Machine Learning, 9, 349–372 (1992)
© 1992 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

A Framework for Average Case Analysis of Conjunctive Learning Algorithms

MICHAEL J. PAZZANI (PAZZANI@ICS.UCI.EDU)
WENDY SARRETT (SARRETT@ICS.UCI.EDU)
Department of Information and Computer Science, University of California, Irvine, Irvine, CA 92717 USA

Editor: David Haussler

Abstract. We present an approach to modeling the average case behavior of learning algorithms. Our motivation is to predict the expected accuracy of learning algorithms as a function of the number of training examples. We apply this framework to a purely empirical learning algorithm, (the one-sided algorithm for pure conjunctive concepts), and to an algorithm that combines empirical and explanation-based learning. The model is used to gain insight into the behavior of these algorithms on a series of problems. Finally, we evaluate how well the average case model performs when the training examples violate the assumptions of the model.

Keywords. average-case analysis, combining empirical and analytical learning

1. Introduction

Most research in machine learning adheres to either a theoretical or an experimental methodology (Langley, 1989). Some attempt to understand learning algorithms by testing the algorithms on a variety of problems (e.g., Fisher, 1987; Minton, 1988). Others perform formal mathematical analysis of algorithms to prove that a given class of concepts is learnable from a given number of training examples (e.g., Valiant, 1984; Haussler, 1987). The common goal of this research is to gain an understanding of the capabilities of learning algorithms. However, in practice, the conclusions of these two approaches are quite different. Experiments lead to findings on the average case accuracy of an algorithm. Formal analyses typically deal with distribution-free, worst-case analyses. The number of examples required to guarantee learning a concept in the worst-case do not accurately reflect the number of examples required to approximate a concept to a given accuracy in practice.

We have begun construction of an average case learning model to unify the formal mathematical and the empirical approaches to understanding the behavior of machine learning algorithms. Current experience with machine learning algorithms has led to a number of empirical observations about the behavior of various algorithms. An average case model can explain these observations, make predictions, and guide the development of new learning algorithms.

1.1. *Mathematical models of human learning*

Several psychologists have created mathematical models of strategies proposed as models of human or animal learning (e.g., Atkinson, Bower, & Crothers, 1965; Restle, 1959). These models have focused on average case behavior of learning algorithms. There are several reasons that these models cannot be directly applied to machine learning algorithms. First, these models typically study learning algorithms restricted to less complex concepts (e.g., single attribute discriminations) than those typically used in machine learning. Second, these models also address complications not present in machine learning programs since they must account for individual differences in human learners caused by such factors as attention, motivation, and memory limitations. Finally, the performance metric predicted by these models is generally the expected number of trials required to learn a concept with perfect accuracy. Typically, experimental studies of machine algorithms report on the observed accuracy of learning algorithms after a given number of examples.

1.2. PAC learning

Valiant (1984) has proposed a model to probabilistically justify the inductive leaps made by an empirical learning program. The probably approximately correct (PAC) model indicates that a system has learned a concept if the system can guarantee with high probability that its hypothesis is approximately correct. Approximately correct means that the concept will have an error no greater then ϵ (i.e., the ratio of misclassified examples to total examples is less than ϵ). The learning system is required to produce an approximately correct concept with probability $1 - \delta$. For a given class of concepts, the PAC model can be used to determine bounds on the number of training examples required to achieve an accuracy of $1 - \epsilon$ with probability $1 - \delta$. The PAC model has led to many important insights about the capabilities of machine learning algorithms. One such result from Blumer, Ehrenfeucht, Haussler, and Warmuth (1989) is that for an algorithm that explores a hypothesis space of cardinality $|h|$, at most

$$\frac{1}{\epsilon}\left(\ln|h| + \ln\frac{1}{\delta}\right)$$

examples are required to guarantee with probability at least $1 - \delta$ that the error is no greater than ϵ. If this equation is solved for ϵ, it yields the upper envelope on the error for all target concepts in the hypothesis space and all distributions of examples as a function of the number of training examples and δ and $|h|$ (Haussler, 1990). For specific values of δ and $|h|$, one can obtain a curve relating ϵ to the number of training examples. There is typically a wide gap between such a curve and the mean observed error reported in experimental studies of machine learning algorithms (Haussler, 1990).

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. For example, theorem 4 of Hembold et al. (1990) shows that the mean error is at most

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

for a particular algorithm (the closure algorithm, a generalization of the one-sided algorithm) that explores an intersection closed hypothesis space h (such as conjunctive concepts), where VCdim(h) is the Vapnik-Chervonenkis dimension of the hypothesis space (Vapnik & Chervonenkis, 1971). This result also appears as theorem 2.1 of Haussler et al. (1990). Theorem 3.1 of Haussler et al. (1990) gives a lower bound on the mean error that can be achieved for any algorithm on the worst case distribution of training examples (for $N >$ VCdim(h), provided VCdim(h) is finite):

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

where e is the base of the natural logarithm.

In figure 1, the points plotted are the mean accuracy computed from one hundred runs of the wholist algorithm (Bruner, Goodnow, & Austin, 1956) as a function of the number of training examples. The wholist algorithm is shown in table 1. After every training example, the accuracy of the current hypothesis was measured by classifying 250 test examples. The errors bars around the points are the 95% confidence interval around the mean (using the t-distribution, Winer, 1971). The concept to be learned was a conjunction of five features from a total of ten Boolean features. Five of these features are irrelevant. The probability that a given irrelevant feature was present in a positive training example was 0.5 and all training (and test) examples were positive. For this problem and this algorithm, the value of $|h|$ is 2^{10} and the VCdim(h) is 10. The curves plotted in this figure are obtained by substituting these values into the equations given above. The curve from Blumer et al. (1989) is for $\delta = 0.05$.

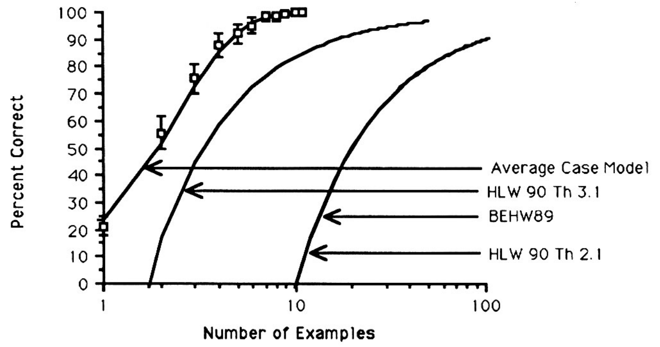


Figure 1. A comparison of the observed accuracy of the wholist algorithm to the worst case accuracy predicted by the average case model presented here, the PAC model (Blumer, Ehrenfeucht, Haussler, & Warmuth, 1989), and theorems 2.1 and 3.1 from Haussler, Littlestone and Warmuth (1991). The boxes represent the empirical means, the y-bars are the 95% confidence interval around those means and the curves are the value predicted by the various models.

Table 1. The wholist algorithm.

```
1. Initialize the hypothesis to the conjunction of all features
   that describe training examples.
2. If the new example is a positive example, and the hypothesis
   misclassifies the new example, then remove all features from the
   hypothesis that are not present in the example.
3. Otherwise, do nothing.
```

There are a variety of reasons for the gap between the observed accuracy and the equation (Blumer et al., 1989) on the PAC model. The first, and most important reason is that the PAC model is not intended to predict the mean, observed accuracy of specific concepts from specific distributions. Rather, the goal of PAC learning has been to provide bounds on the number of examples to learn with high probability any concept from a given class to a given accuracy from any distribution of examples.

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, and perhaps most importantly, both the PAC model and the prediction models presented are distribution-free models. In our simulation there was a single, known distribution of training examples. In this distribution, the values of the irrelevant features are independent. The observed learning accuracy on this distribution is much higher than that which would be observed from a more difficult distribution, such as the worst case distribution used in the proof of theorem 3.1 in Haussler et al. (1990).

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.[1]

In figure 1, we also plot a curve corresponding to the expected accuracy of the average case model described in the next section. This model takes advantage of knowledge of the distribution to produce a predicted accuracy that falls within the 95% confidence intervals around the mean observed on this problem.

2. The average case learning model

We have been developing a framework for average case analysis of machine learning algorithms. The framework for analyzing the expected accuracy of the hypothesis produced by a learning algorithm consists of determining:

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

Clearly, the second requirement presupposes information about the distribution of the training examples. Therefore, unlike the PAC model, the framework we have developed is not distribution-free. Furthermore, to simplify computations (or reduce the amount of information required by the model) we will make certain independence assumptions (e.g., the probabilities of all irrelevant features occurring in training example are independent). Similarly, the third requirement presupposes some information about the test examples and information about the current hypothesis.

Determining conditions under which a learning algorithm changes a hypothesis requires an analysis of the operators used for creating and changing hypotheses. In this respect, the framework is more similar to the mathematical models of human and animal learning strategies than the theoretical results on machine learning algorithms which typically are concerned with the relationship between the size of the hypothesis space and the number of training examples.

The goal of the analysis here differs from that of the PAC model. The average case analysis is intended to predict the accuracy of a given algorithm on a given class of learning problems. The average case analysis can be used to compare the expected accuracy 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 on 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.

The average case analysis uses information that is not available to a learner. For example, the analysis requires specifying the distribution from which the training examples are drawn. In addition, the analysis requires specifying a correct definition of the concept to be learned. Note that the knowledge of the distribution of examples and correct answer are not used by the learning algorithm. Rather, this knowledge is used only in the analysis. To emphasize this difference, we first present below a model of the "learning algorithm" that is occasionally used as a strawman in experimental machine learning research. This algorithm simply guesses the most frequent class. We will restrict our attention to the case in which there are two classes (positive and negative) and there is a fixed probability p that a randomly drawn example is from the most frequent class (i.e., $p \geq 0.5$). If the learner knew which class was more frequent, then the expected accuracy of this approach would be p. However, it is not reasonable to assume that the learner has this information. Rather, the algorithm we analyze will count the number of positive and negative examples encountered so far, and predict that the next example belongs to the most frequent class.[2] In this case, there is a certain probability (p_{freq}) that after i examples the algorithm will correctly identify the most frequent class. This occurs when a greater number of the training examples belong to the most frequent class. This probability is:

$$p_{freq}(i) = \sum_{j=\frac{i+1}{2}}^{i} \binom{i}{j} p^j (1 - p)^{i-j}$$

Therefore, the overall expected accuracy of an algorithm that guesses the most frequent observed class is:

$$p_{freq}(i)\,p + (1 - p_{freq}(i))(1 - p).$$

2.1. *An average case model of wholist*

We will first show how the framework can be applied to the wholist algorithm, a predecessor of the one-sided algorithm for pure conjunctive concepts (Haussler, 1987). This algorithm incrementally processes training examples. The hypothesis maintained is simply the conjunction of all features that have appeared in all positive training examples encountered.

For example, consider the simple case of learning the concept **dime**. We will assume that the training examples are represented by the following binary features: **round**, **silver**, **shiny**, **small**, **edible**, and **warm**. The initial hypothesis is the conjunction of these six features. The first example is a positive example of a dime. It is described by the features **round** $\wedge$ **silver** $\wedge$ **shiny** $\wedge$ **small**. Since the features **edible** and **warm** are not in this positive example they are dropped from the hypothesis. The next example is a quarter, a negative example of the concept **dime**: **round** $\wedge$ **silver** $\wedge$ **shiny** $\wedge$ **warm**. The hypothesis is not revised for negative examples. The third example is another positive example: **round** $\wedge$ **silver** $\wedge$ **warm** $\wedge$ **small**. Since the feature **shiny** is not true of this example, it is dropped from the hypothesis, and the current hypothesis is now the correct conjunctive definition of the concept: **round** $\wedge$ **silver** $\wedge$ **small**.

The following notation will be used in describing the algorithm:

- f_j is the jth irrelevant feature of a training example. A feature is irrelevant if the true conjunctive definition of the concept does not include the feature. For example, the feature **shiny** is irrelevant in the above example.
- N is the number of examples (both positive and negative) seen so far.

The following information is required to predict the expected accuracy of wholist. Note that while this is much more information than required by the PAC model, this is exactly the information required to generate training examples from a product distribution to test the algorithm:

- P is the probability of drawing a positive training example. This value is 1.0 in the data used to generate the graph in figure 1.
- I is the number of irrelevant features. The value of I is 5 in the data displayed in figure 1.
- $P(f_j)$ is the probability that irrelevant feature j is present in a positive training example. For example, P(**shiny**) represents the probability that a dime is shiny. In figure 1, the value of $P(f_j)$ is 0.5 for j from 1 to 5.

Note that the accuracy of this algorithm does not depend upon the number of relevant features that are conjoined to form the true concept definition. Therefore, the analysis does not make use of the total number of features or the number of relevant features.

The wholist algorithm has only one operator to revise a hypothesis. An irrelevant feature is dropped from the hypothesis when a positive training example does not include the irrelevant feature. Therefore, if i positive examples have been seen out of N total training examples, the probability that irrelevant feature j remains in the hypothesis (i.e., feature j has appeared in all i positive training examples) is:

$$REMAIN_{wholist}(f_j,\ i) = P(f_j)^i. \tag{1}$$

The hypothesis created by the wholist algorithm misclassifies a positive test example if the hypothesis contains any irrelevant features that are not included in the test example. The wholist algorithm does not misclassify negative test examples (provided that the true concept definition can be represented as a conjunction of the given features). Therefore, the probability that feature j does not cause a positive test example to be misclassified after i positive training examples is given by:[3]

$$NM_{wholist}(f_j,\ i) = 1 - (REMAIN_{wholist}(f_j,\ \mathrm{i}) * (1 - P(f_j)). \tag{2}$$

If all irrelevant features are independent,[4] then after i positive training examples, the probability that no irrelevant feature will cause a positive test example to be misclassified is:

$$\prod_{j=1}^{I} NM_{wholist}(f_j,\ i) \tag{3}$$

Finally, in order to predict the accuracy of the hypothesis produced by the wholist algorithm after N training examples, it is necessary to take into consideration the probability that exactly i of the N training examples are positive examples for each value of i from 0 to N. Therefore, the accuracy of the wholist algorithm (where accuracy is defined as the probability that a randomly drawn positive example will be classified correctly by an algorithm) can be given by:

$$\sum_{i=0}^{N} \left[\mathrm{b}(i,\ N,\ \mathrm{P}) * \prod_{j=1}^{I} NM_{wholist}(f_j,\ i) \right] \tag{4}$$

where $b(i,\ N,\ P)$ is the binomial formula:

$$b(i,\ N,\ P) = \binom{N}{i} * P^i(1 - P)^{(N-i)}.$$

In figure 1, the predicted and the actual accuracies of the hypothesis produced by the wholist algorithm are plotted. The expected accuracy of wholist algorithm, as predicted by equation (4), is plotted along with the results obtained by running the program on the training data. Since the data is generated from a known distribution that meets the assumptions, the predicted accuracy corresponds very closely with the observed accuracy. A common measure used to describe how well data fit a model is the coefficient of determination[5] (r^2). For this data, r^2 is greater than 0.99.

In the following sections, we will illustrate how the framework we have developed for average case analysis of knowledge-free conjunctive learning algorithms can be used to gain insight on concept formation in the presence of background knowledge. First, we describe a performance task and three learning algorithms that can be used to acquire the knowledge necessary to accomplish that task. Next, we present an average case analysis of each algorithm and compare the predicted and observed accuracy of the algorithms on this task.

2.2. Learning a domain theory

The learning and performance tasks to be analyzed differ from those commonly studied in machine learning. The difference is necessitated by the fact that we are interested in analyzing the use and learning of a domain theory as opposed to the learning of a single concept in isolation. A domain theory consists of a set of inference rules that can be chained together to draw a conclusion. For example, one rule might state that "If a fragile object is struck, the object will break." A second rule might indicate, "If an expensive object breaks, the owner will be angry." If presented with an example of a cat knocking over a glass vase, it would be possible to use both of these rules to conclude that the owner of the vase will be angry. The average case model from the previous section may be used to determine the expected accuracy of a single inference rule learned by the wholist algorithm. In section 2.3, we will use the average case model to determine the expected accuracy of a conclusion inferred with a domain theory if each rule of the domain theory is learned by the wholist algorithm.

A domain theory can be used by explanation-based learning (EBL) (DeJong & Mooney, 1986; Mitchell, Keller, & Kedar-Cabelli, 1986) to create new "macro" rules by combining preconditions of existing rules. For example, the rules in the previous paragraph can be used to form a new rule, "If a person strikes an expensive fragile object, the owner of the object will get angry." If we assume that the rules learned by explanation-based learning are revised whenever the domain theory changes, the model used to predict the expected accuracy of conclusions drawn from a domain theory can also be used to predict the accuracy of rules learned by explanation-based learning. In section 3, we show how the average case model can be used to predict whether a rule learned by explanation-based learning would, on average, be more accurate than a rule learned by the wholist algorithm.

Learning a domain theory requires learning multiple concepts (one for each rule in the domain theory). We will use the notation $G \rightarrow (p_1 \rightarrow p_2)$ to represent the knowledge acquired by the learning system and used by the performance system. This can be read, "If G is true, then p_1 implies p_2." The performance task is to infer whether a predicate, p_2, is true in a situation S given that a predicate p_1 is true. We will assume that a situation is represented as a set of binary features $x_1, x_2 \ldots x_n$. In this paper, variables starting with S will be used to refer to specific situations; variables starting with G will be used to refer to general descriptions of a class of situations. We will assume that each G may be represented as a conjunction of a subset of the features used to describe specific situations.

Training examples are represented as specific instances of inference rules of the form[6] $S \rightarrow (p_1 \rightarrow p_2)$. More general inference rules may be learned by generalizing all of the individual situations in which p_1 implies p_2. A collection of such inference rules, learned from a variety of examples, may serve as the domain theory for explanation-based learning.

In this paper, we consider the case in which there are two possible means to determine if a predicate c is true in situation S when predicate a is true. The first is to acquire a rule ($G_{AC} \rightarrow (a \rightarrow c)$) that allows c to be inferred directly. The second is to acquire two rules ($G_{AB} \rightarrow (a \rightarrow b)$) and ($G_{BC} \rightarrow (b \rightarrow c)$) and allow the performance system to chain rules to infer b from a, and then infer c from b. Note that G_{AC} can also be represented as $G_{AB} \wedge G_{BC}$. For example, a might represent striking an object, b might represent the object breaking and c might represent the owner of the object getting angry. The performance task is to predict when $G_{AC} \rightarrow (a \rightarrow c)$: "If a person strikes an expensive fragile object, then the owner of the object will get angry." Two other rules may help in the prediction: "If a fragile object is struck, the object will break" and "If an expensive object breaks, the owner will be angry." Here, G_{AC} refers to the conditions "expensive and fragile," G_{AB} is the condition "fragile" and G_{BC} is the condition "expensive."

Three distinct groups of training examples are intermixed and presented incrementally to the learning system:

$$S_{AC} \rightarrow (a \rightarrow c)$$
$$S_{AB} \rightarrow (a \rightarrow b)$$
$$S_{BC} \rightarrow (b \rightarrow c)$$

where S_{AC}, S_{AB} and S_{BC} are the sets of features of training examples. We will refer to the first type of training examples as *performance* examples since these examples will permit the learning system to learn a rule that directly solves the performance task. We will refer to the latter two types of examples as *foundational* examples because these examples do not allow the problem to be solved directly but provide a foundation for inferring when c is true.[7]

We will consider three related learning methods that can be used to acquire the knowledge for this performance task:

- wholist: The wholist method can be used to learn G_{AC}, from performance examples. In this case, G_{AC} is the maximally specific conjunction of all examples of S_{AC}. This method ignores foundational examples.
- chaining: The wholist method can be used to learn G_{AB}, and G_{BC} from foundational examples and c can be inferred from $G_{AB} \rightarrow (a \rightarrow b)$ and $G_{BC} \rightarrow (b \rightarrow c)$. In this case, G_{AB} and G_{BC} are the maximally specific conjunction of all examples of S_{AB} and S_{BC}, respectively. Note that this method ignores the performance examples. This method can be viewed as learning the domain theory for explanation-based learning. However, it is irrelevant to the accuracy of the results of the learning whether the rule $G_{AC} \rightarrow (a \rightarrow c)$ is cached by EBL (and updated whenever the domain theory is changed) or the performance task is solved by chaining. If G_{AC} were cached by EBL, it would be the conjunction of all features that are in either G_{AB} or G_{BC}.
- IOSC-TM (Sarrett & Pazzani, 1989a): The result of learning G_{AC} with the wholist algorithm from performance examples, and the result obtained by learning G_{AB} and G_{BC} from foundational examples can be combined to form a composite hypothesis. This is

the technique used by the integrated one-sided conjunctive learning algorithm with truth maintenance[8] (IOSC-TM). Two approximations to G_{AC} are formed. One is the result of wholist on the performance examples. The other is the result of "chaining" on the foundational examples. A composite hypotheses is created by deleting features not present in both. It is possible to combine the result of chaining and the result of learning G_{AC} with wholist because each hypothesis only makes one-sided errors. Unlike the previous two algorithms, IOSC-TM updates its hypothesis when presented with performance or foundational examples. The composite hypothesis formed by IOSC-TM is similar to the S set of the version space merging algorithm (Hirsh, 1989). However, unlike version-space merging, IOSC-TM learns the domain theory from foundational examples.

Table 2 shows an example of the hypothesis produced by each of these three algorithms when run on the same training examples. All three algorithms produce the same hypothesis for G_{AB} and G_{BC}. Note that the hypothesis formed for G_{AC} by IOSC-TM contains only those features that are both in the hypothesis formed by wholist on performance examples and in the hypothesis formed by chaining for this concept.

Table 2. Hypotheses produced by the three algorithms.

Training Examples		
	[x_1= 1, x_2= 1, x_3= 1, x_4= 1, x_5= 1]	→ (a → c)
	[x_1= 1, x_2= 0, x_3= 1, x_4= 1, x_5= 1]	→ (a → b)
	[x_1= 1, x_2= 1, x_3= 0, x_4= 1, x_5= 1]	→ (b → c)
	[x_1= 1, x_2= 1, x_3= 0, x_4= 0, x_5= 1]	→ (a → b)
	[x_1= 0, x_2= 1, x_3= 1, x_4= 0, x_5= 1]	→ (b → c)
	[x_1= 1, x_2= 1, x_3= 1, x_4= 0, x_5= 0]	→ (a → c)
Hypotheses		
All three algorithms:	[x_1= 1, x_5= 1]	→ (a → b)
	[x_2= 1, x_5= 1]	→ (b → c)
wholist:	[x_1= 1, x_2= 1, x_3= 1]	→ (a → c)
chaining {implicit}	[x_1= 1, x_2= 1, x_5= 1]	→ (a → c)
IOSC-TM:	[x_1= 1, x_2= 1]	→ (a → c)

2.3. *An average case model of learning a new domain theory*

In this section, we present a general analysis of the problem of learning and using a domain theory. Some new notation is necessary to describe this problem:

- P_{AC}, P_{AB}, and P_{BC} are the probabilities of drawing a positive training example from S_{AC} → (a → c), and S_{AB} → (a → b), S_{BC} → (b → c), respectively. For brevity, here we only consider the case that $P_{AC} + P_{AB} + P_{BC} = 1$ (i.e., there are no examples that are not one of the three rules).
- $P_{AC}(f_j)$, $P_{AB}(f_j)$, and $P_{BC}(f_j)$ are the probabilities that irrelevant feature j is present in a positive training example from S_{AC} → (a → c), S_{AB} → (a → b), and S_{BC} → (b → c), respectively.

The three algorithms of the previous section differ only in how they update the hypothesis. Therefore, the only difference in the analysis is the probability that an irrelevant feature remains for each algorithm. We will use the notation $\text{REMAIN}_{\text{alg}}(f_j, i_0, i_1, i_2)$ to indicate that a particular algorithm leaves irrelevant feature f_j in the hypothesis for $\mathtt{G_{AC}}$ after exactly i_0 positive training examples of $\mathtt{S_{AC}} \rightarrow \mathtt{(a \rightarrow c)}$, i_1 positive training examples of $\mathtt{S_{AB}} \rightarrow \mathtt{(a \rightarrow b)}$, and i_2 positive training examples of $\mathtt{S_{BC}} \rightarrow \mathtt{(b \rightarrow c)}$. For each algorithm, the probability that feature j does not cause a positive test example of $\mathtt{S_{AC}} \rightarrow \mathtt{(a \rightarrow c)}$ to be misclassified is given by $NM_{\text{alg}}(f_j, i_0, i_1, i_2)$:

$$NM_{alg}(f_j, i_0, i_1, i_2) = 1 - (REMAIN_{alg}(f_j, i_0, i_1, i_2) * (1 - P_{AC}(f_j))) \tag{5}$$

Since there are a total of N training examples, it is necessary to take into consideration the probability that there are exactly i_0 positive training examples of $\mathtt{S_{AC}} \rightarrow \mathtt{(a \rightarrow c)}$, i_1 positive training examples of $\mathtt{S_{AB}} \rightarrow \mathtt{(a \rightarrow b)}$, and i_2 positive training examples of $\mathtt{S_{BC}} \rightarrow \mathtt{(b \rightarrow c)}$ for each value of i_0, i_1 and i_2 from 0 to N. Note that because all examples are of one of three rules, i_2 is equal to $N - (i_0 + i_1)$. The multinomial formula is used to weight the value of $NM_{\text{alg}}(f_j, i_0, i_1, i_2)$ by the probability that various values of i_0, i_1 and i_2 occur. Therefore, on the problem of learning $\mathtt{G_{AC}}$, the expected accuracy of each algorithm, as a function of the total number of training examples can be given by:

$$\sum_{i_o}^{N} \sum_{i_1}^{N-i_0} \binom{N}{i_0,\ i_1, N-(i_0+i_1)} P^{i_0}_{AC} P^{i_1}_{AB} P^{N-(i_0+i_1)}_{BC} * \prod_{j=1}^{I} NM_{alg}(f_j, i_0, i_1, N-(i_0+i_1)) \tag{6}$$

The analysis of the wholist algorithm presented in section 2.1 will apply directly to this problem. The probability that an irrelevant feature remains in the hypothesis for $\mathtt{G_{AC}}$ produced by wholist is given by:

$$REMAIN_{wholist}(f_j, i_0, i_1, i_2) = \text{P}_{\text{AC}}(f_j)^{i_0} \tag{7}$$

Note that since the hypothesis for $\mathtt{G_{AC}}$ produced by wholist is not updated when presented with foundational examples, $REMAIN_{wholist}(f_j, i_0, i_1, i_2)$ does not depend upon the value of i_1 or i_2.

2.3. Average case analysis of using chaining

Chaining $(\mathtt{G_{AB}} \rightarrow \mathtt{(a \rightarrow b)}$ and $(\mathtt{G_{BC}} \rightarrow \mathtt{(b \rightarrow c)}$ requires using the wholist algorithm to learn two conditions ($\mathtt{G_{AB}}$ and $\mathtt{G_{BC}}$). A positive test example will be correctly classified if both $\mathtt{G_{AB}}$ and $\mathtt{G_{BC}}$ do not contain any irrelevant feature that is not present in the test example. The probability that irrelevant feature j remains in the hypothesis for $\mathtt{G_{AC}}$ produced by chaining is:

$$REMAIN_{chaining}(f_j, i_0, i_1, i_2) = 1 - ((1 - \text{P}_{\text{AB}}(f_j)^{i_1}) * (1 - \text{P}_{\text{BC}}(f_j)^{i_2})) \tag{8}$$

Note that since the hypothesis for G_{AC} produced by chaining is not updated when presented with performance examples, $REMAIN_{chaining}(f_j, i_0, i_1, i_2)$ does not depend upon the value of i_0. In Sarrett and Pazzani (1989b), we consider the general case in which the inference chain is of any given length. In section 2.4, we illustrate how well equation (8) models the accuracy of the hypothesis produced by the chaining algorithm.

2.4. Average case analysis of IOSC-TM

The IOSC-TM algorithm combines the hypothesis formed by chaining $G_{AB} \rightarrow (a \rightarrow b)$ and $G_{BC} \rightarrow (b \rightarrow c)$ and the hypothesis produced by wholist learning $G_{AC} \rightarrow (a \rightarrow c)$. A positive test example will be incorrectly classified if it does not contain an irrelevant feature that meets both of the following conditions:

- Either G_{AB} or G_{BC} contains the irrelevant feature.
- G_{AC} contains the irrelevant feature.

If i_o positive training examples of $S_{AC} \rightarrow (a \rightarrow c)$, i_1 positive training examples of $S_{AB} \rightarrow (a \rightarrow b)$, and i_2 positive training examples of $S_{BC} \rightarrow (b \rightarrow c)$ have been seen, the probability that irrelevant feature j remains in the hypothesis for G_{AC} produced by IOSC-TM is:

$$REMAIN_{IOSC\text{-}TM}(f_j, i_0, i_1, i_2) = P_{AC}(f_j)^{i_0} * (1 - ((1 - P_{AB}(f_j)^{i_1}) * (1 - P_{BC}(f_j)^{i_2}))) \quad (9)$$

In order to compare the accuracy of the hypotheses produced by the three algorithms under a variety of conditions, we substituted various values for $P_{AC}(f_j)$, $P_{AB}(f_j)$, $P_{BC}(f_j)$, P_{AC}, P_{AB}, and P_{BC} into equations (7), (8), and (9). In addition, to obtain confirmation of the mathematical result and to visualize the equations, we ran each of the three algorithms on data generated according to the values of the parameters. Figure 2 shows the three algorithms when P_{AC} is 0.4, 0.2, and 0.1. In each case, P_{AB} and P_{BC} are $(1 - P_{AC})/2$. The values of $P_{AC}(f_j)$, $P_{AB}(f_j)$, $P_{BC}(f_j)$ were randomly assigned for each feature from the range (0.01 to 0.80). Since the data generated meets the assumptions, the predicted accuracy corresponds very closely with the observed accuracy. In all cases, r^2 is greater than 0.99. In section 4, we will evaluate how well the model performs when the assumptions are violated.

3. Use of average case models

The goal of our average case analysis is to gain an understanding of how domain knowledge, especially inaccurate domain knowledge, can facilitate learning. It was found experimentally that IOSC-TM consistently achieved an accuracy greater than or equal to the accuracy of chaining or wholist. The average case model can help to explain this empirical finding.

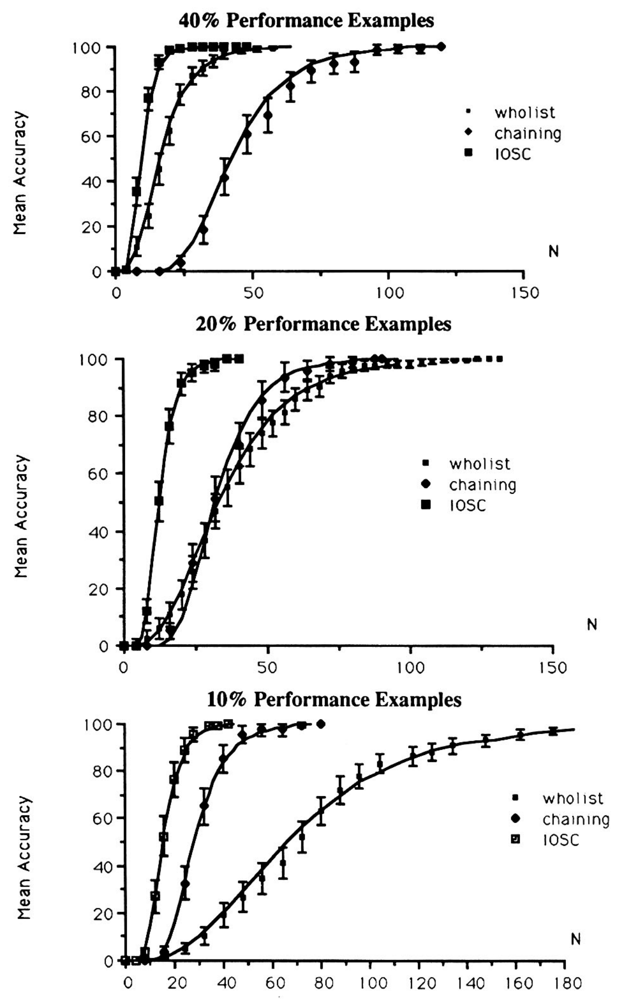


Figure 2. A comparison of the expected and observed accuracy of the three algorithms. The points are the empirical means with 95% confidence intervals and the curves are the value predicted by equations 7, 8, and 9.

The only difference in the model of the three algorithms is in the computation of the probability that a feature remains after a number of positive examples.

$$REMAIN_{wholist}(f_j, i_0, i_1, i_2) = \mathrm{P}_{\mathrm{AC}}(f_j)^{i_0}$$

$$REMAIN_{chaining}(f_j, i_0, i_1, i_2) = 1 - ((1 - P_{AB}(f_j)^{i_1}) * (1 - \mathrm{P}_{\mathrm{BC}}(f_j)^{i_2}))$$

$$REMAIN_{IOSC\text{-}TM}(f_j, i_0, i_1, i_2) = \mathrm{P}_{\mathrm{AC}}(f_j)^{i_0} * (1 - ((1 - \mathrm{P}_{\mathrm{AB}}(f_j)^{i_1}) * (1 - \mathrm{P}_{\mathrm{BC}}(f_j)^{i_2})))$$

From these equations, it is clear that the probability of a feature remaining in the hypothesis produced by IOSC is never greater than the probability of a feature remaining in wholist or the chaining algorithm. Note that $REMAIN_{IOSC\text{-}TM}(f_j, i_0, i_1, i_2)$ could also be expressed as

$$REMAIN_{chaining}(f_j, i_0, i_1, i_2)*REMAIN_{wholist}(f_j, i_0, i_1, i_2).$$

Since $REMAIN_{chaining}(f_j, i_0, i_1, i_2)$ and $REMAIN_{wholist}(f_j, i_0, i_1, i_2)$ never exceed 1, the product cannot be greater than either term. The implication of this result is that the hypothesis produced by this particular combination of empirical and explanation-based learning will be at least as accurate as its empirical and explanation-based components.

The average case model can also be used to gain insight into the purely empirical algorithm. For example, in the wholist algorithm, is it frequently occurring features, or rare features that are likely to lead to misclassifications? Rewriting equation (2) clearly shows that there is a tradeoff since commonly occurring features are more likely to remain in a hypothesis. However, if the hypothesis contains a rare feature, it is more likely to result in misclassification (since rare features are less likely to be in the test example).

$$NM_{wholist}(f_j, i) = 1 - (P(f_j)^i * (1 - P(f_j)).$$

Figure 3 graphs the probability of a feature not resulting in a misclassification as a function of the probability of a feature appearing in a training (and test) example for several values of i (the number of positive training examples). After one training example, the features most likely to result in misclassification are those that occur with probability 0.5. With additional training examples, this function has a minimum at larger values of $P(f_j)$. Therefore, one can conclude that the presence of commonly occurring irrelevant features will result in less accurate hypotheses than the presence of rare irrelevant features.

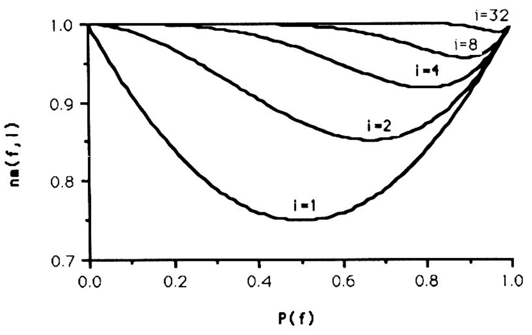


Figure 3. The probability of a feature not resulting in misclassification is plotted as a function of the probability of a feature appearing in a training example.

4. Violating assumptions of the framework

So far, we have presented an average case model of the wholist algorithm and used the average case model to explore the relationship between the expected accuracy of wholist, chaining and IOSC-TM. Here, we consider what happens when the assumptions of the model are violated. In particular, we compare the predicted and observed accuracies when the values of features are not independent. In addition, we run the model on the distribution used in the proof of theorem 3.1 of Haussler et al. (1990). Finally, we compare the observed and predicted accuracy of the wholist algorithm on a naturally occurring data set.

4.1. Violating independence assumptions

Equation (3) computes the probability that no irrelevant feature will cause a positive test example to be misclassified by taking the product of the probabilities that each feature does not cause a misclassification. This assumes that the examples are drawn from a product distribution (i.e., the values of irrelevant features are chosen independently). First, we consider using equation (3) on a simple problem in which there are two relevant features and three irrelevant features. One of the irrelevant features is dependent upon the value of the other two. The goal of these experiments is to evaluate the robustness of equation (3) when the independence assumptions are violated. In all cases, $P(f_1)$ and $P(f_2) = 0.5$ and the values of f_1 and f_2 are independent. In the following two simulations, the values of feature f_3 will depend on f_1 and f_2 as follows:

a. Feature f_3 is true whenever f_1 and f_2 are both true. Therefore the average case model uses the values $P(f_1)$, $P(f_2) = 0.5$ and $P(f_3) = 0.25$. However, the model assumes that $P(f_3 \mid f_1 \text{ and } f_2) = 0.25$ while, in fact, it is 1.
b. Feature f_3 is true whenever either f_1 or f_2 is true. Therefore, $P(f_1)$, $P(f_2) = 0.5$ and $P(f_3) = 0.75$. However, the model assumes that $P(f_3 \mid \overline{f_1} \,\&\, \overline{f_2}) = 0.25$ while, in fact, it is 0. Note that, in effect the k-CNF learning algorithm is identical to the wholist algorithm except that it constructs composite features in this manner. These composite features are disjunctions of k variables used to describe the training examples.[9]

Figure 4 shows the predicted and observed mean accuracy (and 95% confidence interval) of the wholist algorithm under these three conditions. In each case, the accuracy is computed by testing on 250 examples and the accuracy is averaged over 100 simulations. For the conjunctive case, the average case model accounts for 88.9% of the variation in the mean observed accuracy. In the disjunctive case, $r^2 = 0.906$. In the conjunctive case, the predicted accuracy underestimates the observed mean accuracy. This occurs because the presence of f_3 does not affect the accuracy of the algorithm. The average case model, when there are only two irrelevant features (f_1 and f_2), has a very good fit with the data. In the disjunctive case, the predicted accuracy differs from the observed accuracy because the features f_1 and f_2 cannot result in a misclassification unless f_3 also results in a misclassification. The average case model, when there is only one irrelevant feature (f_3), has a very good fit with the data.

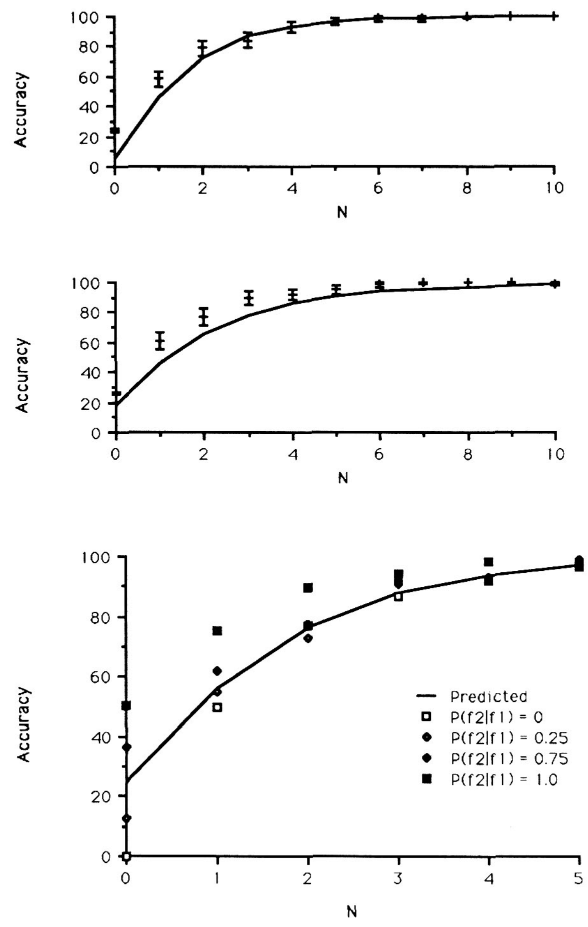


Figure 4. A comparison of the expected (lines) and observed (points) accuracy of the wholist algorithm when one irrelevant feature is unknowningly the conjunction (upper) or disjunction (middle) of two irrelevant features. The lower shows the expected and observed accuracy when $P(f_2|f_1) = 0, 0.25, 0.75$, and 1 but the model assumes $P(f_2|f_1) = P(f_2) = 0.5$.

Next, we consider a problem in which there are two irrelevant features, and one of the features is dependent upon the value of the other. In all cases, $P(f_1)$ and $P(f_2) = 0.5$. Therefore, the average case model uses 0.5 for $P(f_1)$ and $P(f_2)$. The independence assumption assumes $P(f_2 \mid f_1) = 0.5$. However, we will generate data such that $P(f_2 \mid f_1)$ takes on the values 1, 0.75, 0.25, and 0. When $P(f_2 \mid f_1) = 1$, then $f_2 = f_1$; when $P(f_2 \mid f_1) = 0$, then $f_2 = \text{not}(f_1)$. Figure 4 (lower) plots this data. The values for r^2 are 0.468, 0.954, 0.972, 0.975, and 0.930, respectively. Note that when $P(f_2 \mid f_1)$ is greater than 0.5, the prediction of the average case model is less than the observed accuracy, while when $P(f_2 \mid f_1)$ is less than 0.5, the predicted accuracy is greater than the observed accuracy.

4.2. A distribution that varies with time

The proof of theorem 3.1 of Haussler et al. (1990) uses a distribution that varies with time (i.e., with the number of training examples) to give a lower bound on the mean error that can be achieved for any algorithm on this distribution. Here we use a variant on that distribution, which only makes use of $k + 1$ distinct training examples where k is the VCdim(H) and H is a class of concepts. The set of training examples $\{y_1, \ldots, y_k\}$ are those that are shattered by the class of concepts and y_0 is any example that is not a member of this set. For example, for the concept class that is the conjunction of 10 features, the VC dimension is 10. The set $\{y_i : 1 \leq i \leq 10\}$ is shattered by this concept class where in example i, feature i is 0 and all other features are 1. In our simulations, we will use (1,1,1,1,1,1,1,1,1,1) for y_0. The following probability distribution is used:[10]

$$P(y_0) = \frac{1 - k}{N}$$

$$P(y_i) = \frac{1}{N}, \ 1 \leq i \leq k$$

Here, we consider the case when the wholist algorithm is used to learn a conjunction of 5 out of 10 of the features. Table 3 lists the training examples and their classifications for this concept. In this case, there are six positive examples and the probability that an irrelevant feature occurs in a positive example is 0.833. This value will be used for $P(f_1)$, $1 \leq i \leq 5$ by the average case model. Note, however, that since distribution is a function of the number of training examples (N), the probability that one of these irrelevant features is present in a randomly drawn positive example will differ from 0.833.

We ran a series of 11 simulations in which we interpolated between the distribution from Haussler et al. (1990) and the product distribution discussed in section 2. For the product

Table 3. Training data for the simulation in figure 5.

Concept to be learned: $f_6 \wedge f_7 \wedge f_8 \wedge f_9 \wedge f_{10}$

y_0	= (1,1,1,1,1,1,1,1,1,1)	+
y_1	= (0,1,1,1,1,1,1,1,1,1)	+
y_2	= (1,0,1,1,1,1,1,1,1,1)	+
y_3	= (1,1,0,1,1,1,1,1,1,1)	+
y_4	= (1,1,1,0,1,1,1,1,1,1)	+
y_5	= (1,1,1,1,0,1,1,1,1,1)	+
y_6	= (1,1,1,1,1,0,1,1,1,1)	-
y_7	= (1,1,1,1,1,1,0,1,1,1)	-
y_8	= (1,1,1,1,1,1,1,0,1,1)	-
y_9	= (1,1,1,1,1,1,1,1,0,1)	-
y_{10}	= (1,1,1,1,1,1,1,1,1,0)	-

distribution, we used a value of 0.833 for the probability that an irrelevant attribute appears in a positive training example. That was done to match the probability in the time varying distribution. All of the examples from the product distribution were positive examples. In the simulation there was a parameter, α, that determined the probability that an example came from the product distribution. When α was 0, all examples came from the product distribution and when α was 1, all of the examples came from the time varying distribution. The 11 simulations used values of α in increments of 0.1 from 0 to 1. Since the distribution may vary with time, it is not possible to test the accuracy of the algorithm after N examples by testing on a separate set of examples. Instead, the accuracy of the algorithm after N examples was computed by determining the probability that training example N+ is classified correctly. We ran 10,000 trials for each value of α. Figure 5 plots the observed error (i.e., 1-observed accuracy) when α is 0.0, 0.2, 0.5, 0.7, and 1.0. Since theorem 3.1 holds when $N >$ VCdim(H), the graph starts with the eleventh training example. For this graph, example y_1 is a positive example. The graph when y_1 is a negative example is similar. The bottom curve, for $\alpha = 0.0$ closely approximates the expected accuracy ($r^2 >$ 0.99). As conditions diverge from that assumed by average case model, it underestimates the observed error. Figure 6 shows the value of r^2 as a function of α. This graph indicates that predictions of the average case model degrade gracefully as the distribution diverges from the specific distribution we have analyzed.

4.3. Naturally occurring data sets

In this section, we compare the expected accuracy to the observed accuracy on an artificial concept using a naturally occurring data set. Unfortunately, conjunctive concepts are not expressive enough to obtain a completely accurate concept on any of the data sets in the archive of machine learning databases at the University of California, Irvine. Furthermore,

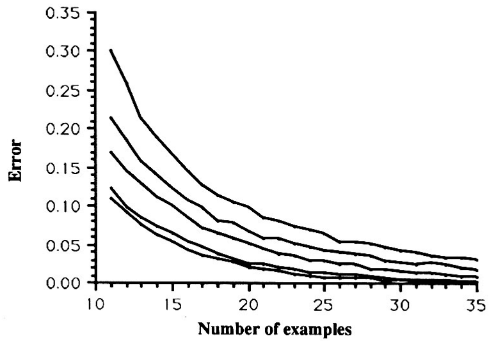


Figure 5. The error of the wholist algorithm for distributions that interpolate between the distribution of theorem 3.1 of Haussler et al. (1990) and the product distribution. From the top, the curves are for α = 1.0, 0.7, 0.5, 0.2, and 0.0.

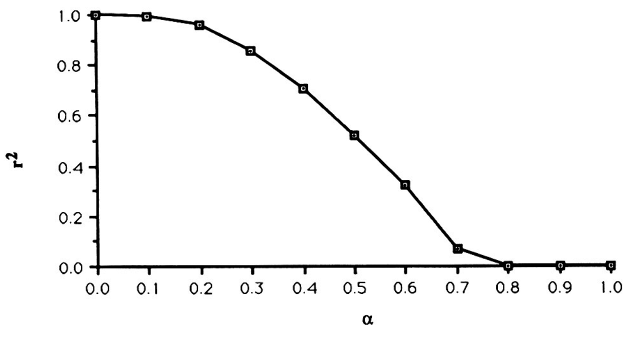
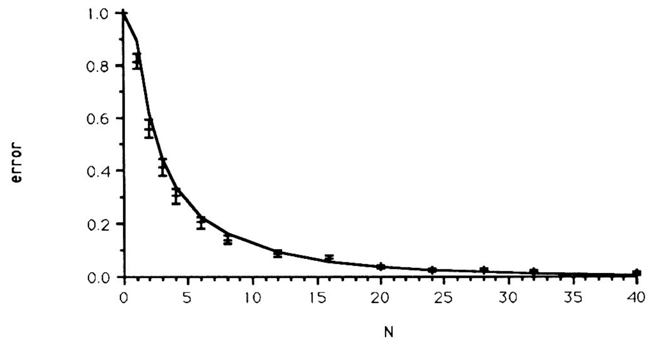


Figure 6. The goodness of fit of the average case model as a function of the probability that an example does not come from the product distribution.

most data sets have a variety of complications that we have not yet addressed. These complications include features that take on more than two values or numeric values. In addition, many naturally occurring data sets contain missing values for some attributes as well as attribute and classification noise.

One data set that nearly meets the specification of the wholist algorithm is the database of congressional voting records. This database has 435 examples. The examples belong to one of two classes (Republicans and Democrats), and contain 16 Boolean features indicating whether the member of Congress voted for or against particular issues, such as aid to El Salvador or research on synthetic fuels. In order to run the wholist algorithm on this data set, we eliminated all examples with missing features. This left a total of 124 Democrats and 108 Republicans. Next, we constructed an approximation to the concept of Democrats by finding the pair of issues that the most number of Democrats voted for and eliminated all examples that were not consistent with this conjunction. We performed the same process on the Republican examples. This left a total of 105 out of the original set of Democrats and also left 105 Republicans. Now, if the wholist algorithm is run on the reduced set of the Democrats, it will produce a conjunction of two binary features. Similarly, the wholist algorithm will return a conjunction of two features when run on the reduced set of Republicans.[11] Note that these manipulations were done to make it possible to run the wholist algorithm on the data and not to make the data conform to the assumptions of the average case model. For each feature of the data sets, we calculated the probability that each irrelevant feature was present in the positive examples. This data was used by the average case model to predict the expected accuracy.

We ran 100 trials of the wholist algorithm on this data. The training examples were selected randomly (with replacement) from the total set of examples. Figure 7 shows the error predicted by the average case model and observed mean error (with 95% confidence intervals around the mean). At each point, the error was calculated on 100 examples selected randomly (with replacement) from the total set of examples. Although there are a variety of dependencies among the irrelevant features, the predictions of the average case model have a reasonably good fit with the observed data. For the reduced Republican examples, r^2 = 0.983 and for the reduced Democrats, r^2 = 0.993. Nonetheless, there are some points at which the predicted accuracy is not within a 95% confidence interval around the observed mean.



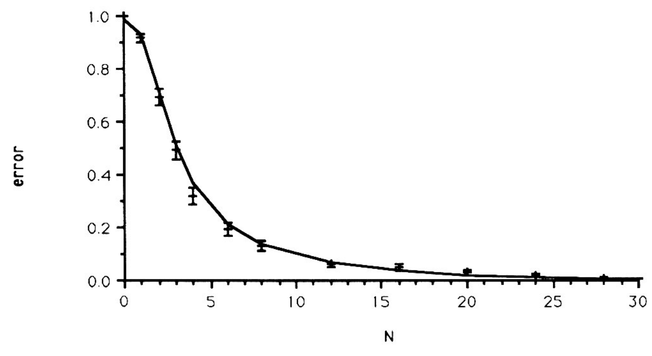


Figure 7. The error of the wholist algorithm on the reduced Republican data set (upper) and the reduced Democrat data set (lower). The points represent the empirical means, the y-bars are the 95% confidence interval around those means and the curves are the value predicted by the average case model.

5. Limitations of the current framework

We view this work as the first step toward a framework for average case modeling of learning algorithms. In the PAC-model, one need only specify the size, (or VC-dimension) of the hypothesis space searched by the learning algorithm. The average case framework requires a detailed analysis of an individual algorithm to determine the conditions under which the algorithm changes the hypothesis. We have performed this analysis for one of the simplest learning algorithms (wholist), and then expanded this analysis to algorithms that use the wholist algorithm for learning a domain theory for explanation-based learning. There are several properties of the wholist algorithm that facilitated this analysis:

- The hypothesis produced by the algorithm is independent of the order in which the examples are encountered. Therefore, calculating the probability that an irrelevant feature remains is a function of the number of training examples, but not the order. It would be possible, but computationally expensive, to compute the probability of a feature remaining for each possible order of training examples. However, due to this complexity, we would predict that a model of the nearest neighbor algorithm (Cover & Hart, 1967) would be easier to construct than a model of a variant of the nearest neighbor algorithm that only retains misclassified instances (Kibler & Aha, 1988), since the latter algorithm is dependent on the order of examples.

- The algorithm maintains a single hypothesis. A model of algorithms that maintain multiple hypotheses (e.g., Mitchell's (1982) version space algorithm) may require computing the probability that each hypothesis in the entire hypothesis space is eliminated from consideration. Although the hypothesis maintained by wholist may be thought of as summarizing a set of more general, consistent hypothesis, in our simulations, classification of test examples is made by this unique most specific hypothesis.
- A revision to the hypothesis maintained by the algorithm cannot make the hypothesis less accurate. The wholist algorithm has only one operator for revising a hypothesis. Algorithms such as POSTHOC (Pazzani, 1989) that have operators for making a hypothesis more general and operators for making a hypothesis more specific may prove more difficult to analyze since the use of an operator is dependent upon the accuracy of the current hypothesis as well as the current training example.
- A necessary and sufficient concept definition is maintained. It may prove difficult to analyze how the hypothesis is revised in algorithms such as COBWEB (Fisher, 1987) that maintain a probabilistic concept definition or neural network algorithms (e.g., Rumelhart, Hinton, & Williams, 1986). In these algorithms, the hypothesis is represented as a complex hierarchy or as weights on connections between nodes.
- Finally, the framework is designed for predicting accuracy, not speed-up. Therefore, the framework is not relevant to systems such as PRODIGY (Minton, 1988) that use explanation-based techniques to improve this aspect of performance.

Here, we have analyzed only algorithms for conjunctive concepts, but we anticipate that the framework will scale to similar algorithms using more complex hypotheses. Hirschberg and Pazzani (1992) have extended this model to k-CNF hypotheses. Our future plans include modeling more complex learning algorithms with more expressive concepts (e.g., k-DNF) and addressing issues of noise in the training data in order to model naturally occurring concepts.

6. Conclusion

In this paper, we have presented a framework for average case analysis of machine learning algorithms. Applying the framework consists of 1) understanding how an algorithm revises a hypothesis for a concept, 2) calculating the probability that a training example will be encountered that causes an inaccurate hypothesis to be revised, and 3) calculating the effect of revising a hypothesis on the accuracy of the hypothesis. The framework 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. We have applied the framework to three different learning algorithms. We have verified through experimentation that the equations accurately predict the expected accuracy and shown how the accuracy of the model degrades when data unknowingly violates these assumptions. Finally, we have shown how the average case model can be used to prove that the hypothesis produced by IOSC-TM will be at least as accurate as the hypothesis maintained by the wholist algorithm or by an algorithm that uses the wholist algorithm to learn a domain theory.

Acknowledgments

We would like to thank Dennis Kibler, David Haussler, Dan Hirschberg, Tom Shultz, Dennis Volper, Kamal Ali, Tim Cain, Jim Wogulis, Nick Flann, Hyam Hirsh, Tom Dietterich, Rob Holte, Jeff Shrager, and two anonymous referees for their helpful comments on this work. 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.

Notes

1. Since the time this paper was submitted for publication there has been further work on distribution specific and average case analysis of learning algorithms in Oblow (1991).
2. We will assume an odd number of training examples to avoid having to specify a strategy when there are equal number of positive and negative examples. There are a variety of such strategies possible (e.g., always guessing positive, guessing randomly, guessing the class of the last example, etc.).
3. *NM* stands for "not misclassified."
4. If irrelevant features are not independent, then the average case learning model would also require conditional probabilities for the irrelevant features. Here, we only consider the case when examples are drawn from a product distribution.
5. The coefficient of determination is frequently used to determine the goodness of fit of regression model to data (see Winer, 1971). It determines the fraction of total variation of a dependent variable that is accounted for by a model. In our case, the dependent variable (O) is the observed mean accuracy and the predicted value (P) is determined by equation 4. The following equation is used to calculate r^2 (where $\bar{O}$ is the mean value of O_i):

$$r^2 = 1 - \frac{\sum_i (O_i - P_i)^2}{\sum_i (O_i - \bar{O})^2}$$

   Note that linear regression uses r^2 to describe the goodness of fit of the equation $O_i = a + bP_i$ while here it is used to describe the goodness of fit of $O_i = P_i$.
6. One possible situation in which training examples of this kind occur is in the induction of causal rules. In this case, a training example of the form $\mathbf{s} \rightarrow (\mathbf{p}_1 \rightarrow \mathbf{p}_2)$ may represent the situation in which an action $\mathbf{p}_1$ was observed to occur in $\mathbf{s}$ and a teacher indicates that $\mathbf{p}_2$ is an effect of $\mathbf{p}_1$.
7. Note that the classification of a training example as a performance or foundational example is with respect to a specific performance task.
8. Truth maintenance implies that the composite hypothesis is updated immediately when a foundational example changes the hypothesis of a foundation rule. IOSC without truth maintenance waits until a performance example is misclassified before updating the composite hypothesis.
9. Note that, due to the feature dependencies, we are not advocating extending the analysis of the wholist algorithm to the k-CNF algorithm by extending the features in this way. See Hirschberg and Pazzani (1991) for an average case analysis of the k-CNF algorithm.

10. Note that this distribution is the worst distribution known and it has a slightly better lower bound than the distribution used in theorem 3.1 of Haussler et al. (1990) since it places heavy weight on an example y_0 outside of the shattered set. The lower bound is:

$$\frac{\text{VCdim}(h)}{2eN}$$

instead of

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

as given in theorem 3.1 of Haussler et al. (1990).
11. If the wholist algorithm is run on all examples of either class, it will eliminate all features from the initial hypothesis, i.e., it learns the function that is always true.

References

Atkinson, R., Bower, G., & Crothers, E. (1965). *An introduction to mathematical learning theory.* New York: John Wiley & Sons.

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.

Bruner, J.S., Goodnow, J.J., & Austin, G.A. (1956). *A study of thinking.* New York: Wiley.

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.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions of Information Theory, 13,* 21-27.

DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternate view. *Machine Learning, 1,* 145-176.

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 Eighth National Conference on Artificial Intelligence* (pp. 564-569). St. Paul, MN: Morgan Kaufmann.

Haussler, D. (1990). Probably approximately correct learning. *Proceedings of the Eighth National Conference on Artificial Intelligence* (p. 1101-1108). Boston: AAAI Press.

Haussler, D., Littlestone, N., & Warmuth, M. (1990). *Predicting {0, 1}-functions on randomly drawn points.* (Technical Report) USCS-CRL-90-54). Santa Cruz, CA: University of California, Santa Cruz.

Hembold, D., Sloan, R., & 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. *Proceedings of the Ninth International Machine Learning Conference* (pp. 206-211). Aberdeen: Morgan Kaufmann.

Hirsh, H. (1989). Combining empirical and analytical learning with version spaces. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 29-33). Ithaca, NY: 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, NY: ACM Press.

Kibler, D., & Aha, D. (1988). Comparing instance-averaging with instance-filtering learning algorithms. *Proceedings of the Third European Working Session on Learning* (pp. 63-79). Glasgow: Pitman.

Langley, P. (1989). Toward a unified science of machine learning. *Machine Learning, 3*(4).

Minton, S. (1988). Quantitative results concerning the utility of explanation-based learning. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 564-569). St. Paul. MN: Morgan Kaufmann.

Mitchell, T. (1982). Generalization as search. *Artificial Intelligence, 18,* 203-226.

Mitchell, T., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based learning: A unifying view. *Machine Learning*, *1*, 47–80.

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

Oblow, E. (1991). Implementation of Valiant's learnability theory using random sets. *Machine Learning*, *8*.

Pazzani, M. (1989). Explanation-based learning with weak domain theories. *The Sixth International Workshop on Machine Learning* (72–75). Ithaca, NY: Morgan Kaufmann.

Restle, F. (1959). A survey and classification of learning models. In R. Bush & W. Estes (Eds.), *Studies in mathematical learning theory*, pp. 415–428. Stanford, CA: Stanford University 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.

Sarrett, W., & Pazzani, M. (1989a). One-sided algorithms for integrating empirical and explanation-based learning. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 26–28). Ithaca, NY: Morgan Kaufmann.

Sarrett, W., & Pazzani, M. (1989b). *Average case analysis of empirical and explanation-based learning algorithms* (Technical Report). Irvine, CA: University of California, Irvine.

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

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

Winer, B. (1971). *Statistical principles in experimental design*. New York: McGraw-Hill.