Induction of Recursive Bayesian Classifiers

Pat Langley

Learning Systems Department, Siemens Corporate Research
755 College Road East, Princeton, New Jersey 08540 USA

Abstract. In this paper, we review the induction of simple Bayesian classifiers, note some of their drawbacks, and describe a recursive algorithm that constructs a hierarchy of probabilistic concept descriptions. We posit that this approach should outperform the simpler scheme in domains that involve disjunctive concepts, since they violate the independence assumption on which the latter relies. To test this hypothesis, we report experimental studies with both natural and artificial domains. The results are mixed, but they are encouraging enough to recommend closer examination of recursive Bayesian classifiers in future work.

1. Introduction

In recent years, there has been growing interest in probabilistic methods for induction. Although much of the recent work in this area [e.g., 6] has focused on unsupervised learning, the approach applies equally well to supervised tasks. Such methods have long been used within the field of pattern recognition [4], but they have only recently received attention within the machine learning community [3, 7, 8, 9].

In this paper we review the most straightforward probabilistic approach to supervised learning – the induction of simple Bayesian classifiers. We also examine this method's apparent drawbacks and propose a revised algorithm that constructs a hierarchy of probabilistic summaries. We present an illustrative domain that this approach can handle but that the simpler scheme cannot, and we report experimental studies of the two algorithms on both natural and artificial induction tasks. Finally, we discuss work on related approaches and some directions for future research.

2. The induction of simple Bayesian classifiers

The most straightforward and widely tested method for probabilistic induction is known as the *simple Bayesian classifier*. This scheme represents each concept with a single probabilistic summary. In particular, each description has an associated class probability or base rate, $p(C_k)$, which specifies the prior probability that one will observe a member of class C_k. Each description also has an associated set of conditional probabilities, specifying a probability distribution for each attribute. In nominal domains, one typically stores a discrete distribution for each attribute in a description. Each $p(v_j|C_k)$ term specifies the probability of value v_j, given an instance of concept C_k. In numeric domains, one must represent a *continuous* probability distribution for each attribute. This requires that one assume some general form or model, and one usually selects the normal distribution. Conveniently, a given normal curve can be represented entirely in terms of its mean μ and variance σ^2. Moreover, the probability for a given numeric value v can be determined from the normal probability density function.

2.1 Prediction in Bayesian classifiers

To classify a new instance I, a Bayesian classifier applies Bayes' theorem to determine the probability of each description given the instance,

$$p(C_i|I) = \frac{p(C_i)p(I|C_i)}{p(I)} \quad .$$

However, since I is a conjunction of j values, one can expand this expression to

$$p(C_i|\bigwedge v_j) = \frac{p(C_i)p(\bigwedge v_j|C_i)}{\sum_k p(\bigwedge v_j|C_k)p(C_k)} \quad ,$$

where the denominator sums over all classes and where $p(\bigwedge v_j|C_i)$ is the probability of the instance I given the class C_i. After calculating these probabilities for each description, the algorithm assigns the instance to the class with the highest overall probability.

In order to make the above expression operational, one must still specify how to compute the term $p(\bigwedge v_j|C_k)$. Typically, one assumes independence of attributes, so that the classifier can calculate it using the equality

$$p(\bigwedge v_j|C_k) = \prod_j p(v_j|C_k),$$

where the values $p(v_j|C_k)$ represent the conditional probabilities stored with each class.

In some cases, the training data may produce a zero for some base rate or conditional probability. Since the classification decision involves multiplication, this overwhelms the effects of other factors. Clark and Niblett [3] avoid this problem by replacing zero entries with $p(C_i)/N$, where N is the number of training examples. The factor $1/N$ represents the belief that this entry has a near-zero value as a function of the size of the training set.

2.2 Learning in Bayesian classifiers

Learning in a Bayesian classifier is an almost trivial matter. The simplest implementation increments a count each time it encounters a new instance, along with a separate count for a class each time it observes an instance of that class. Together, these counts let the classifier estimate $p(C_k)$ for each class C_k. In addition, for each instance of a class that has a given nominal value, the algorithm updates a count for that class-value pair. Together with the second count, this lets the classifier estimate $p(v_j|C_k)$. For each numeric attribute, the method retains and revises two quantities, the sum and the sum of squares, which let it compute the mean and variance for a normal curve that it uses to find $p(v_j|C_k)$. In domains that can have missing attributes, it must include a fourth count for each class-attribute pair.

In contrast to many induction methods, which learn only when they make some error, a Bayesian classifier incorporates information from *every* instance that it encounters. Thus, the action taken by the learning component is entirely independent of whether the performance element makes the correct classification. The basic process can operate either incrementally or nonincrementally, since the order of training instances has no effect on learning. However, one can usefully view the Bayesian classifier as carrying out a hill-climbing search through the space of probabilistic concepts, in that it retains a single summary description at each point during processing. This makes the learning algorithm quite efficient.

2.3 Strengths and limitations of simple Bayesian classifiers

Bayesian classifiers should have advantages over many induction algorithms. For example, they should be inherently robust with respect to noise, due to their collection of class and conditional probabilities. Similarly, their statistical basis should also let them handle large numbers of irrelevant attributes. Langley, Iba, and Thompson [9] present an average-case analysis of these factors' effect on the algorithm's behavior for a specific class of target concepts.

However, the basic approach relies on an important assumption: that the instances in each class can be summarized by a single probabilistic description, and that these are sufficient to distinguish the classes from each other. If we represent each attribute value as a feature that may be present or absent, this is closely related to the assumption of linear separability in early work on neural networks. Other encodings lead to a more complex story, but the effect is nearly the same. Nevertheless, like perceptrons, Bayesian classifiers are typically limited to learning classes that occupy contiguous regions of the instance space.

Figure 1 shows an idealized, noise-free training set that illustrates this difficulty. The domain involves three classes of cells – one from `healthy` patients, another from patients with `lethargia`, and a third from patients with the disease `burpoma`. Four attributes describe the observed cells – the number of `nuclei`, the number of `tails`, the `color` of the cell body, and the `thickness` of the cell walls; each such attribute takes on one of two possible values. Running the 12 training cases in the figure through a simple Bayesian classifier produces one probabilistic summary for each class and, combined with Bayes' rule, these descriptions correctly classify all training instances from the `lethargia` and `burpoma` classes. However, it misclassifies two of the `healthy` cells as instances of `lethargia`. The basic problem is a representational one; the `healthy` cases

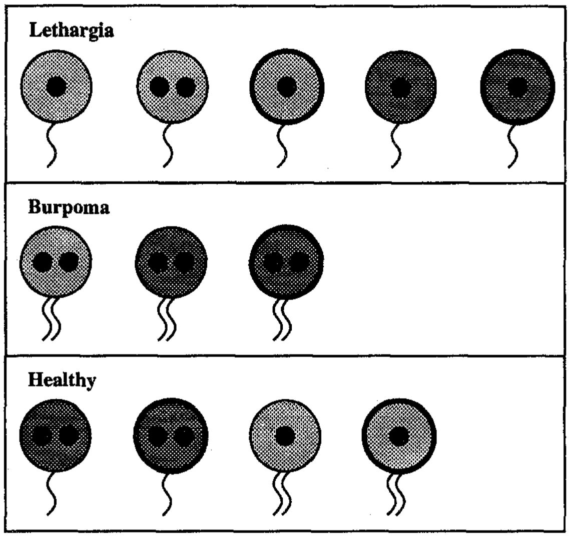


Figure 1. Training instances from an idealized cell domain, involving four attributes and three classes, that cannot be discriminated by a simple Bayesian classifier.

occupy two noncontiguous regions of the instance space, and one cannot represent such disjunctive situations with a single probabilistic summary for each class. In the remainder of the paper, we explore another extension that moves beyond the above representational limitation.

3. The formation of recursive Bayesian classifiers

The assumptions that underlie simple Bayesian classifiers are similar to those commonly made in curve fitting. The technique of linear regression posits that one can approximate the function in question by a single straight line, and this approach fares remarkably well on many tasks. However, in some domains one must resort to *piecewise* linear methods, which fit straight lines only to local portions of the overall curve.

One can apply a similar idea to supervised induction, identifying regions of the instance space in which the independence assumption holds and constructing a simple Bayesian classifier for each such region. One approach to determining such regions, which we will refer to as the RBC algorithm, groups instances by their associated class names, then uses probabilistic averaging to generate intensional descriptions for each cluster. Next, it uses these descriptions to reassign instances, producing a revised partition and summaries. In most situations, the resulting descriptions will not completely discriminate among the classes, so the revised partition and descriptions will differ from the original ones.

If a cluster contains instances from more than one class, this suggests that some refinement is necessary. Thus, the algorithm calls on itself recursively to further subdivide the data and form more specific concept descriptions for each subclass. This approach to partitioning makes less sense if one's goal is a monothetic hierarchy, in which case splitting on one of the predictive attributes is much more direct. Nor can one easily adapt this scheme to handle unsupervised training data, since it relies on class information to form partitions. On the other hand, it provides an elegant approach to generating polythetic hierarchies from supervised data.

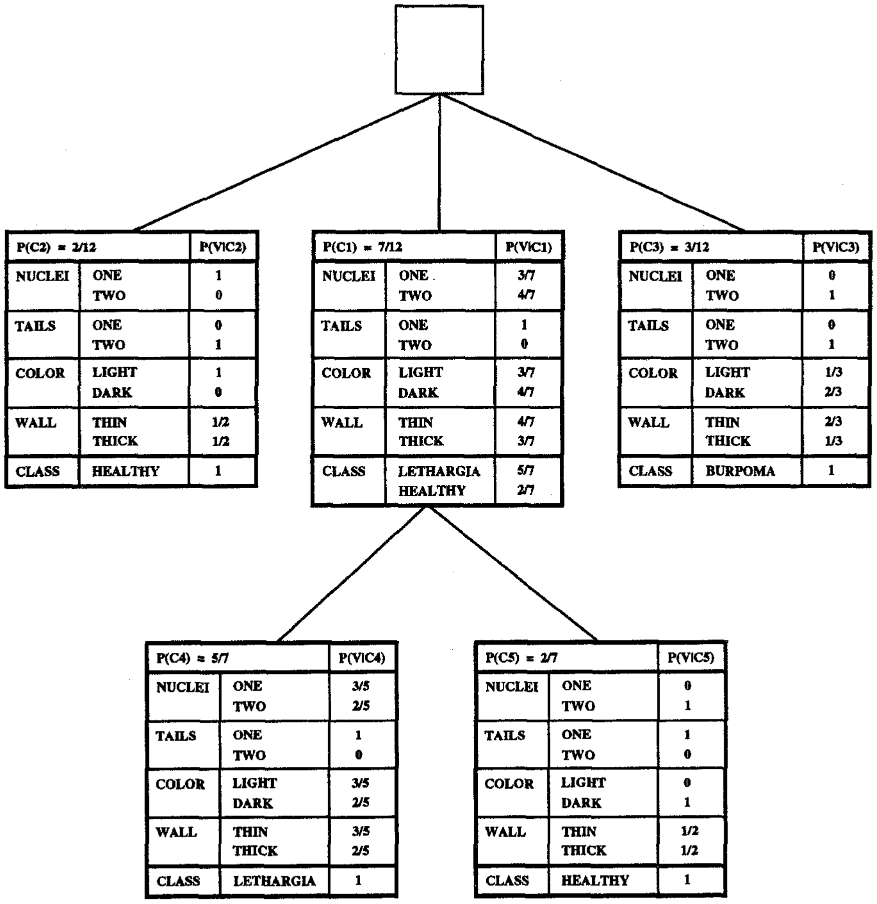
Recall that, when used in isolation, a Bayesian classifier is severely limited in its representational ability, and thus in its ability to learn. The current approach should not suffer from such limitations because the hierarchy stores knowledge at multiple levels of abstraction. Subdivisions at lower levels overcome the representational drawbacks at a given level, letting the hierarchy as a whole represent complex concepts even though each level only describes simple ones.

Figure 2 depicts the concept hierarchy generated by RBC for the training data in Figure 1. As in a simple classifier, each node includes a base rate and a conditional probability for each attribute-value pair. In generating this tree, the algorithm first partitions the training instances into three clusters, one for each class, and then uses a Bayesian classifier to produce a probabilistic summary for each set. The resulting descriptions predict the correct class names for all instances of **lethargia** and **burpoma**, as well as for the second two **healthy** cases. However, it assigns the first two **healthy** cells to the **lethargia** class.

In response, RBC generates a revised partition based on these three groups of instances, then computes a revised probabilistic summary for each one. Because one of the clusters contains instances from two classes, it calls itself recursively on this subset of instances. Thus, the algorithm creates one partition for the five **lethargia** cases and another for the two **healthy** instances, after which it invokes the Bayesian classifier a second time to produce descriptions for each such cluster. This time the probabilistic summaries correctly predict the class associated with each instance, so RBC halts its construction process.

4. Comparative studies of Bayesian classifiers

We have presented intuitive arguments for the superiority of recursive Bayesian classifiers over their simpler cousins, and we have given an illustrative example in which the former outperforms the latter. However, it remains an empirical question whether the RBC algorithm fares better than a simple Bayesian classifier in general. To answer this question, we carried out experiments with both natural domains from the UCI repository and artificial ones designed to test specific hypotheses. For each study, we generated 20 training and test sets, randomly drawn from the domain in question; in each case, we report the average



P(C2) = 2/12		P(V|C2)
NUCLEI	ONE	1
	TWO	0
TAILS	ONE	0
	TWO	1
COLOR	LIGHT	1
	DARK	0
WALL	THIN	1/2
	THICK	1/2
CLASS	HEALTHY	1

P(C1) = 7/12		P(V|C1)
NUCLEI	ONE	3/7
	TWO	4/7
TAILS	ONE	1
	TWO	0
COLOR	LIGHT	3/7
	DARK	4/7
WALL	THIN	4/7
	THICK	3/7
CLASS	LETHARGIA	5/7
	HEALTHY	2/7

P(C3) = 3/12		P(V|C3)
NUCLEI	ONE	0
	TWO	1
TAILS	ONE	0
	TWO	1
COLOR	LIGHT	1/3
	DARK	2/3
WALL	THIN	2/3
	THICK	1/3
CLASS	BURPOMA	1

P(C4) = 5/7		P(V|C4)
NUCLEI	ONE	3/5
	TWO	2/5
TAILS	ONE	1
	TWO	0
COLOR	LIGHT	3/5
	DARK	2/5
WALL	THIN	3/5
	THICK	2/5
CLASS	LETHARGIA	1

P(C5) = 2/7		P(V|C5)
NUCLEI	ONE	0
	TWO	1
TAILS	ONE	1
	TWO	0
COLOR	LIGHT	0
	DARK	1
WALL	THIN	1/2
	THICK	1/2
CLASS	HEALTHY	1

Figure 2. A probabilistic concept hierarchy generated by the RBC algorithm from the training instances in Figure 1. Each node contains a probabilistic summary, which the recursive classifier uses to sort instances and to make predictions.

accuracy of each algorithm on the test set after different numbers of training cases, along with 95% error bars for each point.

Figure 3 (a) presents the results of a comparative study on a domain that involves only nominal attributes. This data set [6] indicates votes ('yea' or 'nea') of the 435 members of the U.S. House of Representatives on 16 issues. The class name corresponds to the member's party, Democrat (267) or Republican (168). By the time they have seen 15 training cases, both algorithms have acquired probabilistic summaries that can predict a member's party with 90% accuracy, though the recursive method does noticeably worse early in the run. However, many induction algorithms produce similar results on this domain, making it useful mainly as an adequacy check. The most interesting result is that the simple Bayesian classifier performs as well as many more sophisticated methods.

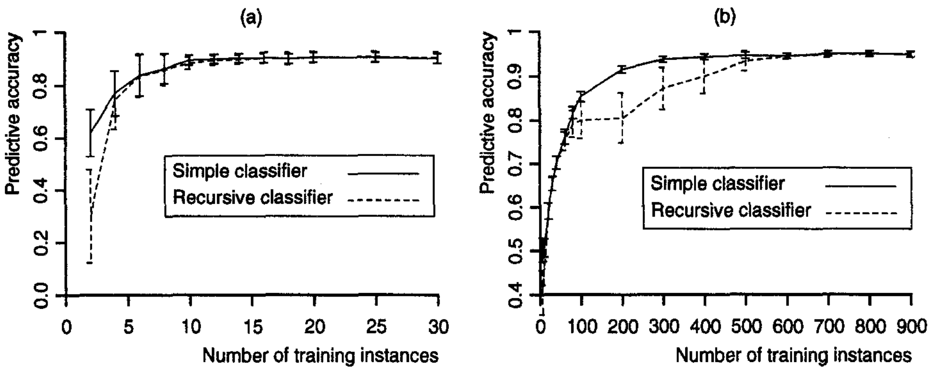


Figure 3. Learning curves for the simple Bayesian classifier and the recursive Bayesian classifier on (a) Congressional voting records and (b) splice junction data.

However, we find very similar effects in Figure 3 (b), which shows learning curves for a more complex domain, this one involving data on "splice junctions" in DNA sequences [12]. The domain includes three classes and some 60 nominal attributes, which specify the nucleotides that precede and follow the position to be classified. The rate of learning in this domain is much slower than for the voting records, but the asymptotic accuracy is slightly higher (95%). As before, the two algorithms reach the same level, but RBC requires more training cases than the simple classifier.

The pattern repeats in Figure 4 (a), which summarizes results on a domain that involves determining, based on nine numeric measures, whether or not a glass fragment came from a window. Again the asymptotic accuracy for both techniques hovers around 90%, with the recursive Bayesian classifier appearing to take slightly more training to reach this level, though the error bars overlap in this case. Analogous learning curves even emerge on the data for heart diseases (originally collected by R. Detrano), which involve two classes and 13 attributes, some numeric and some nominal. In this case the curves are statistically indistinguishable, though RBC's mean accuracy is consistently below that for the simple Bayesian classifier.

Naturally, these results are disappointing and deserve some explanation. Inspection of the induced knowledge structures for these domains reveal that RBC constructs probabilistic decision trees that are quite shallow, in many cases only two levels deep. This suggests that the domains used in our studies closely approximate the independence assumption, making the simple Bayesian classifier appropriate for them, and that other factors (such as noise) are responsible for both methods' imperfect asymptotic accuracies. This does not explain the depression in RBC's learning curve on the splice junction data, but it accounts for most of the observed phenomena.

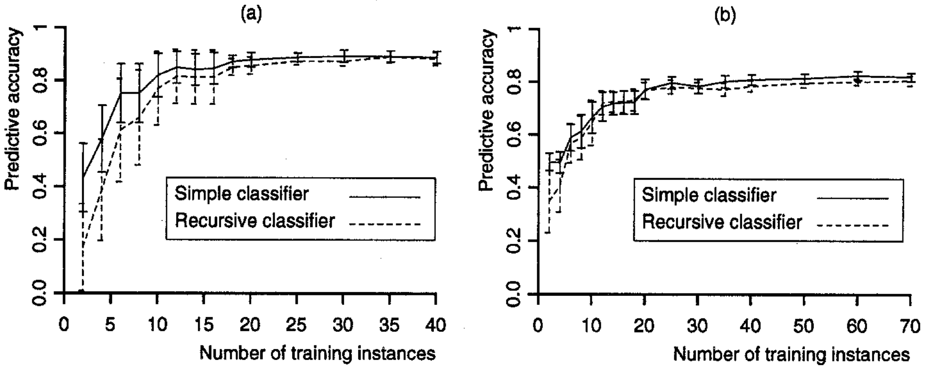


Figure 4. Learning curves for the simple Bayesian classifier and the recursive Bayesian classifier on (a) glass classification and (b) heart disease records.

This theory suggests that we should turn to artificial or synthetic domains to reveal the difference between the two learning algorithms. In response, we ran further comparative studies on a set of noise-free domains that involved two classes, three relevant Boolean attributes (A, B, and C), and three irrelevant Boolean attributes (D, E, and F). We varied the number of disjunctive terms in the target concept from one to four. In particular, we used the target concepts

$$\begin{array}{l}(A \wedge B \wedge C) \\ (A \wedge B \wedge C) \vee (A \wedge \bar{B} \wedge \bar{C}) \\ (A \wedge B \wedge C) \vee (A \wedge \bar{B} \wedge \bar{C}) \vee (\bar{A} \wedge \bar{B} \wedge C) \\ (A \wedge B \wedge C) \vee (A \wedge \bar{B} \wedge \bar{C}) \vee (\bar{A} \wedge \bar{B} \wedge C) \vee (\bar{A} \wedge B \wedge \bar{C})\end{array}$$

which include increasing numbers of nonadjecent vertices of the cube defined by the three relevant features. In each case, adding a new disjunctive term introduces another noncontiguous region to the instance space.

Figure 5 presents the learning curves for both learning algorithms on the first two of these concepts. Each curve is averaged over 20 randomly selected training sets, with the test set including the entire space of 64 instances. As before, the error bars indicate 95% confidence intervals. As expected, the asymptotic accuracies of the simple and recursive Bayesian classifiers approach 100% on the first (conjunctive) domain, as shown in Figure 5 (a), but the more complex algorithm takes more training cases to reach this level, as we found in the natural domains. However, the results for the second (disjunctive) target concept in Figure 5 (b) are quite different. Again, the recursive algorithm starts off worse than its simpler counterpart, but at ten training instances, the learning curves cross over, with the asymptotic accuracy of RBC being significantly higher than that for the simple Bayesian classifier.

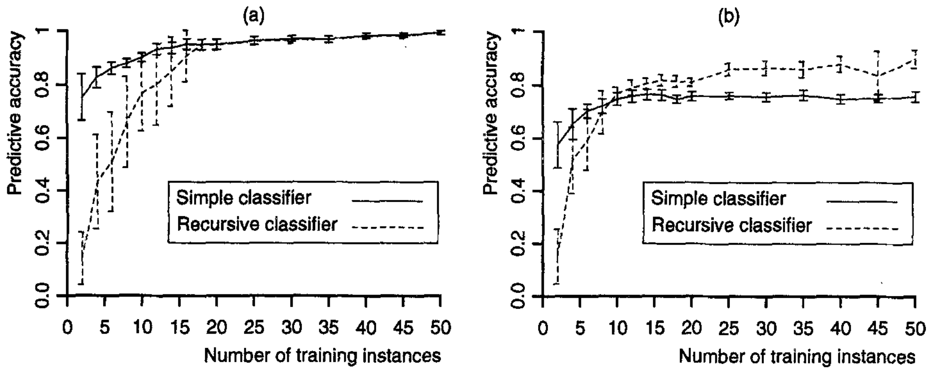


Figure 5. Learning curves for the simple Bayesian classifier and the recursive Bayesian classifier on artificial domains involving (a) a conjunctive target concept and (b) a two-disjunct concept.

The learning curves in Figure 6 provide additional evidence of the recursive scheme's superiority in disjunctive domains. Here we see the same pattern for the three-disjunct and four-disjunct concepts as we found in Figure 5 (b). The RBC algorithm's initial accuracy is lower than that for the simple Bayesian classifier, but crossovers occur between the fifth and tenth training instances, with the more sophisticated method outperforming its simpler counterpart after that point. These results support our original intuitions about the conditions under which RBC will outperform a scheme that relies on the independence assumption. However, the results of the simple Bayesian classifier on the four natural domains remains impressive, and one should by no means abandon it as a useful tool for inductive learning.

5. Related approaches to induction

The idea of organizing knowledge in a tree or hierarchy is far from new, going back to Feigenbaum's [5] EPAM. However, such early algorithms and their descendents, including Quinlan's C4.5 [11] and its relatives, use a single attribute to partition the instance space at each level of the tree. As others have noted, this 'monothetic' approach encounters difficulties when the decision boundaries for the target concept are not parallel to the axes of the instance space. In contrast, 'polythetic' approaches like perceptrons and simple Bayesian classifiers, which consider multiple attributes during a decision, are unaffected by this situation.

On the other hand, the recursive partitioning of the instance space into regions lets decision-tree algorithms deal with disjunctive concepts, which cause problems for perceptrons and simple Bayesian classifiers. A natural response is to combine the recursive structure of decision trees with the discriminating power of polythetic classifiers. Brieman, Friedman, Olshen, and Stone [1] have done some work along these lines, using linear threshold units at each node in the tree.

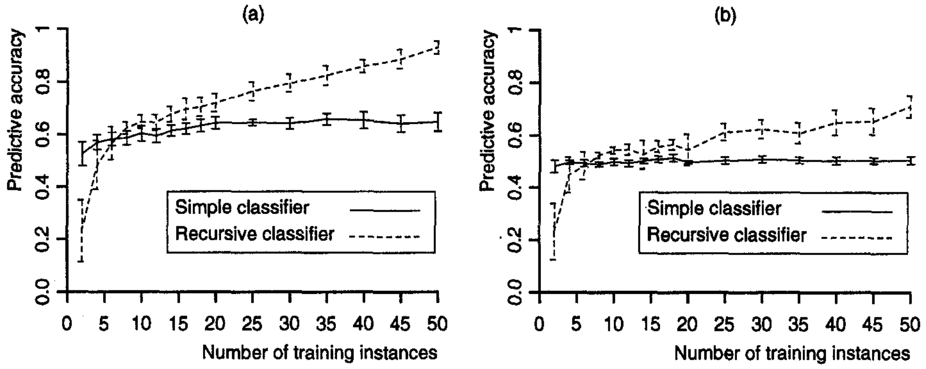


Figure 6. Learning curves for the simple Bayesian classifier and the recursive Bayesian classifier on artificial domains involving (a) a three-disjunct target concept and (b) a four-disjunct concept.

More recently, Utgoff and Brodley [13] have developed LMDT, an incremental method for inducing decision trees in which each nonterminal node specifies a linear machine. As in threshold units, such knowledge structures specify a set of weights for each class-attribute pair, but they operate competitively. The LMDT algorithm requires many passes through the training set, but otherwise it has many similarities to our approach. Sahami (personal communication, 1992) has developed a similar algorithm for inducing polythetic trees.

Fisher [6] has also dealt extensively with methods for organizing probabilistic concepts in a hierarchy. His COBWEB algorithm uses an identical representation and organization of knowledge, but there are also some important differences. COBWEB is unsupervised, attempting to maximize predictive accuracy across all attributes even if class names are present. Also, Fisher's incremental algorithm constructs complex trees that can be strongly affected by training order, which can produce knowledge structures that are difficult to understand.

In contrast, the RBC method is not subject to order effects and recurses only enough to discriminate the class names; thus, we predict that its trees will be more comprehensible than COBWEB's, though we have not tested this hypothesis. Moreover, Kononenko [7] has shown that a simple Bayesian classifier can explain its decisions as the sum of information gain over all attributes, and that physicians prefer such explanations to monothetic decision trees. Given that, in our experience, the RBC algorithm tends to build shallow trees, we expect that at least some domain experts will prefer them to normal monothetic decision trees as well.

Some researchers have explored other approaches to adapting Bayesian classifiers to disjunctive domains that violate the independence assumption. For example, Michie and Al Attar [10] describe a 'sequential Bayesian classifier' that inspects one attribute at a time during performance, selecting the most informative one at each step and halting when the probability of a class exceeds a maximum threshold or falls below a minimum. However, this method's behavior is equivalent to constructing a monothetic decision tree using a probabilistic evaluation function, thus losing the ability to consider evidence from multiple attributes simultaneously.

Kononenko [8] describes a quite different approach that explicitly tests for dependencies among attributes, creating new features based on the conjunctions of correlated values. This 'semi-naive Bayesian classifier' then uses the training data to compute conditional probabilities for these higher-order features, using them to classify test cases rather than the original ones. However, in an experimental comparasion of his algorithm and a simple Bayesian classifier, Kononenko found no differences on two medical domains and only very slight improvement on two others. These findings are consistent with our results on the robustness of the naive method in natural domains.

6. Directions for future research

Despite the promise of recursive Bayesian classifiers, much work remains before we can claim that they constitute a robust approach to induction. We have yet to identify any natural domains on which the method outperforms a simple Bayesian classifier, despite our intuitions about the former's superiority. Moreover, we must still examine the effect of other domain characteristics on the method's behavior. For instance, we should study the influence of both the number of relevant and irrelevant attributes, as well as the effect of class and attribute noise. We should also study the behavior of an incremental version developed by Schlimmer and Hermens (personal communication, 1992).

Some extensions will also be necessary. Like any hierarchical induction algorithm, RBC can overfit the data by constructing an overly detailed tree, and we should install a pruning scheme to counter this tendency. One simple technique would recurse only if the classifier at a given level exceeds a certain error rate. The current version also has difficulty with parity concepts, in which all attributes appear independent of the class name. In such cases, a revised system might resort to an arbitrary monothetic partition to overcome the bottleneck. Finally, future versions should use Cestnik's [2] m estimate, which he has shown improves accuracy over the relative frequencies currently used.

Despite these limitations, we believe that the RBC algorithm constitutes a promising extension to simple Bayesian classifiers which should be less sensitive to domains that violate the assumption of independence. We anticipate that further comparative studies, combined with theoretical analyses, will reveal the conditions under which this method outperforms both the simpler probabilistic algorithm from which it evolved and other techniques that rely on the induction of traditional decision trees.

Acknowledgements

Thanks to Stephanie Sage, Jeff Schlimmer, Carla Brodley, Kevin Thompson, and Wayne Iba for discussions that helped clarify our ideas, and to Nils Nilsson for his support and encouragement. The author carried out this work while at Stanford University and the Institute for the Study of Learning and Expertise.

References

1. Brieman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees.* Belmont: Wadsworth.
2. Cestnik, G. (1990). Estimating probabilities: A crucial task for machine learning. *Proceedings of the Ninth European Conference on Artificial Intelligence* (pp. 147–149). Stockholm, Sweden.
3. Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning, 3*, 261–284.
4. Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis.* New York: John Wiley & Sons.
5. Feigenbaum, E. A. (1963). The simulation of verbal learning behavior. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought.* New York: McGraw-Hill.
6. Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2*, 139–172.
7. Kononenko, I. (1990). Comparison of inductive and naive Bayesian learning approaches to automatic knowledge acquisition. In B. Wielinga et al. (Eds.), *Current trends in knowledge acquisition.* Amsterdam: IOS Press.
8. Kononenko, I. (1991). Semi-naive Bayesian classifier. *Proceedings of the Sixth European Working Session on Learning* (pp. 206–219). Porto: Pittman.
9. Langley, P., Iba, W., & Thompson, K. (1992). An analysis of Bayesian classifiers. *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 223–228). San Jose, CA: AAAI Press.
10. Miche, D., & Al Attar, A. (1991). Use of sequential Bayes with class probability trees. In J. E. Hayes-Michie, D. Michie, & E. Tyugu (Eds.), *Machine intelligence 12.* Oxford: Oxford University Press.
11. Quinlan, J. R. (1993). *C4.5: Programs for induction.* San Mateo, CA: Morgan Kaufmann.
12. Towell, G. G. (1991). *Symbolic knowledge and neural networks: Insertion, refinement, and extraction.* Dissertation, Department of Computer Science, University of Wisconsin, Madison.
13. Utgoff, P. E., & Brodley, C. E. (1990). An incremental method for finding multivariate splits for decision trees. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 58–65). Austin, TX: Morgan Kaufmann.