ARTICLE Communicated by Richard Lippmann

Rule-Based Neural Networks for Classification and Probability Estimation

Rodney M. Goodman
Charles M. Higgins
John W. Miller
Department of Electrical Engineering,
California Institute of Technology, Pasadena, CA 91125 USA

Padhraic Smyth
Communications Systems Research, Jet Propulsion Laboratory,
California Institute of Technology, Pasadena, CA 91109 USA

In this paper we propose a network architecture that combines a rule-based approach with that of the neural network paradigm. Our primary motivation for this is to ensure that the knowledge embodied in the network is *explicitly* encoded in the form of understandable rules. This enables the network's decision to be understood, and provides an audit trail of how that decision was arrived at. We utilize an information theoretic approach to learning a model of the domain knowledge from examples. This model takes the form of a set of probabilistic conjunctive rules between discrete input evidence variables and output class variables. These rules are then mapped onto the weights and nodes of a feedforward neural network resulting in a *directly* specified architecture. The network acts as parallel Bayesian classifier, but more importantly, can also output posterior probability estimates of the class variables. Empirical tests on a number of data sets show that the rule-based classifier performs comparably with standard neural network classifiers, while possessing unique advantages in terms of knowledge representation and probability estimation.

1 Introduction

The rule-based knowledge representation paradigm is well established as a powerful model for higher level cognitive processes (Newell and Simon 1972; Chomsky 1957), whereas the connectionist paradigm seems very well suited to modeling lower level perceptual processes. In particular, rule-based expert systems have proven to be a successful software methodology for automating complex decision-making tasks. Primary advantages of this approach include the facility for *explicit knowledge representation* in the form of rules and objects, and the ability of a rule-based

system's reasoning to be understood by humans. However, current rule-based systems are *fundamentally* restricted in speed of execution, and hence in their applicability to real-time systems, because of the serial computations performed in present inference processing schemes. In addition, current rule-based systems are brittle in their ability to deal with the uncertainties inherent in real-world information and lack any ability to generalize to novel problems. Neural network paradigms, on the other hand, are typically quite adept at modeling problems that occur in pattern recognition, visual perception, and control applications. This ability is due (at least in part) to their inherent robustness in the presence of noise, the lack of which plagues the implementation of rule-based systems in practice. In addition, neural networks are inherently parallel, and special-purpose parallel neural network hardware implementations promise quantum leaps in processing speeds, suitable for real-time systems. However, neural networks, as presently implemented, are poor at explaining their reasoning in human understandable terms because they embed domain knowledge in the implicit form of weights and hidden nodes. The network is thus very much of a "black-box" solution, whose structure and reasoning are relatively inaccessible to higher level reasoning or control processes, such as the human user. In many areas of expertise such as medical, legal, or life-critical domains, it is an absolute requirement that an autonomous reasoning agent be able to explain its decisions to a higher level authority such as a judge. We are therefore led to ask whether it is possible to amalgamate the rule-based and connectionist approaches into a hybrid scheme, combining the better aspects of both, while eliminating the drawbacks peculiar to each in the process. A natural common ground on which to combine these approaches is that of probability. We show that by referencing both rule-based systems and neural networks to the common normative frame of probability, a novel and practical architecture emerges.

In this paper we propose a hybrid rule-based connectionist approach that overcomes many of the problems outlined above. Our ultimate goal is the automatic learning of rule-based expert systems that can perform inference in parallel when implemented on neural network architectures. For the purposes of this paper, however, we concentrate on the problem of classification and posterior probability estimation, implemented on rule-based feedforward neural nets. We show how *probabilistic rules* can be used as a natural method for describing the high-order correlation information in discrete (or categorical) data, and how the hidden units of a feedforward net can easily implement such rules. Furthermore, we show how information theory and minimum description length theory can be used to learn only the most important of these rules, thus *directly* specifying the network architecture in terms of hidden units and connectivity. Finally, we show that output probabilities may be estimated using a parallel Bayesian approach, which is a natural extension of a first-order Bayes classifier. The architecture proposed in this paper is

therefore novel for a number of reasons. First, it avoids iterative network training processes (such as backpropagation) by *directly* specifying network weights in terms of probability estimates derived from the example data. Second, the hidden nodes of the network are automatically learned from the data without having to specify the number of such nodes. This approach leads to the advantage that network parameters are directly interpretable in terms of rules with associated weights of evidence between the nodes. Third, given that it is usually necessary to assume some form of conditional independence among the input variables in order to render the probability estimation problem tractable, the proposed classification scheme is novel in that it uses *data-dependent* conditional independence assumptions only to the extent *justified* by the data.

Networks that learn from example form the basis of many current connectionist paradigms. The success of the backpropagation (Rumelhart *et al.* 1986) and related algorithms is that, given a specific architecture in terms of input, hidden, and output nodes, the connection weights between these nodes needed to model the high-order correlations in the example data can be easily learned. Learning the network *architecture* itself, and generating true output probability estimates is a considerably more difficult task for current neural network paradigms. It is interesting to note that Uttley (1959), conceived of a network in which *all* higher order input–output correlations were stored. This network stored a number of probabilities exponential in the number of input variables, but contained the information necessary for calculating the conditional probability of any set of output variables, given any other set of input variables. In principle, this provided a method of calculating output probabilities at the expense of exponentially many of what we would now call hidden units, many of which were redundant in the sense of not contributing to the output information. Networks whose architectures include high-order connections chosen *randomly* were of course among the very early neural network models (Rosenblatt 1962; Aleksander 1971). At the other extreme, in a previous paper we showed how simple first-order correlations could be used to successfully predict output probabilities (Goodman *et al.* 1989), provided the data were well specified by such low-order information. Between these extremes lie approaches that make subjective prior judgments about conditional independence to decide *which* higher order conjunctive probabilities to store, such as the Bayesian networks described by Pearl (1988), Lansner and Ekeberg (1989), and Kononenko (1989).

This paper develops in the following way. First, we outline our rule-based network architecture. Second, we describe our methodology for learning a set of probabilistic production rules from example data, using an information theoretic approach. Third, we show how these rules are then mapped onto the nodes and links of a feedforward neural network in such a manner that the network computes posterior class probabilities using a Bayesian formalism. We conclude with a comparative evaluation

of the approach using five data sets, including medical diagnosis and protein structure prediction.

2 A Rule-Based Classifier Architecture

We consider the problem of building a classifier that relates a set of K discrete feature variables (or *attributes*) comprising the set $Y = \{Y_1, \ldots, Y_K\}$ to a discrete class variable X. Each attribute variable takes *values* in the alphabet $\{y_l^1, \ldots, y_l^{m_l}\}$, $1 \leq l \leq K$, where m_l is the cardinality of the lth attribute alphabet. The class variable X takes discrete values from the set $\{x_1, \ldots, x_m\}$, where m is the cardinality of the class. We also assume that we are given an initial labeled training set of N examples where each example is of the form $\{Y_1 = y_1^q, \ldots, Y_K = y_K^z, X = x_i\}$. The supervised learning problem we set ourselves is to learn a classifier that when presented with future unseen attribute vectors (which may be either partial or complete) will estimate the posterior probability of each class. We may then wish to output either these probabilities, or the class variable with the highest probability as the decision made by the classifier. Note that we are particularly interested in real data sets in which the classification is often nondeterministic or noisy, that is, there exists class overlap and hence a fundamental ambiguity in the mapping from Y to X. In this case there is no perfect classifier for the problem and the performance of the classifier as measured by its error rate will be nonzero, and bounded below by the optimal Bayes error rate p_e^B.

The rule-based architecture we propose takes the form of a three-layer feedforward network as shown in Figure 1.

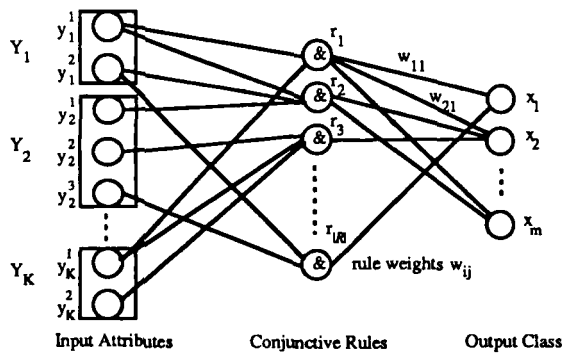

Figure 1: Architecture of the rule-based classifier.

The input nodes correspond to each possible attribute-value pair in the input attribute space. The hidden layer consists of a set of $|\mathcal{R}|$ conjunction detector nodes (or AND gates), one for each rule in the set of rules $\mathcal{R}$. These hidden nodes detect the conjunction of one or more input attribute-value pairs of the form $\{Y_1 = y_1^a, \ldots, Y_l = y_l^z\}$. When a conjunction is detected the rule *fires* and the node outputs a 1. When the node is not firing it outputs a 0. The output layer consists of one node for each output class. The action of a rule firing contributes an activation from the hidden rule node into one or more output class nodes. The contribution into the ith output node from rule r_j is given by the link weight w_{ij}, and represents the weight of evidence for the conclusion, given the occurrence of the left-hand side conjunction of attribute values. Each rule node together with its output link can therefore be considered to be implementing an l*th*-order conjunctive rule of the form

$$\text{IF } \{Y_1 = y_1^a, \ldots, Y_l = y_l^z\} \text{ THEN } \quad X = x_i \text{ with STRENGTH } w_{ij}$$

The rule has a conjunction of input attribute-value pairs on its left-hand side (LHS), and a particular class attribute-value pair on its right-hand side (RHS). The weights w_{ij} can be positive or negative depending on whether the rule supports the truth or falsity of the RHS conclusion. Each output node accumulates the inputs feeding into it from the rules that have fired and outputs a quantity that is a function of the particular activation function and threshold used in the node. Our design problem is then to implement a set of rules and associated weights, together with a suitable set of output activation functions and thresholds, such that the output of each class node is an estimate of the corresponding class probability.

3 Learning Rules Using Information Theory

We now consider how to learn the set of rules $\mathcal{R}$ from the given training data such that the classifier will operate in the desired manner. Clearly we do not want to implement *all* possible conjunctive rules, as the size of the hidden layer will be exponential in the size of the input attributes. Rather we require a *sufficiently good* set of rules that allows the network to both load the training data and to generalize to new data while having a performance that approaches the optimum Bayes risk. Alternatively, given a fixed resource constraint in terms of $|\mathcal{R}|$ allowed hidden units, we should implement the *best* $|\mathcal{R}|$ rules, according to some "goodness" criterion.

Let us rephrase the previously defined rule in terms of a *probabilistic production rule* of the form:

$$\text{If } s_j \text{ then } x_i \text{ with probability } p$$

where p is the conditional probability $p(x_i \mid s_j)$, and s_j represents the particular conjunction of attribute-value pairs found in the LHS of the rule.

We wish to have a measure of the utility or goodness of such a rule. In a Hebbian sense such a rule might be considered good if the occurrence of the LHS conjunction of variables is strongly correlated with the RHS. Alternatively, such a rule might be considered good if the transition probability p is near unity. For example, a rule with $p = 1$ indicates a deterministic rule in which the occurrence of s_j implies $X = x_i$ with certainty. However, we will take an information theoretic approach to this problem, and consider that the goodness of such a rule can be measured by the average bits of information that the occurrence of the LHS s_j gives about the RHS $X = x_i$. We have introduced such a measure, called the J-measure (Goodman and Smyth 1989), which can be defined as

$$J(X; s_j) = p(s_j)\left[p(x_i \mid s_j) \cdot \log\left(\frac{p(x_i \mid s_j)}{p(x_i)}\right) + (1 - p(x_i \mid s_j)) \cdot \log\left(\frac{(1 - p(x_i \mid s_j)}{(1 - p(x_i))}\right)\right]$$

This measure possesses a variety of desirable properties as a rule information measure, not the least of which is the fact that it is unique as a nonnegative measure of the information that s_j gives about X (Blachman 1968). As can be seen the J-measure is the product of two terms. The first is $p(s_j)$, the probability that the LHS will occur. This term can be viewed as a preference for generality or simplicity in our rules; that is, the left-hand side must occur relatively often for a rule to be deemed useful. The other term is the cross-entropy of X and X given s_j, and as such is well-founded measure of the goodness of fit between our a posteriori belief about X and our a priori belief (Shore and Johnson 1980). Hence, maximizing the product of the two terms, $J(X; s_j)$, is equivalent to simultaneously maximizing both the simplicity of the hypothesis s_j and the goodness of fit between s_j and a perfect predictor of X. The simplicity of s_j directly corresponds to the number of attribute-value conjunctions in the rule LHS, that is, the rule *order*. Low-order rules have less LHS conditions, and thus a higher $p(s_j)$. There is a natural trade-off involved here because, typically, one can easily find high-order rules (less probable s_js) that are accurate predictors, but one has a preference for more general low-order rules (more probable s_js). The J-measure thus provides a unique method of not only ranking the goodness of a set of rules, but also being able to tell if a more specialized rule (one with more LHS conditions) is better or worse than a more general rule. This basic trade-off between accuracy and generality (or goodness of fit and simplicity) is a fundamental principle underlying various general theories of inductive inference (Angluin and Smith 1984; Rissanen 1989).

We use the J-measure in a search algorithm (Smyth and Goodman 1992) to search the space of all possible rules relating the LHS attributes Y to the RHS class X and produce a ranked candidate set $\mathcal{S}$ of the $|\mathcal{S}|$ most informative rules that classify X. The search proceeds in a depth

first manner starting with a particular LHS conjunction and progressively specializes the rule until bounds indicate that a higher measure cannot be achieved by specializing further. The search is potentially exponential but in practice is highly constrained by small sample estimators and information theoretic bounds that heavily penalize searching higher order rules (empirical results demonstrating this effect are given in Smyth and Goodman 1992). In addition, higher order rules that have lower information content than a corresponding lower order (more general) rule can be omitted from the final rule list. From this large candidate set of rules $\mathcal{S}$ we next produce the final set of rules $\mathcal{R}$, which will be implemented in the classifier.

4 Rule Pruning Using a Minimum Description Length Model

We have already described how we find an initially large candidate set of rules $\mathcal{S}$ that models the data. It is well known, both empirically and from theory, that there is a trade-off between the complexity of the model and the quality of generalization performance. A model that is too simple will not have the representational power to capture the regularities of the environment, whereas a model that is too complicated may well overfit the training data and generalize poorly. When we speak here of generalization we are referring to the system's mean performance in terms of classification accuracy (or a similar function) evaluated over some infinitely large independent test data set. The notion of Occam's razor has been used to promote model parsimony: choose the simplest model that perfectly explains the data. Unfortunately this presupposes that there exists a model under consideration that can explain the data perfectly in this manner. In practical problems this is unlikely to be the case, since there is often an ambiguity in the mapping from attribute space to the class labels. In this stochastic setting a more general version of Occam's razor has been introduced (Rissanen 1984, 1987) under the title of minimum description length (MDL). The MDL principle is simple to state: choose the model that results in the least description length, where the description length is calculated by first sending a message describing the model [the complexity term $L(M)$], followed by a message encoding the data given the model [the goodness-of-fit term $L(D \mid M)$]. Thus we minimize:

$$L(M) + L(D \mid M)$$

MDL can be viewed as entirely equivalent to a Bayes maximum a posteriori (MAP) principle (where one chooses the model that maximizes the joint probability of the data and the model) by virtue of the fact that the description lengths are directly related to prior probabilities. We will refer primarily to the description length framework as it is somewhat more intuitive.

In the context of applying MDL to the problem at hand we seek a pruned rule set $\mathcal{R} \subseteq \mathcal{S}$, which possesses near-minimal description length among all possible subsets—finding the optimal solution is clearly intractable in the general case. For a more general discussion of search in MDL contexts see Smyth (1991). The algorithm we propose is a simple greedy procedure that, starting from an initially empty set of rules, continues to add the next-best rule to the current set, and terminates at a local minimum of the description length function. In more detail the algorithm is described as follows:

4.1 MDL Rule Pruning Algorithm.

1. Let $\mathcal{R} = \{\ \}$.
2. Find the rule $r \in \mathcal{S}$ such that when $\mathcal{R} \cup r$ is evaluated on the training data as a classifier the sum of the goodness of fit and the complexity of r is minimized.
3. Remove rule r from $\mathcal{S}$.
4. If the description length of $\mathcal{R} \cup r$ is greater than the description length of $\mathcal{R}$ then stop.
5. Else let $\mathcal{R} = \mathcal{R} \cup r$ and return to step 2.

At this point in the discussion we can treat the classifier itself as a "black box" that simply takes a rule set $\mathcal{R}$, a set of unlabeled test data, and produces probability estimates of the class labels. We will describe this "black box" in detail in the next section. Let us first look at the other part of the algorithm, which we have not defined in detail, namely the calculation of description length.

Suppose we have N training samples. For the ith sample, $1 \leq i \leq N$, let $x_{\text{true}}(i)$ be an index to the true class, i.e., the training label. Let $\hat{p}[x_{\text{true}}(i)]$ be the classifier's estimate of this class given the ith attribute vector. Hence, the length in bits to describe the data given the model (the classifier) is

$$L(D \mid M) = \sum_{i=1}^{N} \log \frac{1}{\hat{p}[x_{\text{true}}(i)]}$$

The complexity term, the length in bits to describe the classifier itself, may be arrived at by a number of arguments. In principle we need to send a message describing for each rule its left-hand side component, its right-hand side component, and an estimate of the transition probability of the rule. One of the key factors in proper application of MDL is the *precision* with which these probabilities are stated. It is a well known general result that very often the optimal precision for model parameters is proportional to $\sqrt{N}$, or about $(1/2) \log N$ bits per parameter. In practice this term dominates as N becomes large over the specification of the rule components. Since these specification terms also depend on the

particular coding scheme (or the prior bias in Bayesian terminology), we choose to ignore these terms in the optimization or search and propose that the complexity be simply proportional to the $(1/2)\log N$ precision terms. This penalty scheme has been widely used in practice by other authors (Rissanen and Wax 1988; Tenorio and Lee 1990, et al.). Hence, for rule set $\mathcal{R}$ the complexity is assessed as

$$L(M) = \frac{|\mathcal{R}|}{2}\log N$$

As we shall discuss later in the section on empirical results, this simple pruning algorithm is very effective at discovering parsimonious rule sets that account for the data. Heuristically, for multivalued class problems, we can understand the behavior of the algorithm as initially trying to account for each class by a single accurate rule for each, and then integrating rules that cover regions of the attribute space with high accuracy. In particular, as we shall discuss in the next section, by evaluating the performance of the classifier on each candidate rule, we can match the rule set to the nature of the classifier (conditional independence in this case).

If $|\mathcal{R}|$ is the number of rules in the final rule set then it is straightforward to show that the complexity of the pruning algorithm approximately scales as $N|\mathcal{S}||\mathcal{R}|^2$. Typically $|\mathcal{R}| << |\mathcal{S}|$, the number of rules in the initial rule set. It is difficult to bound $|\mathcal{R}|$ accurately (since it depends on the complexity of the particular classification problem), however, empirical results suggest that it often grows sublinearly with N, perhaps as slowly as $\log N$.

5 Derivation of the Classification Equations

In this section we describe how the network uses the learned rule set to estimate the class probabilities, given a particular set of evidential attribute-values. As before we have m classes $x_1 \ldots . x_m$ and a rule set $\mathcal{R}$. As discussed earlier each rule $r_j \in \mathcal{R}$ specifies a particular lth-order left-hand side attribute conjunction s_j, a class x_i, and the transition probability $p(x_i \mid s_j)$, where we recall that s_j denotes a particular conjunction of input attribute-value terms.

For a particular input vector $\{y_1 \ldots . y_K\}$, a certain subset of rules $\mathcal{F} \subseteq \mathcal{R}$ is said to "fire," i.e., $\mathcal{F}$ is the set of rules whose left-hand sides logically evaluate to true, or, in neural terms, the set of hidden nodes that are activated. The problem is simple: given only knowledge of $p(x_i \mid s_j)$, $1 \leq j \leq |\mathcal{F}|$, how can we estimate $p(x_i \mid s_1 \ldots . s_{|\mathcal{F}|})$? In principle there are strong arguments for using a maximum entropy solution, i.e., viewing the $p(x_i \mid s_j)$ and the particular input vector $\{y_1 \ldots . y_K\}$ as a set of constraints and maximizing the entropy of the joint distribution subject to these constraints (Cheeseman 1983; Miller and Goodman 1990).

However, the direct solution of this nonlinear optimization problem is unattractive from an implementation viewpoint, being both computationally intensive and unnatural to integrate into a system based on explicit knowledge representation.

A better approach in this context is to make a particular simplifying assumption, resulting in a maximum entropy solution that can be directly expressed in closed form (in terms of the component rules). This key assumption is that the left-hand side conjunctions are *conditionally independent* given the class, i.e., for any pair of rules r_j and $r_k \in \mathcal{R}$, which refer to the same class x_i, we have

$$p(s_j . s_k \mid x_i) = p(s_j \mid x_i)p(s_k \mid x_i) \tag{5.1}$$

As described in the previous section, the rule set $\mathcal{R}$ is formed from a large candidate set of rules in a manner such that rules that obey this conditional independence assumption are included and those which violate the assumption are not. Hence, we find a classifier that uses conditional independence only insofar as it can be justified by the training data—this is considerably more robust than making a priori assumptions about independence without any knowledge of the data. Assuming conditional independence of attributes given the class is well motivated, as discussed by Pearl (1988).

By Bayes' rule we have that

$$\begin{aligned} p(x_i \mid s_1 \ldots . s_{|\mathcal{F}|}) &= \frac{p(s_1 \ldots . s_{|\mathcal{F}|} \mid x_i)p(x_i)}{p(s_1 \ldots . s_{|\mathcal{F}|})} \\ &= \frac{\prod_{j=1}^{|\mathcal{F}|} p(s_j \mid x_i)}{p(s_1 \ldots . s_{|\mathcal{F}|})} p(x_i) \\ &\qquad \text{[by the conditional independence assumption in (5.1)]} \\ &= \frac{\prod_{j=1}^{|\mathcal{F}|} p(s_j)}{p(s_1 \ldots . s_{|\mathcal{F}|})} p(x_i) \prod_{j=1}^{|\mathcal{F}|} \frac{p(x_i \mid s_j)}{p(x_i)} \\ &\qquad \text{(by Bayes' rule)} \end{aligned}$$

Let us define the weights w_{ij} as

$$w_{ij} = \log \frac{p(x_i \mid s_j)}{p(x_i)}$$

a bias term for each class as

$$t_i = \log p(x_i)$$

and an (as yet) undetermined constant

$$C = \log \frac{\prod_{j=1}^{|\mathcal{F}|} p(s_j)}{p(s_1 \ldots . s_{|\mathcal{F}|})}$$

Hence, we get that

$$\log p(x_i \mid s_1, \ldots, s_{|\mathcal{F}|}) = C + t_i + \sum_{j=1}^{|\mathcal{F}|} w_{ij} \tag{5.2}$$

Equation 5.2 allows us to estimate the log posterior probability for each output class. From this, the actual probability can be computed, or a classification decision can be made by simply choosing the largest estimate as the output class. Equation 5.2 admits a direct and intuitive interpretation of the operation of the classifier. First, we can ignore the unknown constant C because it can be eliminated by the constraint that the sum of the posterior estimates must equal 1, as shown in the next section. Thus, in the absence of any rules firing ($|\mathcal{F}| = 0$) the estimate for each class is given by the bias value t_i, namely the log of the prior probability of the class x_i. Given a set of rules that fires, each rule contributes a "weight of evidence" into its corresponding output class. This weight of evidence w_{ij} has a direct interpretation as the evidential support for the class provided by the rule—a positive weight implies that the class is true, while a negative weight implies it is false. The w_{ij} thus provide the user with a direct explanation of how the classification decision was arrived at. Each class estimate is then computed by accumulating the "weights of evidence" incident on each class from the rules that fire, which can be done in a parallel manner.

We can relate our weights of evidence to those proposed by Good (1950) and Minsky and Selfridge (1961), namely, our w_{ij} are what Good termed "relative weights of evidence" for the case of multivalued classes. This weights of evidence classifier (which is relatively well known and has appeared in various guises in recent decades) is an intuitively elegant implementation of a linear reasoning scheme, as pros and cons for a particular hypothesis (or class) are tabulated in an additive manner.

6 Neural Architecture

The classification procedure given by equation 5.2 can now be mapped onto the three-layer feedforward network architecture shown in Figure 2. The input layer contains one node for each input attribute value. The hidden layer contains one node for each rule. These nodes are effectively AND gates that output a 1 if the left-hand side of the rule is satisfied, and a 0 otherwise. The third layer contains a node for each value of the class attribute. Each second-layer node representing rule i is connected to a third-layer node j via the multiplicative weight of evidence w_{ij}. Also feeding into and summed by each third-layer node is the bias value t_i. The sum of activations into this node given by

$$\sigma_i = t_i + \sum_{j=1}^{|\mathcal{F}|} w_{ij}$$

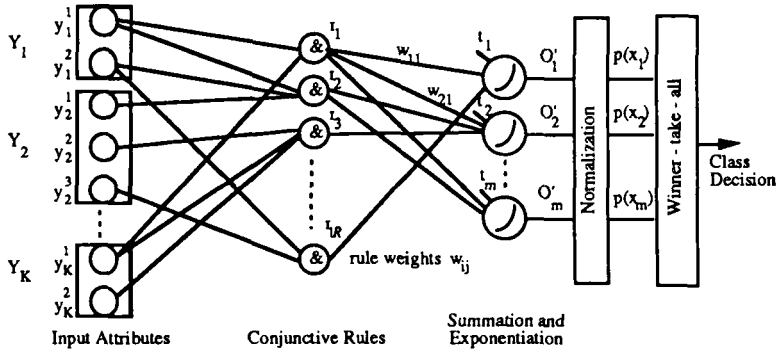

Figure 2: Neural network architecture.

is then *exponentiated* to produce the node output:

$$O'_i = e^{\sigma_i} = e^{-C} \cdot p(x_i \mid s_1, \ldots, s_{|\mathcal{F}|})$$

The output of the exponentiators is then fed into a normalization layer that constrains each output to satisfy:

$$O_i = \frac{O'_i}{\sum_{k=1}^{m} O'_k}$$

This effectively removes the constant C from each input $O'_i = e^{-C} \cdot p(x_i \mid s_1, \ldots, s_{|\mathcal{F}|})$ producing the output probability estimate $O_i = p(x_i \mid s_1, \ldots, s_{|\mathcal{F}|})$ as desired. If required, a winner-take-all stage can be added to decide on the the most likely class.

It is interesting to note that for the special case of a binary class variable, $m = 2$, the resulting circuit may be considerably simplified to that shown in Figure 3. In this case the output is a single node that accumulates the weights of evidence *for* one class value and *against* the other. The rule weights incident on the output are then

$$w_{1j} = \log \frac{p(x_1 \mid s_j)}{p(x_2 \mid s_j)}$$

and the bias is the log-odds of X,

$$t_1 = \log \frac{p(x_1)}{p(x_2)}$$

The exponentiation and normalization steps are combined by noting that for this binary case

$$O_1 = \frac{O'_1}{O'_1 + O'_2} = \frac{1}{1 + e^{-(\sigma_1 - \sigma_2)}}$$

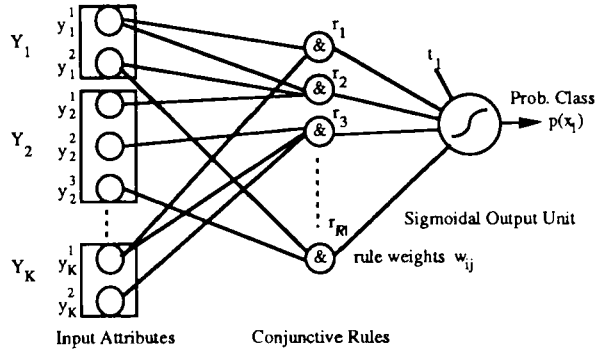

Figure 3: Binary class architecture.

resulting in the output node having the well-known sigmoid activation function. If a classification decision is required, the sigmoid is simply replaced with a hard–limiting node which switches at zero input activation to output 1 for class x_1 = true and 0 for class x_1 = false.

Without getting into details, it is important to note that the above neural architecture is well suited to VLSI implementation. In particular, the weights for the first layer are binary, and the hidden units are simply AND gates. Analog weight storage is required only for the second layer weights, and there are typically many fewer of these than there are first layer weights. Also, exponentiation can easily be performed in VLSI by using MOS transistors in their subthreshold region (Chiueh and Goodman 1990), and the final normalization stages can be performed by variants of the winner-take-all circuit (Lazzaro *et al.* 1990).

7 Empirical Results

We now compare the performance of the proposed classifier with that of two other classifiers, namely a backpropagation-trained neural network and a first-order Bayes model. It is important to point out that the primary goal of these experiments was to see if our rule-based classifier yielded *comparable* performance (in terms of classification accuracy) when compared to standard alternative approaches, rather than demonstrably *superior* performance. It is well known that most well-founded classifier algorithms will come reasonably close to the optimal Bayes classification rate on most reasonable problems (Weiss and Kapouleas 1989; Lee and Lippmann 1990). Hence the goal of the empirical evaluation is to

test whether the rule-based classifier can achieve similar near-optimal performance over a number of different problems.

From this point onward we will refer to the first-order Bayes classifier model as the "first-order classifier," and to the backpropagation trained feedforward neural network as the "neural network." A first-order classifier is a special case of our rule-based network, where the network architecture consists of no hidden units, i.e., the model consists of all possible first-order rules with the weights defined exactly as for the rule-based network as described earlier. This model is also known as a "naive Bayes" model (Kononenko 1989) and amounts to assuming that the joint distribution of the class and the attributes can be factored into first-order terms. We chose this model for comparison purposes both because of its simplicity and the fact that it often provides better classification performance than one has any right to expect.

The particular neural network design algorithm that we used was the conjugate-gradient scheme of Barnard and Cole (1989). Each network has three layers, the first layer containing a single node for each attribute, and the third layer containing a node for each class. The size of the hidden unit layer was typically chosen to be roughly twice the number of input nodes. One of the current problems with neural network techniques is the arbitrary choices that must be made in terms of architecture selection. If one chooses too few hidden units, the network may have too limited a hypothesis space to learn the required concepts, while with too many it may overfit on the training data. Typically, however, for the data sets considered here, we found only minor variations in classification accuracy as long as the number of hidden units was of the same order as the number of input units.

We evaluated the performance of the algorithms on five data sets. Two of the data sets are synthetic (LED digits, and a Boolean function), while the other three are real-world data sets (congressional voting, medical diagnosis, and protein secondary structure). The first data set is the well-known LED digits classification problem with 10% noise added. Essentially this consists of a seven-segment LED, where the seven segments correspond to seven binary features, and the digits "0" though "9" represent 10 classes. The 10% noise consists of reversing the segments from their true value with a probability of 0.1. This renders the classification problem somewhat nontrivial, since the optimal Bayes classification accuracy can be shown to be about 74% (as opposed to 100% for the noiseless case). We generated a database of size 1000 (with 10 equally likely classes) to use for evaluation. The second data set consists of 435 voting records from a session of the 1984 United States Congress. The attributes correspond to 16 different issues on which the politicians voted, such as aid to the Nicaraguan contras and budget cuts. The class variable is party affiliation, i.e., Democrat or Republican. Recognition accuracy up to 95% is known to be achievable on this data set using only a single attribute, the *physician-fee-freeze* attribute. Hence, as suggested by Bun-

Table 1: Comparative Performance Results on Three Data Sets.

	Mean percentage accuracy ± SD				Mean rule
Data set	Trivial	First-order	Neural	Rule-Based	Complexity ± SD
LED digits	10.0	74.1± 5.3	72.2 ± 4.96	73.1± 5.03	40 ± 1.27
Voting	61.4	87.44± 9.66	87.68± 7.06	88.18± 4.41	2.2 ± 0.40
Boolean	64.7	66.66± 5.41	89.99± 2.33	89.06± 2.44	11.7 ± 0.48

tine (1991) and others, the problem is made more interesting by removing this attribute. On the modified data set, Buntine reports accuracies up to 89% using a variety of decision tree techniques. The third data set is artificially generated with size 640, where there are 6 binary attributes, $Y_1, \ldots, Y_6$ and the class is the Boolean function

$$X = \mathrm{OR}[\mathrm{XOR}(Y_1, Y_2),\ \mathrm{AND}(Y_3, Y_4),\ \mathrm{AND}(Y_5, Y_6)]$$

To introduce noise, the class variable X has a 10% random chance of being reversed from its true state. Hence the optimal recognition rate on this problem is 90%. The fourth data set is a real database of breast cancer diagnosis data collected at the University of Wisconsin Hospitals between January 1989 and July 1990. We will describe this data set in more detail later since the performance of the classifiers was evaluated in an incremental manner as if they had been run as the data were collected (in chronological order). The fifth data set is a protein secondary structure problem, also described in more detail later.

For the first three data sets we use the standard evaluation technique of V-fold cross-validation where V was chosen to be 10. This means that the LED, voting, and Boolean function data sets were divided into disjoint test sets of size 100, 43, and 64, respectively. The neural network was a three-layer feedforward network with sigmoid activation functions, and 25, 20, and 8 hidden units for the LED, voting, and Boolean function problems, respectively. Both the mean and standard deviation of the resulting CV estimates are reported in Table 1. In addition we tabulate the mean complexity of the rule-based classifier for each data set, in terms of the mean (over the different training sets) number of weights connected to the output layer (the number of rules in the classifier). One column of the table corresponds to the mean accuracy obtainable for each data set simply by the trivial strategy of always predicting the most likely class label.

Performance on the data is roughly equivalent between the classifiers except for the first-order model on the Boolean function data—one of the motivations for including this data set was to demonstrate the limitations of the first-order model in capturing such high-order concepts. Hence, we can conclude that the rule-based model achieves roughly comparable classification accuracy to the more usual backpropagation model.

The fourth data set considered is the aforementioned medical diagnosis database. A common technique in breast cancer diagnosis is to obtain a fine needle aspirate (FNA) from a patient under examination. Wolberg and Mangasarian (1990) describe the domain in some detail. The FNA sample is evaluated under a microscope by a physician who makes a diagnosis. All patients evaluated as malignant, and some of those labeled as benign, later undergo biopsy, which confirms or disconfirms the original diagnosis—the other patients diagnosed as benign undergo later reexamination to provide a true measurement of their condition. Since biopsy is roughly eight times as costly as the FNA technique, it is important that unnecessary biopsies be kept to a minimum. In addition, Wolberg and Mangasarian report that physicians encounter borderline cases making diagnosis difficult. The approach taken by Wolberg and Mangasarian was to collect training data in the form of nine subjectively evaluated characteristics of the FNA sample for each patient. These features describe general characteristics of the FNA sample as seen under a microscope, such as uniformity of cell size, marginal adhesion, and mitoses. Ground truth in the form of class labels (benign or malignant) was obtained at a later stage by a biopsy or reexamination. A classifier was then designed that takes the physician's description of the FNA sample and produces a diagnosis. In Wolberg and Mangasarian (1990) a successful linear programming technique is introduced for determining the parameters of a neural network classifier for this diagnosis problem.

For our evaluation purposes we used the same database that consists of 535 patient records. As described above there are 9 attributes, each of which takes on a discrete value between 1 and 10. We chose to evaluate classifier performance by training the performance of each classifier on the first $k \times 50$ samples (where $1 \leq k \leq 10$) and testing on the remainder. This gives an idea of the performance as a function of training sample size and is also closer to the manner in which a classifier would be used in practice since the database of patient records is in chronological order. The results are shown in Figure 4. Clearly beyond about 150 training samples, all of the classifiers perform equally well. The excellent performance of the first-order classifier, and the fact that near 100% accuracy can be attained, leads one to suspect that the problem is not a difficult one in terms of classifier design. The relatively poor performance of the rule-based classifier for small sample sizes deserves some comment. In effect the MDL nature of the classifier design algorithm ensures that the model is conservative in its use of parameters when there are few data available. In contrast, both the first-order model and the neural network had a fixed, relatively complex, architecture independent of the amount of training data—the neural network used a single hidden layer with 12 hidden units throughout. In theory, for small sample sizes, both of these networks are too complex to be plausible models in a statistical sense. This phenomenon has been observed elsewhere by Cybenko (1990) and Smyth (1991). Nonetheless, in practice, overcomplex models can outper-

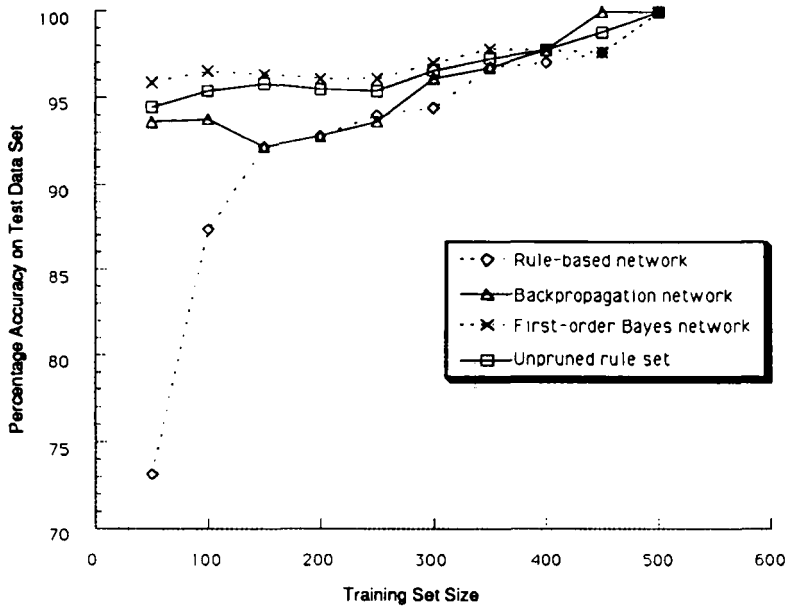

Figure 4: Medical database—performance.

form the more theoretically correct, simpler, models on a particular data set. In particular we note that the performance of our rule-based classifier when used with the more complex (in terms of more rules) unpruned rule set also performs comparably to the other techniques.

Classification accuracy alone is not, however, the only figure of merit of interest. We are particularly interested in the ability of our scheme to provide an estimate of the classifier's confidence in its decision, by using the output probability estimates provided along with the classification decision. For the rule-based and first-order Bayes networks these probabilities are produced directly. For the backpropagation network we normalize the output activation to produce a probability estimate. Figure 5 shows the mean binary entropy computed using these probabilities, for each classifier's decisions on the medical database, as a function of sample size. Entropy provides a measure of the classifier's uncertainty in its decision, and ranges from 0 (completely certain) to 1 bit (maximally uncertain). In practice this uncertainty estimate can provide a useful confidence indicator to a higher level decision maker. Two cases are shown for each classifier. One case corresponds to the uncertainty

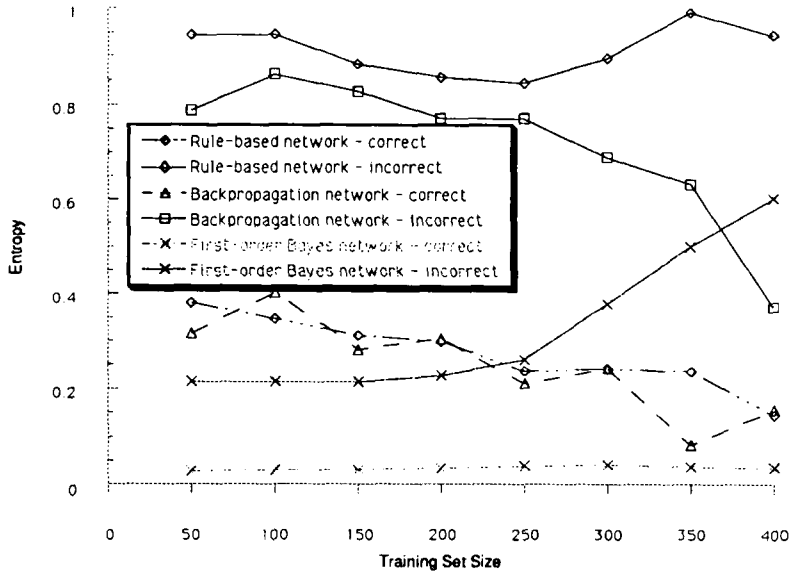

Figure 5: Medical database—classifier uncertainty.

when the classifier's decision was correct, and the other corresponds to the uncertainty when the decision was incorrect. Ideally we would like a classifier to have a low uncertainty (near 0) when it makes a correct decision, and a high uncertainty (near 1) when it makes an incorrect decision. Consider first the three curves that indicate incorrect decisions. From Figure 5 we see that the rule-based system performs well in this regard, being near maximal uncertainty when it makes a wrong decision over the entire range of sample sizes. The neural network does not perform as well. It begins with a reasonable degree of uncertainty and then becomes more definite (in its mistakes) as the training size increases. The first-order Bayes classifier is initially quite definite in its mistaken conclusions, but begins to become more reasonable as the sample size increases. This effect is likely due to the fact that its model becomes more accurate as more data become available (in terms of probability estimates). Also shown are the uncertainty curves for the correct decision. The rule-based and backpropagation networks are comparably low as desired, with the first-order Bayes classifier being even more confident in its decisions.

In Table 2 we list the the actual rules obtained when the algorithm was trained on the first 400 samples of the data set. This set of 11 rules is the final set obtained by the MDL portion of the algorithm after the

Table 2: Medical Database—Rules.

J-Measure	Rule				Strength w_{ij}
0.297	IF	cell size uniformity 1	THEN	DB[a]	5.9
	AND	mitoses 1			
0.289	IF	bare nuclei 1	THEN	DB	6.2
	AND	normal nucleoli 1			
0.271	IF	epithelial cell size 2	THEN	DB	8.0
	AND	bare nuclei 1			
0.231	IF	bare nuclei 10	THEN	DM[b]	−4.4
0.145	IF	clump thickness 10	THEN	DM	−5.7
0.111	IF	cell size uniformity 10	THEN	DM	−5.3
0.103	IF	normal nucleoli 10	THEN	DM	−5.2
0.085	IF	marginal adhesion 10	THEN	DM	−4.2
0.057	IF	cell size uniformity 5	THEN	DM	−4.5
0.056	IF	epithelial cell size 10	THEN	DM	−3.8
0.045	IF	bland chromatin 8	THEN	DM	−4.2

[a]Diagnosis benign.
[b]Diagnosis malignant.

rule search procedure had initially found a candidate set of 500 rules. The rules are ranked in order of decreasing average information content. The rules that confirm the benign condition (positive weights) are somewhat more informative than those that conclude the malignant condition (negative weights), primarily because the malignant rules have a lower prior probability of occurrence, i.e., the left-hand side conditions are less likely.

Figure 6 shows a diagram of the network that results when the rules are implemented on a neural architecture. Note that there are really only three genuine hidden units (the AND gates) corresponding to the three second-order rules. The first-order rules do not need a hidden unit and effectively correspond to a single weighted link between the input and output layers.

The final data set was chosen to test the rule-based approach on a large database. One of the original successes of the neural network classifier model on a large-scale problem was the secondary structure protein prediction problem, as described by Qian and Sejnowski (1988). The objective of this prediction task is to predict the secondary structure of globular proteins from a knowledge of their sequence of amino acids (the primary structure). The secondary structure is comprised of small groups of residues that join together into recognizable local shapes. The secondary structure is classified into one of three types: "helix," "sheet," and "coil," denoted by "h," "e," and "–," respectively. For our experiment we used the same training and testing data as used in Qian and

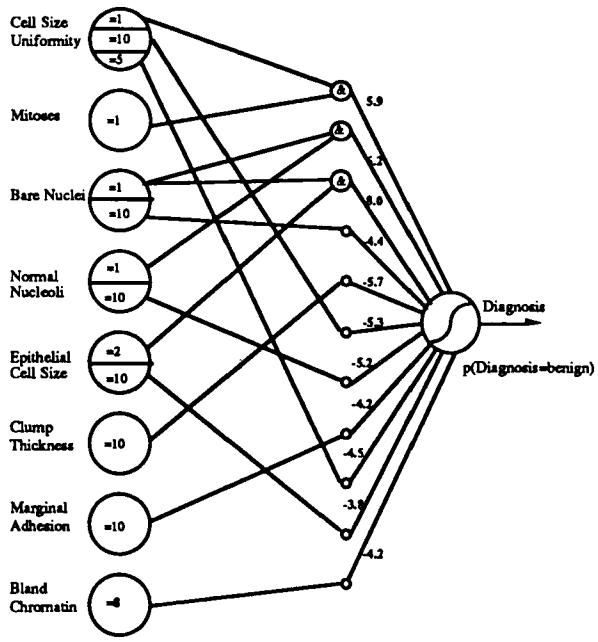

Figure 6: Medical database—network.

Sejnowski's paper, which consists of 18,105 training residues and 3520 testing residues. Each "example" in the database consists of a window of 13 contiguous amino acids, six preceding and six following a particular central amino acid. A total of 20 different amino acid fractions appear in the data, each one denoted by an alphabetic character (A, C, D, ... etc). The objective is to classify the central amino acid into one of the three secondary classes.

Prior to Qian and Sejnowski's work, the best results obtained on the protein data set were in the mid-50% accuracy range. The work of Qian and Sejnowski showed that accuracies in the low 60% range were obtainable using neural network techniques: 62.7% for a single network, and 64.3% when correlations between adjacent elements in the sequence were taken into account using a secondary cascaded network. Subsequent studies by other authors using network models achieved similar accuracies. Indeed Stolorz *et al.* (1991) recently showed that a first-order Bayes classifier can achieve 61.1% accuracy using a window size of 17, indicating that there is limited predictive information in the attributes beyond first-order.

Table 3: Protein Database—Rules.

J-Measure	Rule				Strength w_{ij}
0.016	IF	primary+1 P	THEN	secondary –	0.467
0.011	IF	primary+1 P	THEN	secondary h	−1.542
0.010	IF	primary+0 P	THEN	secondary –	0.394
0.009	IF	primary+0 G	THEN	secondary –	0.296
0.007	IF	primary+2 P	THEN	secondary –	0.340
0.006	IF	primary+0 V	THEN	secondary e	0.309
0.006	IF	primary+2 P	THEN	secondary h	−0.837
0.006	IF	primary+0 G	THEN	secondary h	−0.428
0.006	IF	primary−1 P	THEN	secondary –	0.313
0.006	IF	primary+0 V	THEN	secondary –	−0.348
0.005	IF	primary+0 I	THEN	secondary e	0.360
0.004	IF	primary+0 L	THEN	secondary –	−0.285
0.004	IF	primary+1 V	THEN	secondary e	0.237
0.004	IF	primary−1 G	THEN	secondary –	0.212
0.004	IF	primary+0 P	THEN	secondary e	−1.261
0.004	IF	primary+3 P	THEN	secondary h	−0.594
0.004	IF	primary+1 L	THEN	secondary –	−0.265
0.004	IF	primary+0 I	THEN	secondary –	−0.374
0.004	IF	primary+0 A	THEN	secondary h	0.594
0.004	IF	primary−1 G	THEN	secondary h	−0.273

We ran our algorithm to find the best first-order model, i.e., using only first-order rules on a window size of 13. The final model contained 194 rules and correctly predicted 61.7% of the test samples. In Table 3 we list the 20 most informative rules from the final pruned network model. It is interesting to note that the best rules tend to involve the central amino acid (primary+0) and the ones nearby at positions primary+1, primary−1, etc., rather than those farther away from the center. The rules also tend to be grouped into triplets of rules with the same left-hand side. These rules tend to be a positive rule for the most probable class (–, with a prior of 0.545), and two negative rules for the other two classes.

Again, we note that the point of this experiment was not necessarily to obtain better results than have been previously reported but to demonstrate that results comparable to other "black-box" techniques can be obtained on a large-scale discrete prediction problem, while achieving useful explainability due to the explicit rule-based model.

The experimental results confirm that the rule-based classifier is very competitive in terms of classification accuracy when compared with alternative approaches. The results (particularly those for the cancer diagnosis problem) also clearly demonstrate the unique ability of this approach to produce a hybrid rule-based neural network, wherein units and weights

possess a clear semantic interpretation to the external observer. In domains such as medical diagnosis such a feature makes the likelihood of user acceptance much higher than would be the case with a "black box" algorithm.

8 Conclusions

A novel hybrid rule-based connectionist classifier architecture has been proposed. The architecture of the classifier is directly derived from the example data by an efficient information-theoretic search technique. The classification performance of the hybrid scheme on discrete data has been shown to be comparable with that of conventional neural network classifiers, and the resulting network exhibits an explicit knowledge representation in the form of human-readable rules.

Acknowledgments

This work is supported in part by Pacific Bell, in part by the Army Research Office under Contract DAAL03-89-K-0126, and in part by DARPA under contract AFOSR–90–0199. Part of this research was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The authors thank David Aha of U.C. Irvine for providing the voting data set, and Olvi Mangasarian of the University of Wisconsin for providing the medical diagnosis data.

References

Aleksander, I. 1971. *Microcircuit Learning Computers.* Mills and Boon, London.

Angluin, D., and Smith, C. 1984. Inductive inference: Theory and methods. *ACM Comput. Surveys* **15**(3), 237–270.

Barnard, E., and Cole, R. 1989. *A neural net training program based on conjugate-gradient optimization.* Oregon Graduate Centre Tech. Rep. No. CSE 89–014, Oregon.

Blachman, N. M. 1968. The amount of information that y gives about X. *IEEE Trans. Inform. Theory* **IT–14**(1), 27–31.

Buntine, W. 1991. A theory of learning classification rules. Ph.D. Thesis, University of Technology, Sydney.

Cheeseman, P. 1983. A method of computing generalized Bayesian probability values for expert systems. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (Karlsruhe, West Germany)*, pp. 198–202.

Chiueh, T., and Goodman, R. M. 1990. VLSI Implementation of a high-capacity neural network associative memory. In *Advances in Neural Information Processing 2*, D. Touretzky, ed., pp. 793–800. Morgan Kaufmann, San Mateo, CA.

Chomsky, A. N. 1957. *Syntactic Structures*. Mouton, The Hague.

Cybenko, G. 1990. Complexity theory of neural networks and classification problems. In *Neural Networks*, L. Almeida, ed., pp. 26–45. Springer Lecture Notes in Computer Science.

Good, I. J. 1950. *Probability and the Weighing of Evidence*. Charles Griffin, London.

Goodman, R. M., and Smyth, P. 1989. The induction of probabilistic rule-sets—the ITRULE algorithm. *Proceedings of the 1989 International Workshop on Machine Learning*, pp. 129–132. Morgan Kaufmann, San Mateo, CA.

Goodman, R. M., Miller, J. W., and Smyth, P. 1989. An information theoretic approach to rule-based connectionist expert systems. In *Advances in Neural Information Processing Systems 1* , D. Touretzky, ed., pp. 256–263. Morgan Kaufmann, San Mateo, CA.

Kononenko, I. 1989. Bayesian neural networks. *Biol. Cybernet.* **61**, 361–370.

Lansner, A., and Ekeberg, O. 1989. A one-layer feedback artificial neural network with a Bayesian learning rule. *Int. J. Neural Networks* **1**(1), 77–87.

Lazzaro, J., Ryckebusch, S., Mahowald, M. A., and Mead, C. A. 1990. Winner-take-all networks of O(N) complexity. In *Advances in Neural Information Processing 1*, D. Touretzky, ed., pp. 703–711. Morgan Kaufmann, San Mateo, CA.

Lee, Y., and Lippmann, R. P. 1990. Practical characteristics of neural network and conventional pattern classifiers on artificial and speech problems. In *Advances in Neural Information Processing Systems 2*, D. Touretzky, ed., pp. 168–177. Morgan Kaufmann, San Mateo, CA.

Miller, J. W., and Goodman, R. M. 1990. A polynomial time algorithm for finding Bayesian probabilities from marginal constraints. *Proceedings of the Sixth Conference on Uncertainty in AI*, Cambridge, Massachusetts, July 27–29.

Minsky, M., and Selfridge, O. G. 1961. Learning in random nets. In *Information Theory (Fourth London Symposium)*, C. Cherry, ed., pp. 335–347. Butterworth, London.

Newell, A., and Simon, H. A. 1972. *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, NJ.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA.

Qian, N., and Sejnowski, T. J. 1988. Predicting the secondary structure of globular proteins using neural network models. *J. Mol. Biol.* **202**, 865–884.

Rissanen, J. 1984. Universal coding, information, prediction, and estimation. *IEEE Trans. Inform. Theory* **30**, 629–636.

Rissanen, J. 1987. Stochastic complexity. *J. Royal Stat. Soc. B* **49**(3), 223–239.

Rissanen, J. 1989. *Stochastic Complexity in Statistical Inquiry*. World Scientific, Teaneck, New Jersey.

Rissanen, J., and Wax, M. 1988. Algorithm for constructing tree structured classifiers. U. S. Patent no. 4,719,571.

Rosenblatt, F. 1962. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington, DC.

Rumelhart, D., Hinton, G., and Williams, R. 1986. Learning internal representations by error propagation. In *Parallel Distributed Processing 1* , D. E. Rumelhart and J. L. McCLelland, eds., p. 318. The MIT Press, Cambridge, MA.

Shore, J. E., and Johnson, R. W. 1980. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Trans. Inform. Theory* **IT-26**(1), 26–37.

Smyth, P. 1991. On stochastic complexity and admissible models for neural network classifiers. In *Advances in Neural Information Processing 3*, D. Touretzky, R. Lippman, and J. Moody, eds., pp. 818–824. Morgan Kaufmann, San Mateo, CA.

Smyth, P., and Goodman, R. M. 1992. An information theoretic approach to rule induction from databases. *IEEE Trans. Knowledge Data Engineer.* **14**(4), 301–316.

Stolorz, P., Lapedes, A., and Xia, Y. 1991. *Predicting protein secondary structure using neural net and statistical methods.* Tech. Rep. LA-UR-91-15, Los Alamos National Laboratory, Los Alamos, New Mexico.

Tenorio, M. F., and Lee, W. T. 1990. Self-organizing network for optimum supervised learning. *IEEE Trans. Neural Networks* **1**(1), 100–110.

Uttley, A. M. 1959. The design of conditional probability computers. *Inform. Control* **2**, 1–24.

Weiss, S. M., and Kapouleas, I. 1989. An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. *Proceedings of IJCAI 1989*, Morgan Kaufmann, San Mateo, CA.

Wolberg, W. H., and Mangasarian, O. L. 1990. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proc. Natl. Acad. Sci. U.S.A.* **87**, 9193–9196.

Received 19 April 1991; accepted 26 March 1992.