Automated Induction of Rule-based Neural Networks from Databases

Rodney M. Goodman
California Institute of Technology, Pasadena, CA, USA
Padhraic Smyth
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA

ABSTRACT **This paper describes our approach to the problem of automated knowledge acquisition from large databases of examples using an information-theoretic approach. Our previous research has resulted in practical algorithms (ITRULE) for the automatic induction of rules from large example databases. Utilizing these algorithms, the raw data can be transformed into a set of human readable IF THEN rules, thus giving insight into the knowledge hidden within the data. These rules can then be automatically loaded into an expert system shell. Alternatively, they can be used to build a new type of parallel inference system—a rule-based neural network. This process enables a prototype expert system to be automatically generated and up and running in a matter of minutes, compared with months using a manual knowledge-acquisition approach. The resulting expert system can then be used as a sophisticated search and analysis tool to query the original database capable of reasoning with uncertain and incomplete data.**

INTRODUCTION

Current database technology, in which the USA is a world leader, is being overwhelmed in many applications by the sheer weight of automated information gathering brought about by advances in hardware (Database Systems, 1990; French *et al.*, 1990). Across a variety of scientific, engineering, and business applications, it has become commonplace to collect and store large volumes of data. Examples in the scientific area include the potential storage of terabytes of interplanetary exploration and satellite image data by NASA (French *et al.*, 1990; NASA, 1991), and the large amount of DNA sequence data being contributed to the National Institute of Health (NIH) Human Genome Project by recent advances in automated DNA sequencing (Burks *et al.*, 1985). Commercial examples include the telecommunications industry in which sophisticated networks exist that automatically report a vast array of traffic information, data on module failures, system-performance analyses, and so on. In turn, these reports are automatically logged on a database system as a historical record of network operations. However, although the databases contain a wealth of information in terms of system performance and fault diagnosis, they are often too complex to manually search. Another familiar example is the automatic scanners used at checkout counters in modern-day supermarkets. These 'scan data' are automatically recorded and used for market research purposes. The volume of data available overwhelms what was previously a manual market-analysis task. There exist many other examples in business in which transactions are routinely logged and could provide a wealth of knowledge if only they could be automatically processed.

1055-615X/93/010041-14$12.00

Advances in software have not kept pace with the hardware advances that have caused this data explosion of the 1990s. Current database technology from a user's point of view allows the extraction of small text-based objects that are satisfied by relatively simple logical or declarative keyword queries. In the large commercial databases of the future such simple queries will not be able to extract the relevant *information* with such a query. This is because the data will be increasingly multi-media, and simple logical queries will return far too much data at too low a level to be of use. We predict that each query in the future will have to have the complexity of an expert system which can reason with uncertainty to give the user the *knowledge* that they want from the query, because a return of the 'matching' (in a loose sense) records will simply overwhelm the user with data. Using an example from *Database Systems* (1990) (and assuming it could be done with current technology), the company president could pose the query 'How many employees in the Widget department will retire in the next three years?' What would be returned is that number, or possibly the employee records. Of course, the president really wants to know 'What is the implication of employee retirements on the company pension fund over the next three years?' The answer the president wants is 'The pension fund will be underfunded by 10% next year with certainty 80%, but achieve level funding the year after with certainty 90%.' To answer such a query involves building *at query time* a complete expert system, based on the data in the database, capable of answering the fuzzy and incomplete query. Our approaches are a first step at automating the process of building expert systems from data, which could then be used in query systems of the future.

A major success of the Artificial Intelligence (A.I.) field of research has been the arrival in the 1980 of Rule-Based expert sytems. The basic paradigm behind expert systems is that knowledge, in an explicit form such as rules, is the key to building intelligent machines (Hayes-Roth *et al.*, 1983). The notion of expert systems was developed from the simple idea that human experts can be characterized as having both a large amount of detailed knowledge about a domain and the skill to use this knowledge effectively and efficiently to perform intelligent problem solving. The major advantages of this approach compared with traditional programming techniques are the explicit coding of knowledge in the form of rules and the ability to handle uncertain and incomplete data via inference techniques. However, these advantages do not come for free. Explicit knowledge representation is an important aspect of modeling rational behavior. Thus the power of the system lies in its knowledge base, an explicit representation of the domain knowledge, as opposed to being buried in 'procedural code'. This facilitates modularity so that portions of the knowledge can be added, deleted or modified without worrying about the control structure of the program. Maintenance, debugging and auditing are also easier. This is an important issue in terms of today's large-scale software projects where maintenance and debugging can be very expensive. The knowledge base is also an asset to the developer both as a training tool, a marketable commodity, and as a repository of human expertise which is relatively permanent. Developing knowledge bases is one of the most difficult aspects of building expert systems, and is commonly called the 'knowledge-acquisition bottleneck'. Knowledge acquisition is the process of translating the domain expertise into a form which can be used by the expert system, in our case, rules. Traditionally, this process has been performed by a 'knowledge engineer' interviewing the expert to identify the knowledge. This is problematic as it is very time-consuming (and, hence, expensive) and inefficient. Very often experts are not very good at describing their own expertise, and may be in disagreement with each other. Also, when it comes to acquiring knowledge involving uncertainty such as probabilistic statements, humans are not very good at giving consistent information of this form. In addition, we can conceive of real-time problems (such as plant control) where no human expert exists. The quality of the final expert system is critically dependent on the quality of the knowledge-acquisition process, and there is therefore an urgent need to develop tools to automate and assist this.

Motivated by results from cognitive science and neurobiology, and a desire to develop a

more robust approach to modeling intelligent behavior, a more recent (but related) AI paradigm has emerged—that of artificial neural networks or connectionist machines (Rumelhart and McClelland (1986)). Essentially, this approach can be characterized as modeling intelligent behavior using distributed parallel architectures. However, the implicit coding of knowledge in such machines remains a major stumbling block for applications which require higher-level cognitive processes at the reasoning rather than the perceptual level. This 'implicit' coding of knowledge in the weights and activations of a neural network is just the same problem that was recognized with conventional programming. Notwithstanding these problems, the inherent computational advantages of the connectionist approach make it an appealing candidate for the implementation of fast parallel intelligent machines, provided their structure and operation can be made explicit to humans.

Our previous work has been to develop a novel approach to solving some of the above problems using the formulations of information theory. The field we are developing is the application of information and communications theory techniques to the design and analysis of real-time rule-based expert systems. We have developed a novel information-theoretic approach to the design of rule-based expert systems, in which we can automatically extract an explicit model of the data in the form of rules, automatically load the model into an expert system shell, or onto the weights and nodes of a neural network, and then run the network to perform fast real-time probabilistic inference on unseen data. In particular, we are developing an information-theoretic approach to the following areas:

(1) Automatic derivation of generalized rule graphs from real world example data;
(2) Handling uncertain information and how uncertainty propagates through inference (uncertainty calculus);
(3) Knowledge representation from example data;
(4) Controlling and guiding the inference process;
(5) Fast parallel execution using artificial neural networks.

The ITRULE algorithm (Goodman and Smyth, 1988a, 1989a,b; Smyth and Goodman, in press) is our information-theoretic algorithm for the extraction of rules from example data. ITRULE takes in a database in the form of example vectors of attributes. These attributes can be categorical (e.g. color = red, blue, or green) or analog (e.g. pressure = 5.32). The algorithm then searches for probabilistic rules of the form: IF A = a AND B = b then C = c with probability p and utility U. ITRULE produces a list of rules ranked in accordance with the utility U, which is an information-theoretic measure of the 'goodness' of the rule. The rules produced by ITRULE can be used (and optimized) for several different purposes.

First, as the algorithm has efficient search techniques it can be used on large databases, in order to perform exploratory data analysis or 'database mining'. This is useful for generating an initial understanding of dependencies among variables, causal relationships, and so on, in an interactive and exploratory manner. One of the early successful applications of ITRULE was for a financial database describing the characteristics and performance of a variety of mutual fund investment companies averaged over five years.

The second use of the algorithm is in knowledge acquisition for standard expert systems. The probabilistic rule output can be directly used as the knowledge base for an expert system. Hence, one can use ITRULE to automate the rule-elicitation process, circumventing the often inefficient manual knowledge-acquisition methodologies. Indeed, even when no database is available, one can, in principle, use expert-supplied case studies as a synthetic data set. We have routinely used the algorithm to produce rules from data for various commercial rule-based shells. The ability to go directly from data to a working expert system in a matter of minutes is particularly powerful, allowing for rapid prototyping of a system and iterative improvement by adding new attributes and rerunning the rule induction in conjunction with human experts. The utility measures produced by ITRULE can be used to provide rule priorities to the expert system thus providing a method of performing conflict resolution during inference. A side benefit is that the knowledge engineer can produce a

tentative set of rules and hence a working expert system to show the human expert. As a basis for discussion this is a much more efficient use of the expert's time, and results in significant productivity improvement in the knowledge-acquisition process. We have reported an application of this technique for the development of expert systems for telecommunications network management and control, where this was used to great effect (Goodman *et al.*, 1989; Goodman and Latin, 1991).

Third, the rules can be loaded onto a new type of Rule-Based Neural Network Bayesian classifier which has also been developed in our research (Goodman *et al.*, 1990a). This classifier, which has achieved excellent classification performance in empirical tests, uses appropriate conditional independence assumptions to combine rule probabilities into an estimate of the class probability. In addition, the weight of evidence of each rule that contributes to the estimate can be used to construct an explanation of how the classification decision was arrived at, providing the basis for a decision-support system. This is of great importance in areas such as medical diagnosis, where the final decision must be made by the human user. Our rule-based classifier also has advantages in that the rules are mapped onto a neural network architecture, resulting in a very fast classifier whose weights have explicit meanings (Goodman *et al.*, 1990b). These weights have a direct interpretation as the evidential support provided by the rules—positive weights imply that the class is true while negative ones imply that it is false. Hence we see that while statistical in nature, this scheme possesses the ability to provide direct explanations to the user in terms of how the classification decision was arrived at. Most importantly we have shown that if the rules generated by ITRULE are conditionally independent in the left-hand sides, then the network performs correct Bayesian inference. Furthermore, by the addition of an exponentiation and normalization output step the network can be made to output correct posterior probabilities for the right-hand side variables (Goodman *et al.*, 1992 Miller and Goodman, 1990).

An Information-theoretic Approach

In this section we summarize our information-theoretic approach to expert system design. The techniques fall into two interrelated categories: automated rule induction using ITRULE, and using the induced rules to perform inferencing under uncertainty, using our Bayesian probability-estimation techniques.

The basis of our approach is that of a probabilistic rule, that is, a generalization of the usual deterministic rule. Say there are N attributes under consideration. Thus each sample datum of the database consists of a vector of some N particular *attribute* values, where each attribute has a discrete alphabet, and is a discrete-valued random variable. Real-valued attributes can be handled by an appropriate quantization scheme. A *proposition* is an attribute-value assignment of the form $\mathbf{Y} = y$ where y is in $\mathbf{Y}$'s alphabet. The basis of our model is what we call a *probabilistic production rule*, that is, a rule which has a probability attached to it. In general such an *lth-order* rule contains a conjunction of l variables in its LHS and some other variable as its RHS. For example:

If $\mathbf{Y} = y$ then $\mathbf{X} = x$ with probability p_r and utility U

where $\mathbf{X}$ and $\mathbf{Y}$ are discrete random variables (attributes), the probability p_r is the rule transition probability given by the conditional probability $p[\mathbf{X} = x|\mathbf{Y} = y]$, and U is the informational *utility* or 'goodness' of the rule. The attachment of a probability to the rule allows for the inherent handling of uncertainty with our model, as opposed to other *ad hoc* uncertainty measures.

In order to search the database for 'good' rules of the above form we need to define the utility or goodness measure U. We utilize an information-theoretic measure of the 'goodness' of a rule (Goodman and Smyth, 1988), called the average information content of a rule, or *J-measure*, and define it as:

$$J[\mathbf{X}; \mathbf{Y} = y] = \mathbf{p[y]} \cdot \Sigma_{\mathbf{x}} \mathbf{p[x|y]} . \log \left[\frac{\mathbf{p[x|y]}}{\mathbf{p[x]}} \right]$$

This measure can be interpreted as the information that the event $\mathbf{Y} = y$ yields about the variable $\mathbf{X}$ when it 'fires'. The J-measure is fundamental to our model and is the basis of

the utility functions that we have defined. The actual utility measure U used can be obtained by modifying the basic J measure to account for costs or other subjective preference criteria.

The ITRULE Algorithm

The J measure forms the basis of a rule-induction algorithm that we have proposed, called ITRULE (Information Theoretic Rule Induction) (Goodman and Smyth, 1988a, 1989a, b; Smyth and Goodman, in press). The algorithm takes as input a set of discrete attribute vectors and it produces as output a set of the K most informative probabilistic rules available from the data, as ranked by the utility measure. The algorithm works from general (low-order) rules to more specialized (high-order) conjunctive rules and uses the information-theoretic properties of the J-measure to determine whether or not a particular general rule is worth specializing. Information-theoretic bounds and small sample statistics are used to guide and constrain the search so that an (exponential) exhaustive search of all rules is not required unless supported by the data.

The raw list of rules produced by ITRULE can be filtered and optimized according to a number of different criteria. For example, low-order rules can 'subsume' higher-order rules with less information. Alternatively, an MDL (Minimum Description Length) criterion can be used to trade off the complexity of the model (the number and order of the rules) with the goodness of fit of the model (the classification accuracy of the set of rules). Also, costs can be introduced so that a 'cost per bit' of potential information can be defined. Rules which have a high cost per bit can then be subsumed by those with a low cost per bit, etc. The final result of the ITRULE process is an optimized list of rules that can be used to:

(1) Give a summary of the important information in the data, particularly the relative importance of different attributes;
(2) Be loaded into a standard expert system shell (with the Utility measure being used to assign a 'rule priority' figure to each rule in order to perform conflict resolution) thus allowing an expert system to be directly bootstrapped from data with minimal use of human experts; and
(3) Used to perform Bayesian inference and probability estimation using a Rule-Based Neural Network.

ITRULE has significant advantages over other rule induction methods such as ID3 (Quinlan, 1986), which are essentially decision tree induction algorithms as opposed to ITRULE's *generalized* rule induction. In fact our early research in decision trees (Goodman and Smyth, 1988b) led us to discover the limitations of trees and seek a new approach, the result of which is ITRULE. ITRULE has been used experimentally on many applications, including:

- Medical diagnosis
- Telecommunication trouble ticket analysis and fault finding
- Stock market and mutual funds analysis
- Real-time telephone network alarms analysis
- Mineral classification using satellite synthetic aperture radar returns
- Antenna fault diagnosis on the JPL deep space network
- Sonar returns signature analysis
- Supermarket product/sales analysis
- Census and questionnaire data

Using ITRULE

ITRULE requires a flat database text file in a rows and columns format. The columns or fields of the database are the 'attributes' or 'variables' of the domain. Each row is an 'example' or instance of the problem. The entries in the matrix are the 'values' taken by the attributes.

Attributes (data variables, field names) can be either categorical or continuous. Categorical (or discrete) attributes take a finite set of values; for example, the attribute 'Rank' may take the values 'Private', 'Corporal', and 'Sergeant'. A continuous attribute takes either integer values or real number values. For example, the attribute 'Temperature' takes real number values such as 28.45°. An attribute such as 'number of shipments' takes integer number values such as 28 or 65. Continuous attributes need to be quantized (made into discrete attributes by splitting into ranges). This can be automatically performed by ITRULE, but it is much better if

'meaningful' ranges of the data can be identified given the context of the data. For example, within a particular context it may make sense to split a continuous variable called 'Temperature' into three significant ranges, thus resulting in a discretized 'Temperature' variable which takes the values 'below_zero', 'zero_to_boiling', and 'above_boiling'.

ITRULE optionally needs to know which attributes are 'right-hand-side' (RHS) or 'hypothesis' attributes, and which are 'left_hand_side' (LHS) or 'data only' attributes. RHS attributes will appear both in the conclusion part of rules and in the LHS of other rules. Data only attributes appear only in the LHS of rules. By default, ITRULE uses all variables as RHSs.

ITRULE's output can take several forms: printed rules, rules in a format suitable for a number of standard expert system shells, rules that can be loaded onto a number of neural network simulators, or rules for ITRULE's own internal probabilistic Rule-Based Network inference mechanism. The rules can then be used to perform prediction on new examples.

Rule-Based Neural Networks

ITRULE uses the rules it has discovered to build a parallel probabilistic inference network rather like a neural network. However, unlike a neural network which is an *implicit* 'black box' predictor, the ITRULE Rule-Based Network has an *explicit* architecture and operation. The architecture is explicit because links in the network correspond to rules. The inference is explicit because the weights on the links correspond to the 'weight of evidence' associated with each rule. That is, our belief in the truth of the rule RHS, given that the rule LHS has fired.

The explicit nature of the Rule-Based Network allows all its decisions to be audited by humans, and, if necessary, shown to a third party or judge to prove that it is operating in the desired manner.

The Rule-Based Network is a powerful new extension of a simple first-order Bayesian classifier. The network is capable of acting as a classifier or, much more powerfully, outputting probability or 'confidence' estimates for each output decision. This enables a higher-level decision maker (such as a human) to make the final decision. The advantage here is that situations in which a completely unknown input is presented can be identified by low confidence on all the outputs. This alerts the system to the fact that more training, or more input attributes, are necessary to derive rules to handle the new situation.

The structure of the rule-based network is

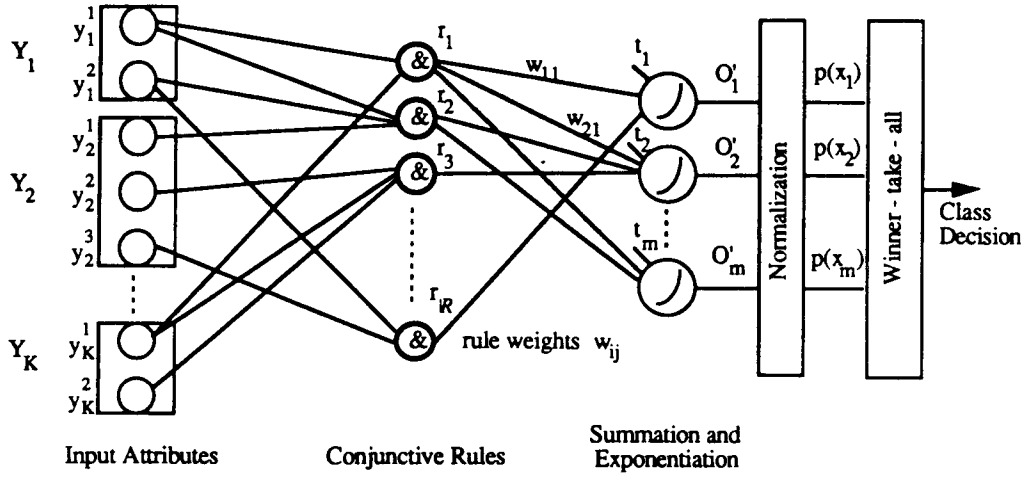

Figure 1 Rule-based network.

shown in Figure 1. 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 which 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 a rule is connected to a third-layer node via a multiplicative weight of evidence w_{ij}. Also feeding into and summed by each third-layer node is a bias value t_i. This sum of activations is then *exponentiated* to produce the node output O'_i. The output of the exponentiators is then fed into a normalization layer which constrains the outputs to sum to one. The output of this layer is the desired probability estimate of each class. If a classification decision is required, a winner-take-all stage can be added to decide on the most likely class.

The network uses the rules found by ITRULE to perform probabilistic inferencing in a data-driven forward-chaining manner (Goodman *et al.*, 1990a). Given an input vector consisting of a known subset of evidence variables, we will have in general a set of rules R_i which 'fire' (i.e. their left-hand sides are true) and which have a particular attribute-value x_i on the right-hand side. The network estimates the conditional probability $p(x_i|R_i)$, that is, the posterior probability of each RHS attribute-value given the rules that fire, using

$$\log(\mathbf{p}(\mathbf{x}_i|\mathbf{R}_i)) = \Sigma_j \mathbf{W}_{ij} + \log(p_i)$$

where $W_{ij} = \log(p(x_i|\mathrm{LHS}_j))/\log(p_i)$ has the direct interpretation of being the weight of evidence provided by a particular rule's LHS about the RHS attribute, positive for true, negative for false. This weight is equal to the log ratio of the rule transition probability p_r and RHS prior probability, and is found by the ITRULE process. The $\log(p_i) = t_i$ term is the prior bias; thus if no rules fire for a particular RHS attribute the estimate of its posterior probability is equal to the prior. The above equation thus specifies the inference process. The weights of evidence provided by the j rules which fire, together with the prior evidence, is accumulated for each RHS attribute-value. The exponentiation and normalization layers serve to constrain the sum of evidence for each RHS attribute to be ≤ 1, thus allowing the actual posterior probability of each class via:

$$\log(\mathbf{p}(\mathbf{x}_i|\mathbf{R}_i)) = e^E/\Sigma_j e^E, \text{ where } E = \log(\mathbf{p}(\mathbf{x}_i|\mathbf{R}_i)).$$

A classification decision on the posterior value for each RHS attribute can then be made by choosing the value with the largest positive value of $p(x_i|R_i)$ in the winner-take-all layer. The output of the network is thus the most likely value taken by each RHS attribute, given the rules that fired due to the evidence variables that were presented. Note that a complete evidence vector is not necessary, that is, as little or as much evidence information can be presented to the system, which will then compute the best decision given the evidence available.

The inference process outlined above computes the best Bayesian estimate of the values taken by each RHS attribute, given the evidence that was presented, and the particular rule set found by ITRULE. Note that this can be a dynamic process in that as new evidence appears, the inference process is rerun to provide the new estimates. In addition, backward chaining is possible by observing which output conclusions are most uncertain (by observing their posterior probability) and this allows for the system to suggest that certain evidence variables should be determined in order to be able to fire rules which will increase the confidence in those RHS conclusions. The incorporation of these advanced features would result in an expert system based on sound probabilistic and information-theoretic principles, capable of inferencing with uncertain, conflicting (and missing) evidence on real-world problems.

An Example of ITRULE Analysis using a Mutual Funds Database

As an example of using ITRULE, we show how rules can be generated from a database of mutual funds information. Table 1 shows a portion of the raw database of mutual funds (AAII, 1990). Note there are several thousand funds in the whole database. Each row represents an example of a particular mutual fund (the name is omitted). The column headings are the attributes of interest when thinking of

Table 1 Raw mutual funds database.

Portion of the mutual funds raw database:

Fund type	5-year return	Diversity	Beta (risk)	Bull perf.	Bear perf.	Stocks (%)	Largest holding	Investment income	Dividends	Investment gain	Capital distrib.	Net asset value	Expense ratio (%)	Turnover rate (%)	Assets ($M)
Growth	135.6	C	0.8	B	D	87	Information	0.67	0.5	5.35	6.07	37.27	0.79	34	414.6
Growth	32.5	C	1.05	E	B	81	Manufacture	−0.02	0.11	−0.76	0	12.49	1.4	200	16.2
Gth&Income	88.3	A	0.96	C	D	82	Financial	0.14	0.16	2.12	0.41	11.92	1.34	127	26.8
AggrGth	24.4	A	1.23	E	E	95	Oil	0.02	0	−0.14	0.6	6.45	1.4	161	64.2
Gth&Income	172.2	E	0.59	A	B	73	Office equip	0.53	0.52	2.88	0.84	13.64	1.09	31	112.5
Gth&Income	80.4	A	0.58	D	B	98	Computers	0.45	0.45	1.06	2.2	13.7	1.18	64	76.4
Growth	52.7	B	0.86	E	D	95	Airlines	0.14	0.24	1.17	0	8.72	1.6	54	11.7
Gth&Income	143.8	C	0.71	B	B	51	Materials	0.72	0.66	2.9	0.7	13.03	0.98	239	190
AggrGth	91.7	B	1.24	C	E	100	Technology	0.01	0.04	0.96	1.8	8.11	1.28	218	96.2
Growth	105.4	A	0.99	B	D	98	Energy	0.34	0.55	3.64	3.21	13.62	0.75	20	253.8
Growth	93	C	0.91	C	B	94	Drugs	−0.22	0	7.35	0	27.44	3.5	8	3.9

Table 2 Quantized database.

Quantized database as input to ITRULE:

Fund type	5-year return	Diversity	Beta	Stocks	Yield	Expns ratio (%)	Turnover	Assets	Cap. gain
	S&P = 138.5%	<100% >100%				<1.5% >1.5%	>100% <100%	<$100M >$100M	
Growth	Below S&P	Low	Under 1	Over 75%	Under 3%	Low	Low	Large	10–20%
Growth	Below S&P	Low	Over 1	Over 75%	Under 3%	Low	High	Small	Under 10%
Gth&Income	Below S&P	High	Under 1	Over 75%	Under 3%	Low	High	Small	10–20%
AggrGth	Below S&P	High	Over 1	Over 75%	Under 3%	Low	High	Small	Under 10%
Gth&Income	Above S&P	Low	Under 1	Under 75%	Over 3%	Low	Low	Large	Over 20%
Gth&Income	Below S&P	High	Under 1	Over 75%	Over 3%	Low	Low	Small	Under 10%
Growth	Below S&P	High	Under 1	Over 75%	Under 3%	High	Low	Small	10–20%
Gth&Income	Above S&P	Low	Under 1	Under 75%	Over 3%	Low	High	Large	Over 20%
AggrGth	Below S&P	High	Over 1	Over 75%	Under 3%	Low	High	Small	10–20%
Growth	Below S&P	High	Under 1	Over 75%	Over 3%	Low	Low	Large	Over 20%
Growth	Below S&P	Low	Under 1	Over 75%	Under 3%	High	Low	Small	Over 20%
Growth	Above S&P	High	Under 1	Over 75%	Under 3%	Low	Low	Large	10–20%
AggrGth	Below S&P	High	Over 1	Over 75%	Under 3%	High	Low	Small	Under 10%
Gth&Income	Below S&P	Low	Under 1	Under 75%	Over 3%	High	Low	Small	10–20%

Table 3 Mutual fund rules.

The top 20 most informative rules as output by ITRULE:

									Prob.	Info.	Utility	Strength
1 IF	Fund type	Gth&Income	AND	Beta	under 1	THEN	Yield	over 3%	0.80	0.21	1000	1.17
2 IF	Beta	Under 1	AND	Yield	over 3%	THEN	Fund type	Gth&Income	0.80	0.2	952	1.13
3 IF	5-yr return	Above S&P	AND	Yield	over 3%	THEN	Stocks	under 75%	0.85	0.17	810	−1.61
4 IF	Assets	Large	AND			THEN	Expns ratio (%)	low	0.99	0.135	643	−0.35
5 IF	Stocks	Under 75%	AND	Expns ratio (%)	low	THEN	5-yr return	above S&P	0.74	0.127	605	1.26
6 IF	Yield	Over 3%	AND			THEN	Beta	under 1	0.95	0.117	557	−0.49
7 IF	Fund type	Gth&Income	AND	Cap. gain	10–20%	THEN	Stocks	under 75%	0.68	0.105	500	−1.29
8 IF	5-yr return	Above S&P	AND	Expns ratio (%)	low	THEN	Assets	large	0.78	0.097	462	0.78
9 IF	Yield	Over 3%	AND	Expns ratio (%)	low	THEN	Assets	large	0.78	0.097	462	0.78
10 IF	Diversity	Low	AND	Assets	small	THEN	Expns ratio (%)	high	0.54	0.096	457	1.27
11 IF	Yield	Under 3%	AND			THEN	Stocks	over 75%	0.91	0.096	457	0.33
12 IF	Fund type	Gth&Income	AND			THEN	Beta	under 1	0.92	0.092	438	−0.44
13 IF	Beta	Over 1	AND	Assets	small	THEN	Fund type	AggrGth	0.60	0.089	424	1.29
14 IF	Fund type	AggrGth	AND	5-yr return	below S&P	THEN	Beta	over 1	0.65	0.072	343	1.01
15 IF	Assets	Small	AND			THEN	5-yr return	below S&P	0.87	0.068	324	−0.33
16 IF	Beta	Under 1	AND	Expns ratio (%)	low	THEN	Assets	large	0.67	0.067	319	0.54
17 IF	Stocks	Over 75%	AND			THEN	Yield	under 3%	0.81	0.067	319	−0.33
18 IF	Stocks	Over 75%	AND	Assets	large	THEN	Fund type	Growth	0.69	0.066	314	0.78
19 IF	5-yr return	Below S&P	AND			THEN	Stocks over	over 75%	0.86	0.058	276	0.26
20 IF	5-yr return	Above S&P	AND			THEN	Expns ratio (%)	low	0.95	0.052	248	−0.29

investing in a mutual fund. An example of a categorical attribute is 'Fund Type' which takes three possible values: 'Growth', 'Growth &Income', and 'Aggressive Growth'. An example of a continuous attribute is the 'Beta' or 'risk' variable of the fund.

Table 2 shows a derived database in which all the continuous variables have been quantized.

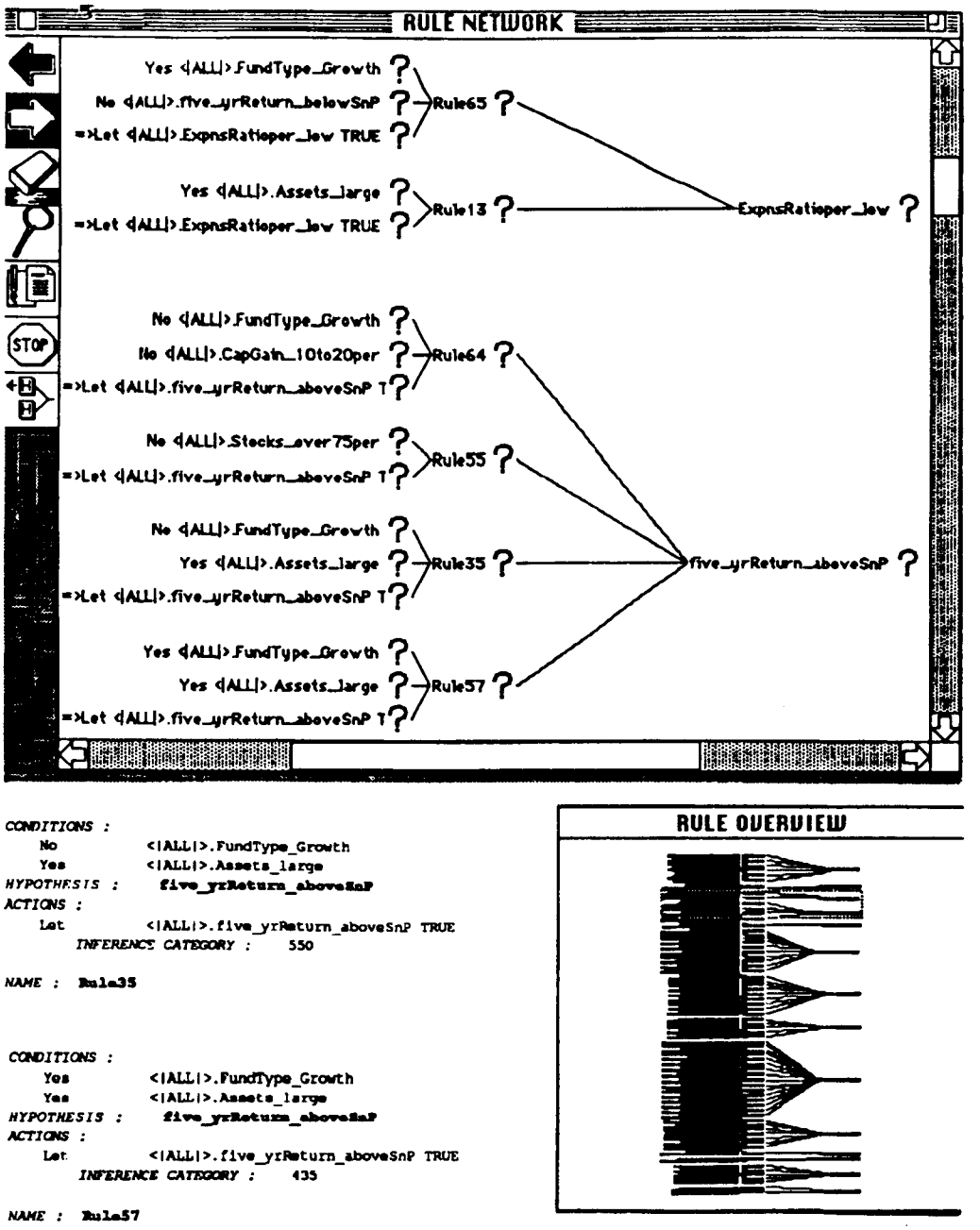

Figure 2 Rules loaded into NEXPERT®

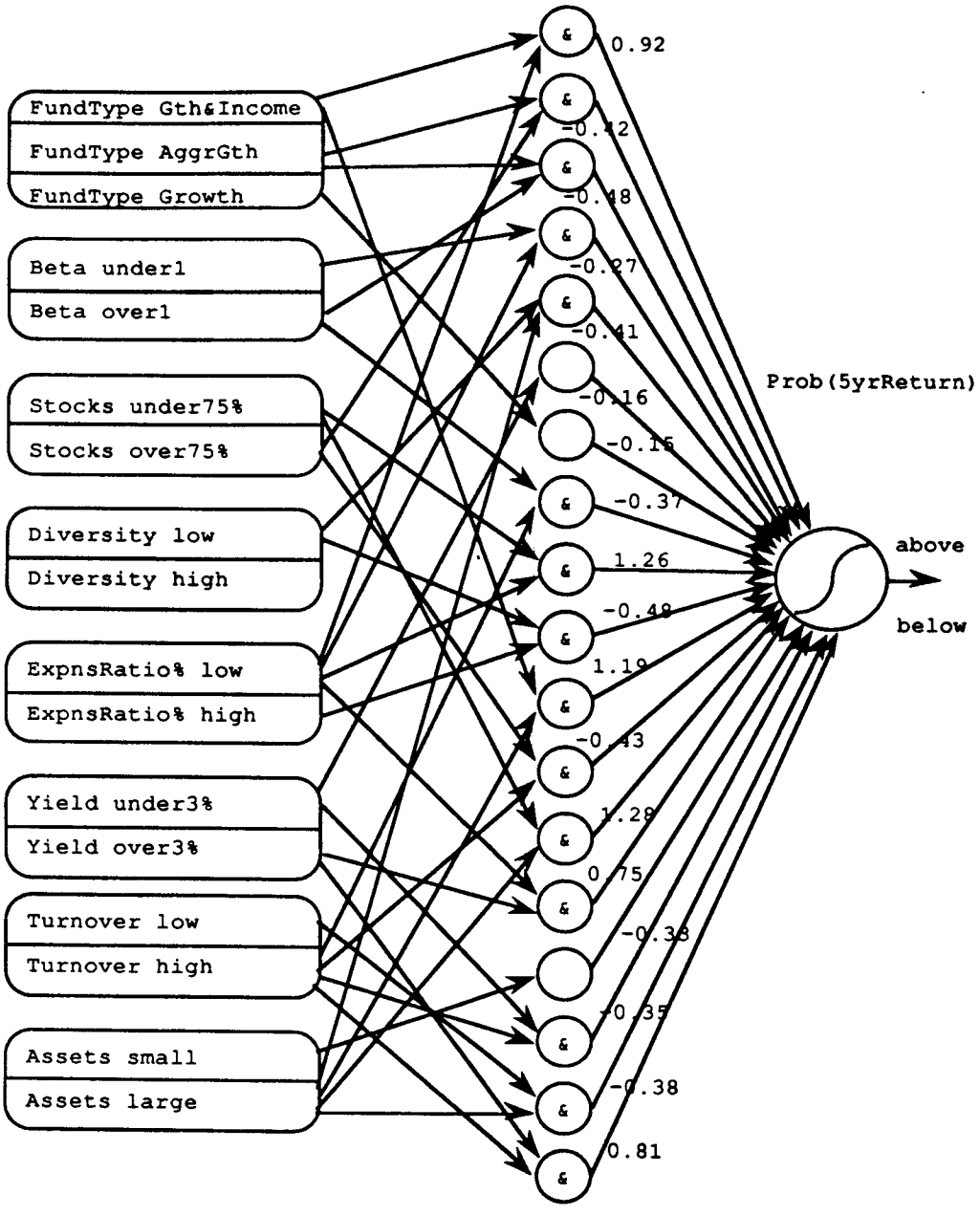

Figure 3 Mutual funds rule network.

The expert has a significant say in this initial process which serves to both categorize numerical data, and select attributes of interest. Note that over-specification of the problem is acceptable, because non-informative attributes will be penalized by low J measures. Numerical data can be categorized by the expert, or by using automatic statistical 'binning' algorithms that we have developed (Smyth and Goodman, in press). The expert can accept this advice or modify the value to make the categorization more meaningful. Some of these quantizations

are 'obvious', and some require expert contextual knowledge of the database domain. For example, the 'Beta' or 'risk' attribute is naturally defined as above and below 1, where 1 is the market risk. On the other hand, the Capital Gain attribute has been quantized to three ranges on expert advice.

Table 3 shows the rules output by ITRULE ranked in order of information content. The probability value is a measure of the 'reliability' of the rule; for example, a completely deterministic rule (a certainty) has probability 1. The Utility value in this case is the relative information of the rule relative to the most informative rule. The 'strength' value is the weight of evidence of the rule, and is used in the rule-based neural network. The rules display information of interest to the investor, and can be used to predict the performance of new funds. For example rule 15 states that IF Assets are small THEN the 5 year Return of the fund will be below the Standard and Poor's Market average return with high probability (0.87).

Figure 2 shows the rules derived by ITRULE loaded into a standard expert system shell, NEXPERT®, in this case. This is almost 'instant' knowledge engineering.

Figure 3 shows a rule-based network which has been built to predict the value of the variable Five-Year Return. The network has been optimized by a rule pruning process that selects the best set of rules to predict this RHS.

CONCLUSIONS

In this paper we have described a novel hybrid approach to knowledge acquisition and designing expert systems from example data. The rule-based neural networks outlined have significant advantages over 'black-box' neural networks in that the network is directly derived from the data by an efficient information-theoretic search technique. Furthermore, the classification performance of the rule-based scheme on discrete data has been shown to be comparable with that of conventional neural network classifiers, while its probability prediction performance tends to be superior. Most significantly, the resulting network exhibits an explicit knowledge representation in the form of human-readable rules.

Acknowledgements

This work is supported in part by Pacific Bell, in part by the Army Research Office under Contract No. DAAL03–89–K–0216, and in part by DARPA under contract no. AFOSR–90–0199. In addition this work was carried out in part by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

AAII (1990) *The Individual Investors Guide to No-Load Mutual Funds*, AAII, International Publishing Corporation.

Burks, C.J. *et al.* (1985) 'The Genbank nucleic acid database', *Comp. Applic. Biosci., No. 1, 225–33.*

Database Systems: Achievements and Opportunities (1990) Report of the NSF Invitational Workshop on the Future of Database Systems Research, Palo Alto, California, February.

French, J.C., Jones, A.K. and Pfaltz, J.L. (1990) 'Summary of the Final Report of the NSF Workshop on Scientific Database Management', SIGMOD Record, December.

Goodman, R.M. and Smyth, P. (1988a) 'Information-theoretic rule induction', Proceedings of European Conference on Artificial Intelligence, 2–5 August, Pitman Publishing, London.

Goodman, R.M. and Smyth, P. (1988b) 'Decision tree design from a communication theory standpoint', *IEEE Trans. on Information Theory*, **34**, No. 5, 979–994, September.

Goodman, R.M. and Smyth, P. (1989a) 'The induction of probabilistic rule sets—the ITRULE algorithm', *Proceedings of the Sixth International Workshop on Machine Learning*, Los Altos, CA: Morgan Kaufman pp 129–132.

Goodman, R.M. and Smyth, P. (1989b) 'The ITRULE algorithm for rule induction', 1989 IJCAI Workshop on Knowledge Discovery in Databases, 20 August.

Goodman, R.M., Higgins, C. and Smyth, P. (1990a) 'A hybrid rule-based/Bayesian classifier', *Proceedings of the 1990 European Conference on Artificial Intelligence*, Pitman Publishing, London, pp 610–615, August.

Goodman, R.M., Higgins, C., Miller, J. and Smyth, P. (1990b) 'A rule-based approach to neural network classifiers', INNC 90 Paris—International Neural Network Conference, Palais Des Congrés, Paris, France, 9–13 July.

Goodman, R.M. and Latin, H. (1991) 'Automated knowledge acquisition from network management databases', Second IFIP-IEEE International Symposium on Integrated Network Management, Washington, DC, 1–5 April.

Goodman, R.M., Smyth, P., Higgins, C. and Miller, J. 1992 'Rule-based neural networks for classification and probability estimation', in press, *Neural Computation.*

Goodman, R.M., Smyth, P. and Latin, H. (1989) 'Real time autonomous expert systems in network management', Invited paper—First IFIP International symposium on Integrated Network Management, Boston, 14–17 May.

Hayes-Roth, F., Waterman, D.A. and Lenat, D.B. (1983) 'An overview of expert systems', in Hayes-Roth, F., Waterman, D. and Lenat, D.B. (eds), *Building Expert Systems,* Reading, MA: Addison-Wesley, pp. 3–30.

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

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

NASA (Jorgenson C., Buntine W.) (1991) Workshop on Pattern Discovery in Large Databases, 14–15 January, NASA Ames, CA.

Rumelhart, D.E. and McClelland, J.J. (1986) *Parallel Distributed Processing,* Volume 1, Cambridge, MA: The MIT Press.

Smyth, P. and Goodman, R.M. (in press) 'An information theoretic approach to rule induction from databases', *IEEE Transactions on Knowledge and Data Engineering.*