paper were run in the following manner. In each simulation, 100 learning trials were run and report the average accuracy of these trials as a function of the number of training examples. The algorithm was presented with training examples until it has learned the concept completely, i.e., has eliminated all the irrelevant features from the hypothesis. After every two training examples, the current hypothesis was used to classify 100 positive test examples as either positive or negative. In the figures in this section the scatter points are the average accuracy obtained in the simulation, the Y-Bars are the 95% confidence interval, and the curve is the theoretical estimates obtained by the Average Case Learning Model.

3.3 EBL-FIRST-TM

3.3.1 Performance and Foundational Examples

The goal of presenting EBL-FIRST-TM, a simplified version of explanation-based learning [4],[5], is to understand the analytical learning component of the integrated algorithm (IOSC-TM) we will present in the next section. In order to understand EBL-FIRST-TM and IOSC-TM, two different kinds of training examples must be defined. *Performance examples* are training examples of the complete performance task. For example, imagine a small child learning when other people will become angry. Performance examples would be examples of people becoming angry when the child performs some action (e.g., breaking Daddy's watch). *Foundational examples* are training examples from which a domain theory can be learned. Learning a domain theory can be viewed as a subproblem of the performance task. One way of decomposing the performance task of predicting what will anger a person is to learn what actions will cause an object to break and to learn the class of objects which when broken will anger a class of people. More formally, assume the domain knowledge is of the form: X_1 and $X_{1,2} \rightarrow X_2$, X_2 and $X_{2,3} \rightarrow X_3$ and we wish to acquire a predictive relationship: X_1 and $X_{1,3} \rightarrow X_3$. If the goal is to learn the relationship "If you drop an expensive, glass object, the owner will become angry.", then X_1 represents dropping an object, X_2 represents an object breaking, and X_3 stands for a person getting angry; $X_{1,2}$ represents a number of conditions that restrict the class of objects that are broken when dropped (e.g., objects composed of glass); $X_{2,3}$ refers to additional conditions that are needed to determine what class of persons will become angry when what class of objects breaks (e.g., the owner becomes angry when an expensive object breaks). These conditions are not specified in the domain theory; they must be acquired empirically from foundational examples. The goal of learning is to acquire $X_{1,3}$. This can be learned empirically from performance examples, or analytically ($X_{1,3} = X_{1,2} \wedge X_{2,3}$) from a domain theory acquired from foundational examples.

3.3.2 EBL-FIRST-TM Algorithm

While Wholist learns only from performance examples, EBL-FIRST-TM learns only from foundational examples.

EBL-FIRST-TM:

```
Input: h -- Hypothesized definition of the concept being learned.
       B -- domain theory explaining h.
       t -- A training example consisting of a set of features.
       Member?-Boolean, indicates if E is a training example.
Output: h' -- Updated current hypothesis of concept definition

EBL-FIRST-TM(h, B, t, Member?)
  begin
    C = Classify(h,t);
    If (C = negative and Member? = True)
       Then  If Explained(t,B)
                 Then Let h' = EBL(t,B);
                 Else
                   begin
                      Let h' = Drop_Features(h,t);
                      For each h2 that depends on h
                          Let h2' = EBL(h2,B');
                   end
              endif;
    endif;
  end
```

EBL-FIRST-TM prefers EBL but uses an empirical method (Wholist) when the training example is not explained by the domain theory. Note that, by definition, the domain theory can explain performance examples but not foundational examples. The domain theory is a set of implications whose antecedents are pure conjunctions learned by Wholist. Note that TM stands for truth maintenance. An explanation-based learning algorithm with truth-maintenance will update a hypothesis for a performance concept whenever a performance example is misclassified or whenever the domain knowledge is updated. In contrast, an algorithm without truth maintenance will only update a hypothesis for a performance concept when an error is made in classifying a performance example. A simple truth maintenance algorithm is used to rederive the definition of any performance concept that depends on a foundational concept when the definition of a foundational concept is changed.

In EBL-FIRST-TM B' is the domain theory with the hypothesis h replaced by h'. When h is a hypothesis for a foundational concept, the explanation-based learning with the new domain theory will produce a new hypothesis for the performance concept. The function $EBL(t, B)$ creates a hypothesis by taking the conjunction of the hypothesized definitions of the concepts in B that explain t.

3.3.3 EBL-FIRST-TM Model

Extending the results of the previous sections to EBL-FIRST-TM requires the following changes: First, the binomial formula must be replaced by the multinomial formula, an extension of the binomial formula to multiple "success" events to account for the fact that, in EBL-FIRST-TM, training examples for several foundational concepts and for a performance concept are intermixed; and second, the formula for the probability that $f_j = 1$ after i positive examples must be replaced by a formula reflecting how EBL-FIRST-TM removes features from the hypothesized definition of $X_{1,L+1}$

Before presenting these results some notational extensions must be noted:

- $X_{m,n}$ = those conditions that allow X_n to be inferred if X_m is known. $X_{m,m+1}$ is abbreviated as X_m. m goes from 1 to L where L is the length of the inference chain used to explain a performance concept. $X_{1,L+1}$ denotes the performance concept.

- h_m = the current hypothesis for foundational concept X_m. $h_{1,L+1}$ denotes the current hypothesis for the performance concept $X_{1,L+1}$.

- i_m = the number of the N training examples that are examples of foundational concept X_m. $i_{1,L+1}$ denotes the number of the N training examples that are examples of the performance concept.

- S_m = the set of positive examples of foundational concept m. m goes from 1 to L where L is the number of foundational concepts. $S_{1,L+1}$ denotes the set of examples of the performance concept.

- T_m = an element of S_m. $T_{1,L+1}$ denotes an element of $S_{1,L+1}$

- $\Pr(S_m)$ = the probability that a training example is a positive example of S_m. $\Pr(S_{1,L+1})$ denotes the probability that a training example is a positive example of $S_{1,L+1}$. Note that $Pr(S_{1,L+1}) + \Pr(S_1) + \cdots + \Pr(S_L) \leq 1$. The probabilities need not sum to 1 because these are the only probabilities of drawing a positive training example.

- $\Pr_m(f_j)$ = the probability that f_j is 1 in a example of concept m ($T_m \in S_m$). $\Pr_{1,L+1}(f_j)$ denotes The probability that $f_j = 1$ in a example of the performance concept ($T_{1,L+1} \in S_{1,L+1}$).

- $m(i_{1,L+1}, i_1, \cdots, i_L, N, \Pr(S_{1,L+1}), \Pr(S_1), \cdots, \Pr(S_L))$ = the multinomial formula [4] for calculating the probability that after N training examples $i_{1,L+1}$ are selected from $S_{1,L+1}$ and i_1 are selected from S_1, etc.

Lemma 3 *Given N training examples where $i_{1,L+1}$ are elements of $S_{1,L+1}$, i_1 are elements of S_1, i_2 are elements of S_2, $\cdots$ i_L are elements of S_L the probability that $f_j = 1$ in $h_{1,L+1}$ is $(1 - \prod_{m=1}^{L}(1 - (\Pr_m(f_j))^{i_m}))$.*

Proof:

When EBL-FIRST-TM sees N training examples where $i_{1,L+1}$ are examples of $T_{1,L+1}$, i_1 are examples of T_1, etc. By definition of the algorithm, $f_j = 1$ in h unless it has been eliminated by all L foundational concepts. The probability that $f_j = 1$ in h_m after i_m examples drawn from S_m are seen is, by a symmetric argument to Lemma 1, $(\Pr_m(f_j))^{i_m}$. Thus the probability that $f_j = 0$ in h_m is $(1 - (\Pr_m(f_j))^{i_m})$. Since these probabilities are independent, the probability that $f_j = 0$ for all $h_m \in H$ is $\prod_{m=1}^{L}(1 - (\Pr_m(f_j))^{i_m})$. Therefore, the probability that $f_j = 1$ in at least one h_m and thereby equals 1 in $h_{1,L+1}$ is $(1 - \prod_{m=1}^{L}(1 - (\Pr_m(f_j))^{i_m}))$. □

For EBL-FIRST-TM, we are primarily concerned with accuracy of the hypothesis for the performance concept. (Of course, the accuracy of the performance concept is related to the accuracy of the foundational concepts.) Therefore, accuracy is defined here as the probability that a randomly generated $T_{1,L+1} \in S_{1,L+1}$ is classified correctly. The major result of this section is the following extension of Theorem 1 :

[4] An extension of the binomial formula:
$\begin{pmatrix} N \\ i_{1,L+1} i_1 \cdots i_L \end{pmatrix} * \Pr(S_{1,L+1})^{i_{1,L+1}} \cdots \Pr(S_L)^{i}_{L}$ where $\Pr(S_m)$ and i_m are, as with the binomial formula, respectively the probability of event m occurring and the number of such events in N trials.

Theorem 2 *After N training examples, the expected accuracy of the hypothesis maintained by EBL-FIRST-TM is:*

$$\sum_{i_{1,L+1}=0}^{N} \sum_{i_1=0}^{N-i_{1,L+1}} \sum_{i_2=0}^{N-(i_{1,L+1}+i_1)} \cdots \sum_{i_L=0}^{N-(i_{1,L+1}+\sum_{m=1}^{L-1} i_m)} [m(i_{1,L+1}, i_1, \cdots, i_L, N, \Pr(S_{1,L+1}), \Pr(S_1), \cdots, \Pr(S_L)) * A]$$

Where

$$A = \prod_{j=1}^{|I|}(1-(1-\prod_{m=1}^{L}(1-(\Pr_m(f_j))^{i_m}))*(1-\Pr_{1,L+1}(f_j)))$$

Proof: Suppose EBL-FIRST-TM sees N training examples where $i_{1,L+1}$ were drawn from $S_{1,L+1}$, i_1 are drawn from S_1, etc. By Lemma 3 we know the probability that a given $f_j = 1$ in $h_{1,L+1}$is $(1-\prod_{m=1}^{L}(1-(\Pr_m(f_j))^{i_m}))$. Furthermore by a symmetric argument to Theorem 1 we then know the probability of classifying a $T_{1,L+1}$ correctly is A: $(\prod_{j=1}^{|I|}(1-(1-\prod_{m=1}^{L}(1-(\Pr_m(f_j))^{i_m}))*(1-\Pr_{1,L+1}(f_j)))$ (i.e., the probability no irrelevant feature, f_j, is 1 in $h_{1,L+1}$ and 0 in $T_{1,L+1}$). By a symmetric argument to Theorem 1, we weight each combination of $i_{1,L+1}, i_1, \cdots, i_L$ by the probability of that combination occurring when N examples are seen. This probability comes from the multinomial formula $(m(i_{1,L+1}, i_1, \cdots, i_L, N, \Pr(S_{1,L+1}), \Pr(S_1), \cdots, \Pr(S_L)))$. □

Figure 2 shows a simulation verifying this result. In this experiment, the performance concept is a conjunction of five features. In addition, there are five irrelevant features. The probability of an irrelevant feature appearing ranges from 5% to 35%. The probability of drawing a positive training example of the performance concept is 40%. There are two foundational concepts; the probability of drawing a positive example of a foundational concept is 30%.

[Figure 2 goes here]

3.4 IOSC-TM

3.4.1 IOSC-TM Algorithm

The IOSC-TM algorithm combines both the EBL-FIRST-TM and Wholist algorithms.[6] The EBL-FIRST-TM component eliminates features not supported by the domain theory, while the Wholist component eliminates features absent from a training example. IOSC-TM can eliminate a feature for either of these two reasons. In the case of a performance example, IOSC-TM operates by forming an empirical hypothesis using the Wholist algorithm and an analytical hypothesis using *EBL*. A composite hypothesis is formed by dropping any feature not appearing in both hypotheses. In the case of a foundational example, when the domain theory is updated the truth maintenance component of IOSC-TM will update the hypothesis for the performance concept by eliminating any irrelevant feature no longer appearing in the hypothesis created by applying the *EBL* algorithm to the updated domain theory.

IOSC-TM:

```
Input: h -- Hypothesized definition of the concept being learned.
       B -- Domain theory explaining h.
       t -- A training example consisting of a set of features.
       Member?-Boolean, indicates if E is a training example.
Output: h' -- Updated current hypothesis of concept definition

IOSC-TM(h, B, t, Member?)
  begin
    C = Classify(h,t);
    If (C = negative and Member? = True)
       Then  If Explained(t,B);
                 Then Let h' = Drop_features(Drop_Features(h,t),EBL(t,B))
                 Else
                   begin
                      Let h' = Drop_Features(h,t);
                      For each h2 that depends on h
                         h2' = Drop_Features(H,EBL(h2,B'));
                   end
             endif;
    endif;
  end
```

IOSC-TM relies on the fact that the hypothesis formed by Wholist is never more general than the true concept definition. Similarly, the hypothesis formed by EBL with a domain theory learned by Wholist is also never more general than the true hypothesis. Note that the result of EBL is a conjunction of the preconditions of the rules used to explain a performance training example. Since the antecedents of these rules are conjunctions (because the antecedents are learned by Wholist), the result of EBL is also a conjunction. Since the analytical and empirical hypotheses are never more general than the true concept definition and they are both represented as conjunctions of surface features, they may be merged into a composite hypothesis that is also never more general than the true definition.

3.4.2 IOSC-TM Results

Extending our previous results to IOSC-TM is straightforward. The only change required is changing the formula for the probability of $f_j = 1$ in $h_{1,L+1}$ to reflect the fact that for f_j to equal 1 in $h_{1,L+1}$ it must equal 1 in the hypothesis as created by $EBL(t, B)$ and the hypothesis created empirically.

Lemma 4 *Given N training examples: $i_{1,L+1}$ from $S_{1,L+1}$, i_1 from S_1, i_2 from S_2, $\cdots$ i_L from S_L the probability that $f_j = 1$ in $h_{1,L+1}$ is* $\mathrm{Pr}_{1,L+1}(f_j)^{i_{1,L+1}}(1-\prod_{m=1}^{L}(1-(\mathrm{Pr}_m(f_j))^{i_m}))$

Proof: Suppose IOSC-TM has seen N training examples. From Lemma 3 we know the probability of $f_j = 1$ in the hypothesis created analytically (by EBL(t,B)) is $(1-\prod_{m=1}^{L}(1-(\mathrm{Pr}_m(f_j))^{i_m}))$. Furthermore from Lemma 1 we know the probability of $f_j = 1$ in the hypothesis created empirically is $\mathrm{Pr}_{1,L+1}(f_j)^{i_{1,L+1}}$. Since these are independent (as $\mathrm{Pr}_{1,L+1}(f_j)$ and all L $\mathrm{Pr}_m(f_j)$ are independent) we can take the product giving us $\mathrm{Pr}_{1,L+1}(f_j)^{i_{1,L+1}}(1-\prod_{m=1}^{L}(1-(\mathrm{Pr}_m(f_j))^{i_m}))$ □

For IOSC-TM, the Average Case Learning Model makes the following prediction about the expected accuracy of a hypothesis learned by IOSC-TM:

Theorem 3 *After N training examples, the expected accuracy of the hypothesis maintained by IOSC-TM is:*

$$\sum_{i_{1,L+1}=0}^{N}\sum_{i_1=0}^{N-i_{1,L+1}}\sum_{i_2=0}^{N-(i_{1,L+1}+i_1)}\cdots\sum_{i_L=0}^{N-(i_{1,L+1}+\sum_{m=1}^{L-1}i_m)}[m(i_{1,L+1},i_1,\cdots,i_L,N,\Pr(S_{1,L+1}),\Pr(S_1),\cdots,\Pr(S_L))*A]$$

Where

$$A = \prod_{j=1}^{|I|}(1 - (\mathrm{Pr}_{1,L+1}(f_j)^{i_{1,L+1}}(1 - \prod_{m=1}^{L}(1 - (\mathrm{Pr}_m(f_j))^{i_m}))) * (1 - \mathrm{Pr}_{1,L+1}(f_j)))$$

Proof:

From Lemma 4 and a symmetric argument to Theorem 3. Note that the extra $\mathrm{Pr}_{1,L+1}(f_j)^{i_{1,L+1}}$ in A comes from substituting Lemma 4 for Lemma 3 in defining A. □

Figure 3 shows a simulation verifying this result. The test was run in the same manner, using the same concepts, as the simulation of EBL-TM.

[Figure 3 goes here]

4 Using the Model to Compare Algorithms

In this section, we describe how the Average Case Learning Model can be used to compare different algorithms' behavior under a number of different conditions.

To gain an understanding of the class of situations under which EBL-FIRST-TM is more accurate than Wholist, we used the model to compare the algorithms under a variety of circumstances. In addition, we ran simulations of these circumstances to verify that our analysis is accurate. Clearly, if the domain theory is accurate, it is best to use explanation-based learning techniques to acquire a new concept since these techniques can create an accurate hypothesis from a single example. However, if the domain theory is inaccurate (e.g., if the domain theory is being learned by Wholist from foundational examples at the same time that the domain theory is being used to explain performance examples), then it may be best to simply ignore the inaccurate domain theory and use an empirical technique such as Wholist.

We ran our first group of simulations on a concept with 40 features. The performance concept to be learned consisted of a conjunction of 10 of these features. There were foundational concepts constructed from the same 40 features. The ten features deemed relevant in the performance concept were those features deemed relevant in at least one foundational concept. The probability of a given irrelevant feature appearing ranged from 1% to 80%. For each algorithm, we ran two simulations. For the first simulation, we set the probability of a performance example at 50%, two of the foundational examples at 15% and the third at 20%. For the second simulation, we set the probability of a performance example at 25% and all three foundational examples at 25%. Figures 4, 5 and 6 display the results of these simulations.

[Figures 4-6 go here]

Wholist converges to 100% accuracy more quickly when there is a larger proportion of performance examples, while EBL-FIRST-TM converges more quickly when there is a larger proportion of foundational examples. IOSC-TM only slightly outperforms Wholist in the 50% case but does significantly better than both EBL-FIRST-TM and Wholist in the 25% case.

Our second group of simulations was run on a concept with 75 features. The performance concept consisted of a conjunction of 12 of these features. Two foundational concepts were constructed from the same 75 features. As before, the set of relevant features in the performance concept were those features deemed relevant in at least one foundational concept. The probability of a given irrelevant feature appearing ranged from 1% to 30%. For each algorithm we ran two simulations. For the first simulation, we set the probability of a performance example at 80%, and both of the foundational examples at 10%. For the second simulation, we set the probability of a performance example at 20% and both foundational examples at 40%. Figures 7, 8 and 9 display the results of the analysis and simulation. Although there are more irrelevant features in this set of simulations than there are in the previous set of simulations, faster learning rates were achieved in this set because the probability of an irrelevant feature appearing is smaller. The analysis and simulation also replicates the finding of the previous situation in which Wholist learns faster as the proportion of performance examples is increased and EBL-FIRST-TM learns more quickly as the proportion of foundational examples is increased.

[Figures 7-9 go here]

The Average Case Learning Model can make general predictions that would require running a large number of simulations to produce. For example, it can be shown by simple algebra that for a given probability distributions, IOSC-TM will always achieve an accuracy greater than or equal to the accuracy of EBL-FIRST-TM or Wholist. This is an intuitive result in that IOSC-TM learns from both performance and foundational examples, while

EBL-FIRST-TM only learns from foundational examples and Wholist only learns from performance examples.

For any given distribution of foundational and performance examples, we can solve for the expected accuracy of each algorithm and predict which learning algorithm will be more accurate. Furthermore, given the length of the inference chain, we can find the proportion of foundational to performance examples at which both algorithms will have an equal expected accuracy. When there are more foundational examples, then EBL-FIRST-TM will be more accurate; when there are fewer, then Wholist will be more accurate.

5 Previous Work

Comparisons can be made between the Average Case Learning Model and Valiant's *Probably Approximately Correct* (PAC) model[7],[8],[3]. There are two major differences:

- The goals of the Valiant model are different from the goals of our model.
- The Valiant model is distribution free. We make specific assumptions about the distribution from which the training and test examples are selected, enabling us to avoid some of the limitations of the Valiant model.

The goals of our analysis differ from those of the Valiant model. Haussler [3], building upon the Valiant model, developed formulae for calculating the upper bound on the sample size necessary to attain a given learning accuracy with high probability. Since these results are upper bounds, the sample size predicted for a given accuracy is actually much larger than what would be needed on average. This is very different from our model. We model the expected behavior of a learning algorithm. We wish to estimate the classification accuracy expected after giving a learning algorithm N training examples. Under adverse circumstances accuracy could be significantly lower. Under fortuitous circumstances, accuracy may be much higher. Clearly, these two types of analysis answer very different questions.

Our analysis assumes a known sampling distribution and determines the expected accuracy of the concept definition after seeing a sample of N training examples. To demonstrate how our results differ from those produced by the Valiant model, consider the convergence rate of the one-sided conjunctive algorithm [3]. Suppose we have a conjunctive concept with 15 relevant features and 85 irrelevant features. If we want to be extremely confident ($\delta = 0.01$) that the error rate will be at most 5%, the Valiant model would require a sample size of some constant times ~ 35200. Our analysis, in contrast, predicts that on average it takes less than 200 examples in order to learn the concept with an accuracy of greater than 95%.

[Figure 10 goes here]

Figure 10 compares the mean accuracy, the accuracy predicted by the Average Case Learning Model, the minimum accuracy achieved from 100 learning trials and the bounds predicted by the Valiant model. Note the large difference between the bounds derived from the Valiant model and the expected accuracy predicted by our model. Furthermore, note the difference between the Valiant bounds and the minimum accuracy achieved on any of the 100 learning trials.

Valiant's model, and indeed most of the recent work in computational learning theory, is distribution free. This results in certain limitations. Valiant's definition of *learnable* requires that the bounds hold for all distributions D of positive examples of concept F [7],[8],[3]. Thus, the Valiant model does not model an algorithms' expected behavior. Rather, it provides a lower bound on sample size. In addition, the model does not allow for inductive bias. As pointed out by Buntine [9], these limitations result in the PAC model producing overly-conservative estimates of error (as we have found via simulation). The Probably Approximately Correct model has resulted in many valuable findings on the relative difficulty of various learning tasks. However, this model does not match the induction process as it is commonly implemented in machine learning systems.

By taking advantage of the information available to machine learning programs, we are able to accurately predict their average case performance. We utilize knowledge about the distribution of training examples and assume training examples are drawn from a known uniform distribution and each feature appears in positive examples with an independent uniform distribution. This allows us to more accurately model an algorithm's behavior. Finally, we incorporate inductive bias. For example, we assume that the concepts are representable as pure conjunctions and thus limit the hypothesis space to pure conjunctive hypothesis.

Although our model addresses some of the limitations of the Valiant model, it is very restrictive. Our analysis requires both prior knowledge of the distribution and an independence assumption. Furthermore, our analysis requires knowledge about the number of relevant and irrelevant features. In spite of these restrictions, the Average Case Learning Model has proven useful. We have demonstrated that it allows us to compare the performance of different learning algorithms on the same problem. In addition, it allows us to better understand how the distribution of examples affects a learning algorithm's performance. Our future plans are to apply the model to more complex learning algorithms and problems:

- Apply the Average Case Learning Model to more powerful representation languages such as $k-CNF$. This is a relatively straightforward extension of the model since there are one-sided algorithms analogous to Wholist for this representation.
- Extend our results to incorporate noise. The work of Angluin and Laird [10] may provide a good starting point.

We are particularly interested in forging a compromise between Valiant's distribution free model and our distribution specific model. By estimating the true distribution during the training process we can remove one assumption and increase the class of situations in which the model may be applied.

6 Conclusions

In this paper, we have presented the Average Case Learning Model that is capable of predicting the performance of machine learning algorithms. The key aspect of applying the model is to understand the probability that a training example will be encountered that causes an inaccurate hypothesis to be revised and to understand the effect of revising a hypothesis on the accuracy of the hypothesis. We have applied the model to Wholist, a simple empirical learning algorithm; EBL-FIRST-TM, a simple analytical learning algorithm; and IOSC-TM, an algorithm that combines empirical and analytical learning.

The Average Case Learning Model is unique in that it is capable of predicting the performance of a class of learning algorithms. We have verified both formally and through simulation that the model accurately predicts the expected behavior of learning algorithms. We have used the model to compare learning algorithms and gain insight into the conditions under which each algorithm is most accurate.

References

[1] M. J. Pazzani, *Learning Causal Relationships by Integrating Analytical and Empirical Methods.* Doctoral dissertation, University of California, Los Angeles, 1988.

[2] J. S. Bruner, J. J. Goodnow, and G. A. Austin, *A Study of Thinking.* New York: Wiley, 1956.

[3] D. Haussler, *Applying Valiant's Learning Framework To AI Concept Learning Problems*, (Technical Report No. UCSC-CRL-87- 11). Santa Cruz: University of California, Department of Computer Science, 1987.

[4] G. DeJong and R. Mooney, " Explanation-Based Learning: An Alternative View," *Machine Learning*, 1(2), pp. 145-176, 1986.

[5] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli, "Explanation-Based Generalization: A Unifying View," *Machine Learning*, 1(1), pp. 47-80, 1986.

[6] W. E. Sarrett and M. J. Pazzani, "One-Sided Algorithms for Integrating Empirical and Explanation-Based Learning," *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, NY: Morgan Kaufmann, pp. 26–28, June 1989.

[7] L. G. Valiant, "A Theory of the Learnable," *Communications of the ACM*, 27(11), pp. 1134–1142, Nov. 1984.

[8] L. G. Valiant, "Learning Disjunctions of Conjunctions," In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA: Morgan Kaufmann, pp. 560–566, Aug. 1985.

[9] W. Buntine, "A Critique of the Valiant Model," *Proceeding of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, Michigan: Morgan Kaufmann, pp. 837–842, Aug. 1989.

[10] D. Angluin and P. Laird, " Learning From Noisy Examples," *Machine Learning*, 2(4), pp. 343-370, April 1988.

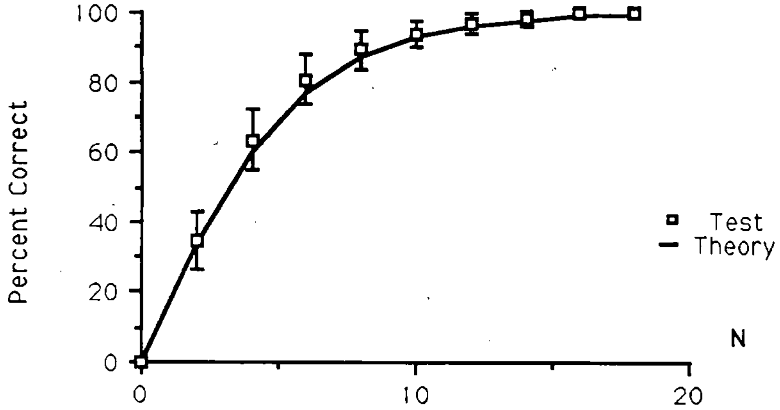

Figure 1: A comparison of the expected and actual accuracy of the Wholist algorithm. The x-axis is the number of training instances and the y-axis is the percent of test instances correctly classified. The boxes represent the empirical means, the y-bars the 95% confidence interval around those means and the curve is the value predicted by the Average Case Learning Model.

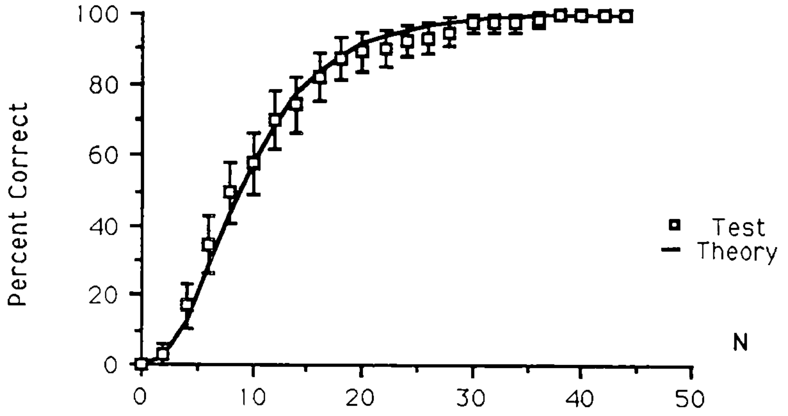

Figure 2: A comparison of the expected and actual accuracy of the EBL-FIRST-TM algorithm.

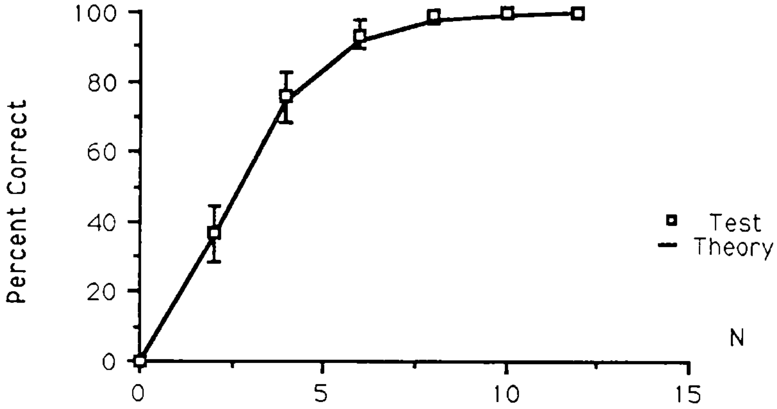

Figure 3: A comparison of the expected and actual accuracy of the IOSC-TM algorithm.

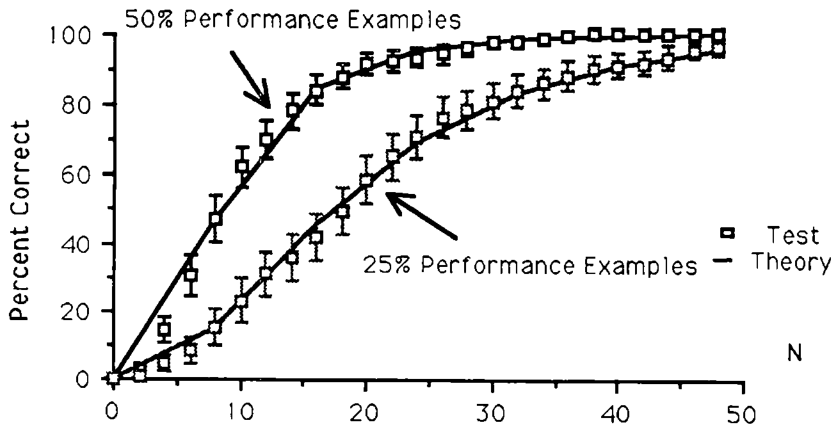

Figure 4: Test showing the behavior of Wholist with 25% and 50% performance examples.

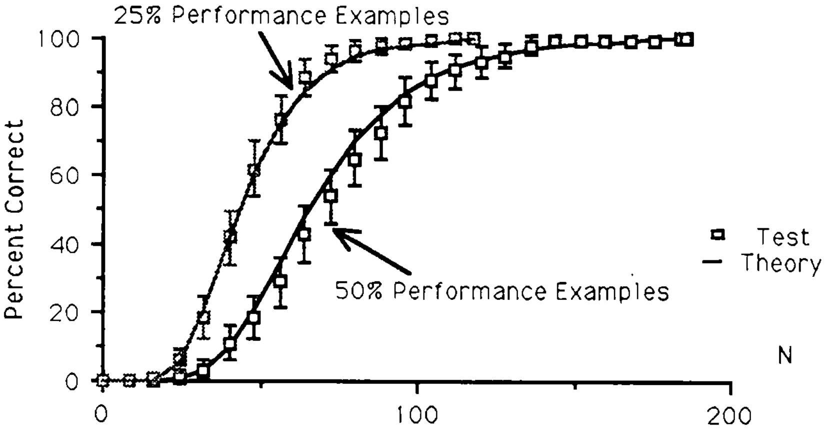

Figure 5: Test showing the behavior of EBL-FIRST-TM with 25% and 50% performance examples.

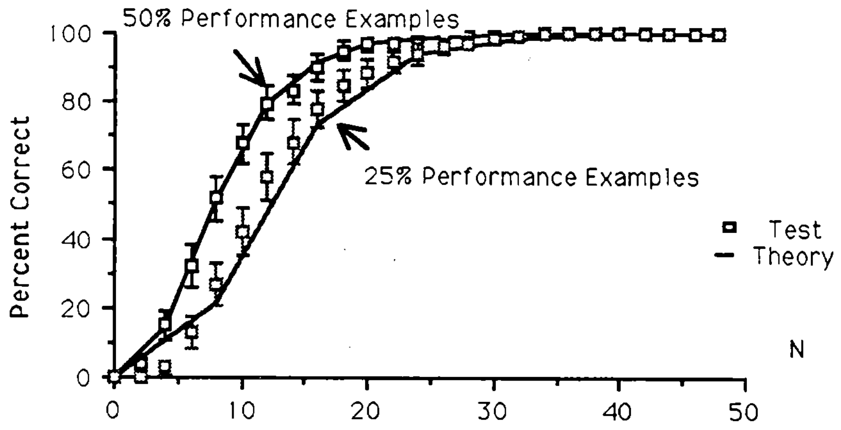

Figure 6: Test showing the behavior of IOSC-TM with 25% and 50% performance examples.

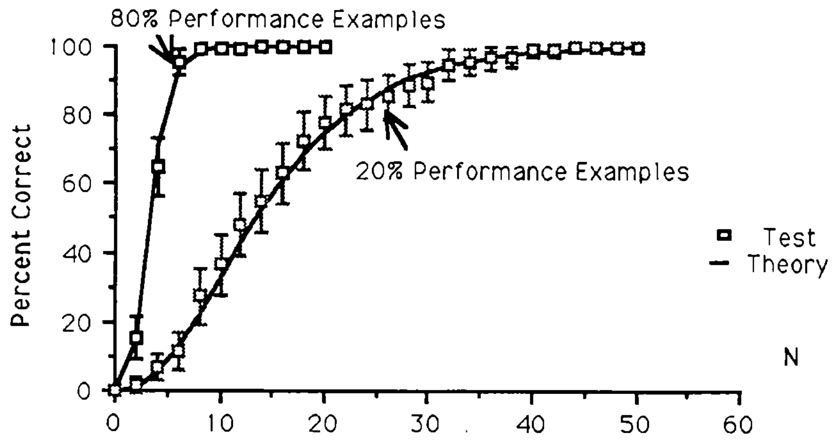

Figure 7: Test showing the behavior of Wholist with 20% and 80% performance examples.

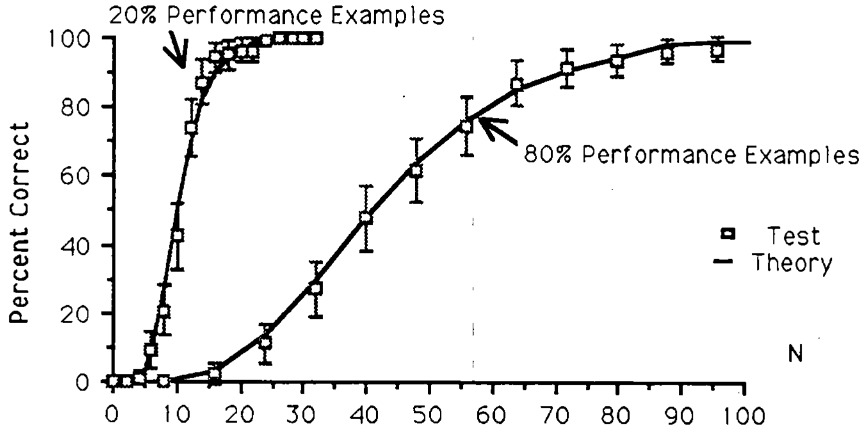

Figure 8: Test showing the behavior of EBL-FIRST-TM with 20% and 80% performance examples.

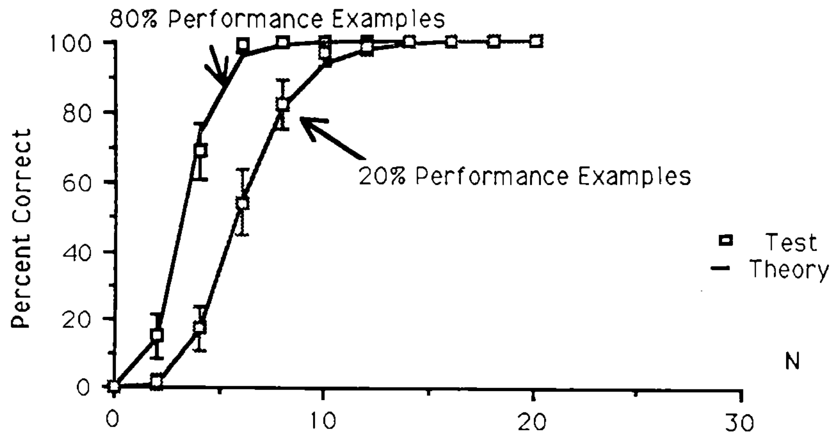

Figure 9: Test showing the behavior of IOSC-TM with 20% and 80% performance examples.

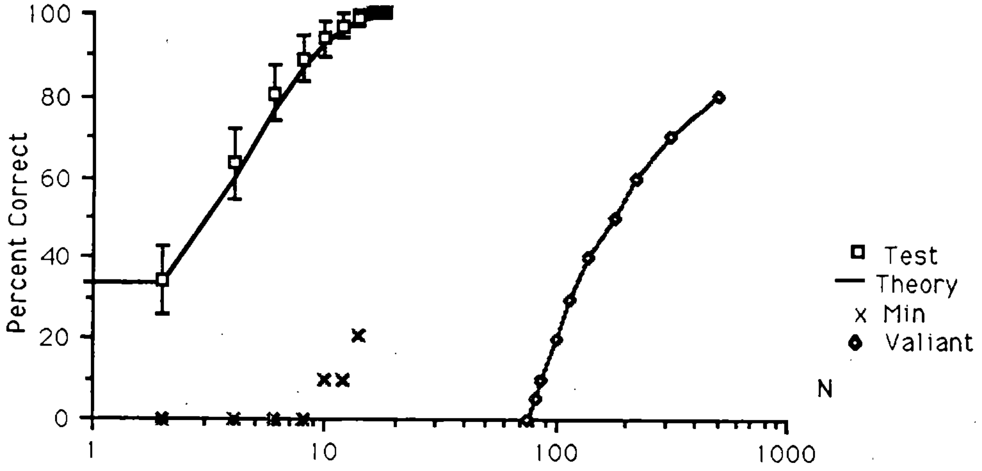

Figure 10: This figure compares the average case performance with the minimum accuracy in the empirical test and the Valiant model. For the Valiant calculations we assumed a feature space of ten (the total number of features in our concept) and δ = 0.05. The X-axis is the required N predicted by the Valiant model and the Y-axis is 1-ε. Note that the scale of the X axis is logorithmic.