BOOSTING AND NAIVE BAYESIAN LEARNING

Charles Elkan

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, California 92093-0114

elkan@cs.ucsd.edu

Technical Report No. CS97-557, September 1997.

First version May 1997.

Abstract: Although so-called "naive" Bayesian classification makes the unrealistic assumption that the values of the attributes of an example are independent given the class of the example, this learning method is remarkably successful in practice, and no uniformly better learning method is known. Boosting is a general method of combining multiple classifiers due to Yoav Freund and Rob Schapire. This paper shows that boosting applied to naive Bayesian classifiers yields combination classifiers that are representationally equivalent to standard feedforward multilayer perceptrons. (An ancillary result is that naive Bayesian classification is a nonparametric, nonlinear generalization of logistic regression.) As a training algorithm, boosted naive Bayesian learning is quite different from backpropagation, and has definite advantages. Boosting requires only linear time and constant space, and hidden nodes are learned incrementally, starting with the most important. On the real-world datasets on which the method has been tried so far, generalization performance is as good as or better than the best published result using any other learning method. Unlike all other standard learning algorithms, naive Bayesian learning, with and without boosting, can be done in logarithmic time with a linear number of parallel computing units. Accordingly, these learning methods are highly plausible computationally as models of animal learning. Other arguments suggest that they are plausible behaviorally also.

1 Introduction

So-called "naive" Bayesian classification is the optimal method of supervised learning if the values of the attributes of an example are independent given the class of the example. Although this assumption is almost always violated in practice, recent work has shown that naive Bayesian learning is remarkably effective in practice and difficult to improve upon systematically [Domingos and Pazzani, 1996]. This paper shows that it is possible to improve reliably the generalization ability of naive Bayesian classifiers using a technique called boosting due to Freund and Schapire [1995]. The paper also argues that with and without boosting, naive Bayesian classification is computationally and phenomenologically plausible as a model of animal and human learning. Specifically, the paper shows that when the AdaBoost algorithm of Freund and Schapire [1995] is used to combine several naive Bayesian classifiers, the resulting combination is mathematically equivalent to a feed-forward neural network with sparse encoding of inputs, a single hidden layer of nodes, and sigmoid activation functions.

As a learning algorithm, boosting in combination with naive Bayesian learning has important advantages. First, the generalization ability of boosted naive Bayesian classifiers is excellent. On the real-world example datasets discussed below (and on many other datasets not discussed here) such a classifier gives better test set accuracy than any known method, including backpropagation and C4.5 decision trees. Second, these classifiers can be learned very efficiently. Given e training examples over f attributes, the time required to learn a boosted naive Bayesian classifier is $O(ef)$, i.e. linear. No learning algorithm that examines all its training data can be faster. Moreover, the constant factor is very small. On a modern workstation a boosted naive Bayesian classifier can be learned from 40000 examples of dimension 25 in under one minute, with most of the time devoted to processing strings in the dataset.

2 A review of naive Bayesian learning

Let A_1 through A_k be attributes with discrete values used to predict a discrete class C. Given an example with observed attribute values a_1 through a_k, the optimal prediction is class value c such that $Pr(C = c \mid A_1 = a_1 \ \wedge \ \ldots \ \wedge \ A_k = a_k)$ is maximal. By Bayes' rule this probability equals

$$\frac{Pr(A_1 = a_1 \ \wedge \ \ldots \ \wedge \ A_k = a_k \mid C = c)}{Pr(A_1 = a_1 \ \wedge \ \ldots \ \wedge \ A_k = a_k)} Pr(C = c).$$

The background probability or base rate $Pr(C = c)$ can be estimated from training data easily. The example probability $Pr(A_1 = a_1 \ \wedge \ \ldots \ \wedge \ A_k = a_k)$ is irrelevant for decision-making since it is the same for each class value c. Learning is therefore reduced to the problem of estimating $Pr(A_1 = a_1 \ \wedge \ \ldots \ \wedge \ A_k = a_k \mid C = c)$ from training examples. Using Bayes' rule again, this class-conditional probability can be written as

$$Pr(A_1 = a_1 \mid A_2 = a_2 \ \wedge \ \ldots \ \wedge \ A_k = a_k, C = c) \cdot Pr(A_2 = a_2 \ \wedge \ \ldots \ \wedge \ A_k = a_k \mid C = c).$$

Recursively, the second factor above can be written as

$$Pr(A_2 = a_2 \mid A_3 = a_3 \ \wedge \ \ldots \ \wedge \ A_k = a_k, C = c) \cdot Pr(A_3 = a_3 \ \wedge \ \ldots \ \wedge \ A_k = a_k \mid C = c)$$

and so on. Now suppose we assume for each A_i that its outcome is independent of the outcome of all other A_j, given C. Formally, we assume that

$$Pr(A_1 = a_1 \mid A_2 = a_2 \ \wedge \ \ldots \ \wedge \ A_k = a_k, C = c) = Pr(A_1 = a_1 \mid C = c)$$

and so on for A_2 through A_k. Then $Pr(A_1 = a_1 \ \wedge \ \ldots \ \wedge \ A_k = a_k \mid C = c)$ equals

$$Pr(A_1 = a_1 \mid C = c) Pr(A_2 = a_2 \mid C = c) \cdots Pr(A_k = a_k \mid C = c).$$

Now each factor in the product above can be estimated from training data:

$$\hat{Pr}(A_j = a_j \mid C = c) = \frac{count(A_j = a_j \ \wedge \ C = c)}{count(C = c)}. \tag{1}$$

It can be shown that Equation (1) gives "maximum likelihood" probability estimates, i.e. probability parameter values that maximize the probability of the training examples.

The induction algorithm explained above is called naive Bayesian learning; the earliest reference known to this author is Chapter 12 of the celebrated *Perceptrons* book by Minsky and Papert [1969].

3 Naive Bayesian classifiers, perceptrons, and logistic regression

Suppose that there are just two possible classes, called 0 and 1, and let a_1 through a_k be the observed attribute values for a test example. Let $b_0 = Pr(C = 0)$, let $b_1 = Pr(C = 1) = 1 - b_0$, let $p_{j0} = Pr(A_j = a_j \mid C = 0)$, and let $p_{j1} = Pr(A_j = a_j \mid C = 1)$. Then

$$p = Pr(C = 1 \mid A_1 = a_1 \ \wedge \ \ldots \ \wedge \ A_k = a_k) = \Big(\prod_{j=1}^{k} p_{j1} \Big) b_1 / z$$

and

$$q = Pr(C = 0 \mid A_1 = a_1 \ \wedge \ \ldots \ \wedge \ A_k = a_k) = \Big(\prod_{j=1}^{k} p_{j0} \Big) b_0 / z$$

where z is a normalizing constant. Taking logarithms gives

$$\log p - \log q = (\sum_{j=1}^{k} \log p_{j1} - \log p_{j0}) + \log b_1 - \log b_0.$$

Letting $w_j = \log p_{j1} - \log p_{j0}$ and $b = \log b_1 - \log b_0$ gives

$$\log \frac{1-p}{p} = -\sum_{j=1}^{k} w_j - b.$$

Exponentiating both sides and rearranging gives

$$p = \frac{1}{1 + e^{-\left(\sum_{j=1}^{k} w_j\right) - b}}.$$

In general, suppose the attribute A_j has $v(j)$ possible values. Let

$$w_{jj'} = \log Pr(A_j = a_{jj'} \mid C = 1) - \log Pr(A_j = a_{jj'} \mid C = 0)$$

where $1 \leq j' \leq v(j)$. Then

$$Pr(C(x) = 1) = \frac{1}{1 + e^{-\left(\sum_{j=1}^{k} \sum_{j'=1}^{v(j)} I(A_j(x) = a_{jj'}) w_{jj'}\right) - b}}. \tag{2}$$

Here I is an indicator function: $I(\phi) = 1$ if ϕ is true and $I(\phi) = 0$ if ϕ is false.

Equation (2) describes exactly the functioning of a perceptron with a sigmoid activation function and sparse encoding of attributes, i.e. with a separate input for each of the $v(j)$ possible values of each attribute A_j. Naive Bayesian classifiers are therefore equivalent in representational ability to perceptrons with localist input representations. Of course maximum likelihood learning as given by Equation (1) in general gives parameter values different from those obtained by other training methods.

It is also possible to show that naive Bayesian classification generalizes logistic regression, the most widely used statistical method for probabilistic classification from numerical attribute values. Consider again Equation (1). If A_j has discrete values then $count(A_j = a_j \ \wedge \ C = c)$ can be computed directly from training examples. If the values of A_j are numerical, the standard practice is to quantize or discretize the attribute. The simplest quantization method is to divide the range of the attribute into a fixed number M of equal-width bins. In the experiments described below $M = 10$ is chosen arbitrarily. Previous experimental work has shown that the benefits of more complex quantization methods are small at best [Dougherty *et al.*, 1995]. Using bins of equal width tends to work well because they allow good non-parametric approximation of skewed, multimodal, and/or heavy-tailed probability distributions.

Let each A_j be a numerical (integer-valued or real-valued) attribute. Logistic regression assumes the model

$$\log \frac{Pr(C = 1 \mid A_1 = a_1, \ldots A_k = a_k)}{Pr(C = 0 \mid A_1 = a_1, \ldots A_k = a_k)} = \sum_{j=1}^{k} b_j a_j + b_0. \tag{3}$$

This equation describes the functioning of a perceptron with a sigmoid activation function and a single input node for each attribute, i.e. with attribute values encoded as their magnitude. Discretization of A_j corresponds to replacing the linear term $b_j a_j$ by a piecewise constant function of a_j. If the range of A_j is divided into M intervals where the ith interval is $[c_{j(i-1)}, c_{ji}]$ then this function is

$$f_j(a_j) = \sum_{i=1}^{M} b_{ji} I(c_{j(i-1)} \leq a_j < c_{ji}) \tag{4}$$

where each b_{ji} is a constant. Combining Equations (3) and (4) yields a version of Equation (2). Hence naive Bayesian classification is a nonparametric, nonlinear extension of logistic regression; the standard logistic regression equation can be approximated by setting $b_{ji} = (c_{j(i-1)} + c_{ji})/(2b_j)$.

4 Boosting

The general idea of boosting is to learn a series of classifiers, where each classifier in the series pays more attention to the examples misclassified by its predecessor. Specifically, after learning the classifier H_k, boosting increases the weight of training examples misclassified by H_k, and learns the next classifier H_{k+1} from the reweighted examples. This process is repeated for T rounds. The final boosted classifier outputs a weighted sum of the outputs of the individual H_k, with each H_k weighted according to its accuracy on its training set.

Formally, the AdaBoost algorithm of Freund and Schapire [1995] is as follows. Let $w_i^{(1)} = 1/N$ for example $i = 1, \ldots, N$. For round $t = 1$ through T do:

- Given weights $w_i^{(t)}$ obtain a hypothesis $H^{(t)} : X \to [0, 1]$.
- Let the error of $H^{(t)}$ be $\epsilon^{(t)} = \sum_{i=1}^{N} w_i^{(t)} |y_i - h_i^{(t)}(x_i)|$.
- Let $\beta^{(t)} = \epsilon^{(t)}/(1 - \epsilon^{(t)})$ and let $w_i^{(t+1)} = w_i^{(t)}(\beta^{(t)})^{1-|y_i - h_i^{(t)}(x_i)|}$.
- Normalize $w_i^{(t+1)}$ to sum to 1.0.

Suppose that each individual classifier is actually useful, i.e. $\epsilon^{(t)} < 0.5$. Then $\beta^{(t)} < 1$, and as $|y_i - h_i^{(t)}(x_i)|$ increases, $w_i^{(t+1)}$ increases, so the algorithm satisfies the intuitive description given above. Experiments with variants of the AdaBoost algorithm show that none of its details are critical to its success–the basic idea of upweighting misclassified examples is all that is important in practice.

One proposal of Freund and Schapire [1995] is to let the final combination hypothesis be H given by

$$H(x) = \frac{1}{1 + \prod_{t=1}^{(t)} (\beta^{(t)})^{2r(x)-1}}$$

where the linear combination of individual classifiers is

$$r(x) = \frac{\sum_{t=1}^{(t)} (\log 1/\beta^{(t)}) H^{(t)}(x)}{\sum_{t=1}^{(t)} \log 1/\beta^{(t)}}.$$

We now show that a boosted naive Bayesian classifier is representationally equivalent to a multilayer perceptron with a single hidden layer. Let $\alpha = \prod_{t=1}^{T} \beta^{(t)}$ and let the vote of classifier t be $v^{(t)} = \log \beta^{(t)} / \log \alpha$. Then

$$\begin{aligned} H(x) = \frac{1}{1 + \alpha^{2\left(\sum_{t=1}^{T} v^{(t)} H^{(t)}(x)\right)-1}} &= \frac{1}{1 + e^{\left(\sum 2 \log \alpha v^{(t)} H^{(t)}(x)\right) - \log \alpha}} \\ &= \frac{1}{1 + e^{\sum 2 \log \beta^{(t)} H^{(t)}(x) - \sum \log \beta^{(t)}}}. \end{aligned}$$

In words, the output of the combined classifier is found by applying a sigmoid function to a weighted sum of the outputs of the individual classifiers. Since individual naive Bayesian classifiers are equivalent to perceptrons, the combined classifier is equivalent to a perceptron network with a single hidden layer.

The derivation above shows that boosted naive Bayesian learning can be viewed as an alternative training algorithm for feedforward neural networks. (The argument can be extended to cover feedforward networks with multiple hidden layers.) As such, it is most similar to algorithms such as the "tiling" method of Mezard and Nadal [1989] that incrementally create new hidden nodes to correct errors made by previously created nodes.

5 Computational complexity

Suppose that examples are over f attributes, each with v values. Then a naive Bayesian classifier as in Equation (2) has $fv+1$ parameters. These parameters are learned by accumulating $2fv+2$ counts. Each attribute value of each training example leads to incrementing exactly one count. Hence with e examples training time is $O(ef)$ independent of v. This time complexity is essentially optimal: any learning algorithm that examines every attribute value of every training example must have the same or worse complexity. For comparison, learning a decision tree without pruning requires $O(ef^2)$ time. (The algorithm that underlies this result is non-trivial.) The time required to update weights under boosting is also $O(ef)$, so T rounds of boosting use $O(Tef)$ time in total, which is also $O(ef)$ if T is constant.

When accumulating the counts on which a naive Bayesian classifier is based, training examples may be processed sequentially directly from disk or tape storage. The amount of random-access memory required is $O(fv)$ independent of the number of training examples. With boosting, the random-access memory needed is $O(Tfv)$. Therefore, boosted naive Bayesian classifiers are well-suited for knowledge discovery in very large databases: these databases can remain on disk or even on tape, and need not fit into main memory.

It is straightforward to show that learning a boosted naive Bayesian classifier is in the parallel complexity class NC. This class consists of problems that can be solved in polylogarithmic time with a polynomial number of processors. With communication patterned after a binary tree of height $\log e$, e processors can compute $count(A_j = a \wedge C = 0)$ and $count(A_j = a \wedge C = 1)$ for a given j and a in $O(\log e)$ time. Therefore naive Bayesian learning requires $O(\log e)$ time using $O(efv)$ processors.

In practice, with f and v constant and many examples allocated to each processor, a parallel implementation could use far fewer than e processors and still run in $O(\log e)$ time. With a constant number of rounds of boosting, boosted naive Bayesian learning is also in NC. A literature search reveals that no other practical learning problem is known to have an NC algorithm. Training simple linear threshold neural nets optimally is NP-complete [Blum and Rivest, 1992].

6 Experimental results

It is important to stress that the experiments described here are exploratory in nature. They use datasets selected as likely to yield particularly informative results.

The "German credit" dataset. This dataset has 700 positive and 300 negative examples. Seventeen features are discrete, while three are continuous. The best published prediction error rate is 25.3%, for a naive Bayesian classifier with five-fold cross-validation [Friedman and Goldszmidt, 1996]. Although boosting improves the performance of C4.5 on most datasets, on this dataset boosting increases the prediction error rate of C4.5 from 28.44% to 29.14% [Quinlan, 1996].

The table below shows the results of boosted naive Bayesian learning on this dataset. Combining three naive Bayesian classifiers through boosting gives an error rate of 24.0%, a useful improvement.

rounds	*error (%)*
1	25.1
2	24.3
3	24.0
4	24.4
5	24.7
6	24.9
7	25.2
8	25.1
9	25.3
10	25.4
$\vdots$	
100	25.7

In the table above, the reduction in test set error due to boosting can be seen to come after just a few rounds. Experiments done so far show a similar phenomenon on all datasets, although usually more than three rounds are beneficial. Boosted naive Bayesian learning is therefore a practical "anytime" learning algorithm. Depending on how much time and memory is available, learning a combination classifier may be stopped after any number of components have been learned.

As the number of rounds of boosting is increased beyond a dataset-dependent threshold, the error of the combined classifier increases slowly. This phenomenon is not surprising, but it has not been observed when boosting is used with other base learning methods [Schapire *et al.*, 1997]. Future work will aim to explain this phenomenon. For practical applications, cross-validation is a good way of deciding how many rounds of boosting to use.

The monk's problems. In 1991 Sebastian Thrun organized a comparison of dozens of learning algorithms on three synthetic datasets generated from propositional formulas over six attributes. The formulas can be stated as (i) $A_1 = A_2$ or $A_5 = a_{51}$, (ii) exactly two of A_1 through A_6 have their first value, and (iii) ($A_5 = a_{53}$ and $A_4 = a_{41}$) or ($A_5 \neq a_{54}$ and $A_2 \neq a_{23}$). For the third dataset 5% classification noise was added.

The first two formulas above involve comparing attributes with each other, and hence have an "xor" or first-order (relational as opposed to propositional) flavor. Only backpropagation and a version of Michalski's AQ with the builtin ability to use appropriate new features succeeded on these problems. On the third problem, the only successful methods were other versions of AQ, one of several decision tree algorithms, one of several inductive logic programming algorithms, and backpropagation with weight decay.

A boosted naive Bayesian classifier succeeds on the third monk's problem, but not on the first two. In general, boosted naive Bayesian learning appears to perform very well on realistic tasks where the true concept resembles a disjunction of conjunctions, with noise. Failure on the first two monk's problems can be explained qualitatively: naive Bayesian learning cannot achieve over 50% accuracy on "xor", and boosting requires that each intermediate hypothesis, including the first, have over 50% accuracy. With noise added to training examples, a naive Bayesian classifier should achieve over 50% accuracy on "xor". Future experiments will investigate whether boosting can amplify this advantage.

Diabetes in Pima Indians. This dataset is used by Ripley [1996] throughout his book. It is typical of difficult medical diagnosis problems where a limited number of training examples are available, and some attribute values are missing in many examples. Under naive Bayesian learning, even if some attribute values are unknown for a training example, its other attribute values can still contribute to counts. Our implementation of boosted naive Bayesian learning takes advantage of this fact. (An alternative is to view "unknown" as just another possible discrete value for each attribute.)

Examples in the Pima Indians diabetes dataset consist of eight numerical attributes:

number of pregnancies
glucose concentration in a tolerance test
diastolic blood pressure (mm Hg)
skin triceps skin fold thickness (mm)
serum insulin (micro U/ml)
body mass index (kg/m^2)
diabetes pedigree function
age in years

Using 200 complete training examples, the best learning method reported by Ripley [1996] is standard linear discrimination, which realizes 20.3% error on a separate test set. Backpropagation using one hidden unit achieves 22.6%; more hidden units give worse performance. Using an additional 100 training examples with missing data, C4.5 achieves 22.3% error rate. As shown in the table below, using the 200 complete training examples and the 100 examples with missing attribute values, boosting with ten rounds achieves an error rate of 18.7%. This result is a substantial improvement over any method tried by Ripley, and shows the practical importance of being able to use training examples with missing attribute values.

round	*using only 200 complete examples error (%)*	*including 100 extra incomplete examples error (%)*
1	24.7	20.2
2	23.8	20.5
3	22.9	19.9
4	22.6	19.6
5	22.9	19.6
6	22.9	19.3
7	22.9	19.0
8	22.9	19.0
9	22.6	19.0
10	22.0	18.7

7 Neurocomputational and psychological plausibility

There is strong experimental evidence that humans are highly sensitive to class-conditional probabilities; see for example [Bechara *et al.*, 1997]. Learning a single naive Bayesian classifier can be done incrementally, so it is computationally straightforward for the brain to do this in the course of everyday experience. Learning a boosted naive Bayesian classifier can be done by rehearsing past experiences. Boosting involves paying most attention to training examples that have been classified incorrectly previously; it is plausible that these are precisely the examples most likely to be accessible to memory.

At the behavioral level, naive Bayesian learning, with and without boosting, is a plausible human and animal learning method.

- The method can tolerate many irrelevant attributes: if some or all values of some attributes are uncorrelated with the class of an example, the log-odds weights attached to these attribute values will tend to zero as the number of training examples increases.

- With multiplicative updating of counts instead of additive updating, target concepts that change over time can be tracked efficiently and effectively.

- Either by changing the initial weights of training examples, or by changing the probability threshold used to make decisions, different costs for different misclassification errors (which may vary example by example) can be taken into account in an optimal manner.

The extended version of this paper will contain a detailed exposition of how naive Bayesian learning, with and without boosting, is a better model of human learning behavior than backpropagation. This discussion will show that naive Bayesian learning reproduces each of the phenomena of human category learning discussed by Kruschke [1993]. In particular, naive Bayesian learning performs better at so-called “filtration” tasks than at “condensation” tasks [Garner and Sutliff, 1974]. While humans also find filtration tasks easier, backpropagation works equally well for tasks of both types. The corresponding mathematical phenomenon is that backpropagation with a hidden layer learns a decision plane equally well regardless of its rotation, while naive Bayesian learning is less accurate when the optimal decision plane is not axis-parallel.

Lehky and Sejnowski [1988] expressed a standard opinion when they wrote “No biological significance is claimed for the algorithm (back propagation) by which the network developed”. On the contrary, naive Bayesian learning is neurocomputationally plausible. In a nutshell, this type of learning can be done either in logarithmic time with a linear number of neurons, or incrementally in constant time per example, with fixed, sparse, connections between neurons, with low precision connection strengths between neurons, without feedback connections, and without any sort of associative memory.

One major assumption of naive Bayesian learning is a sparse, localist encoding of attributes, i.e. of stimulus dimensions. Such an encoding, which is also called a population code, is quite plausible in many domains. For example, rats have place cells that fire when and only when the animal is in a specific spatial location [Bures *et al.*, 1997] and they have localist neural representations of other environmental features also. In humans, face recognition may be based on localist features such as Gabor transforms of small facial regions that are in fixed positions relative to each other [Kalocsai *et al.*, 1996].

It is important to note that the claim here is not that humans or animals implement the AdaBoost algorithm precisely. Rather, the claim is that it is computationally feasible for low-level brain processes to implement some generally similar algorithm. Since little is known about the operation of memory and learning at the neural level, one way to make progress in understanding these phenomena is to identify algorithms that meet basic constraints of computational feasibility. The computational complexity class NC is a useful first approximation of the computational capacity of low-level brain processes. With about 10^{10} neurons and about 10^5 synapses per neuron, the human brain may possess about 10^{15} information processing elements, but these elements are relatively slow. Therefore learning methods that require sublinear processing time, using a linear number of computing units, are more neurocomputationally plausible than intrinsically sequential methods–even if those methods have the same sequential time complexity. Formally speaking, it is cognitively plausible for low-level brain mechanisms to be solving problems in NC, while it is not plausible for them to be solving problems that lie outside NC.

References

[Bechara *et al.*, 1997] A. Bechara, H. Damasio H, D. Tranel, and A. R. Damasio. Deciding advantageously before knowing the advantageous strategy. *Science*, 275:1293–1295, February 1997.

[Blum and Rivest, 1992] A. L. Blum and R. L. Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5(1):117–127, 1992.

[Bures *et al.*, 1997] J. Bures, A. A. Fenton, Y. Kaminsky, and L. Zinyuk. Place cells and place navigation. *Proceedings of the National Academy of Sciences*, 94(1):343–350, January 1997.

[Domingos and Pazzani, 1996] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 105–112. Morgan Kaufmann Publishers, Inc., 1996.

[Dougherty *et al.*, 1995] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann Publishers, Inc., 1995.

[Freund and Schapire, 1995] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, 1995.

[Friedman and Goldszmidt, 1996] Nir Friedman and Moises Goldszmidt. Building classifers using Bayesian networks. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1277–1284, Portland, Oregon, August 1996. AAAI Press (distributed by MIT Press).

[Garner and Sutliff, 1974] W. R. Garner and Donna Sutliff. The effect of goodness on encoding time in visual pattern discrimination. *Perception and Psychophysics*, 16(3):426–430, 1974.

[Kalocsai *et al.*, 1996] P. Kalocsai, I. Biederman, and C. von der Malsburg. Predicting face recognition from a spatial filter similarity space. Personal communication, 1996.

[Kruschke, 1993] John K. Kruschke. Human category learning: Implications for backpropagation models. *Connection Science*, 5(1):3–36, 1993.

[Lehky and Sejnowski, 1988] S. R. Lehky and T. J. Sejnowski. Network model of shape-from-shading: neural function arises from both receptive and projective fields. *Nature*, 333(6172):452–454, 1988.

[Mezard and Nadal, 1989] M. Mezard and J. P. Nadal. Learning in feedforward layered networks: The tiling algorithm. *Journal of Physics A*, 22(12):2191–2203, 1989.

[Minsky and Papert, 1969] Marvin Minsky and Seymour Papert. *Perceptrons; an introduction to computational geometry*. MIT Press, 1969.

[Quinlan, 1996] John R. Quinlan. Boosting, bagging, and C4.5. In *Proceedings of the National Conference on Artificial Intelligence*, pages 725–730, Portland, Oregon, August 1996. AAAI Press (distributed by MIT Press).

[Ripley, 1996] Brian Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[Schapire *et al.*, 1997] R. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin; a new explanation for the effectiveness of voting methods. In *Proceedings of the Fourteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers, Inc., 1997. To appear.