An Analysis of Bayesian Classifiers

Pat Langley
Wayne Iba
Kevin Thompson

AI Research Branch (M/S 269-2)
NASA Ames Research Center
Moffett Field, CA 94035 USA

January 15, 1992

Abstract

In this paper we present an average-case analysis of the Bayesian classifier, a simple probabilistic induction algorithm that fares remarkably well on many learning tasks. Our analysis assumes a monotone conjunctive target concept, Boolean attributes that are independent of each other and that follow a single distribution, and the absence of attribute noise. We first calculate the probability that the algorithm will induce an arbitrary pair of concept descriptions; we then use this expression to compute the probability of correct classification over the space of instances. The analysis takes into account the number of training instances, the number of relevant and irrelevant attributes, the distribution of these attributes, and the level of class noise. In addition, we explore the behavioral implications of the analysis by presenting predicted learning curves for a number of artificial domains. We also give experimental results on these domains as a check on our reasoning. Finally, we discuss some unresolved questions about the behavior of Bayesian classifiers and outline directions for future research.

Function: learning; *Knowledge:* experience
Domain: classification; *Foundation:* mathematical/experimental

Primary contact: Pat Langley *Phone:* (415) 604–4878
Electronic mail: LANGLEY@PTOLEMY.ARC.NASA.GOV

Note: Without acknowledgements and references, this paper fits into 12 pages with dimensions 5.5 inches $\times$ 7.5 inches using 12 point LaTeX type. However, we find the current format more desirable. We have not submitted the paper to any other conference or journal.

1. Simple Bayesian Classifiers

One goal of research in machine learning is to discover principles that relate algorithms and domain characteristics to behavior. To this end, many researchers have carried out systematic experimentation with natural and artificial domains in search of empirical regularities (e.g., Kibler & Langley, 1988; Rendell & Cho, 1990). Others have focused on theoretical analyses, often in the paradigm of probability approximately correct learning (e.g., Kearns, Li, Pitt, & Valiant, 1987; Haussler, 1990). However, most experimental studies are based only on informal analyses of the learning task, and most formal analyses are worst case in nature, bearing only a distant relation to empirical results.

A third approach, proposed by Cohen and Howe (1988), involves the formulation of average-case models for specific algorithms and testing them through experimentation. Pazzani and Sarrett's (1990) work on conjunctive learning provides an excellent example of this technique, as does Hirschberg and Pazzani's (1991) work on inducing k-CNF concepts. By assuming information about the target concept, the number of irrelevant attributes, and the class and attribute frequencies, they obtained detailed predictions about the behavior of induction algorithms and used experiments to check their analyses. However, this research did not focus on algorithms that are typically used by the experimental and practical sides of machine learning, and it is important that such average-case analyses be extended to such methods.

Recently, there has been growing interest in Bayesian approaches to inductive learning. For example, Fisher (1987) has described Cobweb, an incremental algorithm for conceptual clustering that draws heavily on Bayesian ideas, and the literature reports a number of systems that build on this work (e.g., Allen & Langley, 1990; Iba & Gennari, 1991; Thompson & Langley, 1991; Yoo & Fisher, 1991). Cheeseman et al. (1988) have outlined AutoClass, a nonincremental system that uses Bayesian methods to cluster instances into groups. Other researchers have taken a different approach that focuses on the induction of Bayesian inference networks (e.g., Geiger, Paz, & Pearl, 1990).

These recent Bayesian learning algorithms are complex and not easily amenable to analysis, but they share a common ancestor that is simpler and more tractable. This supervised algorithm, which we will refer to as a *Bayesian classifier*, comes originally from work in pattern recognition (Duda & Hart, 1973). The method stores a simple probabilistic summary for each class; this summary contains the conditional probability of each attribute value given the class, as well as the probability (or base rate) of the class. This data structure has approximately the same representational power as a perceptron; it describes a single decision boundary through the instance space. Each time the algorithm encounters a new instance, it updates the probabilities stored with the specified class. Neither the order of training instances nor the occurrence of classification errors have any effect on this process. Upon being given an unclassified test instance, the classifier uses an evaluation function to rank the alternative classes, based on their probabilistic summaries, and assigns the instance to the class with the highest score.

Both the evaluation function and the summary descriptions used in Bayesian classifiers assume that attributes are statistically independent. Since this seems unrealistic for many natural domains, researchers have often concluded that the algorithm will behave poorly in comparison to other induction methods. However, no studies have examined the extent to which violation of this assumption leads to performance degradation. Moreover, the probabilistic approach should be quite robust with respect to both noise and irrelevant attributes. Clark and Niblett (1987) present evidence of the practicality of Bayesian classifiers. Although they originally included the method as a 'straw man' in comparisons with algorithms for decision-tree and rule induction, they found that it performed as well as the more sophisticated techniques on several domains. Table 1 presents the results for some experiments that demonstrate the utility of Bayesian classifiers. We randomly selected 25 separate test and training sets for each of five domains, using a 80% – 20% split. The table shows accuracy results and 95% confidence intervals for each domain, reporting asymptotic accuracy on test sets only. We show results for a Bayesian classifier, a simulation of C4 (Buntine & Caruana, 1991), and a frequency-based algorithm that simply guesses the modal class. The domains include the "small" soybean domain (Fisher & Schlimmer, 1988), the king-rook-king-pawn chess end game domain (Shapiro, 1987), a domain for predicting lymphography diseases (Cestnik, Konenenko, & Bratko, 1987), and two biological domains, one for predicting splice junctions in DNA sequences (Towell, Craven, & Shavlik, 1991), and another for predicting DNA promoters (Towell, Shavlik, & Noordweier, 1990).

Note that in four of the five domains tested, the mean accuracy for the Bayesian classifier was at least as high as that for the C4-like system. In addition, in the splice-junction domain, the Bayesian classifier performs comparably to the more knowledge-intensive KBANN algorithm (Towell et al., 1991). Note that we are not claiming superiority for the Bayesian classifier, but that it performs reasonably well across a variety of domains, in comparison to other well-known (and more recent) algorithms. Thus, there remain

Table 1. Percentage accuracies for two induction algorithms on five classification domains, along with default accuracy.

ALGORITHM	SOYBEAN DISEASE	CHESS END GAME	LYMPHOGRAPHY DIAGNOSIS	SPLICE JUNCTION	DNA PROMOTERS
BAYES	100.0 ± 0.0	$86.7 \pm$ X.X	$74.3 \pm$ X.X	94.0 ± 2.7	$89.6 \pm$ X.X
C4	97.9 ± 4.2	$97.7 \pm$ X.X	$72.1 \pm$ X.X	89.9 ± 3.4	$71.1 \pm$ X.X
FREQUENCY	36.2 ± 0.0	52.2 ± 0.0	56.7 ± 0.0	53.2 ± 0.0	50.0 ± 0.0

ambiguities about the behavior of Bayesian classifiers that a careful analysis might answer.

To simplify matters, we limit our analysis to the induction of conjunctive concepts. Furthermore, we assume that there are only two classes, that each attribute is Boolean, that attributes are independent of each other, that all relevant attributes follow the same probability distribution, and that all irrelevant attributes follow another. We divide our average-case study into two sections. In Section 2 we determine the probability that the algorithm will learn a particular pair of concept descriptions. After this, in Section 3 we derive the accuracy of an arbitrary pair of descriptions. Taken together, these expressions give us the overall accuracy of the learned concepts. We find that a number of factors influence behavior of the algorithm, including the number of training instances, the number of relevant and irrrelevant attributes, the amount of class and attribute noise, and the class and attribute frequencies. In Section 4 we examine the implications of our analysis by predicting behavior in specific domains, and we check our reasoning with experiments in these domains.

2. Probability of Induced Concepts

Consider a concept C defined as the monotone conjunction of r features $A_1, \ldots, A_r$ (i.e., in which none of the relevant features are negated). Also assume there are i irrelevant features $A_{r+1}, \ldots, A_{r+i}$. Let $P(A_j)$ be the probability of feature A_j occurring in an instance. The concept descriptions learned by a Bayesian classifier are completely determined by the n training instances it has observed. Thus, to compute the probability of each such concept description, we must consider different possible combinations of n training instances.

First let us consider the probability that the algorithm has observed exactly k out of n positive instances. If we let $P(C)$ be the probability of observing a positive instance and we let x be the observed fraction of positive instances, then we have

$$P(x = \frac{k}{n}) = \binom{n}{k} P(C)^k [1 - P(C)]^{n-k} \quad .$$

This expression also represents the probability that one has observed exactly $n - k$ negative instances. Since we assume that the concept is monotone conjunctive and that the attributes are independent, we have

$$P(C) = \prod_{j=1}^{r} P(A_j) \quad ,$$

which is simply the product of the probabilities for all relevant attributes.

A given number k of positive instances can produce many alternative descriptions of the positive class, depending on the instances that are observed. One can envision each such concept description as a cell in an $r + i$ dimensional matrix, with each dimension ranging from 0 to k, and with the count on dimension j representing the number of positive instances in which attribute A_j was present. In addition, one can envision a similar matrix for the negative instances, again having dimensionality $r + i$, but with each dimension ranging from 0 to $n - k$, and with the count on each dimension j representing the number of negative instances in which attribute A_j was present.

In both matrices, one can index each cell or concept description by a vector of length $r + i$. Let $P(cell_{\vec{u}})_k$ be the probability that the cell in the positive matrix indexed by vector $\vec{u}$, is generated by the algorithm; let $P(cell_{\vec{v}})_{n-k}$ be the analogous probability for a cell in the negative matrix. Then a weighted product of these terms gives us the probability that the learning algorithm will generate any particular pair of concept descriptions, which is

$$P(k, \vec{u}, \vec{v})_n = P(x = \frac{k}{n}) P(cell_{\vec{u}})_k P(cell_{\vec{v}})_{n-k} \quad .$$

In other words, one multiplies the probability of seeing k out of n positive instances, the probability of encountering cell $\vec{u}$ in the positive matrix, and the probability of encountering cell $\vec{q}$ in the negative matrix.

However, we must still determine the probability of a given cell from the matrix. For those in the positive matrix, this is straightforward, since the attributes remain independent when the instance is a member of a conjunctive concept. Thus, we have

$$P(cell_{\vec{u}})_k = \prod_{j=1}^{r+i} P(y_j = \frac{vecu_j}{k})$$

as the probability for $cell_{\vec{u}}$ in the positive matrix, where y_j represents the observed fraction of the k instances in which attribute A_j was present. Furthermore, the probability that one will observe A_j in exactly m out of k such instances is

$$P(y = \frac{m}{k}) = \binom{k}{m} P(A_j|C)^m [1 - P(A_j|C)]^{k-m} \quad .$$

In the absence of noise, we have $P(A_j|C) = 1$ for all relevant attributes and $P(A_j|C) = P(A_j)$ for all irrelevant attributes. We will return to the issue of noise shortly.

The calculation is more difficult for cells in the negative matrix. One cannot simply take the product of the probabilities for each index of the cell, since for a conjunctive concept, the attributes are not statistically independent. However, one can compute the probability that the $n - k$ observed negative instances will be composed of a particular combination of instances.

If we let $P(I_j|\bar{C})$ be the probability of I_j given a negative instance, we can use the multinomial distribution to compute the probability that exactly u_1 of the $n - k$ instances will be instance I_1, u_2 will be instance I_2, ..., and u_w will be instance I_w:

$$\frac{(n-k)!}{u_1!u_2!\ldots u_w!} P(I_1|\bar{C})^{u_1} P(I_2|\bar{C})^{u_2} \ldots P(I_w|\bar{C})^{u_w} \quad .$$

This expression gives us the probability of a particular combination of negative instances, and from that combination we can compute the concept description (i.e., cell indices) that result. Of course, two or more combinations of instances may produce the same concept description, but one simply sums the probabilities for all such combinations to get the total probability for the cell. All that we need to make this operational is $P(I_j|\bar{C})$, the probability of I_j given a negative instance. In the absence of noise, this is simply $P(I_j)/P(\bar{C})$.

We can extend the framework to handle class noise by modifying the definitions of three basic terms – $P(C)$, $P(A_j|C)$, and $P(I_j|\bar{C})$. Class noise involves the corruption of class names (i.e., replacing the actual class with its opposite) with a certain probability z between 0 and 1. The probability of the class after one has corrupted values is

$$P'(C) = (1-z)P(C) + z(1-P(C)) = P(C)[1-2z] + z \quad ,$$

as we note elsewhere (Iba & Langley, 1991).

For an irrelevant attribute A_j, the probability $P(A_j|C)$ is unaffected by class noise and remains equal to $P(A_j)$, since the attribute is still independent of the class. However, the situation for relevant attributes is more complicated. By definition, we can reexpress the corrupted conditional probability of a relevant attribute A_j given the (possibly corrupted) class C as

$$P'(A_j|C) = \frac{P'(A_j \wedge C)}{P'(C)} \quad ,$$

where $P'(C)$ is the noisy class probability given above. Also, we can rewrite the numerator to specify the situations in which corruption of the class name does and does not occur, giving

$$P'(A_j|C) = \frac{(1-z)P(C)P(A_j|C) + zP(\bar{C})P(A_j|\bar{C})}{P'(C)} \quad .$$

Since we know that $P(A_j|C) = 1$ for a relevant attribute, and since $P(A_j|\bar{C}) = [P(A_j) - P(C)]/P(\bar{C})$ for conjunctive concepts, we have

$$P'(A_j|C) = \frac{(1-z)P(C) + z[P(A_j) - P(C)]}{P(C)[1-2z] + z} \quad ,$$

which involves only terms that existed before corruption of the class name.

We can use similar reasoning to compute the post-noise probability of any particular instance given that it is negative. As before, we have

$$P'(I_j|\bar{C}) = \frac{P'(I_j \wedge \bar{C})}{P'(\bar{C})} = \frac{(1-z)P(\bar{C})P(I_j|\bar{C}) + zP(C)P(I_j|C)}{1 - (P(C)[1-2z] + z)} \quad ,$$

but in this case the special conditions are somewhat different. For a negative instance, we have $P(I_j|C) = 0$, so that the second term in the numerator becomes zero. In contrast, for a positive instance, we have $P(I_j|\bar{C}) = 0$, so that the first term disappears. Taken together, these conditions let us generate probabilities for cells in the negative matrix after one has added noise to the class name.

After replacing $P(C)$ with $P'(C)$, $P(A_j|C)$ with $P'(A_j|C)$, and $P(I_j|\bar{C})$ with $P'(I_j|\bar{C})$, the expressions earlier in this section let us compute the probability that a Bayesian classifier will induce any particular pair of concept descriptions (cells in the two matrices). The information necessary for this calculation is the number of training instances, the number of relevant and irrelevant attributes, their distributions, and the level of class noise. This analysis holds only for monotone conjunctive concepts and in domains with independent attributes, but many of the ideas should carry over to other domains.

3. Accuracy of Induced Concepts

To calculate overall accuracy after n training instances, we must sum the expected accuracy for each possible instance weighted by that instance's probability of occurance. More formally, the expected accuracy is

$$K_n = \sum_{j}^{I} P(I_j) K(I_j)_n \quad .$$

To compute the expected accuracy $A(I_j)_n$ for instance I_j, we must determine the accuracy for each pair of cells in the positive and negative matrices.

As mentioned earlier, a Bayesian classifier uses an evaluation function to assign an instance I_j to a class. The algorithm computes the score of each class description for I_j, and then places the instance in the class with the highest score, selecting one randomly in case of ties. We will define $accuracy(I_j)_{n,k,\vec{u},\vec{v}}$ for the pair of concept descriptions $\vec{u}$ and $\vec{v}$ to be 1 if this scheme correctly predicts I_j's class, 0 if it incorrectly predicts the class, and $\frac{1}{2}$ if a tie occurs. Since we assume the concept is conjunctive, an instance should be assigned to the positive class only if all relevant attributes are present, and to the negative class otherwise.

Following our previous notation, let n be the number of observed instances, k be the number of observed positive instances, m_j be the number of positive instances in which attribute A_j occurs, and q_j be the number of negative instances in which A_j occurs. For a given instance I_j, one can compute the score for the positive class description as

$$score(C)_j = \frac{k}{n} \prod_{j=1}^{r+i} \frac{u_j}{k}$$

and the score for the negative class description as

$$score(\bar{C}_j) = \frac{n-k}{n} \prod_{j=1}^{r+i} \frac{v_j}{n-k} \quad .$$

Although we assume that the classifier only stores probabilities $\frac{u_j}{k}$ for 'positive' attributes A_j, it can handle a negated attribute $\bar{A}_j$ in an instance by using $1 - \frac{u_j}{k}$. Of course, u_j will be 0 if an instance contains A_j but one has never observed attribute for a given class, and a similar problem occurs if one has never encountered a class. To avoid an inordinate number of ties, we follow Clark and Niblett's (1987) suggestion of replacing 0 with a small value, such as $\frac{1}{2n}$.

To compute the expected accuracy for instance I_j, we multiply the accuracy for each pair of cells by the probability of each pair, multiply this by the probability of observing exactly k out of n positive instances, and sum over all possible values of k and pairs of cells. Thus, we have

$$K(I_j)_n = \sum_{k=0}^{n} P(x = \frac{k}{n}) \sum_{\vec{u}}^{\mathbf{U}} P(cell_{\vec{u}}) \sum_{\vec{v}}^{\mathbf{V}} P(cell_{\vec{v}}) \; accuracy(I_j)_{n,k,\vec{u},\vec{v}} \quad ,$$

which refers to the probabilities we derived in the previous section but where the second and third sums are over the possible vectors that index into the positive and negative matricies $\mathbf{U}$ and $\mathbf{V}$. To complete our calculations, we need an expression for $P(I_j)$, which is simply the product of the probabilities of the features present in I_j.

4. Implications for Learning Behavior

Although the equations in the previous two sections give a formal description of the Bayesian classifier's behavior, their implications are not transparent. In this section, we examine the effects of various domain characteristics on the algorithm's classification accuracy. In each case, we present multiple learning curves, showing how the number of training instances interacts with another of the factors mentioned in our analysis. However, because the number of possible concept descriptions grows exponentially with the number of training instances and the number of attributes, our predictions have been limited to 20 instances and five attributes.

In addition to theoretical predictions, we report learning curves that summarize runs on 100 randomly generated training sets. Each curve reports the

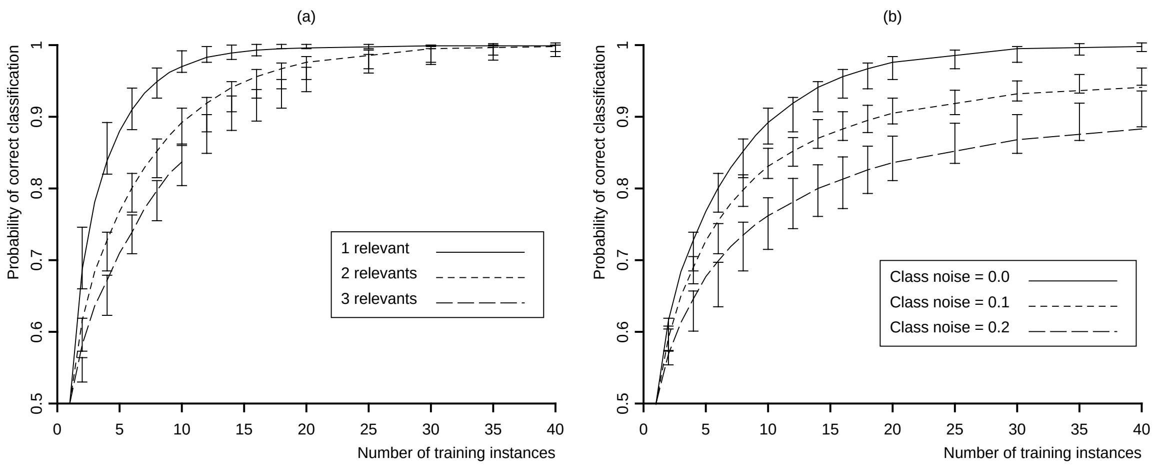

Figure 1. Predictive accuracy of a Bayesian classifier in a conjunctive concept, assuming the presence of one irrelevant attribute, as a function of training instances and (a) number of relevant attributes and (b) amount of class noise. The lines represent theoretical learning curves, whereas the error bars indicate experimental results.

average classification accuracy over these runs on a single test set of 200 randomly generated instances that contained no noise. In each case, we bound the mean accuracy with 95% confidence intervals to show the degree to which our predicted learning curves fit the observed ones. These experimental results provide an important check on our reasoning, and they revealed a number of problems during development of the analysis.

For instance, Figure 1 shows the effects of concept complexity on the rate of learning in the Bayesian classifier when no noise is present. In this case, we held the number of irrelevant attributes i constant at one and we held their probability of occurrence $P(A)$ constant at $\frac{1}{2}$. We varied both the number of training instances and the number of relevant attributes r, which determine the complexity of the conjunctive target concept. To normalize for effects of the base rate, we also held $P(C)$, the probability of the concept, constant at $\frac{1}{2}$; however, this meant that, for each of the r relevant attributes, $P(A)$ was $P(C)^{\frac{1}{r}}$, and thus varied for the different conditions.[1]

As typical with learning curves, the initial accuracies begin low (at $\frac{1}{2}$) and gradually improve with increasing numbers of training instances. The effect

1. An alternative approach would hold $P(A)$ constant for relevant attributes, causing $P(C)$ to become $P(A)^r$. This nudges the initial accuracies upward but otherwise has little effect on the learning curves.

of concept complexity also agrees with our intuitions; introducing additional features into the target concept slows the learning rate, but it does not affect asymptotic accuracy, which is always 1.0 for conjunctive concepts on noise-free test cases. Also, the rate of learning appears to degrade gracefully with increasing complexity. The predicted and observed learning curves are in close agreement, which lends confidence to our average-case analysis. Theory and experiment showed similar effects when we varied the number of irrelevant attributes; learning rate slowed as we introduced more misleading features, but the algorithm gradually converged on perfect accuracy.

Figure 2 presents similar results on the interaction between class noise and the number of training instances. Here we held the number of relevant attributes constant at two and the number of irrelevants constant at one, and we examined three separate levels of class noise. Following the analysis, we assumed the test instances were free of noise, which normalizes accuracies and eases comparison. As one might expect, increasing the noise level z decreases the rate of learning. However, the probabilistic nature of the Bayesian classifier leads to graceful degradation, and asymptotic accuracy should be unaffected. As before, we find a close fit between the theoretical behavior and the experimental learning curves. Although our analysis does not incorporate attribute noise, experiments with this factor produced similar results. In this case, equivalent levels led to somewhat slower learning rates, as one would expect given that attribute noise can corrupt multiple values, whereas class noise affects only one.

5. Discussion

In this paper we developed an analysis of a simple Bayesian classifier. The analysis requires that the concept be monotone conjunctive, and that the number and freqencies of relevant and irrelevant attributes be known. Given this information, the equations compute the expected classification accuracy after a given number of training instances. We plotted the predicted behavior of the simple Bayesian classifier, varying the number of relevant attributes and the level of class noise. As a test of our analysis, we ran the algorithm on artificial domains having the same characteristics. These empirical results proved to be invaluable as we discovered several errors in our reasoning.

We also compared the behavior of the Bayesian classifier to that of a more sophisticated classification mechanism, C4. In at least one of these domains, the Bayesian classifier out-performed C4 and in most, performed comparably. We interpret these results as supporting evidence that simple mechanisms should not be summarily rejected. However, this is not to say that everyone should be using Bayesian classifiers. Instead, the results reveal the difficulty in finding challenging real-world data sets and in assessing the significance of subsequent results on such a data set. This paper hopefully underscores the importance of multiple types of evaulation – comparative, theoretical, and empirical.

In the future, there are a number of directions that we intend to continue this work – extend the analysis and run additional experiments. Most importantly, the analysis should be extended in several ways. Currently, we have derived equations for analyzing only class noise, but as others have shown (Angluin & Laird***, 1987), attribute noise is more problematic to a learning algorithm. We have developed the equations for dealing with attribute noise but the treatment is more complicated than that reported here and does not fit within the space constraints of the current paper.

We also intend to relax some of the assumptions made by the present analysis. In particular, attributes in real domains do not all follow a single frequency distribution; instead they each have their own probabilities. Following Hirschberg and Pazzani (1991) we should extend the analysis to treat each attribute individually. Although this will complicate matters somewhat, we believe it will amount to converting terms raised to powers in the current equations, to product distributions in an extended version. Similarly, we need to relax the constraint that target concepts must be monotone conjunctive. Again, we cannot expect target concepts in the real world to satisfy this assumption.

The second general direction in which we can extend the present work involves running additional and more extensive experiments. Even with the current analysis and its assumptions, we could run experiments with more complex domains and see to what extent violations of the assumptions alter the observed behavior of the algorithm. Additionally, we could analyze the attribute frequencies in several of the popular domains and see how well the analytic model, given the computed frequencies as inputs, predicts the empirical behavior on the data set. This is an important direction we intend to pursue with the general type of research represented in this paper. Overall, we are quite encouraged by the results obtained here; we have demonstrated that a simple bayesian classifier compares favorably with a more complex learning and classification mechanism, and, for a restricted class of domains, that the simple mechanism is amenable to an average-case analysis based on a set of characteristics describing the domain.

Acknowledgements

Thanks to Kimball Collins and Andy Philips for discussions that helped clarify the ideas in this paper and for software that helped produce timely results.

References

Allen, J. A., & Langley, P. (1990). Integrating memory and search in planning. *Proceedings of the 1990* DARPA *Workshop on Innovative Approaches to Planning, Scheduling, and Control* (pp. 301–312). San Diego, CA: Morgan Kaufmann.

Buntine, W., & Caruana, R. (1991). *Introduction to IND and Recursive Partitioning* (NASA Technical Report FIA-91-28). Moffett Field, CA: NASA Ames Research Center.

Cestnik,G., Konenenko,I, & Bratko,I. (1987). Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users. In I.Bratko & N.Lavrac (Eds.) *Progress in Machine Learning*, 31-45, Sigma Press.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AUTOCLASS: A Bayesian classification system. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 54–64). Ann Arbor, MI: Morgan Kaufman.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*, 261–284.

Cohen, P. R., & Howe, A. E. (1988). How evaluation guides AI research. *AI Magazine*, *9*, 35–43.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis.* New York: John Wiley & Sons.

Fisher,D. H. & Schlimmer, J. C. (1988). Concept simplification and predictive accuracy. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 22-28). Ann Arbor, Michigan: Morgan Kaufmann.

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

Geiger, D., Paz, A., & Pearl, J. (1990). Learning causal trees from dependency information. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 770–776). Boston, MA: AAAI Press.

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

Iba, W., & Gennari, J. H. (1991). Learning to recognize movements. In D. H. Fisher, M. J. Pazzani, & P. Langley (Eds.), *Concept formation: Knowledge and experience in unsupervised learning*. San Mateo, CA: Morgan Kaufmann.

Iba, W., & Langley, P. (1992). *Induction of one-level decision trees*. Unpublished manuscript, AI Research Branch, NASA Ames Research Center, Moffett Field, DA.

Kearns, M., Li, M., Pitt, L., & Valiant, L. G. (1987). Recent results on Boolean concept learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 337–352). Irvine, CA: Morgan Kaufmann.

Hirschberg, D. S., & Pazzani, M. J. (1991). *Average-case analysis of a k-CNF learning algorithm* (Technical Report 91-50). Irvine: University of California, Department of Information & Computer Science.

Kibler, D., & Langley, P. (1988). Machine learning as an experimental science. *Proceedings of the Third European Working Session on Learning* (pp. 81–92). Glasgow: Pittman.

Pazzani, M. J., & Sarrett, W. (1990). Average-case analysis of conjunctive learning algorithms. *Proceedings of the Seventh International Conference on Machine Learning/* (pp. 339–347). Austin, TX: Morgan Kaufmann.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.

Rendell, L., & Cho, H. (1990). Empirical learning as a function of concept character. *Machine Learning*, *5*, 267–298.

Shapiro, A. (1987). *Structured induction in expert systems*. Reading, MA: Addison Wesley.

Thompson, K., & Langley, P. (1991). Concept formation in structured domains. In D. H. Fisher, M. J. Pazzani, & P. Langley (Eds.), *Concept formation: Knowledge and experience in unsupervised learning*. San Mateo, CA: Morgan Kaufmann.

Towell, G. G., Craven, M.W., & Shavlik, J. (1991). Constructive induction in knowledge-based networks. *Proceedings of the Eighth International Machine Learning Workshop* (pp. 213–217). Evanston: Morgan Kaufmann.

Towell, G., Shavlik, J, & Noordewier, M.O. (1990). Refinement of approximate domain theories by knowledge-based neural networks. *Proceedings of the Eighth National Conference of the American Association for Artificial Intelligence* (pp. 861–866). Boston: AAAI Press.

Yoo, J., & Fisher, D. (1991). Concept formation over problem-solving experience. In D. H. Fisher, M. J. Pazzani, & P. Langley (Eds.), *Concept formation: Knowledge and experience in unsupervised learning*. San Mateo, CA: Morgan Kaufmann.