THE APPLICATION OF INFORMATION THEORY TO PROBLEMS IN DECISION TREE DESIGN AND RULE-BASED EXPERT SYSTEMS

Thesis by

Padhraic Smyth

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1988

(Submitted May 17, 1988)

Acknowledgements

Getting to the thank the people who helped me is probably the best part of writing a thesis. It would be impossible to thank everyone who contributed in one way or another. There isn't enough room and, more to the point, I'm sure that I will forget to thank several important people anyway. To the forgotten ones, thanks.

A very sincere thanks must go to Professor Rod Goodman, my advisor, whose tremendous enthusiasm for the subject, and general encouragement, have been a great help. His ideas and willingness to experiment have contributed a great deal to my work. I would also like to thank Professor Ed Posner for allowing me to participate in various telecommunications projects at Caltech and Pacific Bell, which, in my position as a foreign graduate student trying to get home occasionally, greatly helped my often-ailing bank account. Mention must also be made of Professor Posner's wit and demeanour, which makes the EE systems group a very enjoyable group to work in. Also, I would like to thank Professor Bob McEliece for persuading me to come and work at Caltech back in 1984, a move which turned out to be very fortuitous. And thanks also, to Professor's Yaser Abu-Mostafa and John Hopfield for kindly agreeing to read this over-long manuscript.

Thanks to the various people in the research group, Tzi-Dar Chieuh, Seán Coffey and John Miller, who helped me in discussing research ideas, proof-reading the thesis, letting me use the new Sun workstation every day for the past two months, and generally being fun to work with. Credit also goes to Truong Nguyen for always helping out when I tried to get the VAX to do something it hadn't done before. Mention has to be made of the Caltech administrative staff and secretaries who always let me get things done with a minimum of red-tape and two weeks after the last official deadline.The contribution of Pacific Bell who have consistently supported our research is also very much appreciated.

That takes care of the people who helped me directly (I hope). There's also a supporting cast of other players. Thanks to Caltech's soccer players, my four years at Caltech have allowed me to pursue athletic endeavours far more than I might have imagined. I also want to thank my merry band of friends who attempted to sink my research by dragging me skiing in the daytime, and to the Loch Ness, the King's Head and other Los Angelean dens of iniquity by night. Their valiant attempt to keep me at Caltech *ad infinitum* failed,

but only just. A note of thanks also goes to the "Train to Sligo" band (now defunct, alas) both for weekly entertainment and for occasionally letting me keep my fingers in shape.

Special thanks to my family who encouraged me in every possible manner to pursue my goals, and more importantly, sent me the football scores and the Sunday paper every week.

Last, but by no means least, a very special thank you to Crista for teaching me how to avoid rattlesnakes while hiking in the San Gabriel mountains, but also for providing love, support and friendship which I can only strive to repay. Crista, thanks for everything.

Abstract

This thesis examines the problems of designing decision trees and expert systems from an information-theoretic viewpoint. A well-known greedy algorithm using mutual information for tree design is analysed. A basic model for tree design is developed leading to a series of bounds relating tree performance parameters. Analogies with prefix-coding and rate-distortion theory lead to interesting interpretations and results. The problem of finding termination rules for such greedy algorithms is discussed in the context of the theoretical models derived earlier, and several experimentally observed phenomena are explained in this manner. In two classification experiments, involving alphanumeric LEDS and local edge detection, the hierarchical approach is seen to offer significant advantages over alternative techniques.

The second part of the thesis begins by analysing the difficulties in designing rule-based expert systems. The inability to model uncertainty in an effective manner is identified as a key limitation of existing approaches. Accordingly, an information-theoretic model for rules and rule-based systems is developed. From a simple definition of rule information content, the ability to specialise and generalise (akin to cognitive processes) in a quantitative manner is demonstrated. The problem of generalised rule induction is posed and the ITRULE algorithm is described which derives optimal rule sets from data. The problem of probabilistic updating in inference nets is discussed and a new maximum-likelihhod rule is proposed based on bounded probabilities. Utility functions and statistical decision theory concepts are used to develop a model of implicit control for rule-based inference. The theory is demonstrated by deriving rules from expert-supplied data and performing backward and forward chaining based on decision-theoretic criteria. The thesis concludes by outlining the many problems which remain to be solved in this area, and by briefly discussing the analogies between rule-based inference nets and neural networks.

Table of Contents

List of Figures and Tables

Ceo draíoctha i gcoim oiche do sheol mé
trí thíorthaibh mar óinmhid ar strae
Eoghan Rua Ó Súilleabháin, c.1770

This thesis is dedicated to my mother and father, who gave me
the gift of being able to appreciate learning and discovery.

Introduction

The emergence of electronic and magnetic storage media over the last few decades as convenient and affordable methods to store large amounts of data, has led to the coining of the phrase "information revolution." The popular notion appears to be that the widespread availability of information will considerably accelerate man's technological progress. With the continued progress in increasing the information capacity of both storage media (e.g., VLSI memory chips, optical discs) and transmission media (e.g., optical fibers), one can only predict that the quantity of electronic data in existence will continue to grow at a phenomenal rate. Yet, despite the progress in *handling* this information from a hardware standpoint, progress in *using* the information continues to lag far behind.

One of the primary reasons is the sheer quantity and volume of the data. Simply put, there is not enough manpower to analyse and examine the typical large database. For example, in the telecommunications industry at present, very sophisticated networks are being installed which automatically report a vast array of traffic information, data on module failures, system performance analyses, etc. These reports are automatically "logged," in turn, on a database system, as a historical record of the network operation. However, although the databases contain a wealth of information in terms of fault diagnosis, they are often too complex to search manually. This scenario is common in many different applications, where users are being overwhelmed by information.

The fundamental goal of this thesis is to propose and investigate new techniques and theories for the *automated* analysis of data, with a view to creating a model of the environment from which the data came. Our basic tools will be probability and information theory. In particular we will look at the problem of designing systems or algorithms which can *generalise* from existing data, in order to classify, or make decisions about, future data. The thesis is divided into two chapters, which deal with the particular problems of designing decision trees and rule-based expert systems, respectively. Each chapter is relatively self-contained and the reader may read them in any order. Chapter 1, on decision trees, assumes a basic familiarity with the principles of information theory, in particular, mutual information, prefix coding, and the concept of rate-distortion functions. Given that the thesis is quite lengthy, tutorial-type discussions of these ideas are not included. Instead, the interested reader is referred to the first few chapters of any basic information theory

textbook, such as McEliece [123] or Blahut [126]. Chapter 2 does not assume any prior background on the part of the reader, except perhaps a grasp of basic probability theory.

Chapter 1

An Information-Theoretic Approach to Decision Tree Design

1.1 The mutual information decision tree design algorithm

1.1.1 Background and motivation

Hierarchical classifier design is a subject which has received considerable attention in recent literature. In designing any classifier, hierarchical or not, one seeks to classify some future samples of unknown class based on what one has *inferred* from labelled training samples. The training samples are described by feature vectors located in a multi-dimensional feature space. In addition, each sample has a label, i.e., the class to which it belongs. Classifier design can thus be viewed as a statistical inference procedure for partitioning the multi-dimensional feature space into class regions. Thus, any future sample is assigned to a particular class region (classified) according to some function of its feature vector in this space. It is important to note that the above problem is quite general and could equally well describe problems in speech recognition, image analysis, data compression, medical diagnosis, equipment maintenance strategies, etc.

The more traditional techniques of partitioning the feature space, as described in Duda and Hart [1], rely on extensive knowledge of the joint probability distributions between the classes and the feature vectors. In addition, they yield computationally complex classifiers where, typically, an unknown sample must be compared to each class to find the "most likely" classification. Estimation of the joint distributions when either the dimensionality of the feature space or the number of classes is very large requires impractically large training sets. Indeed, increasing the number of features while keeping the number of samples fixed can actually lead to a decrease in the accuracy of the classifier [2].

An alternative approach is to use a non-parametric hierarchical partitioning of the feature space. Clearly, as in any hierarchy, the *order* in which the hierarchy is implemented is of fundamental importance for any optimality criterion which might be used. This hierarchical approach can be traced back to Wald's original work on sequential statistical decision theory [3]. The idea of using the mutual information between the features and classes to select the best features has received considerable attention recently (cf. [7] through [28]), but, in fact, the idea was initially put forward in 1962 in a paper by Lewis [4]. In 1968, Fu [5] formulated Wald's ideas in terms of the classifier design problem with reference to Lewis' information criterion. More recently there has been renewed interest

in this approach, more commonly referred to as decision tree design (since a hierarchical classifier can be viewed as a decision tree). The primary reason for this renewed interest is the need to derive more efficient classifiers, particularly with respect to classification speed. More detailed rationale on the motivation for using hierarchical classifiers can be found, for example, in Kanal [6] and Swain and Hauska [7].

Some of the earlier work (e.g., Ganapathy and Rajaraman [8], Hartmann et al. [9]) deals with the conversion of decision tables to decision trees. In particular, Moret [10] provides a very comprehensive survey of this area. However, decision table conversion in some respects is a subset of the more general problem of tree design from a table of arbitrary sample data, as outlined by Chou [11]. In decision table conversion it is essentially assumed that there is no noise in the problem, i.e., that the classes are completely separable using the given features and that the sample data completely exhausts the possible inputs. While this is true for decision table problems such as automated software design, it is not generally true in typical pattern recognition problems. We will focus primarily on the more general case with noise, although all results are equally valid for the special case of zero noise.

Furthermore we will deal with a particular approach to this more general problem of probabilistic decision tree design, namely using the average mutual information between the features and the classes to design the tree. The basic principle of this approach revolves around choosing the "best" feature at any node in the tree (or equivalently at any stage in the sequential decision process), *conditioned* on which features were chosen previously and the outcomes of evaluating those features. From an intuitive viewpoint, any sequence of feature evaluation, or path from the root node to a leaf (i.e., a terminal node), involves only those features which are most relevant towards the classification decision made at the leaf. Clearly, there must be some trade-off between the number of features evaluated and the accuracy of classification. Recently, Chou and Gray [12] have formulated this in terms of rate-distortion theory where the classification error versus average depth trade-off can be directly related to the rate-distortion curve for a given problem. The overall effect one expects by using the average depth approach is for the average number of feature evaluations to be considerably reduced with consequent advantages in time and cost. Among the successful applications of the mutual information approach to decision-tree design reported already are medical diagnosis [13], alpha-numeric character recognition [14,

15], Chinese character recognition [16] and natural-language understanding [17]. In addition, there have been several successful applications of tree-classifiers designed manually without using information-theoretic concepts, e.g., in aerial image analysis [18] and speech recognition [19].

Quinlan proposed a tree design algorithm called ID3 based on the mutual information criterion [20, 21], which has since been widely applied and developed in various forms, particularly as an expert system development tool [22, 23, 24, 25] (although we shall see later in Chapter 2 that decision trees are too simple to merit the title "expert system"). Michie recently reported [26] on several diverse applications of the ID3 tree design algorithm, including Dow Jones forecasting, space shuttle engine analysis (Rockwell International), and the design of offshore gas-oil separation systems (British Petroleum). Other applications include severe thunderstorm forecasting for the National Severe Storms Forecasting Center (NSSFC) [27], troubleshooting minicomputer systems for Texas Instruments [28], and an automated diagnosis assistant for thyroid diseases [29].

Within the domain of the mutual information approach, there exist several variations on the same principle, e.g., Breiman et al. have developed extensive algorithms based on pruning large or complete trees back to much smaller trees [13]. In this thesis, we will deal with algorithms which are strictly *top-down* where the tree is essentially derived in a single iteration. In addition, we will define our optimisation criteria as being the simultaneous minimisation of the misclassification rate and the average depth of the tree. In fact, we will focus almost exclusively on the trade-off between these two measures. More general measures of tree optimality can be defined which incorporate costs, but we choose the error and average depth for two reasons. First, costs and other criteria (e.g. number of leaves (storage), longest path, etc.) are often very domain-dependent, and as such their investigation may not give much general insight. The second, and more important,reason is that the error rate and average depth are the two "classic" performance parameters in decision tree analysis and, generally speaking, results based on these parameters can form the basis for results for more complicated performance criteria.

The primary motivation for the decision tree work in this thesis stems from the lack of quantitative, theoretical results for algorithms which generally fall under the heading of mutual information. As witnessed by the plethora of applications of algorithms in this

category, everyone agrees in practice that the average mutual information criterion is a good splitting criterion to use in tree design. But there are little or no results of a theoretical nature to back this claim. Our goal will be to develop a more complete understanding of the algorithm. In particular, we will be interpreting the problem of tree design as a communications problem, developing analogies with prefix coding and rate-distortion theory, and identifying the basic limitations and problems of this approach.

1.1.2 The mutual information algorithm for tree design

In the introduction we discussed very vaguely the basic algorithm. Let us now be somewhat more specific. Consider that one is given a table of data as in Figure 1.1.

Samples	*Attributes* A_1 A_2 ..... A_N			*Classification label*
1	0	35	red	c_4
2	1	16	blue	c_2
3	1	42	green	c_{11}
⋮	⋮	⋮	⋮	⋮
M	0	32	red	c_4

Figure 1.1: A typical table of sample data

The table consists of M samples drawn from the population at large. Each sample is described in terms of N feature values (or *attribute* values) and a class label. The class

labels are letters from the alphabet of the discrete random variable **C**. For example, the samples could be plants described in terms of height, number of flowers, etc., or medical case-histories in terms of presence or absence of certain symptoms. It is important to realise that the table is probabilistic rather than deterministic, i.e., there may be "overlap" in the class-feature conditional probability distributions.

Note that the problem of feature selection is not dealt with here. Rather, we take whatever features are given for the problem and do the best we can. Feature selection is a different problem, but nevertheless directly related. Indeed, if all known features are given then the tree design algorithm will *automatically* select the most relevant features and ignore the irrelevant ones. This is an additional advantage of the mutual information tree design approach over conventional classifiers, since it yields valuable information on the relative importance of the various features. In applications such as medical diagnosis this can be quite useful [13]. We implicitly assume that the sample size is sufficiently large to yield reliable estimates of the class distribution conditioned on the values of the N features (we will return to this topic later). In this manner all probabilities are assumed to be estimated directly from the data. We also assume that the feature values have been quantised, i.e., that the sample data is discrete-valued.

Let **C** be a discrete random variable, henceforth referred to as the class variable. The finite set of class labels $\{c_1, c_2, \ldots, c_K\}$ comprise the letters of the alphabet of **C**. The probability that **C** assumes the value c_j is denoted as $p(\mathbf{C} = c_j)$ and $\sum_{j=1}^{K} p(\mathbf{C} = c_j) = 1$. We will adopt the convention that $p(c_j) = p(\mathbf{C} = c_j)$.

In addition we have N features or attributes. Each feature $\mathbf{A_i}$, $1 \leq i \leq N$, is also defined as a discrete random variable, i.e., the i-th feature or variable has an alphabet $\{a_1^i, a_2^i, \ldots, a_{n_i}^i\}$, where n_i is the cardinality of the alphabet of $\mathbf{A_i}$. Effectively we can assume that n_i is some small integer. The probability of the event $\mathbf{A_i} = a_j^i$, $1 \leq j \leq n_i$ is denoted as $p(\mathbf{A_i} = a_j^i)$, where $\sum_{j=1}^{n_i} p(\mathbf{A_i} = a_j^i) = 1$, $1 \leq i \leq N$.

Now we consider the algorithm itself. Essentially, it deals with just one tree-node at a time. The initial node (root node) consists of the original table of data samples, i.e., unconditioned on any feature values. Subsequent nodes result from evaluating certain features and obtaining sub-tables conditioned on the outcome of all prior evaluations. The

```
tree()
        begin
        if(node-list is not empty)
                { node = node-list[topnode]
                  if(leaf(node))
                        { increment leaf-list
                          tree() }
                else
                        { A_k = maximum information attribute
                          for(each value a_i of A_k)
                                { sort(sample data table, a_i)
                                  increment(node-list) }
                        }
                }
        else return()
        end
```

Figure 1.2: Pseudocode description of the basic algorithm.

algorithm continues to process nodes until no candidate nodes remain, i.e., the tree has been grown and all leaves and internal nodes defined. The algorithm may be implemented recursively using a stack to store unprocessed nodes. New nodes which are not leaves are pushed onto the stack and "popped" off later to either yield more new nodes (child nodes) or be declared leaves. The algorithm is defined in pseudocode in Figure 1.2.

Apart from the bookkeeping aspects, two functions in the algorithm remain to be defined, namely the "leaf-or-not" and "max-information-feature" functions. It is these two functions which characterise any *top-down* tree derivation algorithm, i.e.,

(1) determine if the node is a leaf, and

(2) if it is not a leaf then determine the feature which yields the most information at that node.

Later we shall deal with the first problem which concerns finding appropriate termination rules. Initially, however, we will focus on the second problem, namely which feature yields the most information. The criterion can be stated quite simply, but we shall be more interested in what *implications* this approach has for the tree as an *efficient* classifier. We can state the criterion in terms of our previously defined notation. Choose the feature A_k

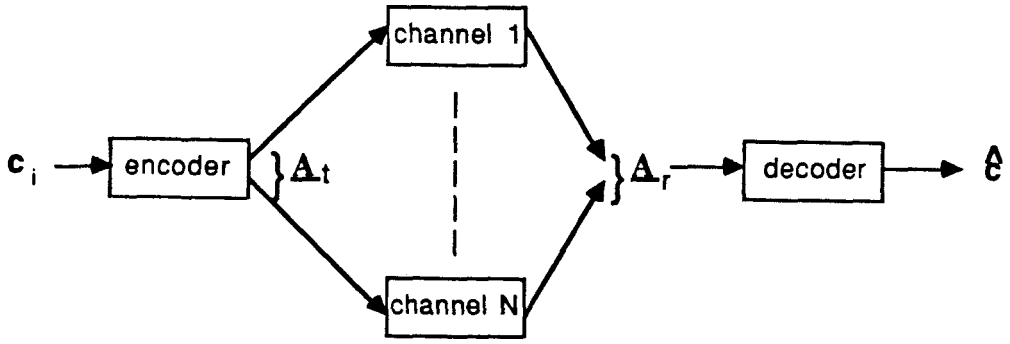


Figure 1.3: Communications channel analogy

such that

$$I(\mathbf{C};\mathbf{A_k}) \geq I(\mathbf{C};\mathbf{A_i}) \qquad 1 \leq i \leq N,\ i \neq k$$

or equivalently,

$$H(\mathbf{C}|\mathbf{A_k}) \leq H(\mathbf{C}|\mathbf{A_i}) \qquad 1 \leq i \leq N,\ i \neq k\ , \tag{1-1}$$

where $H(.)$ and $I(.)$ are the well-known entropy and average mutual information functions, respectively. This is essentially the same as the criterion originally proposed by Lewis in 1962 [4]. In his paper Lewis justifies theoretically why the mutual information criterion is appropriate for feature selection. The interested reader should refer to the original.

We now introduce an interesting analogy with a communications problem. The transmitted information is c_i, the unknown class (from the receiver's viewpoint), a member of what one might term the class alphabet. What is being transmitted is essentially a coded version of this "letter" c_i, namely $\underline{A_t}$ the transmitted feature vector. This feature vector (or message sequence) contains N components and each component can assume n_i different elements in each alphabet. Consider that the features are transmitted over N different discrete memoryless noisy channels, one for each feature component and its alphabet. Figure 1.3 shows the overall picture. For example feature a_1 could be binary valued ($n_1 = 2$) and so channel 1 could be a binary symmetric channel. These N noisy channels are analogous to the feature measurement process and essentially represent a model for the combined ef-

fects of overlap in the joint class-feature distributions, transducer noise, quantisation noise, etc.

The receiver's problem is to compute $\hat{c}$ as a best estimate for whichever c_i was transmitted. This is equivalent to decoding the received feature vector $\underline{A_r}$.

Clearly then, having a set of training samples available in the classification problem is analogous to having a number of test transmissions with side-information available, namely the true value of $\hat{c}$. The problem is to design a decoder which is optimal or near-optimal in some sense, given what one has inferred from the test transmissions. Traditionally, the approach is to use all of the available channels (i.e., all of the features) in the decoding algorithm. But perhaps some channels are costly to operate, just like some features are difficult to measure, e.g., if one of the channels is an expensive satellite link. The hierarchical approach here is to devise a decoding algorithm which uses the channels sequentially and terminates with an estimate of $\hat{c}$, using on average somewhat fewer than N channels. The mutual information extension of this is simply to sequentially choose the channels which yield the most average mutual information between $\underline{A_r}$ and c_i.

1.1.3 Tree derivation as a form of prefix coding

As outlined in the previous section the tree algorithm can be reduced to a recursive procedure which only deals with nodes. Each node in the designed tree has an associated test or feature evaluation. In our notation this feature is a random variable. We adopt the convention (for notational convenience) of using $node_j$ to represent the feature variable associated with the j-th node, where j is an index over the internal nodes. In other words, wherever the $node_j$ symbol is used it represents the random variable associated with that node. Hence, we can think of the node itself as a random variable whose outcomes are members of the associated feature alphabet. In a similar manner we define the discrete random variable **T** to be a function of the internal nodes. Henceforth, we refer to **T** as the tree. The alphabet of **T** consists of all the possible paths (or sequences of feature values) through the designed tree. Since these paths are disjoint and the sum of their probabilities is 1, **T** is indeed a random variable.

We define the average mutual information which the tree yields about the classes, $I(\mathbf{C};\mathbf{T})$, as

$$I(\mathbf{C};\mathbf{T}) = \sum_{j \in S} p(node_j) I(\mathbf{C}; node_j) \ , \tag{1-2}$$

where S is the set of internal nodes, $p(node_j)$ is the probability that one traverses a particular internal node (j is an arbitrary index) in the tree and $I(\mathbf{C}; node_j)$ is the information which the node yields about $\mathbf{C}$, i.e., by traversing a node one evaluates a feature and descends a branch to a child node conditioned on the outcome of the feature evaluation. The average mutual information gained is the average information one receives about the classes when one evaluates this feature. Note that to calculate $I(\mathbf{C};\mathbf{T})$ in Equation (1-2) we need a specific designed tree, i.e., $I(\mathbf{C};\mathbf{T})$ depends on the sample data and the algorithm used. A natural question to ask is "what exactly is $I(\mathbf{C}; node_j)$?"

Theorem 1.1:

The average information gained by evaluating a feature at a given node is simply the entropy of the probabilities of the branches leaving that node.

$$\begin{aligned} I(\mathbf{C}; node_j) &= H_m(q_1, q_2, \ldots, q_m) \\ &= \sum_{i=1}^{m} q_i \log(\frac{1}{q_i}) \ , \end{aligned} \tag{1-3}$$

where m is the number of branches at the node, q_i $(1 \leq i \leq m)$ is the probability of descending the i-th branch, and ϵ the noise level $= 0$. The proof is given in Appendix 1.

We will define ϵ, the noise level, shortly. It suffices to say at this point that $\epsilon = 0$ corresponds to completely separable classes in terms of the features given.

The result can be worded as follows: the information we gain from traversing a node in the tree is simply equal to the m-ary entropy of the branch probabilities emanating from that node. As a numerical example if $m = 2$ and $q_1 = 0.4$ at some $node_j$, then by Theorem 1.1, $I(\mathbf{C}; node_j) = H_2(0.4) = 0.971$ bits.

We can state the following corollary.

Corollary to Theorem 1.1:

For an optimal feature set the top down or "greedy" algorithm for designing trees using mutual information is directly equivalent to prefix coding of a certain type, namely a form

of Shannon-Fano prefix coding.

Proof:

From Theorem 1.1, we see that the tree-design algorithm tries to maximise the quantity $H(p_1, \ldots, p_m)$. This is equivalent to determining the m-ary partition of the component p_i's such that the p_i are as equal as possible. This is the same criterion as used in a form of Shannon-Fano prefix coding [30], and since trees are directly equivalent to prefix codes, [31] then the tree-design algorithm and the prefix coding algorithm are in fact the same. It should be noted, however, that the tree-design is really equivalent to "constrained" prefix coding in a practical sense, since there is no guarantee that the features exist to perform the appropriate partitions. With actual prefix coding no such constraints exist. Hence, the corollary is conditioned on the fact that we have an *optimal* feature set, i.e., at each internal node in the tree we have a feature which defines the optimal split or partition of the classes. This concludes the proof.

An immediate consequence of this result is the fact that we have proven that the top down tree-design algorithm using mutual information is necessarily sub-optimal. This is because of the equivalence to prefix coding where the optimal code derivation algorithm is that of Huffman [32]. We are, of course, interpreting optimal here in the sense of minimum average tree length which is consistent with our overall criterion of optimality when the noise, and consequently the misclassification rate, equal zero.

So if the top-down "greedy" algorithm is sub-optimal, should we abandon it and search for the optimal one? Most probably not, for a variety of reasons, not least the fact that optimal tree-derivation has been shown to be NP-complete [33]. In addition, it is well-known that Shannon-Fano prefix coding yields near-optimal length codes in practice. One suspects then that this simple algorithm may behave quite well from a practical point of view. Rather than viewing the proof of sub-optimality of the algorithm as a negative result, one should interpret its equivalence to Shannon-Fano coding as a positive argument in its favour. The only weakness in the argument is the lack of quantitative results available on how "near-optimal" this prefix coding scheme really is — although it is known to work well in practice one would like to quantify this. We will work towards establishing some results in this regard.

Theorem 1.1 was derived on the unrealistic assumption of zero noise. The reason for doing this was to clarify the basic equivalence to prefix coding. Indeed, the following theorem, Theorem 1.2, includes the result of Theorem 1.1 as a special case, but we have chosen to separate the two results for the purposes of clarity.

Theorem 1.2:

$$I(\mathbf{C};node_j) = H_m(q_1,q_2,\ldots,q_m) - H_m(1-\epsilon,\frac{\epsilon}{m-1},\ldots,\frac{\epsilon}{m-1})$$
$$= \sum_{i=1}^{m} q_i \log(\frac{1}{q_i}) - (1-\epsilon)\log(\frac{1}{1-\epsilon}) - \epsilon\log(\frac{m-1}{\epsilon}) \ , \qquad (1\text{–}4)$$

where m is the number of branches at the node, q_i $(1 \leq i \leq m)$ is the probability of descending the i-th branch, as before. Now ϵ is the probability that one will not descend the "correct" branch and $\frac{\epsilon}{m-1}$ is the probability that one descends any particular one of the other $m-1$ branches. In this sense, ϵ is a noise parameter for an m-ary symmetric channel, where the noisy channel represents the partitioning of the classes at the node using a particular feature. For a proof of this result using a noisy channel analogy see Appendix 1. From the proof we note that this is simply the information equation for a discrete memoryless channel, so that the noise term can be replaced by a general equivocation term $H(\mathbf{Q}|\mathbf{C})$ for any noise chracteristics, symmetric or not, where $\mathbf{Q}$ is the m-ary partition random variable as defined in the Appendix. Let us say that $m = 2$ and $q_1 = 0.4$ as before, and we let $\epsilon = 0.1$. From Equation (1–4) we then find that $I(\mathbf{C};node_j) = H_2(0.4) - H_2(0.1) = 0.502$ bits. In other words, the introduction of a noise level of $\epsilon = 0.1$ resulted in a loss of information of 0.469 bits.

Here we see that the noise places a fundamental limit on the amount of information a node can yield, i.e.,

$$I(\mathbf{C};node_j) \ \leq \ \log_2(m) - \ (1-\epsilon)\log(1/(1-\epsilon)) \ - \ \epsilon\log(m-1/(\epsilon)) \ . \qquad (1\text{–}5)$$

From the original equation it might seem that $I(\mathbf{C};node_j)$ could be negative if the noise were large enough. Of course, this does not happen since the q_i's themselves are noise-dependent and with complete randomness they must in fact be equal, giving the $\log_2(m)$ term for the first expression on the right-hand side. We see later that this limit has direct implications for using threshold-type termination rules in the tree algorithm.

A word on the noise term ϵ is in order at this point. We can think of ϵ as the combined effect of class-feature conditional density overlap, quantisation noise, measurement noise, etc. It represents the quantitative, cumulative effect of these noise sources, and the degree to which the "true" classes as they exist in nature have been obscured by our observation mechanisms. It must be acknowledged that defining a single, feature-independent, noise term constitutes a rather simple model. Yet, as we shall shortly see, it buys us both the ability to model the problem theoretically and implement tree-design experiments where we can control the noise directly.

1.1.4 Bounds on the average mutual information available at a node

We have seen that the average information to be gained from a node is equal to $H_m(q_1, q_2, \ldots, q_m)$, where q_i is the probability of descending to the i-th child given that one is at the parent node. Clearly, this entropy term attains a maximum when all q_i are equal to $\frac{1}{m}$, i.e., when the branches are perfectly balanced in terms of branch weights. But on average, given an arbitrary class probability distribution at the node, how well can one do in terms of maximising the information?

Consider a binary node, i.e., a node where one wishes to partition the members of the class alphabet into two disjoint subsets. Or to look at it from another angle, how much information can one get by asking a purely binary question? Let us define p_{max} as the maximum probability component in the discrete distribution of the class random variable $\mathbf{C}$. Then we have the following theorem.

Theorem 1.3:

Given a discrete random variable $\mathbf{C}$ with K possible outcomes, one can always define a partition of the outcomes of $\mathbf{C}$ into 2 disjoint subsets such that

$$I(\mathbf{C};\mathbf{Q}) \;\geq\; H_2(\,\max\{p_{max}, \frac{1}{3}\})\,, \tag{1–6}$$

where $I(\mathbf{C};\mathbf{Q})$ is the average mutual information between the class variable $\mathbf{C}$ and the optimal partition variable $\mathbf{Q}$. The optimal partition variable $\mathbf{Q}$ is in turn defined as the

binary random variable whose two possible outcomes consist of the optimal partitioned subsets of the alphabet of **C**, i.e., the two branches. The optimal split will be that for which the probabilities of the components of **Q** are closest to 0.5. The proof of this result is given in Appendix 2.

We plot this bound as a function of p_{max} in Figure 4. This leads to the following corollary.

Corollary to Theorem 1.3:

$$\text{If } p_{max} < \frac{2}{3} \text{ , then } I(\mathbf{C};\mathbf{Q}) > H_2(\frac{1}{3}) = 0.918 \text{ bits .} \tag{1–7}$$

The proof follows directly from Theorem 1.3.

Theorem 1.3 and its corollary yield a surprising result. For example, the corollary states that if the maximum probability component is less than $\frac{2}{3}$, then one can always define a partition or question which yields 0.918 bits of information. This is quite large when one considers that the upper bound is, of course, 1 bit, so that even under the most adverse conditions we lose only 0.082 bits. For $p_{max} \leq \frac{2}{3}$ the loss of information is even smaller. Already one can begin to see that "top-down" node-splitting may well be near-optimal under certain conditions. Later, we shall use this theorem to show why Shannon-Fano prefix coding and, more relevantly, *top-down* tree design, almost always yield near optimal results.

Consider first, though, the question of how restrictive the assumption that $p_{max} < p$ may be, i.e., how likely it is that an arbitrary distribution will have p_{max} less than some value p, say $\frac{2}{3}$, $\frac{9}{10}$ or some such nunber. Let us assume that no constraints have been put on the distribution in a Maxwell-Boltzmann statistical type of argument.

Theorem 1.4:

$$\text{prob}\{p_{max} > p\} = k(1-p)^{k-1} \text{ ,} \tag{1–8}$$

where $0 < p_{max} \leq 1$, $0.5 < p \leq 1$, and k is the number of components of the distribution, e.g., $k = K$ for the class variable **C**. The proof is given in Appendix 3 using a combinatorial argument.

Consider a numerical example where $k = 10$ and $p = 0.9$. From Equation (1–7) we get $\text{prob}\{p_{max} > p\} = 10.(1-0.9)^{10-1} = 10^{-8}$.

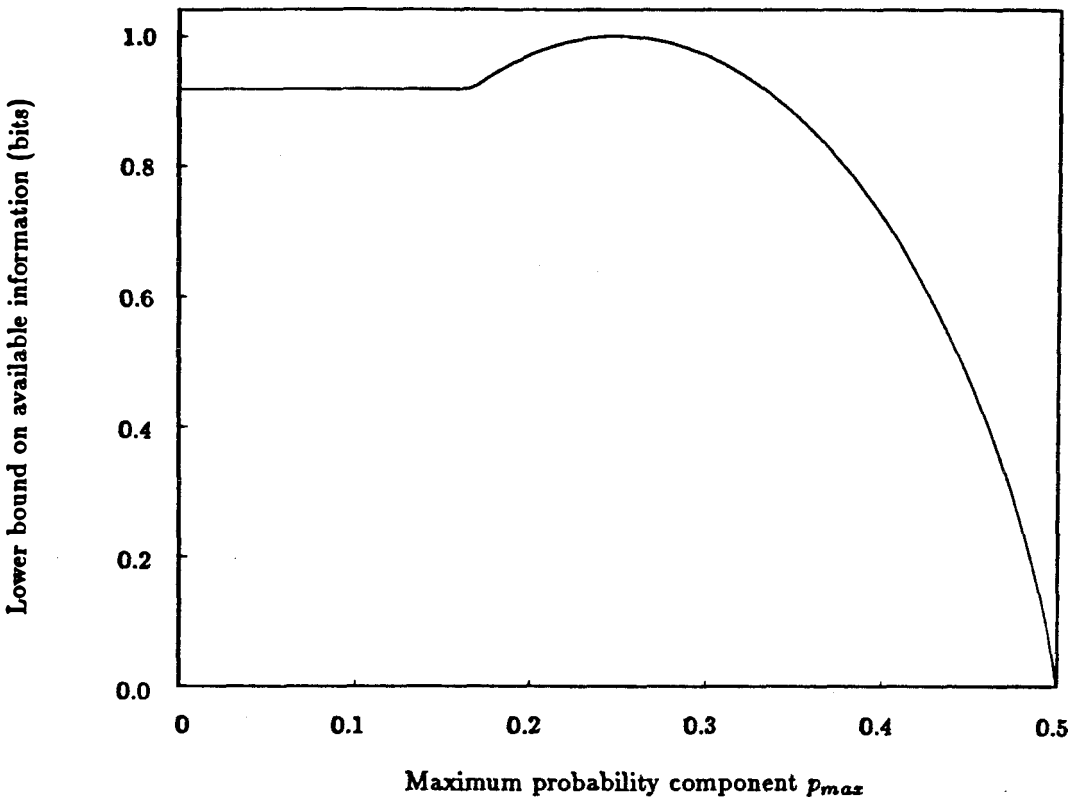


Figure 1.4: Lower bound on the most average information available by asking a binary question, as a function of p_{max}.

Clearly, the probability depends primarily on the number of classes. As k, the number of classes, increases, the probability tends to zero. Of course, just as in solid-state physics, the assumption that all the "states" or distributions are equally likely may be an over-simplification. Nevertheless, the result is a positive indication that as the number of classes increases, "almost all" distributions have no components greater than some p, i.e., for any arbitrary δ there exists a p, $0.5 \leq p < 1$, such that

$$\text{prob}\{p_{max} > p\} \;=\; k(1-p)^{k-1} < \delta \;. \tag{1–9}$$

Returning to the main topic of interest, we now extend the result of Theorem 1.3 to account for any noise which may be present at the node. Consider a noise level of ϵ at the node as previously defined.

Theorem 1.3 extended:

$$I(\mathbf{C};\mathbf{Q}) \;\geq\; H_2(p+\epsilon(1-2p)) - H_2(\epsilon) \;\;, \tag{1–10}$$

where $p = \max\{p_{max}, \frac{1}{3}\}$. The proof is given in Appendix 2.

The consequences of this result can be seen in Figures 1.5 and 1.6. Figure 1.5 relates to $p_{\max} = 0.9$, while Figure 1.6 is for $p_{\max} = \frac{2}{3}$. The upper bound in each graph is $1 - H_2(\epsilon)$, the maximum amount of information available for a noise level of ϵ. The lower bound is the minimum amount of information which can be achieved by the optimal partition, *given* that the maximum probability component is less than some p_{max}, i.e., constraining the maximum probability to be less than $\frac{2}{3}$ is stronger than constraining it to be less than 0.9, which explains why the bound is much tighter in the former case. The noise level ϵ is only allowed to 0.5 since beyond that point the curve is symmetric. This is consistent with the idea of communication channel type noise. The extended result of Theorem 1.3 gives a direct quantitative measure of the decrease in node information due to noise.

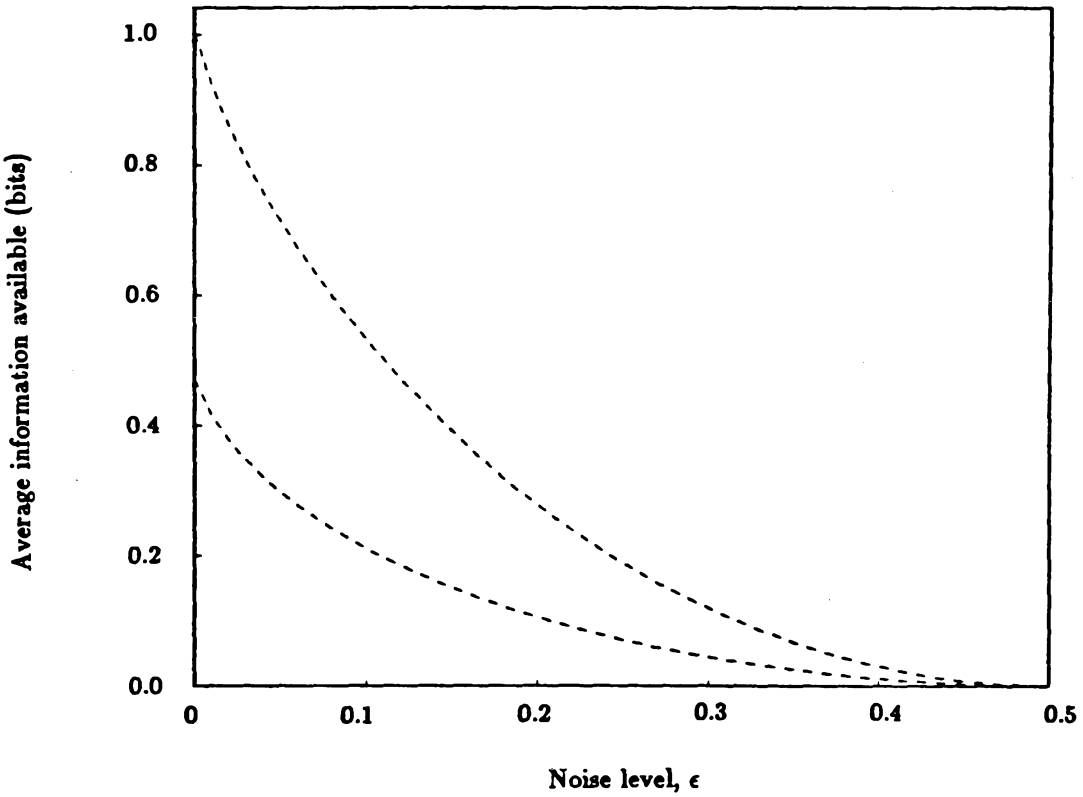


Figure 1.5: Upper and lower bounds on the decrease in information available as a function of noise (ϵ), with $p_{max} = \frac{9}{10}$

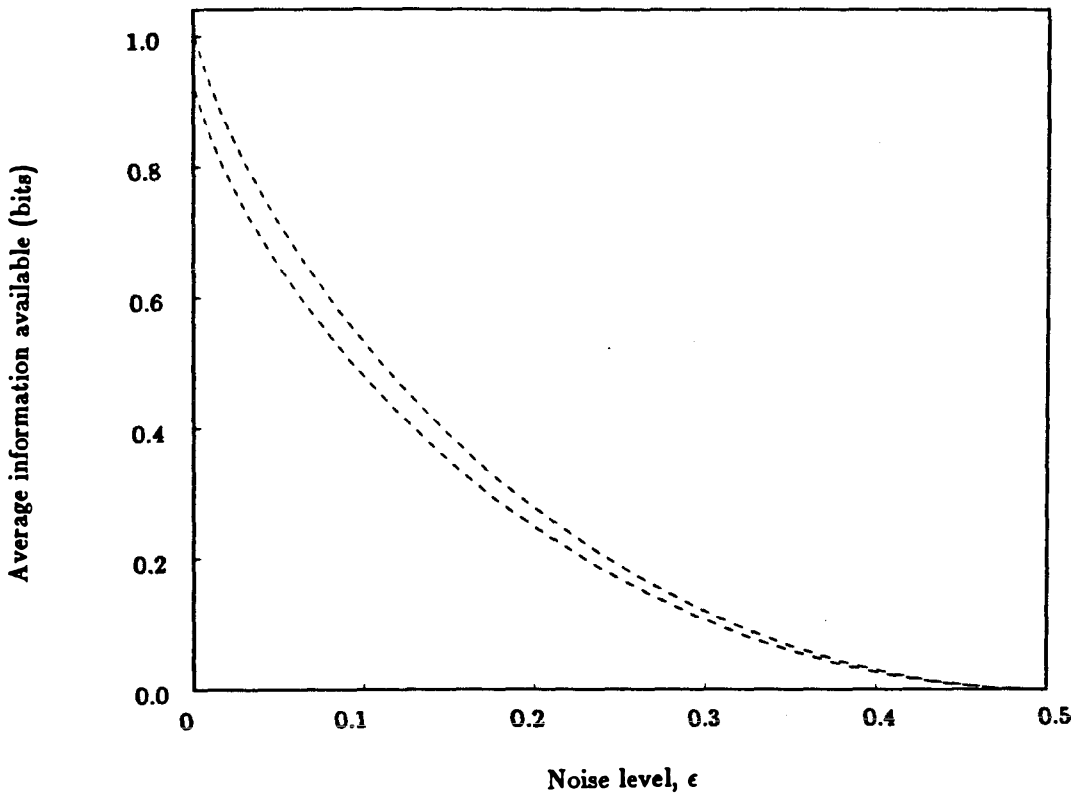


Figure 1.6: Upper and lower bounds on the decrease in information available as a function of noise (ϵ), with $p_{max} = \frac{2}{3}$

We now pose the question of how the tree information, the probability of misclassification, and the tree depth, relate to each other. Clearly the first two parameters can be compared by Fano's inequality (cf. [16] and, originally, [30]). The third parameter, tree depth, will be introduced using some of the results we derived earlier. Let us define $\bar{d}$ to be the average tree depth and P_e to be the average probability of misclassification for the tree. We define p_{min} as the Bayes' risk for the given feature set with uniform losses, i.e., p_{min} is the minimum achievable probability of classification error for a given classification problem and a given feature set. It is an inherent characteristic of a given problem. Its use is tempered by the fact that for many practical problems it cannot be determined exactly. For completeness, let us define p_{min} in terms of the notation we introduced in Section 1.1.2.

$$p_{min} = 1 - \sum_{i=1}^{K} \sum_{j_1=1}^{n_1} \cdots \sum_{j_N=1}^{n_N} \left(\max_i \{ p(a_{j_1}^1, \ldots, a_{j_N}^N | c_i) . p(c_i) \} \right) \quad . \tag{1–11}$$

We will not actually use this formula since, for the type of problem we are investigating (large number of features typically), the joint probabilities are impossible to estimate empirically from the data.

In addition, note that $H(\mathbf{C})$ is the entropy of the class distribution (before using the tree), $I(\mathbf{C};\mathbf{T})$ is the average mutual information which the tree yields about the classes, and $H(\mathbf{C}|\mathbf{T})$ is the average remaining uncertainty about the classes having used the tree.

It is informative to review what type of relationships one might reasonably expect to hold. Consider a situation where there is some noise ϵ, and view this in terms of Chou and Gray's rate-distortion model [12]. In their model they established an equivalence between the communication rate of a channel and the average tree depth, and between the expected distortion from using the channel and the probability of misclassification in using the tree. The tree is equivalent to a variable length code, the act of using the tree and measuring some features being equivalent to transmitting a particular message (class) through a noisy channel. In addition, the original class information has been corrupted by noise prior to coding. Wolf and Ziv [34] established that in such situations the asymptotic limit of the distortion (as the rate is increased) is not zero, but some finite positive distortion which is a function of the noise between the source and the encoder. This asymptotic limit is directly equivalent to the minimum attainable error probability (Bayes' risk for equal losses) we discussed earlier, if we define the distortion measure in terms of classification error. Figure 7 depicts the type of distortion-rate curve one might expect for this model (assuming a

memoryless source), where the original inputs (the original class information) are distorted (represented by an imperfect set of features) prior to being source coded (decision tree) and transmitted (measured by an observer) before decoding (classification at a leaf).

Some general comments are in order at this point. The hierarchical classification approach is based on the principle of finding an "operating point" somewhere above the distortion-rate curve itself at a particular rate where the distortion is "near" the Bayes misclassification rate. If there is substantial redundancy in the feature set with respect to the class distribution (i.e., $H(\mathbf{C})$ is very much less than N, the number of features), then one expects the distortion-rate curve to approach the Bayes misclassification rate for $\bar{d}$ somewhere above $H(\mathbf{C})$ but still "far away" from N. For these problems, in principle, a good tree should be designable with $\bar{d} \ll N$, but having P_e near p_{min}.

Essentially, the notion of $H(\mathbf{C})$ being much less than N is problem dependent and can be taken to hold in a wide variety of well-known classification tasks, such as image classification, where a wealth of features exists, many of which contain similar information, e.g., Conners and Harlow [35] have shown that many measures used in image processing to discriminate between textures contain the same information.

First we will bound the average tree depth. The lower bounds are quite general and involve no assumptions whatsoever, while the upper bounds depend on the type of termination rule used in the tree-design algorithm, among other factors.

Theorem 1.5:

For a binary tree with average depth $\bar{d}$, designed using the *top-down* mutual information algorithm, we have that

$$\text{(a)} \qquad \bar{d} \geq \frac{H(\mathbf{C}) - H(\mathbf{C}|\mathbf{T})}{1 - H_2(\epsilon)} \quad , \tag{1-12}$$

which holds in general, and

$$\text{(b)} \qquad \bar{d} \leq \frac{H(\mathbf{C}) - H(\mathbf{C}|\mathbf{T})}{H_2(p + \epsilon(1-2p)) - H_2(\epsilon)} \, , \tag{1-13}$$

where a threshold termination rule of the form "stop splitting when the maximum probability component at a node is greater than p" is used in the algorithm, and the assumption is made that an optimal feature set exists at each internal node. For a proof of this result see Appendix 4.

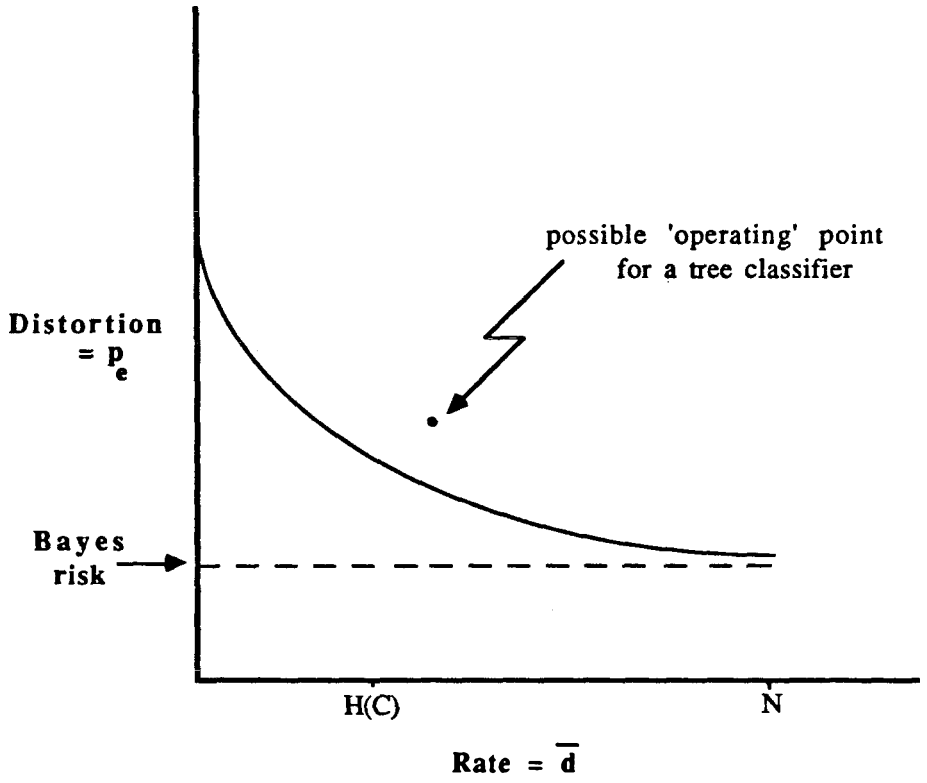


Figure 1.7: Typical distortion-rate characteristic for a hierarchical classification problem.

Example:

Consider a uniformly distributed class distribution with 32 classes, the noise level ϵ is zero, and we use the threshold termination rule of $p_{max} > \frac{2}{3}$. Making the further assumption that the appropriate node-splits can be defined,

$$I(\mathbf{C};\mathbf{T}) \leq \bar{d} \leq \frac{I(\mathbf{C};\mathbf{T})}{0.918} = 1.09 I(\mathbf{C};\mathbf{T}) \ . \tag{1–14}$$

But $I(\mathbf{C};\mathbf{T}) = H(\mathbf{C}) = 5.0$ since $\epsilon = 0$, and so

$$5.0 \leq \bar{d} \leq 5.45.$$

The assumption of the existence of an optimal feature set is certainly satisfied if one were prefix-coding the classes. Hence, the tightness of the bound explains why Shannon-Fano coding almost always seems to work as well as it does, i.e., the average length of the code is lower bounded by $H(\mathbf{C})$ as always and upper bounded by $1.09H(\mathbf{C})$, provided that $p_{max} < \frac{2}{3}$ at each node.

The lower bound on $\bar{d}$ is fixed for *any* tree classifier. The upper bound holds only if we use a particular type of termination rule and with restrictive assumptions. However, it does provide us with a quantitative indication of the behaviour of the algorithm. In particular, we get a bound which relates $\bar{d}$, ϵ and the noise level.

$H(\mathbf{C}|\mathbf{T})$ can be related to P_e as follows:

$$H_2(P_e) + P_e \log(K-1) \geq H(\mathbf{C}|\mathbf{T}) \geq 2P_e \ , \tag{1–15}$$

where the upper bound is due to Fano and the lower bound has been derived by Kovalevsky [36], among others. K is the number of classes.

Consequently, one can derive lower and upper bounds on P_e in terms of $\bar{d}$, $H(\mathbf{C})$ and ϵ, using Equations (1–12), (1–13) and (1–15). We get

$$H_2(P_e) + P_e \log(K-1) \geq H(\mathbf{C}) - \bar{d}(1 - H_2(\epsilon)) \tag{1–16}$$

and

$$P_e \leq \frac{1}{2}\Big(H(\mathbf{C}) - \bar{d}\Big(H_2(p + \epsilon(1-2p)) - H_2(\epsilon)\Big)\Big) \ , \tag{1–17}$$

where again the lower bound is general, but the upper bound is based on the restrictive assumptions of Theorem 1.5. As a numerical example of the upper bound, let $p = \frac{2}{3}$ and $\epsilon = 0.1$. Let us say that the variable $\mathbf{C}$ is uniformly distributed and has an alphabet size K of 8 so that $H(\mathbf{C}) = 3.0$. We then get that $P_e \leq 2.5 - 0.240\bar{d}$.

These bounds are not particularly tight in many instances due to the well known fact that Fano's and Kovalevsky's bounds relating error probability and uncertainty are loose. In addition, it should be noted that $H(\mathbf{C}|\mathbf{T})$ is that value which one has estimated from the available training samples. In this sense it is directly analogous to the resubstitution estimate for the misclassification rate in conventional classifier design problems. It is a *biased* estimate. Given the nature of the algorithm, i.e., minimising the remaining uncertainty, this estimate $H(\mathbf{C}|\mathbf{T})$ may be slightly lower than the true value. It follows then that the bounds derived above are actually for the *estimated* value of $H(\mathbf{C}|\mathbf{T})$ and the *estimated* misclassification rate (or the resubstitution estimate), so that the true probability of misclassification may be higher. Previous bounds derived in the literature [15] based on Fano's inequality have not emphasised this point. Unfortunately, there is no general relationship between the resubstitution estimate and the true misclassification rate, but in practice it has been found that if the sample size is sufficiently large, the true value of P_e is not much greater than the resubstitution estimate [13].

The upper bound in Equation (1–17) is a quantitative affirmation that, at least for the assumptions we made, the error probability of the tree is upper bounded by a decreasing linear function of $\bar{d}$ the average tree depth. It might seem that the upper bound on P_e indicates that by arbitrarily increasing $\bar{d}$ one can reduce P_e to zero. But the distortion-rate model tells us that this cannot be true, or equivalently, no classifier can reduce P_e below the Bayes misclassification rate. The anomaly is resolved by remembering that the upper bound depends on the assumptions made in deriving Theorem 1.5 and, in particular, it is obvious that one cannot keep asking questions *ad infinitum* and continue to receive much information.

However, the lower bound is completely general. Since $H_2(P_e) \leq 1$ then one can write the bound in a looser, but more informative, manner as follows:

$$P_e \geq \frac{H(\mathbf{C}) - \bar{d}(1 - H_2(\epsilon)) - 1}{\log(K-1)}, \qquad (1\text{–}18)$$

e.g., for $\epsilon = 0.1$, $K = 8$ and $H(\mathbf{C}) = 3.0$ as before, we get $P_e \geq 0.712 - 0.189\bar{d}$. This is in fact a weak lower bound on the distortion-rate characteristic. It is interesting that the bound is a linear function of the average depth of the tree, which confirms experimental results that increasing the average depth (given that the $1 - H_2(\epsilon)$ term is not zero, i.e., $\epsilon > 0$) will reduce the misclassification rate at least until the Bayes misclassification rate is approached. The increase in tree depth is necessary to exploit the inherent redundancy in the features. This is completely consistent with our communications analogy where the rate must be increased to reduce distortion.

As a final observation on bounding, we note that for the case of no noise (decision table conversion), Garey and Graham [37] have shown by construction that there exists a particular problem with a particular feature set for which

$$\bar{d} \geq H(\mathbf{C}) \left(\frac{\log_2 K}{10 \log_2 \log_2 K} \right) , \tag{1–19}$$

where K is the number of classes and each class is equally likely. However, in order for this bound to apply, i.e., to overtake the trivial bound of

$$\bar{d} \geq H(\mathbf{C}) , \tag{1–20}$$

we require that

$$K \sim O(10^{17}) . \tag{1–21}$$

Hence, the Garey and Graham bound is not of practical interest, since even for pattern recognition problems with a large number of classes (large-vocabulary speech recognition, Chinese character text classifiers),

$$K \sim O(10^5)$$

at most. Theoretically, however, the bound shows us that as the number of classes becomes infinitely large (with no noise), degenerate worst-case problems exist for which a particularly worst-case feature set yields

$$\bar{d} \geq d^* \left(\frac{\log_2 K}{10 \log_2 \log_2 K} \right) \tag{1–22}$$

where d^* is the average tree depth of the optimal solution for that same feature set found by dynamic programming or some such technique. There are, however, no results indicating the *likelihood* of occurrence of such worst-case situations. Based on the results we

have obtained here, one may conjecture that, in fact, such cases are extremely rare. An interesting direction for future work would be to establish some results in this regard.

1.1.6 Termination rules

We now address the problem of defining a suitable stopping rule for the algorithm, one of the two fundamental criteria for any *top-down* tree design algorithm. This stopping rule is the algorithm's leaf termination criterion or the criterion used to decide whether a node should be a leaf or not. Before looking at the details of this problem, it is worth referring briefly to the general distortion-rate curve of Figure 1.7 once again. The distortion-rate curve is, by definition, a lower bound on the performance of any hierarchical classifier designed for that particular problem. It is reasonable to expect that tree classifiers will exhibit similar characteristics to the bound, i.e., as P_e, the probability of misclassification, is reduced closer and closer to p_{min} (the Bayes misclassification rate), then $\bar{d}$ must be increased accordingly.

A greedy algorithm traces out a series of operating points above the bound. As each internal node is split at each step of the algorithm, $\bar{d}$ increases and P_e decreases. Hence, the purpose of the termination rule in an automated algorithm is to somehow decide a good operating point at which to stop. It is clear from what we have discussed earlier that some types of termination rules will not work well in the presence of noise. Placing an upper bound on P_e and continuing to split nodes until the resubstitution estimate is below this bound is not a good idea, since given a certain amount of noise (which we assume we don't know much about) and, for a given problem, there exists a fundamental lower limit on the misclassification rate as represented by the asymptotic limit of the distortion-rate curve as the rate approaches N, the number of features, as N becomes arbitrarily large. So if P_e is chosen to be less than p_{min}, even a complete tree (i.e., all features evaluated on the path to each leaf) may not satisfy the threshold condition. More generally, this approach is liable to yield very large values for $\bar{d}$ as P_e approaches p_{min}.

Similarly, one could choose to continue splitting until the total average information $I(\mathbf{C};\mathbf{T})$ from the tree had exceeded a certain value or equivalently $H(\mathbf{C}|\mathbf{T})$ had decreased

below a certain value. But this is really equivalent to placing a threshold on P_e, since we have already seen that $H(\mathbf{C}|\mathbf{T})$ is bounded above and below by linear functions of p_{min}. In other words, as well as there being a minimum misclassification rate p_{min} for a given problem with a certain amount of noise, there is a corresponding $H(\mathbf{C}|\mathbf{T})_{min}$ where

$$2p_{min} \leq H(\mathbf{C}|\mathbf{T})_{min} \leq 1 + p_{min}\log(K-1) \; . \tag{1-23}$$

So thresholding on $H(\mathbf{C}|\mathbf{T})$ or $I(\mathbf{C};\mathbf{T})$ will not yield satisfactory results in the presence of noise for the same reason that thresholding on P_e will not work in the same situation.

Yet another approach which potentially looks somewhat more promising is to stop splitting nodes whenever $I(\mathbf{C};\mathbf{Q})$ falls below a certain value, i.e., determine $I(\mathbf{C};node_j)$ at $node_j$ for the best split and if it falls below some threshold parameter t then declare the node to be a leaf. In the presence of noise ($\epsilon \neq 0$), the various results derived earlier in terms of $I(\mathbf{C};node_j)$ and ϵ tell us that if t is too large, one will get a lot of splitting and, hence, very large trees, while if t is too small, very little splitting will occur because of the $H_2(\epsilon)$ term. The resulting tree will have $\bar{d}$ less than $H(\mathbf{C})$ with a resultant high probability of misclassification as can be inferred from the distortion-rate characteristic. In practice this precise phenomenon has been observed by other authors, ([13],[38]) as a result of experimentation with designing trees for a particular problem and increasing the value of the noise level ϵ. In the simulation results presented in the next section, we will not include results for trees designed using threshold rules, since in the presence of noise, such rules were found to be practically unworkable.

We have seen so far that threshold rules by themselves are not sufficient to form noise-independent termination rules. In addition, they have the added drawback of requiring some external intervention in order to supply the threshold levels, which is undesirable in the overall context of *automated* tree design.

Consider the basic problem again. One has a class probability distribution defined at a node, conditioned on the outcomes of feature (node) evaluations along the path from the root to the current node. The question is whether it is worth splitting this node or declaring it to be a leaf, i.e., making a classification decision at that point or evaluating another feature. Form the null hypothesis that the node is in fact worth splitting, i.e., the hypothesis is "another feature should be evaluated before making a classification decision."

There are potentially many ways to test this hypothesis. The fixed threshold rules we have already considered are tests which are insensitive to any noise which may be present. An approach used by Quinlan [38] is to use the Chi-square test to test for the independence of the candidate features (for splitting) and the classes. The reasoning is that only the features which are rejected with a high degree of confidence are subsequently considered for splitting and so the effect of noise is minimised. However, as the number of classes and features increase, the number of samples at the node required to use the Chi-square test becomes large. Since the test is for leaf nodes, then it is precisely in the situation where nodes are deep in the tree and the number of samples at the node is relatively small, that its use is critical to the success of the algorithm. For problems with many classes and features, this test may not be *sufficient* as a termination criterion. It should be noted, however, that for a particular two-class problem, this rule was found to be quite useful in practice [38].

Another approach is to use a statistical confidence interval termination test on p_{max}. The idea is to specify W_α as a parameter of the design algorithm, where W_α is the minimum confidence interval width for p_{max} one is willing to tolerate at the α% confidence level. In Appendix 5, a confidence interval on p_{max} is derived so that if

$$W_\alpha < 2\Delta \tag{1-24}$$

$$= 2\lambda\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \tag{1-25}$$

the algorithm declares the node to be a leaf and terminates that particular path based on the fact that there are not enough data samples available. Equivalently, we can write the condition directly in terms of the number of samples so that termination occurs if

$$n < n_{critical} \ , \tag{1-26}$$

where

$$n_{critical} = \frac{4\lambda^2}{{W_\alpha}^2}\hat{p}(1-\hat{p}) \ . \tag{1-27}$$

For example, with a 99% confidence interval of 0.2 ($W_\alpha = 0.2$), we have

$$n_{critical} = 664\hat{p}(1-\hat{p}) \tag{1-28}$$

$$< 166 \ , \tag{1-29}$$

whereas with less restrictive criteria of a 95% confidence interval of size 0.4,

$$n_{critical} = 96\hat{p}(1-\hat{p}) \tag{1-30}$$

$$< 24 \; . \tag{1-31}$$

We note that this simple test does not take into account any noise which may be present. While we were able to *model* the noise in our earlier theoretical analysis, it is not nearly as clear how to account for noise in a *practical* situation. However, since the test operates as a lower bound, this point does not affect its validity. This statistical test is a *necessary* but *non-sufficient* termination procedure.

Another termination test which we will now consider is called the Delta-entropy rule. There are no external parameters required and no constraints as to the number of samples required for it to work. In addition, it is easy to compute and intuitively appealing from an information-theoretic viewpoint.

The Delta-entropy rule:

$$\text{If } \; H'(\mathbf{C}') \geq H'(\mathbf{C}) \quad \text{and} \quad p_{max} \geq 0.5 \; , \tag{1-32}$$

then declare the node a leaf, where $\mathbf{C}$ is the original class random variable at the node, $\mathbf{C}'$ is the normalised class distribution if p_{max} is deleted from the component set, and $H'(\mathbf{P})$ is the usual entropy function divided by the logarithm to the base 2 of the number of non-zero probability components of $\mathbf{P}$.

There is no fundamental theoretical basis for this rule but it has been found to be very useful in practice. One interpretation is that we consider the hypothesis that the node is a leaf node, i.e., if we continue to split then we will not reduce P_e any further. This is equivalent to the hypothesis that, in principle, any partition (defined by the given attributes) of the classes at the node will *not* effect a decrease in the probability of misclassification, i.e., apart from the class component corresponding to p_{max} the other class components present at the node are due to noise. If $H'(\mathbf{C}') \geq H'(\mathbf{C})$ then this is evidence in favour of accepting this hypothesis. In the absence of any other evidence, the only course of action is to declare the node to be a leaf. Another way of looking at it is that our relative uncertainty about $\mathbf{C}'$ is large enough that there is reason to believe that we should not continue splitting (where "large enough" is measured relative to the uncertainty about $\mathbf{C}$).

For computational purposes the rule can easily be manipulated to yield the equivalent condition for declaring a node to be leaf, i.e.,

$$p_{max} \geq \frac{H_2(p_{max})}{H(\mathrm{C})} - \frac{\log_2(n) - \log_2(n-1)}{\log_2(n-1)}, \qquad (1\text{–}33)$$

where n is the number of non-zero class components at the node. In effect, one has a dynamic threshold on p_{max}. The rule as outlined here is for the case where the class variable is uniformly distributed. The extension to the general case is given in Appendix 6. It should be noted that this rule cannot be used unless $n \geq 3$, where n is the number of classes and that it works best when the noise is evenly distributed among the classes.

No claims of optimality are made for this rule. Rather, we have a useful and practical test for establishing terminal nodes with the top-down tree design algorithm. In particular, the rule requires no externally-supplied parameters and does not necessarily require large data sets to work properly. In the next section experimental results are given which indicate that the Delta-entropy rule works well in practice.

To conclude this section some final observations on the problem of termination tests are in order. Finding general termination rules for a top down tree design algorithm which can be applied to different problems with different noise levels is quite difficult. The fundamental problem lies in the concept of the greedy algorithm, which, by definition, can not "look ahead." The question is whether it is worth continuing to split or not. Clearly, a purely greedy algorithm can never be totally satisfactory in this regard while, on the other hand, all look-ahead strategies must be either sub-optimal or NP-complete. To complicate the problem, there may be varying levels of noise present which cannot be distinguished except by continuing to split the node further. In addition the data set at the node may be too small to permit any confidence in the use of statistical tests. One might conjecture that the exact form of the termination rule to be used is inevitably dictated by the nature of the particular problem at hand, e.g., approaches such as "pruning" [12] (where subtrees are grown and later possibly removed by a "pruning" algorithm) may be more appropriate in applications where a more complicated tree-derivation algorithm is acceptable. Recent results in this area [39] suggest that the pruning algorithm of Breiman et al. [13] is, in fact, optimal in a certain sense. Nonetheless, the problem of growing the *initial* tree from which to prune, using a greedy algorithm, is still a fundamental problem.

For the case of strictly top-down algorithms, we have established from a theoretical point of view that threshold-type rules exhibit deficiencies. Hence, we confirmed earlier experimentally-based conclusions. The Delta-entropy rule is an alternative termination rule to overcome the deficiencies of the other currently known rules.

1.1.7 Experimental results using the mutual information algorithm

Simulations were carried out by designing trees at various noise levels based on the following problem. Consider the 16-segment alpha-numeric display as shown in Figure 1.8.

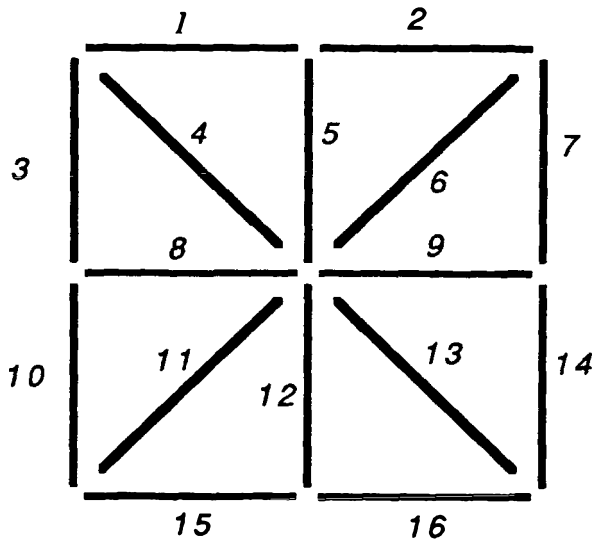


Figure 1.8: 16-segment alpha-numeric display used for simulations with numbered LED segments corresponding to features.

The 16 segments correspond to 16 binary features. The 26 upper-case letters and 10 digits make up the 36 classes. This particular problem was chosen because one can directly control the parameters of the simulated data set (e.g., noise, number of samples) while, in addition, it fits into the category of having a relatively large number of both features

and classes. A more complicated problem with real data (as opposed to simulated data) might yield less insight into tree design *per se* than it would about the problem itself. The noise parameter ϵ has an easily interpretable meaning. Each segment has a probability ϵ of "doing the wrong thing" at the time the features are being measured, i.e., being off when it should be on or being on when it should be off. In fact, this is just a more complicated version of a 7-segment problem which has been simulated by others [12, 13]. The range of values chosen for ϵ during the simulations were 0.0 to 0.1 in increments of 0.01. It is interesting to note the reason for keeping $\epsilon \leq 0.1$. With $\epsilon = 0.1$ we found that it was quite difficult for a human to recognise the characters and the Bayes misclassification rate was estimated at about 0.25. It seemed reasonable to expect then that the area of greatest interest with respect to tree performance would be for $\epsilon \leq 0.1$, since above this figure the data was extremely noisy. The distribution of the alphabetic classes was taken to be that of English text as defined in [40], while the digits were simply assigned probabilities of $\frac{1}{36}$ each. The number of data samples used for each tree design was 5000. Based on an order of magnitude estimate for the total number of samples required, it is shown in Appendix 5 that 5000 is in the correct range. The misclassification rate P_e was estimated by using the tree to classify an independent data set also of size 5000. Using a test set of size 5000 to estimate P_e yields estimates of very high confidence levels for P_e according to [41].

The noise level of the test set and the design set was the same. It has been experimentally observed by both the author and others [38] that using less noisy data than that to be encountered in practice while designing a tree classifier will actually lead to a higher misclassification rate than if the same type of data is used for both design and use. This conclusion can be inferred from the distortion-rate model as shown in Figure 1.9. The situation is equivalent to having an artificially low distortion-rate bound during the *design* while actually *operating* with a bound which is higher up on the misclassification rate axis. Designing the tree with a given algorithm will result in $\bar{d}$ being fixed. P_e in practice will then be higher than the desired P_e or the resubstitution estimate. However, were one to design with data having the true (higher) value for ϵ, one should be able to determine a lower P_e for a correspondingly higher $\bar{d}$. This is what happens in practice, the artificial case yielding trees which are too short and do not exploit the redundancy in the features as the noise increases. On the other hand, if the situation is reversed and one is designing with noisier data than that to be encountered in practice (as might happen if at some later

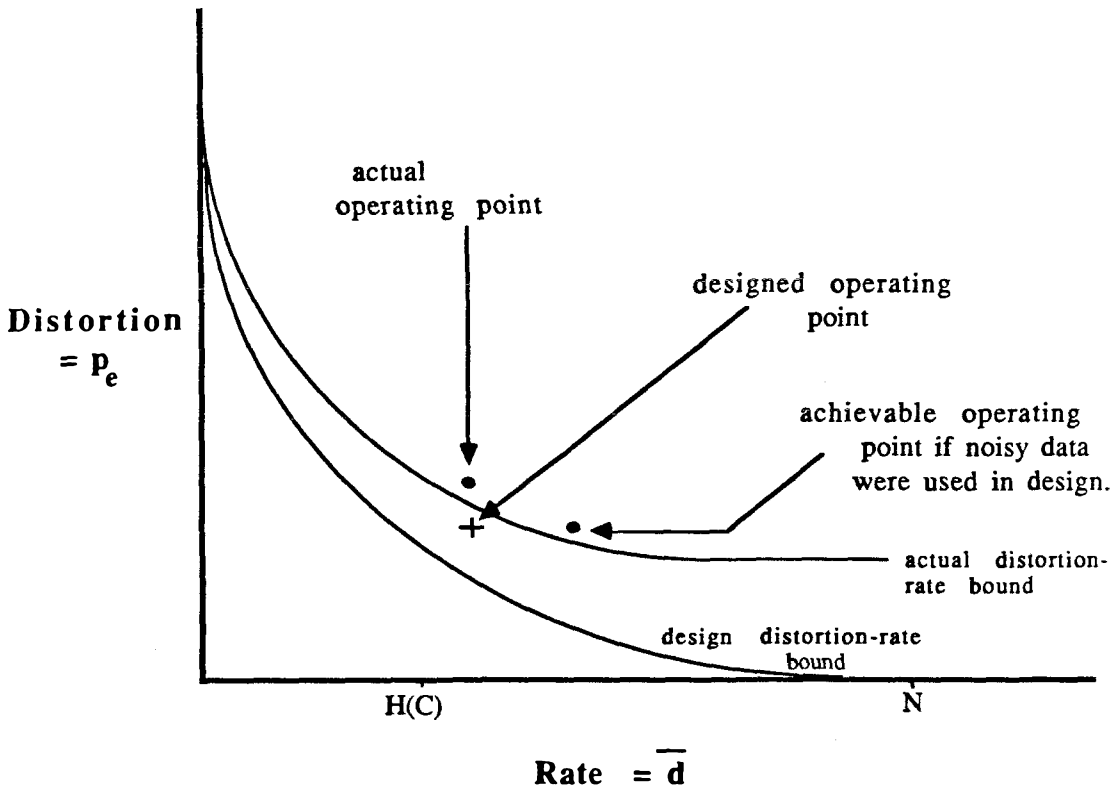


Figure 1.9: Distortion-rate curve showing how good design data can lead to a bad classifier.

point in time one's feature measurements were to become less noisy), one can conjecture that for a designed $\bar{d}$, the actual misclassification rate in practice might well be lower than the resubstitution estimate but only because the $\bar{d}$ is significantly larger than it needs to be.

In order to give an idea of what is involved, Figure 1.10 shows a tree designed for the noiseless case, i.e., $\epsilon = 0$. The leaves (or terminal nodes) are labelled with the class decision at that leaf while the internal nodes are labelled with the feature (segment) to be evaluated at that node. Branches going to the right in the tree correspond to a segment being lit while those to the left mean that the segment is not lit. We see that common letters such as E and T correspond to shorter paths, while less likely letters are on longer paths.

It can easily be shown that for the special case of $\epsilon = 0$, the Delta-entropy rule will always yield a "correct" tree in the sense that there is a 1-1 correspondence between leaves and classes. This property is necessary for any termination rule which is to be used on data where it is not known *a priori* whether $\epsilon = 0$ or not. Threshold rules for example do not exhibit this property in general.

Figure 1.11 shows the increase in average depth plotted against the noise level. Note that for $\epsilon = 0$ the average depth is only slightly greater than the entropy of the class distribution ($\bar{d} = 4.9692$, $H(\mathbf{C}) = 4.7567$) and does not increase significantly above $H(\mathbf{C})$ as the noise level increases, i.e., even for $\epsilon = 0.1$, less than 6 of the 16 possible features need to be evaluated on the average. In fact, we found the Delta-entropy rule to be quite conservative in our simulations, consistently yielding trees with an average depth quite close to the entropy of the class distribution. However, average depth alone is not significant. Figure 1.12 shows the estimated probability of misclassification for the trees of Figure 1.11. On the same graph is a plot of the misclassification rate obtained using the same data with the single nearest neighbour algorithm, i.e., where all 16 features were used. The tree classifier is seen to do very well in comparison with the nearest neighbour technique. In Figure 1.13 we plot a merit function for each algorithm versus the noise level. The merit function was chosen to be $\frac{(1-P_e)H(\mathbf{C})}{\bar{d}}$, where $\bar{d}$ is fixed at 16 for the nearest neighbour (even though effectively it is much greater than this). This merit function can be interpreted as a simple "benefit over cost" ratio where the benefit is $1 - P_e$ and the cost is defined as $\frac{\bar{d}}{H(\mathbf{C})}$. Hence, with $P_e = 0$ and $\bar{d} = H(\mathbf{C})$, the merit function = 1 which is its maximum

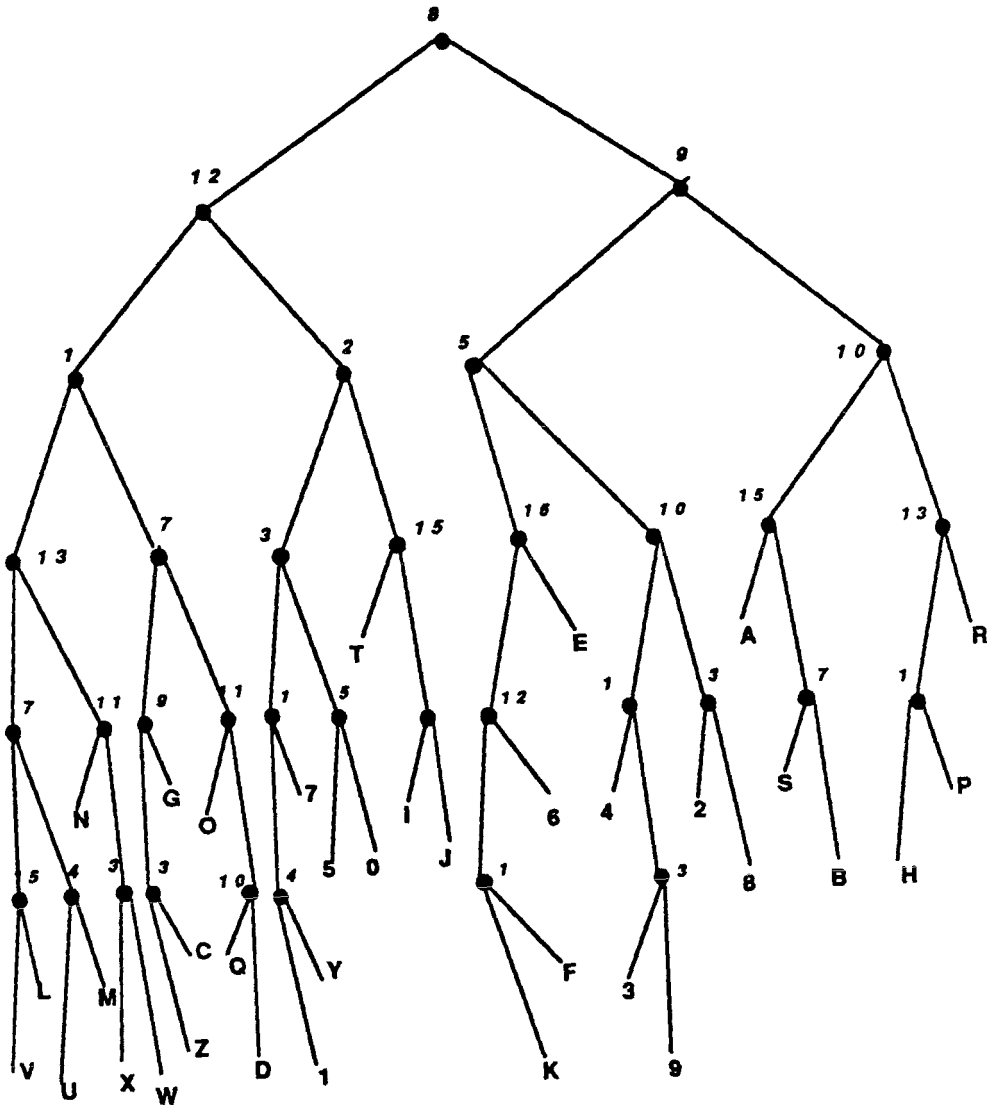


Figure 1.10: A tree designed using the Delta-entropy rule for the alpha-numeric problem.

attainable value. The tree classifier is clearly better under this criterion and indeed would be superior for any criterion which gives the average number of features evaluated at least equal weight with the misclassification rate.

That the tree classifier does well on this problem is a good indication of the fact that 16-segment alphanumeric displays exhibit a lot of redundancy. Similar success can be expected in classification problems where there is considerable redundancy in the feature data. We now consider a more realistic application of the tree-design approach, the problem of local edge detection.

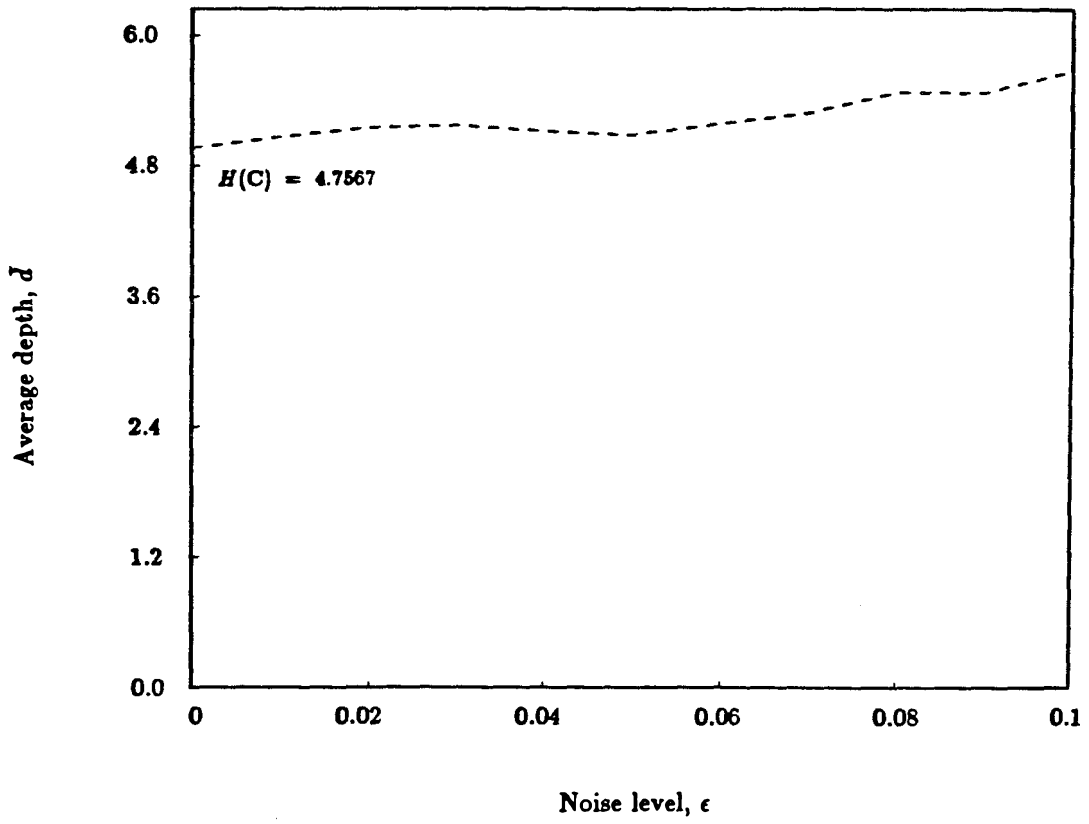


Figure 1.11: Average tree depth verus noise, for the alpha-numeric problem.

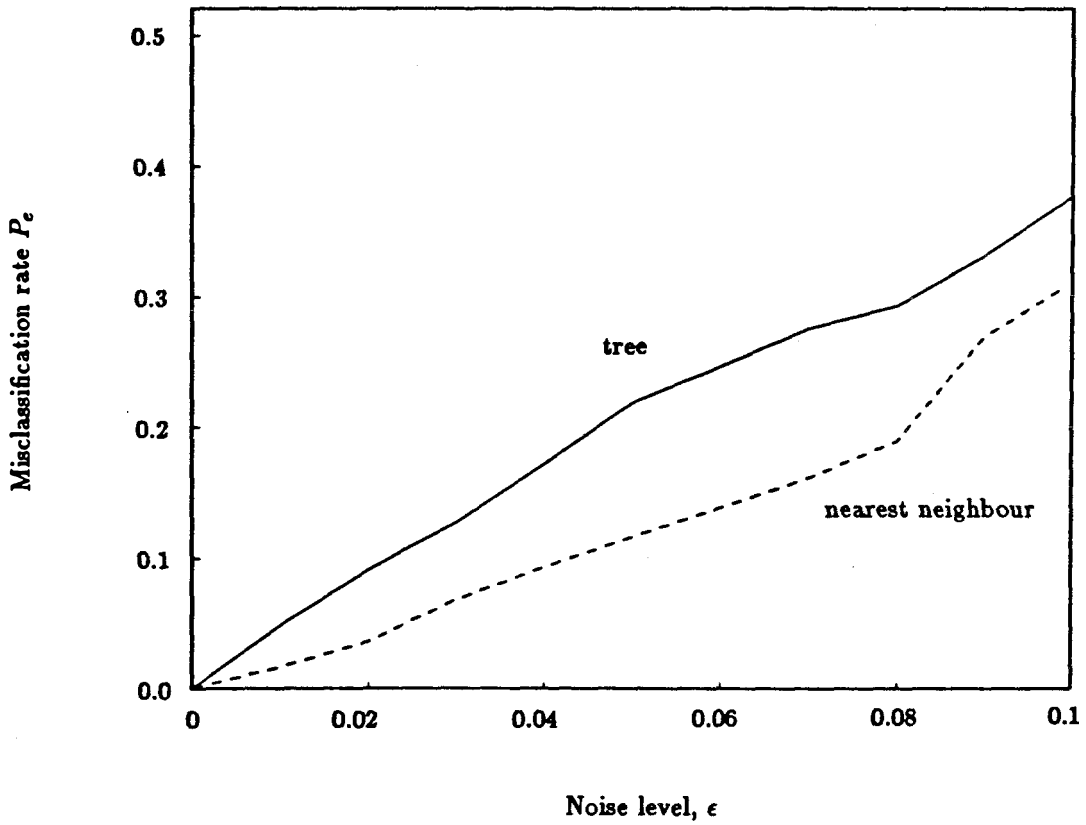


Figure 1.12: Misclassification rates of the tree classifiers and the nearest neighbour classifiers versus noise for, the alpha-numeric problem.

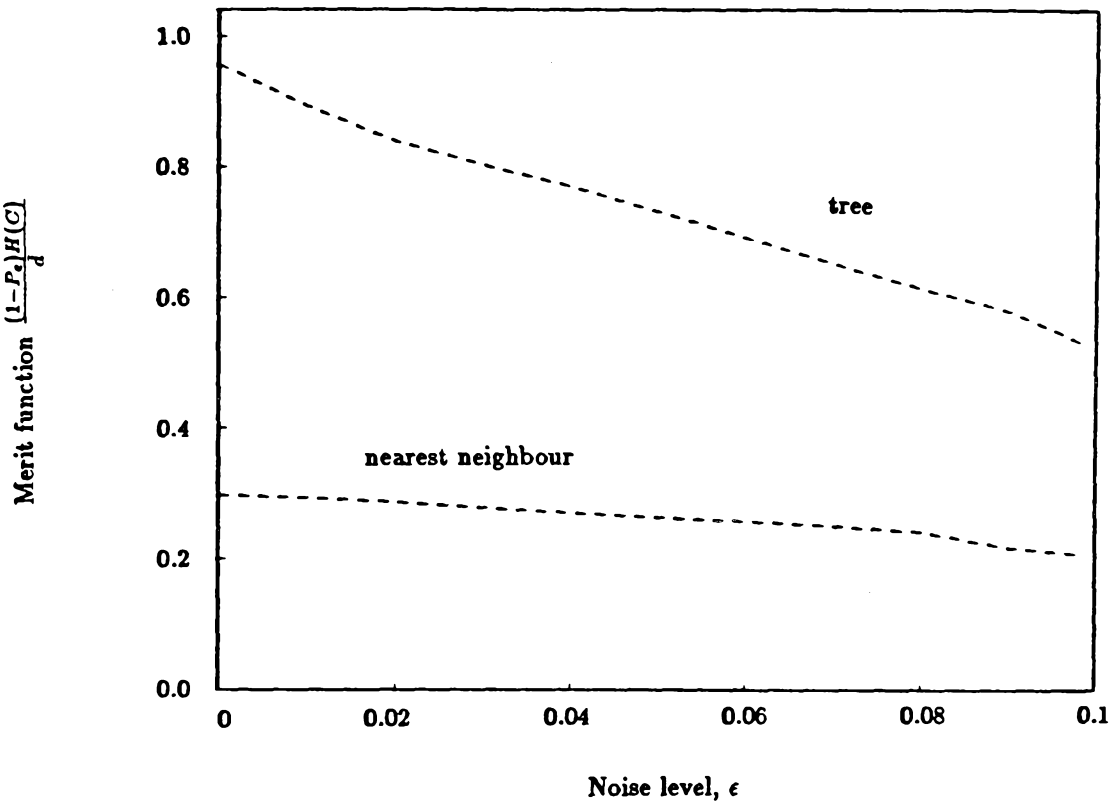


Figure 1.13: Merit function for both the tree classifier and the nearest neighbour classifier versus noise, for the alpha-numeric problem.

1.2 The application of decision trees to edge detection

1.2.1 Current techniques in edge detection

Edge detection is a classic problem in the field of computer vision and image processing. We will use the description "local discontinuity in image luminance" to define an edge. An image is sampled and stored as a finite set of discrete grey values. It is not surprising, then, that most edge detection schemes are essentially digital filters through which the image is passed or convolved. The length of the filter is defined by the size or extent of the window operator used to implement the filter. The filters tend to be non-causal and non-recursive, i.e., finite impulse response. Despite the fact that the window sizes of typical edge-detector filters are relatively small in comparison with the size of the image itself, two-dimensional convolution is still an expensive operation in computational terms, even with improvements in special-purpose hardware [42]. Recent advances in edge detection theory (such as those of Marr and Hildreth [43] and Canny [44]) have led to the derivation of filters which are optimal under certain criteria. Unfortunately, the window sizes of these filters are large enough so that for the present they are very demanding (computationally) for applications such as real-time automated inspection. Instead, the smaller window size operators such as the Sobel [1] continue to be used in the applications environment ([45], [46]). However, even for these smaller sized windows (e.g., 3×3) two-dimensional convolution is still relatively slow. There is considerable motivation to investigate any techniques which can reduce the fundamental computation requirements in edge detection, particularly for real-time vision applications such as robotics [47].

We will investigate the application of decision trees to edge filtering, with the goal of significantly decreasing the computation cost of local linear edge operators (i.e., small window sizes). Essentially, we formulate edge detection as a classification problem where pixels are to be classified as either "edge" or "non-edge." Because the mutual information induction algorithm yields a rule *hierarchy*, it is ideally suited to the problem of deriving faster operators. The important aspect of this experiment will be to illustrate the *principle* of using decision trees in a particular application to effect an improvement in classification performance over existing techniques. We will focus on local, 3×3–sized windows, for simplicity. However, it should be noted that from a practical point of view, the real benefit of this approach is for computationally more intensive tasks, such as larger window sizes.

After defining a general procedure for designing hierarchical edge operators we will proceed to derive such operators on training images and evaluate their performance. For comparison purposes, we use two other operators, namely the aforementioned Sobel operator and the recently introduced dispersion operator [48] based on order statistics.

1.2.2 The Sobel and dispersion edge operators

Let us briefly review these two edge detection schemes. The two masks for the Sobel operator are defined in figure 1.14.

1	2	1
0	0	0
-1	-2	-1

1	0	-1
2	0	-2
1	0	-1

Figure 1.14: The vertical and horizontal Sobel edge masks.

The procedure is to obtain 2 edge-enhanced images by convolving these masks with the original image. Essentially, one mask enhances vertically oriented edges while the other enhances horizontally oriented edges. If we define $EI(i,j)$ to be the grey-scale value of the enhanced image at pixel (i,j) and adopt a numbering scheme which begins at the top left-hand corner (number 1) and proceeds from left-to-right and top-to-bottom to number

9 (defined further ahead in Figure 1.15), then

$$EI_1(i,j) = |\sum_{i=1}^{3}(x_i - x_{i+6})| \tag{1-34}$$

$$EI_2(i,j) = |\sum_{i=1}^{3}(x_{3i-2} - x_{3i})| \ , \tag{1-35}$$

so that

$$EI(i,j) = \sqrt{EI_1(i,j)^2 + EI_2(i,j)^2} \ . \tag{1-36}$$

A suitable threshold t is determined, and a pixel is classified as an edge if

$$EI(i,j) > t \ , \tag{1-37}$$

otherwise it is classified as a non-edge. Abdou and Pratt [49] have shown experimentally that, of the local 3×3 operators, the Sobel performs best over a wide range of images.

The dispersion operator, based on order statistics, was recently introduced by Pitas and Venetsanopoulos [48] and has the advantage of being, in principle, computationally faster than any other known 3×3 operator. To obtain the order statistics, it is first necessary to sort the N pixels in the window in order of increasing magnitude so that

$$x_{(1)} \leq x_{(2)} \leq , \ldots\ldots, \leq x_{(N)} \tag{1-38}$$

where $N = 9$ for a 3×3 window. The range of the random variables $x_1, \ldots, x_N$ is defined as

$$w_{(1)} = x_{(N)} - x_{(1)} \ . \tag{1-39}$$

Quasi-ranges are defined as

$$w_{(i)} = x_{(N+1-i)} - x_{(i)} \quad , \quad 2 \leq i \leq \lfloor \frac{N}{2} \rfloor \ . \tag{1-40}$$

The range and quasi-ranges give an indication of the grey-scale variation within the local area of the pixel and, hence, indicate the possible presence of an edge. However, due to the absence of spatial information they are susceptible to noise, and in particular, to impulse noise, i.e., noise which shows up in the image as random pixels whose grey-scale values have no correlation with nearby pixels. The dispersion operator was defined as an average over the quasi-ranges to combat the effect of noise [48], i.e.,

$$W = \sum_{i=1}^{\lfloor \frac{N}{2} \rfloor} w_i \ . \tag{1-41}$$

The W value for each pixel (i, j) is thresholded to determine if it contains an edge or not. It is shown in [48] that this operator performs almost as well as the Sobel operator over a variety of comparison measures and can be implemented much faster if one uses a parallel sorting scheme.

1.2.3 Formulating edge detection as a decision tree problem

The idea of pixel-based classification into "edge" and "non-edge" classes is easy to visualise. As in any classification problem the classes are described in terms of features. A classifier is designed by taking a sufficiently large sample of pixels in order to estimate the conditional probabilities of the classes given the various features, e.g., the probability that the class is "edge" given that the i-th feature has value j. From the set of training samples relating features and classes we must infer some general rules for classifying pixels based on their feature description.

First the problem of *feature selection* must be addressed, i.e., before we can consider designing a classifier the features must be defined. Remember that our ultimate goal is to reduce the computation cost in defining a local edge detector. Consequently, in order to compete with (for example) the Sobel operator, whatever features we choose must be computationally simple to implement. The term "primitive" provides a convenient description. Window operators of the type we have been discussing can be considered as "composite" binary features in the sense that they are composed of combinations of primitive features. They are binary-valued because the pixels are essentially classified as "edge" or "non-edge," according to whether the composite feature is greater than or less than a certain threshold. For edge detection the difference operator is the most important type of primitive operator. The rationale for this claim can be stated in many different ways, but in simple terms, the presence or absence of an edge will lead to larger or smaller difference values, although the converse may not be true due to the presence of noise. Primitive features can be defined by supplying pairs of arguments to the difference operator.

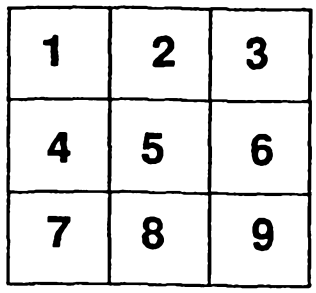


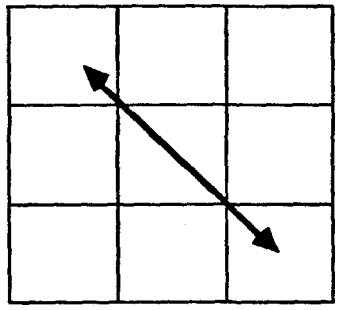
Figure 1.15: Definition of the window pixel notation.

We could, for example, define the following primitive feature,

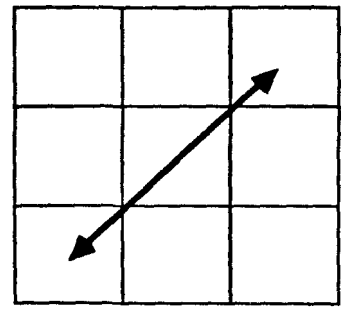
$$\begin{aligned} \text{feature 1} &:= x_1 - x_9 \\ \text{feature 2} &:= x_4 - x_6 \\ \text{feature 3} &:= \max_i\{x_i\} - \min_i\{x_i\} \ , \end{aligned}$$

where the i ranges over the window as described in Figure 1.15 and x_i is the grey level of the i-th pixel. It is easy to see that the Sobel operator can be defined in terms of primitive features like the ones above. Let us use F_i to represent the i-th feature and the random variable f_i to denote the value of F_i. f_i is a discrete random variable, which for the difference-type operators we are discussing here, is restricted in range to the maximum grey level of the image. The problem, of course, is that there is an extremely large set of potential pairwise features to choose from, e.g., we are not necessarily constrained to using pixels within a 3×3 window. As in any classification-type problem, feature selection is almost an art form. However, one of the advantages of the mutual information algorithm is to delete irrelevant features from consideration, making the feature selection process easier.

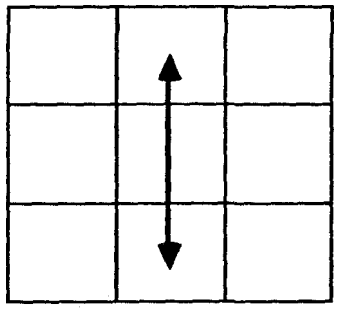
For the particular hierarchical operator in this experiment, let us define the primitive features as the range and the quasi-ranges of the order statistics model, and the corner differences and main axes differences of the Sobel operator as defined in Figure 1.16. The rationale for choosing the latter 4 features (as defined in Figure 1.16) was to include some primitive features which had spatial dependence and could indicate edges in various orien-



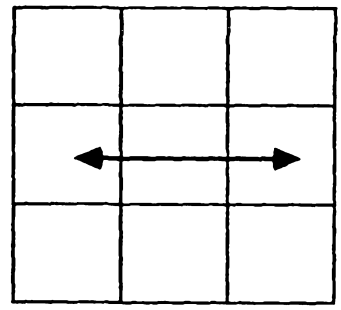
x_1 - x_9



x_3 - x_7



x_2 - x_8



x_4 - x_6

Figure 1.16: Spatial primitive features.

tations. The features chosen may not be the set of optimal primitive features but rather are a set of easily computable operations which are easy to interpret. However, it is important to remember that any redundancy in the feature set, or features which yield no relevant information, will by definition be suppressed by the tree derivation algorithm. The goal here is not to necessarily to derive *optimal* hierarchical operators, but rather to demonstrate the feasibility of hierarchical operator design and investigate the resulting performance.

Having defined the appropriate attributes, the next step is to obtain a set of labelled or classified samples, i.e., training samples. Define a "typical image" to be one that contains typical edge content, luminance and noise of those images to be encountered in practice. If the variation in application images is significant, then from a practical point of view it may be necessary to train the algorithm using sample data from different images. As with any classifier design problem, we must ensure that the sample data is a representative random sample from the overall population of samples. Since the individual samples are pixels then it quite easy to obtain a very large sample. The only constraints imposed may be by the computing facilities available to run the classifier design algorithm, e.g., memory constraints. The sample data is then classified via standard edge-detection techniques using the best edge-detection algorithm available and, hence, the *classified* table of training samples is obtained. Each datum corresponds to a pixel described by the chosen primitive attributes evaluated at that pixel and tagged with an "edge" or "non-edge" label. One approach which might merit further investigation would be to get a human subject to classify the edges (e.g., by delineating them on a screen or a hard-copy). This approach has not been investigated in this thesis but might potentially lead to interesting results.

We can now begin to appreciate the potential advantages of the hierarchical operator. If, for example, the typical image contains low edge content, as indeed is the case with many images, then the optimal hierarchical operator should first try to classify each pixel as "non-edge," using some primitive attribute if possible. For those pixels which are not immediately so classified, then apply other attributes to determine if an edge is present. Since many routine images have at least 90% of their pixels classified as "non-edge," the computational advantages are obvious.

We note again that the composite operators themselves (Sobel and dispersion) are essentially binary-valued, i.e., greater or less than some threshold. In computational terms,

this amounts to a simple comparison. Hence, we must also implement our primitive features in this simple manner (i.e., with thresholds) if we wish the hierarchical operator to be competitive in computational terms. The old problem of threshold selection once again rears its ugly head.

We will use an approach which circumvents the problem by using the tree derivation algorithm. Rather than defining the thresholds *a priori* and designing the tree using these fixed thresholds, we will select thresholds at the *node* level while designing the tree. The thresholds for each feature are defined by using the Stoller criterion based on the Kolgomorov-Smirnoff distance for discriminating between two classes [50]. Essentially, the threshold of maximum separability in terms of the classes is that which maximises the distance between their cumulative distributions conditioned on a given feature. The procedure has the advantage of being relatively simple to implement computationally and is well-defined in the original work by Friedman [51] applying this idea to decision trees. In this sense we are implementing a dynamic threshold scheme, which should be advantageous since each threshold will be locally optimal for a given node. Having found the best threshold for each attribute in this manner, the features are effectively quantised into binary features. The resulting binary feature which provides the most information about the edge classes is selected by the greedy mutual information algorithm as the feature for that node in the usual manner.

1.2.4 Experimental evaluation of the hierarchical approach

In this section we compare the performance of the Sobel, dispersion and hierarchical operators on 3 different images. We are interested in three aspects of the results, namely,

(1) The performance of the induced edge operator,

(2) The possibility of inducing new "knowledge" about edge detection in general,

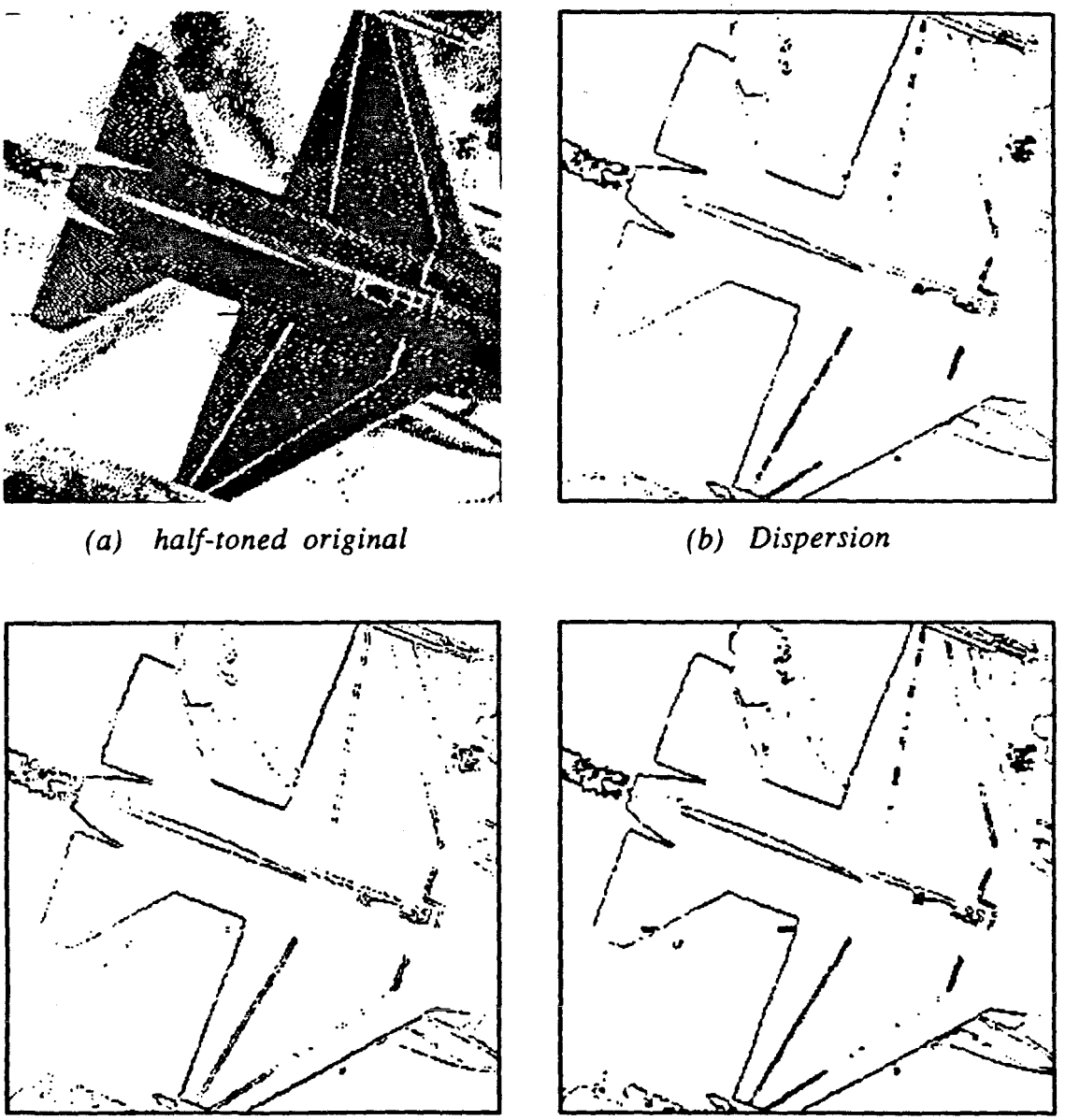
(3) Overall implications regarding the mutual information tree design algorithm and its application.

The images were generated via a scanning device using 32 grey levels. Each image is 256 pixels by 256 pixels.

Before considering the results it is worth mentioning some practical aspects in deriving the hierarchical operators. The operator used to *classify* the samples initially, is the Sobel since we found that it yielded slightly higher quality edge maps than the dispersion operator. The threshold t for both operators (dispersion and Sobel) was chosen heuristically based on visual evaluation of the effects on the edge maps of varying t. Because of memory limitations in the software, the entire image could not be used for training purposes. Consequently, 80 pixel by 80 pixel sub-images were used as training images. Provided the sub-images were reasonably typical of the image as a whole, it made little difference in the experiments where they were located within the overall image. In each of Figures 1.17, 1.19, and 1.20, sub-figure(a) is a half-toned output on a printer of the original grey-scale image.

The first example we consider is the airplane image shown in Figure 1.17(a). The output of the Sobel operator in Figure 1.17(c) is clearly better than that of the dispersion operator in 1.17(b). The hierarchical output in 1.17(d) is also better than the dispersion. But it is difficult to distinguish whether the Sobel or hierarchical operator is better. While the Sobel picks up finer detail, the hierarchical operator contains more edge information, particularly that of the airplane itself. The lack of fine detail in 1.17(d) can be explained by the actual tree structure of the operator as shown in figure 1.18. It uses mainly the quasi-ranges as primitive components, which, as we can see in 1.17(b), tends to merge edges which are close together (this difference can be seen by considering the effect of passing the 2 operators over alternating black and white pixel-wide vertical lines). Obviously, the average number of additions/subtractions performed by the tree is much less than that of either of the other 2 methods, i.e., only 1.34 as compared to 13 for the Sobel and 7 for the dispersion.

Figure 1.19 contains equivalent information to Figure 1.17 but for the "bin of tools" image. Interestingly enough, the hierarchical operator derived for this image was quite similar to that derived from the airplane image. All 3 operators perform similarly here with perhaps the order of merit being dispersion, Sobel, hierarchical. The difference in performance is slight. The image itself does not have clearly defined edges, hence, the noisiness towards the top of all three edge maps. For this image we see that the hierar-



(a) half-toned original *(b) Dispersion*

(c) Sobel *(d) Hierarchical*

Figure 1.17: "Airplane" image, (a) is half-toned, (b), (c) and (d) are edge maps.

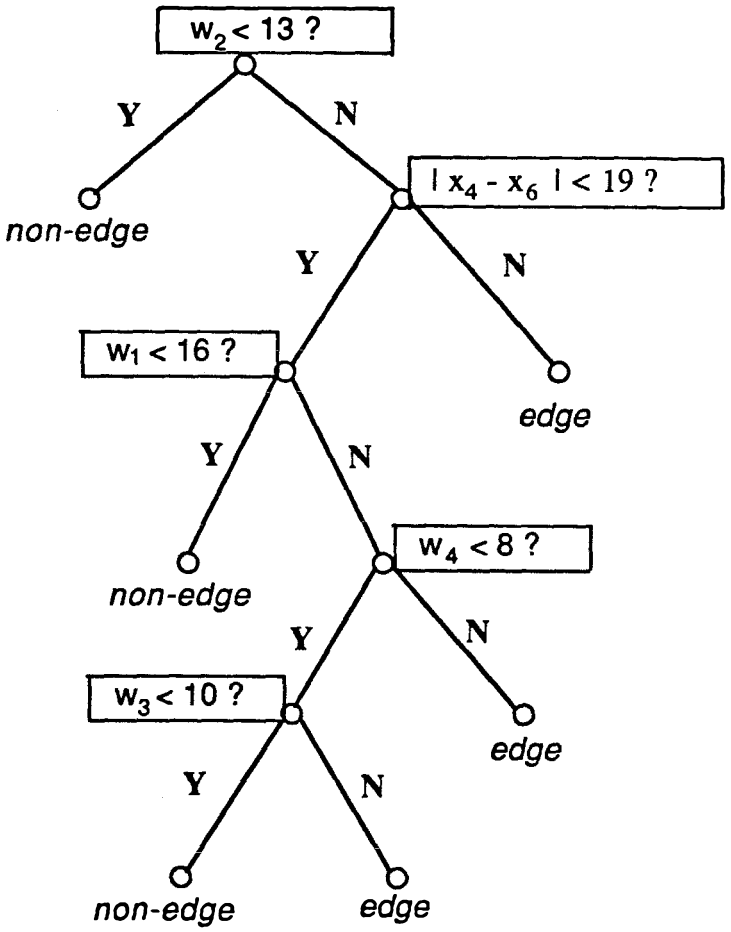


Figure 1.18: Hierarchical operator for the airplane image.

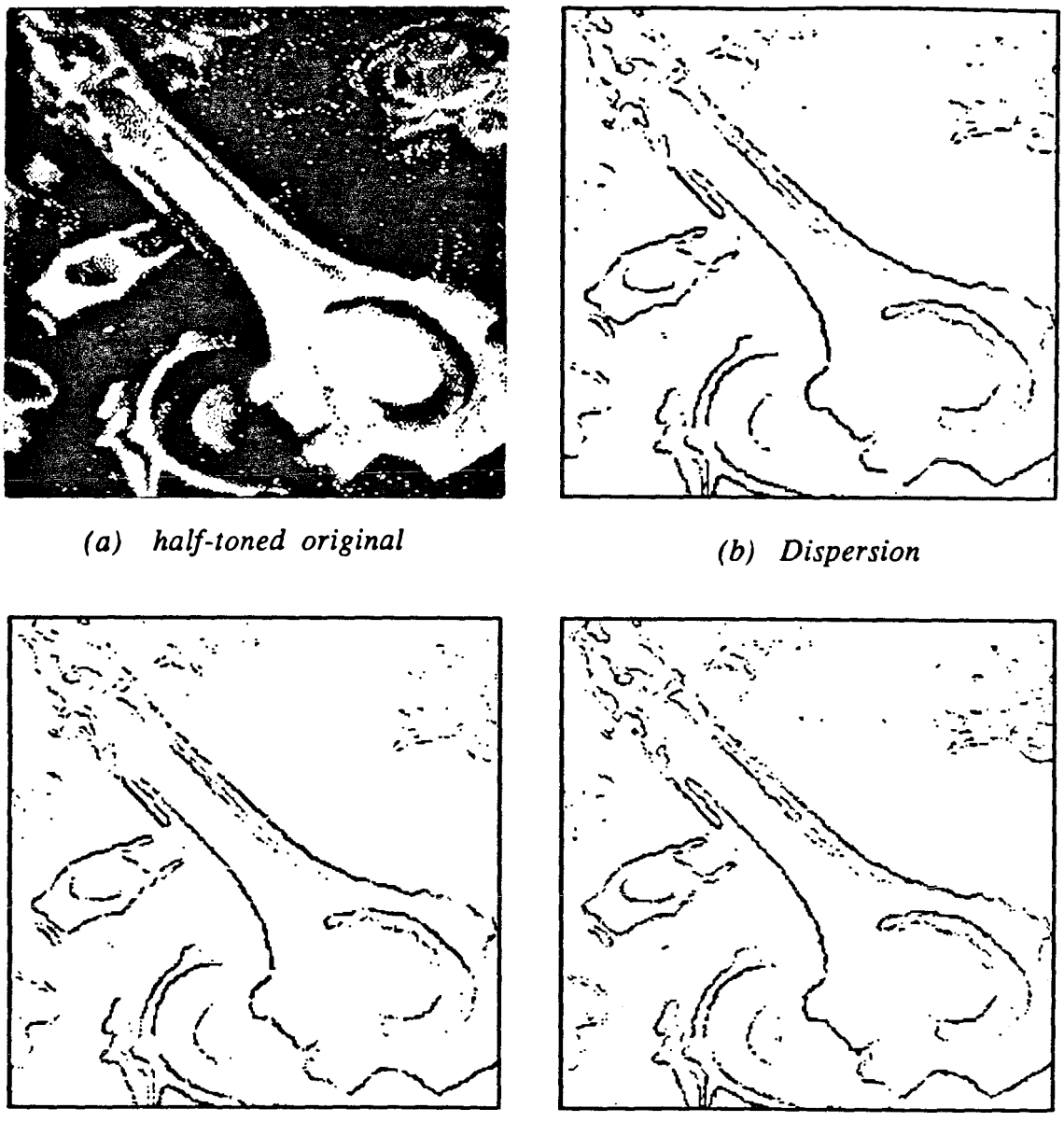
chical operator can only resolve edges as well as its primitive features will allow. Better performance could be achieved by using a higher quality classifier than the Sobel operator and defining more primitive features.

The third image is shown in Figure 1.20(a), an image of an integrated circuit wafer. There is a significant difference between this image and the previous two in that the edge content is significantly higher. Nonetheless, the algorithm yielded a tree operator quite similar to the two derived previously. The dispersion operator succeeds in finding the edges of the various squares but the edges of the numerals are hardly detected. The Sobel, on the other hand, yields some numeral edges but also responds to a lot of impulse noise. While the hierarchical does respond to some noise, and does not distinguish clearly several of the numerals, its edges are nonetheless better defined than either of the other two. The improved performance is due to the fact that the hierarchical operator combines the more discriminating features of *both* other operators.

It is interesting to note that we tried out the operator which had been trained on the airplane image on the other two images. While it performed well on the bin of tools image, it gave quite noisy results on the wafer image. The latter phenomenon is due to the difference in edge content and edge orientation between the two images. Hence, the importance of selecting a representative training data during the classification phase of the design is emphasised.

For each of the tree operators, the feature selected at the root node was the same, namely the feature $w_{(2)}$ as defined earlier. Given that in a decision tree context, the feature at the root node is the most discriminating of all features by definition, one can conjecture that $w_{(2)}$ possesses better edge discrimination properties than any of the other primitive features we defined. In particular, it is better than the range, $w_{(1)}$. The reason for this is simply that low values of this feature indicate no edges (just as with the range), whereas a high value is a good indication of an edge (unlike the range where a high value may just be impulse noise). This is a good example of how the hierarchical approach can yield new insights into the problem at hand.

Now let us consider the relative *computational* performance of the three operators. The results are tabulated in Table 1.1. Since the hierarchical operators for both images were



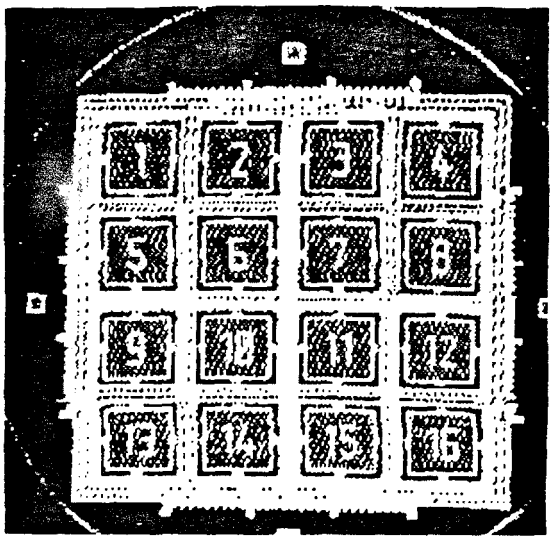
(a) half-toned original

(b) Dispersion

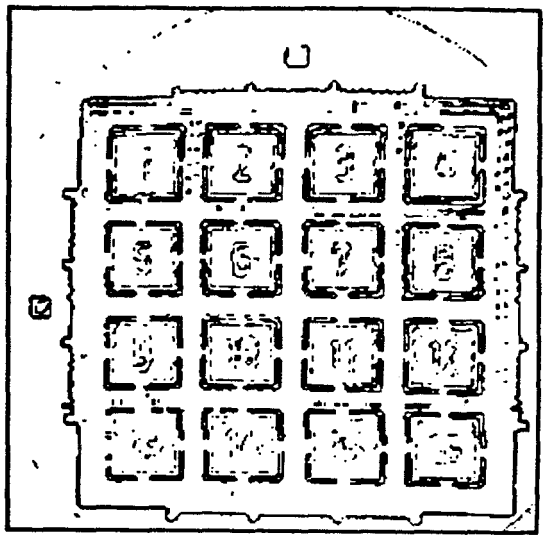
(c) Sobel

(d) Hierarchical

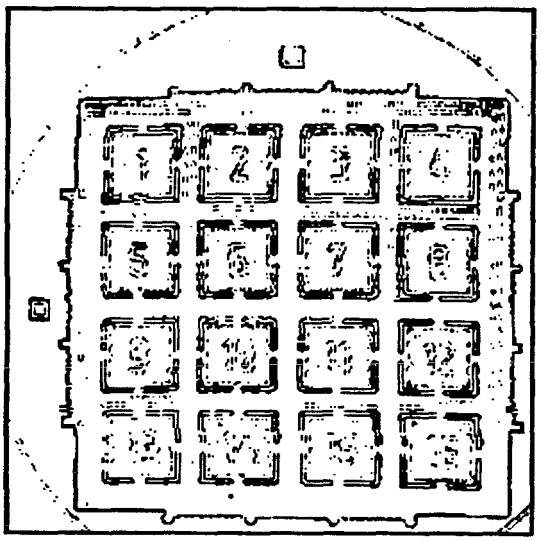
Figure 1.19: "Bin of tools" image, (a) is half-toned, (b), (c) and (d) are edge maps.



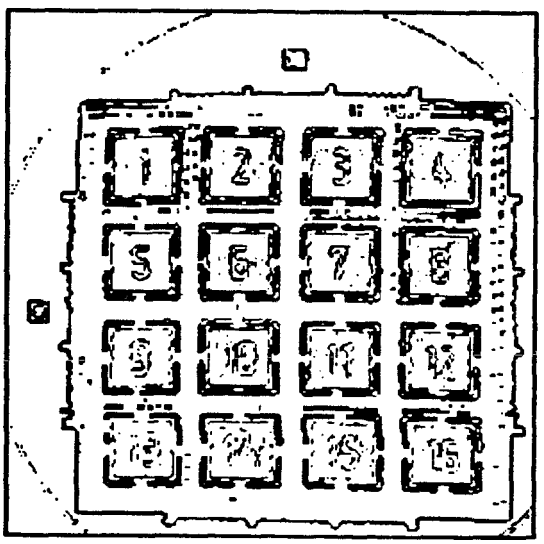
(a) half-toned original



(b) Dispersion



(c) Sobel



(d) Hierarchical

Figure 1.20: "Wafer" image, (a) is half-toned, (b), (c) and (d) are edge maps.

	Sobel	Dispersion	Hierarchical
Additions	13	7	1.34
Multiplications	0	0	0
Absolute value operations	2	0	0.46
Conditional evaluations	1	1	1.34
Sorting comparisons			
Parallel	0	8	8
Serial	0	19	15.8

Table 1.1: Comparison in computational terms of the different edge detection schemes.

very similar, the figures in the hierarchical column are an average over both. The conditional evaluations are tests of the form "is $x > y$?" and do not include the computation of x if x is a compound expression. Such computations are included under "additions." We have used the minimum delay of 8 comparisons for a parallel sorting structure as given in Knuth [52]. For sorting in serial form the number in the dispersion column corresponds to the minimum number of comparisons required to sort 9 numbers, again as given by Knuth [52]. Because we do not need to sort all the numbers each time the hierarchical operator is used, the figures in the hierarchical column are the experimentally obtained average number of comparisons required. Any comparison in this manner is obviously dependent on the method of implementation, e.g., hardware or software, serial or parallel, relative speed of the various primitive operations. Nevertheless, *independent of the implementation method*, the hierarchical operator will always be considerably faster than the dispersion

operator particularly for images of very low edge content. Comparing the Sobel and hierarchical operators will depend on the implementation and especially on the relative speed of comparisons and additions. If one assumes (as Pitas and Vetsanopoulos do [48]) that one were to implement the sorting operation using a parallel VLSI sorting network, then the hierarchical operator would indeed be much faster. As a matter of interest, using programs written in C, to classify all pixels in a 256 by 256 size image took 30.8 seconds with the dispersion operator, 23.7 seconds with the Sobel, and 16.4 seconds with the hierarchical. It was apparent that both the dispersion and hierarchical operators spent most of the time sorting the pixel values, so that one might expect a parallel implementation to run very quickly indeed.

1.2.5 Concluding remarks on the edge detection experiment

From a practical viewpoint we have met with success in this edge detection experiment. It is important to realise the general implications we can draw from our practical results.

(1) The algorithm yields efficient and accurate edge detectors. In addition, edge operator design using this technique is *automated*, non-parametric and relatively flexible. The notion of an algorithm which can "learn" edge detection capabilities from edge-classified data is potentially a useful idea in applications such as robotics. We note that the general *principle* is far more important than the particular experiment we looked at here. Indeed, there is no reason to limit the approach to image-processing applications. Any problem which can be cast in a suitable classification light is amenable to the hierarchical approach.

(2) Feature selection is an important aspect of experiment design. Indeed, it is about the only part of the design procedure which may require domain-dependent expert knowledge. However, because the mutual information approach ignores irrelevant features, then one can over-specify the number of features. Hence, using this technique, image-processing "novices" can achieve relatively "expert" results.

(3) We noted that in all the hierarchical operators derived by the algorithm (for the 2 examples given and several other images not shown in this paper), the attribute $w_{(2)}$ was

always at the root node, i.e., $w_{(2)}$ was the most informative attribute for every image we used. In addition, the spatial attributes rarely appeared in any of the operators. Hence, the conjecture is that order statistics are more useful for edge detection than spatial operators, and for 3×3 windows $w_{(2)}$ is the single best statistic. This is an example of new domain-dependent knowledge resulting from the algorithm.

1.3 Discussion and conclusions

1.3.1 A summary and some suggestions on future research directions

The mutual information tree design algorithm has received much attention recently in terms of practical applications. Yet, little or no work has been devoted to some of the underlying theoretical problems. We have established a basic understanding of the algorithm and its behaviour in terms of noise, misclassification rate, average depth, etc. In particular, we were able to establish a direct correspondence between this algorithm and prefix coding. In addition, the general problem of tree design in the presence of noise is seen to be equivalent to the problem of designing a variable-rate code for a source which has already been corrupted by noise. By considering the rate-distortion model of Wolf and Ziv for this communications problem, we were able to better understand the fundamental limitations and trade-offs for tree design.

We began by deriving equations for the information available at a node in the presence of noise, eventually leading to upper and lower bounds on the average tree depth and misclassification rate for particular termination rules. In terms of the bounds at least, we determined that the algorithm exhibits the appropriate behaviour, trading-off increased depth for decreased misclassification rate. The problem of finding termination rules for the greedy algorithm was discussed and our theoretical models demonstrated clearly the inadequacies of any form of threshold rules. Two new termination rules were proposed, one statistical and the other more heuristic in nature. The theory also enabled us to predict in a qualitative manner that "good" data can lead to bad classifiers and confirm earlier conjectures of this nature.

In terms of experimental results in using the algorithm, the algorithm performed comparably well with existing approaches in both a controlled experiment and a practical application. In general, it resulted in similar performance to the other approaches with significant savings in computation. The overall conclusion is that for certain problems where a hierarchical approach makes sense, this tree design algorithm can obtain near-optimal solutions in an efficient manner.

It is worth mentioning some aspects of tree design which we did *not* cover and that are worthy of future investigation.

(1) Continuous-valued attributes are common enough in practical situations that the quantisation problem needs to be addressed. Aspects of this problem include the number of quanta for a given attribute, whether to quantise before or during the design algorithm, criteria for quantisation, etc. The Kolmogorov-Smirnov criterion used in the edge detection experiment is an interesting approach which allows one to define quantisation levels at each particular node in the tree. Unfortunately, it can only be used in the 2-class case. Friedman [51] proposed extending the technique by artificially breaking the n-class problem into n binary class problems. However, this is not a very elegant or efficient solution. Interestingly enough, it is easy to establish that the Kolmogorov-Smirnov criterion yields partitions which also maximise the entropy of the quantised variable for certain cases.

(2) The models we developed in this chapter were quite simple in nature. The upper bounds in particular on $\bar{d}$ were based on some rather unrealistic assumptions. The worst-case analysis of Garey and Graham [37], however, represents the other extreme. A more appropriate model would be to average the performance of the algorithm over various problems and attribute sets. This is an impossible task if approached in a brute-force manner, but it may be possible to considerably simplify matters by insightful arguments and assumptions. For example, an interesting problem is to consider how much information on average an attribute yields in terms of partitioning a sample space, i.e., for the case with no noise. If one assumes that all possible probability distributions are equally likely (Maxwell-Boltzmann), and that all partitions (attributes) are equally likely, one finds (after some calculation) that the expectation over all attributes is 0.69 bits. Hence, "typical" binary partitions yield quite a lot of information.

(3) Another common occurrence in practice is the unavailability of a large amount of truly random sample data. Either the sample size is too small, or the data is biased in some manner, or both situations occur. While much work has been carried out in applied statistics for problems of this sort, there still exists a gap between finding useful results which can be applied to the hierarchical classifier problem and existing theoretical results in the statistical literature.

(4) In parallel with the work described here, recent results of Breiman et al. [13] and Chou et al. [39] have demonstrated both theoretical advances and practical applications in

the field of "pruning" algorithms. Essentially, in a pruning algorithm, a series of operating points are found along the distortion-rate curve for many different rates. Then, beginning with the largest tree, the trees are pruned back to find a tree which meets some criteria of optimality. Questions remain as to how much "nearer-optimal" pruning solutions are when compared to the greedy approach. What is the trade-off in computation between these two approaches? What are the statistical implications of growing large initial trees? This approach certainly merits further research.

1.3.2 On the limitations of trees

Consider the following scenario. A company asks us to design an algorithm which can classify any one of 100 possible chemical spills by instructing a user or human tester to perform a series of tests and by interpreting the results of these tests. We decide that a decision tree is an appropriate solution, since, the tests are expensive (and hence minimising the expected number of tests is important), the test results are discrete-valued, and a large body of historical data is available.

We design the tree using the mutual information top-down algorithm and install the tree program on-site. The tree works fine initially. A few months later, we discover that the tree is no longer being used and the testing process has reverted to the slow, inefficient manual trial-and-error techniques that the tree was designed to replace. On further investigation, we find that several factors led to the abandonment of the tree.

(1) Quite often a test (or attribute), corresponding to a particular node in the tree, either could not be performed (the particular tester did not know how to perform this test, or the test materials had run out), or else the test results were not available due to the test having been performed incorrectly. In this situation, the user (tester) could not proceed any further in using the tree program. The tree representation possessed *no redundancy or flexibility.*

(2) Another frequent situation was that initial data would be available in the form of strong odours, oily appearance or particular location of the spill. This initial data

would normally enable the tester to make a good initial guess as to which subset of classes the spill might belong. However, the tree could not make use of this initial information because of its pre-determined structure and would proceed to begin with the same test regardless of initial information. The tree was *insensitive to context.*

(3) Some new testing procedures had become available which could easily be characterised in terms of their classification performance and their relation to existing tests. However, this information could not be utilised without re-designing the tree, which the company did not know how, or particularly want, to do. The tree was *not easily modifiable.*

(4) The company management had hoped to use the tree as a training tool to train novice testers and also as an analysis tool to better the company's understanding of this problem. But the tree was *obscure and difficult to understand.* Furthermore, because of the obscurity of the tree's decision-making rationale, the testers using the tree did not trust it as a tool, especially since the tree program was unable to communicate or explain why it was proceeding in a certain manner.

The aggregation of all these effects led to the abandonment of the decision tree approach in this hypothetical scenario. These effects highlight some of the limitations of trees in the realm of general automated decision procedures. Essentially, trees represent a "hard-wired" solution to the sequential decision problem which may or may not be appropriate depending on the particular application. The edge experiment was a good application in this sense. The environment is static, all the tests can always be performed, no initial data is available and the no user interface is required.

Many decision problems however fall into the category described by the chemical spill problem, where the automated decision procedure needs to be able to handle variable inputs (missing, uncertain, or changing data), variable outputs (different goal specifications), and have an explicit representation of its "knowledge" for user interaction. Chapter 2 of this thesis deals entirely with this intriguing problem of building machines which can mimic intelligent decision-making behaviour. In essence, we will see that such machines *must* have an internal *model* of their external environment.

From this point of view a tree represents a very restrictive, specialised internal model. Hence, giving decision trees the name of "expert systems" (as has recently been popular)

is completely inappropriate, since, a *tree* structure is far too simple to model the decision process of an expert. We shall see later how a more general *graph* structure, called an "inference net", is a much more realistic model. There has been some recent work in the area of adapting tree design algorithms for more general purposes. White [53] and Quinlan [54] have developed algorithms for trees which could handle missing or uncertain information, Arbab and Michie [55] worked on designing "linear trees" which are easier for humans to understand, and Quinlan [56] proposed an algorithm which derives more general graph components from a collection of trees. However, these algorithms represent a very indirect approach to the problem by seeking to map the restrictive tree representation into more general schemes. The fundamental limitation remains that trees are too limited to model general decision behaviour. In this thesis we will adopt the approach that a basic conceptual leap must occur from decision trees to decision *graphs* in order to solve the types of problems which are beyond the scope of trees. Chapter 2 is devoted to the learning and implementation of such general decision structures.

Chapter 2

An Information Theory Model for Rule-Based Expert Systems

2.1 Building intelligent machines

Since the arrival of relatively inexpensive computational power in the 1960's, the notion of building machines which display some form of intelligent behaviour has caught the fancy of academic and industrial research communities alike. Indeed ideas concerning the simulation and modelling of intelligence can be traced back to the field of *cybernetics* in the 1940's and the work of Wiener, Turing and many others. Early implementation efforts in the 1960's focussed primarily on statistical techniques, under the broad heading of statistical pattern recognition [57], a field that has become gradually depleted as more and more of the specialised problems get solved and the realisation grows that specific algorithms and statistical models may be too narrow and restrictive in their assumptions to model true intelligent behaviour. Statistical techniques are necessary but not sufficient.

Some statistical pattern recognisers then abandoned the purely statistical approach and began to establish other fields, many of which fell under the umbrella of "artificial intelligence." Artificial intelligence encountered difficulties, as solutions proposed for simple toy-world problems (e.g., the work on checkers-playing by Samuel [58] and blocks-world computer vision by Winston [59]) were difficult to extend to real-world problems. However, with the arrival and relative success, in the late 1970's, of expert systems, and their practical application to real problems, artificial intelligence techniques regained popularity. The basic paradigm behind expert systems is that *knowledge*, in an explicit form, is the key to building intelligent machines. The general idea is that, rather than embodying machines with general problem-solving and search techniques, it is better to explicitly encode detailed knowledge (i.e., expert-level knowledge [60]) into the program. Programs which derived their power from the encoding of expert knowledge became known, somewhat optimistically, as expert systems. The notion of expert systems was developed from the simple idea that human experts can be characterised 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. This basic paradigm of explicit knowledge representation has many advantages over implicit techniques. However, there are limitations also, which we shall discuss later.

Motivated by results from cognitive science, and a desire to develop a more robust approach to modelling intelligent behaviour, a third paradigm of intelligent behaviour which

emerged was that of neural networks or connectionist machines (cf. Feldman and Ballard [61] and Hopfield [62]). Essentially this approach can be characterised as modelling intelligent behaviour using distributed parallel architectures. As indicated by the current spate of interest in this topic, it holds considerable promise. Yet the problem of implicit knowledge representation remains a stumbling block for certain applications, i.e., problems which require higher-level cognitive processes at the *reasoning* level rather than the *perceptual* level (Rumelhart and McClelland [63]). For the present, neural networks seem more appropriate for perceptual tasks rather than the higher-level decision problems to be considered in this thesis. However, the inherent computational advantages of the connectionist approach make it an appealing candidate for implementation purposes, as we shall see.

This second part (Chapter 2) of the thesis intends to deal with the problem of how one might go about building an intelligent machine. The general outline will be to use probabilistic ideas to form a necessary basic framework, and then to use this framework to construct what to all intents and purposes is a rule-based expert system. We will borrow and adapt ideas from several fields to achieve our goal, e.g., information theory, artificial intelligence, cognitive science, statistical decision theory, etc. Curiously enough we will eventually arrive at a proposal for an architecture which bears some resemblance to a connectionist machine. We will arbitrarily refer to terms such as expert systems, artificial intelligence, rational agents, etc., etc., throughout the discussion. For our purposes they all mean the same thing: our model of rational behaviour which we are about to define. The reader should be aware that our approach will not necessarily be biased for or against any of the various fields whose ideas we use.

Of course a caveat is in order. Building an intelligent machine (we haven't even properly defined the problem yet) is not the sort of problem one can solve in a thesis, assuming it can be solved at all. We will not get involved in the philosophical implications of machine intelligence but instead adopt a more pragmatic, engineering approach. Let us define what we mean by an expert system. Consider the following *gedanken* experiment. We are given a well-defined problem domain (i.e., with well-defined inputs and outputs, e.g., weather forecasting), and a well-defined language which can be used to communicate concepts in this domain. There are two rooms with which we are in communication, via the defined language. A machine or program is placed in one room and a human expert in the other. We are not told which is in which room. Both have access to the same initial data and both

can obtain further data if they so choose. If for all intents and purposes, using the same language to communicate with both rooms, we cannot determine beyond reasonable doubt which room contains the machine, the machine can be deemed an "expert system." Note that this "definition" hinges critically on the nature of the communication language used. In particular if we can query for explanations about the decisions made, then straightforward algorithmic techniques will not suffice, e.g., a differential equation may implement a model for weather prediction but on its own it will not pass our test if we are allowed to query *why* it makes certain predictions. The data may be changing from situation to situation so the system must be able to handle different data inputs with varying levels of certainty, e.g., one day there may be a lot of exact meteorological data available, while the next day the exact data may have been lost so that the system has to rely on older data and uncertain information from other forecasters. A standard branching type of sequential algorithm is clearly unsuited to the task. To incorporate features like explanation capabilities and the ability to be data-driven, the system must have its own internal representation or model of the domain. In AI terms this model is the domain knowledge. This is a key characteristic of "intelligent" algorithms or expert systems. One of the contributions of this thesis is to extend the AI notion of a knowledge-based representation to a *probabilistic* knowledge-based representation.

We will focus on a particular type of problem, a basic inference and decision problem. The system must reach a conclusion or a decision about a given situation in a given domain, based on historical knowledge of the domain (e.g., rules), initial situation-dependent data, and data which it can obtain from its environment (i.e., it is allowed to query for additional information in some manner). Common reasoning modes such as diagnosis, prediction, interpretation, monitoring and control easily fall under this description while more creative modes such as planning and design may or may not be interpreted as such. Applications range from medical diagnosis to legal reasoning to interpretation of images to network management, etc. Suitable applications can generally be identified as tasks which require human expertise in the form of reasoning for their solution. In particular, any application which does not fit the above description, but can be better modelled by a more direct algorithmic approach, should use the direct algorithmic solution rather than an expert system.

The popular distinction which has been drawn between artificial intelligence and stand

-ard algorithms is generally not valid. After all, any AI program can be viewed as "just an algorithm" while any algorithm can be interpreted as a manifestation of "artificial intelligence". Who is to say that an adaptive digital filter doesn't display artificial intelligence? The point is that these two schemes, algorithms and AI, can be thought of as representing different extremes along continuous dimensions, rather than completely disconnected approaches. The assertion that statistical pattern recognition, artificial intelligence, and connectionist modelling are all mutually exclusive to some degree is counter-productive. They are all, after all, trying to solve the same "machine in the room" problem. It seems appropriate that if we can identify the dimensions (along which these approaches represent extremes), we can then explore new paradigms of modelling intelligence by combining the advantages of each scheme. Two such dimensions we can identify are that of knowledge representation (AI: explicit, statistical: implicit, connectionist: implicit), and the modelling of uncertainty (AI: very little, connectionist: somewhat, statistical: yes). An example of this type of approach is the recent work of Geffner and Pearl [64], where they develop a correspondence between the implicit knowledge representation of the connectionist model and what they term "the better understood semantics of probabilistic networks". The work presented in this thesis is of a similar vein, except that we will primarily focus on bridging the gap between artificial intelligence and probability.

The first dimension, explicit knowledge representation, is an important aspect of modelling rational behaviour. Perhaps the key contribution of artificial intelligence in the last decade has been the identification of this concept. If the machine is to appear intelligent then it must be able to explain, reason and interact with its environment at the *knowledge* level. Implicit knowledge, such as we have in a decision tree, is fine if all we want is a black box which implements an input/output transfer function. But such black boxes will not be deemed intelligent by the user. Explicit knowledge representation has the advantage that the power of the system lies in its *knowledge base*, an explicit and separate representation of the domain knowledge. For example, in a rule-based system, the knowledge base is the set of rules in the system. In a typical clever algorithm, however, the knowledge may be hidden away implicitly in the code. The explicit representation 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 relatively expensive. An added benefit of establishing a knowledge base is that it is an asset to the developer both as a training tool and as a repository of human expertise which is relatively permanent. In areas such as telecommunications network management and marketing strategy development, knowledge bases are themselves valuable commodities. As we shall see, developing knowledge bases is one of the more difficult aspects of building expert systems. We will later develop new information-theoretic techniques for automated knowledge-acquisition which significantly improve on existing approaches and make possible the learning of rule-based knowledge bases directly from raw data.

The most widespread form of knowledge representation in use at present is the rule-based representation (which we will deal with in detail in the next section), partly because of its origins in cognitive modelling (Newell and Simon [65]) but primarily due to the modularity advantages which ensue from rule-based programming. Rule-based expert systems began to appear in the 1970's and displayed considerable prowess in certain domains. Let us consider some examples of "true" expert systems, i.e., systems that would probably pass our "machine in a room" test. One of the first success stories was the PROSPECTOR system (Duda et al. [66]), which aids geologists in searching for ore deposits. Its most notable achievement was in correctly predicting the location of an ore deposit reportedly worth over $100 million dollars [67]. Many of the early systems were developed in the field of medical diagnosis, including MYCIN [68], which was able to give consultative advice on diagnosis and therapy for infectious diseases, advice which was rated slightly better than national-level experts in the domain, and considerably better than ordinary physicians, in independent statistical tests [69]. Another early project was the DENDRAL program for inferring molecular structure from mass spectrographic information (Buchanan and Feigenbaum [70]), which outperformed experts on certain problems [71]. From these initial successes we can grasp the appeal and potential of rule-based expert systems. We are not alone. Over the last decade expert systems have been developed in almost every conceivable domain, e.g., XCON/R1 configures VAX systems for Digital Equipment Corporation [72], DART diagnoses faults in computer hardware systems [73], ACE identifies faulty cables in telephone networks for AT&T [74], COMPASS analyses telephone switching systems for GTE [75], PALLADIO assists in VLSI design [76], DELTA performs diagnoses of diesel/electric locomotives for General Electric [77], DIPMETER interprets dipmeter logs to aid oil drilling [78], LDS aids legal experts in settling product liability cases [79],

MACSYMA performs mathematical tasks [80], WILLARD forecasts thunderstorms [81], MECHO solves problems in applied mechanics [82], LES monitors the loading of liquid oxygen for the space shuttle at Kennedy space center [83], etc.

The potential benefits of expert systems are obvious. Productivity and efficiency are increased, with the resultant cost benefits, by augmenting and automating scarce expertise in critical problem areas. Yet critical problems remain unsolved. It is a fact, not widely advertised, that the vast majority of expert systems have never gotten beyond the stage of a prototype. One of the main reasons is the fact that real-world problems often involve uncertainty in one form or another. While expert system researchers have proposed various uncertainty calculi to deal with this problem, their schemes tend to be ad hoc and unrealistic in their assumptions. As such they are avoided by practitioners. Indeed the most common type of expert system, of the subset that actually gets *fielded* (as opposed to just prototyped), is the system that solves a deterministic type of problem where uncertainty is not required.

The real potential lies in the more general type of problem which includes uncertainty as a primary component. Humans solve such uncertain decision problems all the time in a practical manner. This brings us to the second dimension along which AI and statistics can be related, namely, the modelling of uncertainty. Let us state the position to be adopted in this thesis. We will adhere to the basic axioms of probability theory and Bayes' formula as far as possible. The debate over the appropriateness of probability theory as a scheme for modelling uncertainty in the real world has raged for quite a while in the AI, philosophy, and statistical research communities. It is not a debate which we will join. For example, fuzzy logic has been proposed as a more accurate representation of real world uncertainty (Zadeh [84], Gaines [85], Prade [86]) with claims both for (Zadeh [87]), and against (Cheeseman [88]), its use. The question asked by Zadeh in [87], "is probability theory *sufficient* for dealing with uncertainty in AI problems?," may be difficult to answer. We will take the approach here that probability is certainly a *necessary* component in dealing with uncertainty and, hence, probabilistic problems in AI need to be addressed. Dempster-Shafer belief theory (Dempster [89], Shafer [90]) is another scheme which has been proposed as being a generalisation of point-valued probability theory. Its relation to standard Bayesian approaches is as yet the topic of research, e.g., Kyburg [91] has recently asserted that Dempster-Shafer theory is actually a special case of the Bayesian approach.

Probability theory retains the advantages of being the only simple and theoretically well-defined scheme of representing uncertainty, and has recently been defended as such (Cheeseman [92], Pearl [93]). Indeed it has been shown by Lindley [94], that probability is unique as a measure on which to base decisions in the sense that using any other scheme will inevitably be worse in a "competitive" environment. We will interpret probability in the broadest sense, simply as a set of weights which satisfy the basic axioms. How these weights are obtained is not our primary concern (e.g., whether by objective or subjective assignment), but rather how to *use* the probabilistic information to good advantage.

We will proceed in our investigation roughly as follows. We will first look in somewhat more detail at rule-based systems, outlining advantages and disadvantages of existing approaches to design and implementation. The lack of a well-defined *quantitative* theoretical model for rule systems will be identified as a source of several problems. We will proceed to define information-theoretic rule measures as the basis for a new quantitative theory of rule-based probabilistic reasoning. The problem of *learning* rules will be discussed, and a new information-theoretic rule learning algorithm (ITRULE) will be proposed. We will then deal with the considerable problem of linking the rules together to perform probabilistic *inference*. Finally we will describe a statistical decision theory model for *controlling* the inference. The individual pieces of the theory will be fitted together to form both a theoretical basis and a practical model for rule-based expert systems. The approach is unique in several aspects, not least the fact that the model represents a new and interesting application of information theory.

We note that because of the broad scope of the problem, and given the obvious constraints, necessarily this thesis does not provide "all the answers." Rather it is intended to formulate a well-defined framework, provide *some* of the answers, and identify and discuss future directions as to where the other answers might be found.

2.2 Rule-based systems: principles and problems

2.2.1 The historical origins of rule-based systems

A rule-based system is a special case of a *production system*. To quote Davis and King [95]:

> A production system may be viewed as consisting of three basic components: a set of rules, a data base, and an interpreter for the rules. In the simplest design a rule is an ordered pair of symbol strings, with a left-hand side and a right-hand side (LHS and RHS). The rule set has a predetermined, total ordering, and the data base is simply a collection of symbols. The interpreter in this simple design operates by scanning the LHS of each rule until one is found which can be matched against the data base. At that point the symbols matched in the data base are replaced with those found in the RHS of the rule and scanning either continues with the next rule or begins with the first. A rule can also be viewed as a simple conditional statement, and the invocation of rules as a sequence of actions chained by *modus ponens*.

Production systems were originally defined by Post for symbolic logic in 1947 [96] and subsequently reappeared under the guise of Markov algorithms [97] in 1954. Chomsky in 1957 used production systems for linguistic modelling with some success, after which they were applied to the general problem of psychological modelling in the 1960's with the work of Newell and Simon [98, 99, 100]. Newell and Simon, in their major exposition on the topic in 1972 [65], proposed production systems as a good candidate for modelling human cognitive processes, particularly because of the functional analogies between human memory and the production system model. Other work followed in terms of cognitive modelling (Newell [101, 102], Thorndyke [103], Anderson [104]). More recently the work of Holland et al. [130] provides new evidence and ideas supporting production systems and rules as cognitive models. However, both production system proponents, and proponents of other schemes such as the connectionist approach, agree that it is difficult to establish conclusively the validity of any one particular model over another (Newell and Simon [65], pp.803-804: Rumelhart and McClelland [63], pp.143-144). While in this thesis we are primarily concerned in a more performance-oriented computational approach to reasoning

than in the validity of cognitive models *per se*, it can be noted that at present there appears to be a consensus that rule-based models are more appropriate for higher-level decision processes, while neural models are better suited to modelling perceptual processes. We may conjecture that the differences between the two approaches are not as great as is commonly perceived, and in particular, as we shall see later, there may be common ground which is apparent at the *implementation* level.

In addition to the cognitive work of Newell, Simon and others, there was a similar and parallel effort to develop computational models based on production systems for the purposes of solving problems in artificial intelligence, culminating in the widespread application of rule-based expert systems (as we discussed earlier). One of the advantages of the rule-based production system is the fact that it has the computational generality of a universal Turing machine (Anderson, [105]). There have been recent suggestions in the AI community that rule-based systems are not general enough to model rational behaviour, and that "second-generation" expert systems are required [106]. Such arguments are not relevant since (in theory at least) a production system can model any other programming representation. The perceived limitations of rule-based systems in problems like uncertainty modelling are more due to the fact that the problem has not been approached properly yet, rather than being due to any *inherent* limitations.

2.2.2 Principles of rule-based systems

Rule-based expert systems are by and large the only expert system paradigm which has gained widespread acceptance in practice. This is due mainly to its inherent advantages of modularity, and ease of design and maintainability as a programming methodology. Let us consider how a simple rule-based system might be designed and implemented. As mentioned earlier there are three components: the data base (short-term memory), the rules (long-term memory or knowledge base) and the inference "engine" or control scheme. Typically to begin with, the system designer (or "knowledge engineer") discusses the problem domain with a domain expert. A representation scheme is defined by identifying relevant attributes and values for the problem. The next step is to define rules or construct the knowledge

base. Typically this may involve lengthy discussions on case problems with the expert as the knowledge engineer tries to "tease out" rules. Rules are logical implications of attribute-value propositions, e.g.,

$$\text{If } \textit{valve} = \textit{open} \text{ and } \textit{temp} = \textit{high} \text{ then } \textit{flow} = \textit{normal} \ .$$

Rules may also contain actual actions in their conclusions, e.g.,

$$\text{If } \textit{alarm} = \textit{on} \text{ then } \textit{shut.down.plant} \ .$$

Sequences of rules can be used to establish logical inference paths between propositions. In the process, one is (hopefully) modelling the expert's reasoning chain.

The basic inference scheme is a simple "match and fire" or "recognise-act" loop. When a rule is applied or "fired" the attribute on the RHS is instantiated. This instantiation is recorded in the data base or working memory. The control scheme then scans the updated data base to search (or match) for left-hand sides of rules which evaluate to true, i.e., it determines a list of candidates for firing on the next cycle. It then decides which particular rule to fire, typically using some heuristic measure to order the rules. This ordering is termed *conflict resolution*. The chosen rule is then "fired," the data base updated, and the cycle repeats. Control information is very implicit in this scheme, since essentially it resides in the rules. The state of the data base on each cycle represents the state of the world as the system knows it. In a sense the rule-based approach is a generalisation of standard procedural algorithms since, in principle, we can replace the antecedent and consequent by more complicated expressions, even by sub-programs. However, rule-based systems are *not* very efficient for solving procedural, algorithmic types of problems.

The cycle we have just described is known as antecedent-driven inference or *forward chaining*. This mode of operation is useful when there is a lot of data available and the system must search the rule space for a conclusion. This is particularly useful in planning and design applications (e.g., it is used in the XCON/R1 system for configuring VAX machines), and for data interpretation and monitoring. Another useful mode of inference was found to be consequent, or hypothesis-driven, reasoning, better known as *backward chaining*. Here the system may have several hypotheses which it will try to validate by querying for evidence. The control scheme is basically to scan the RHS's of rules rather than the LHS's. Rules which conclude the current goal hypothesis are initially put on a

list, and one of these rules is then selected, typically by some heuristic scheme. The LHS propositions of this selected rule are established as sub-goals or intermediate hypotheses, and backward chaining occurs recursively until the system arrives at a proposition which can be determined by the external environment, e.g., by performing a test or querying the user. The best-known example of backward-chaining is the MYCIN system, which works backwards from hypotheses of infections and treatments to ask the user for relevant information. In general a system may have to perform a judicious combination of both forward and backward chaining since many problems fit both categories, i.e., there is both initial data and the need to acquire more.

The advantages of the rule-based approach are based on the fact that the knowledge is explicitly represented, is easily accessible and modular, and is directed by a simple, implicit and elegant control scheme. Successful applications, such as the MYCIN [68] and PROSPECTOR [66] systems, bear testimony to the practicality of the rule-based approach.

2.2.3 Problems and issues in rule-based systems

So far we have focussed on the advantages of the rule-based representation. But our attention now turns to disadvantages and issues in rule-based systems as currently implemented. In particular we identify three areas which form major stumbling blocks for the application of what is obviously a promising paradigm. The three problem areas are knowledge acquisition, modelling of uncertainty, and unpredictability of control.

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 [107]. The situation has become known as the "knowledge acquisition bottleneck," as the quality of the expert system is limited by the quality of this interviewing process. Much attention [108, 109] has been directed towards using psychological techniques for the interview process. Protocol analysis [110] and repertory grid techniques [111] are among the more widely

favoured techniques. However, when it comes to acquiring knowledge involving *uncertainty*, such as the elicitation of subjective probabilities, there is no way around the problem, as it is well known that humans are not very good at giving consistent information of this form [112].

Our basic approach will be to focus on problems where initial domain data exist, and to use this data to *automate* the knowledge acquisition process, i.e., learn the rules directly from the data. In many problems suited to expert system applications there exist large amounts of domain data (e.g., local area network management or computer vision) and little human expertise. The role of the expert in the automated process will change considerably. He/she can focus on structuring, quantising and interpreting the data with the knowledge engineer. This approach does not preclude the case where little or no data may exist, e.g., medical applications. In such instances the expert can provide small sample data in the form of particular cases, which can be transformed into subjective probability estimates using the appropriate statistical techniques. The notion of automated knowledge acquisition is already receiving some attention but existing tools often use non-statistical approaches [113, 114] and use restrictive knowledge representation media such as decision trees [115, 116, 117]. We will investigate theoretically sound and practical techniques which learn general rule-based representations using probabilistic models.

The second problem with current expert systems is with regard to the modelling of uncertainty. The basic production system model described earlier is fundamentally *logical* in nature. Very often, however, rules are not logical but uncertain in the sense that A implies B with some degree of certainty, e.g., if the sky is cloudy then it will rain with some probability. As we mentioned earlier, prior work in this area has deviated somewhat from basic probability theory, and it can safely be said that no-one has developed a rigorous and robust probabilistic inference scheme using rules. We would like to retain the basic advantages of the production system architecture (explicit knowledge representation and implicit control) as far as possible. To do this will require breaking new ground in rule-based probabilistic inference.

The third main problem is the lack of a well-defined theory of inference control. Current schemes for choosing the direction of reasoning (i.e., choosing which rule to fire next) are rather ad hoc and qualitative in nature. Operating under the paradigm of "fitting the

model to the expert," system designers frequently have to add extra rules or "tweak" their heuristic control scheme in a domain-dependent manner, in order to get the inference to mimic the expert. The end result is that control becomes less and less implicit and more algorithmic in nature with the resulting software complexity problems. Making rule-bases mimic algorithms is a bad idea. We adopt a different paradigm, that of "fitting the expert to the model." By defining a theoretical domain-independent model of rational behaviour based on rules, we will reduce the system design problem to defining the knowledge base, as our theoretical model of behaviour will implicitly control the reasoning.

In essence we will replace the qualitative approach to control with a quantitative one. In fact one can identify this as the common underlying problem in all three areas, namely the lack of a quantitative model. Expert system design techniques have inherited from artificial intelligence the usual preference for symbolic modelling rather than quantitative rigour. The future application of expert systems will require an engineering approach to design, particularly if such systems are to outperform standard solutions which are perhaps less imaginative but at least have the virtue of being well-understood.

2.3 Quantitative modelling of rules

In this section we develop a quantitative measure for the goodness of a rule. A rule is considered to be of the form

$$\text{If } \mathbf{Y} = y \text{ then } \mathbf{X} = x \text{ with probability } p$$

where $\mathbf{X}$ and $\mathbf{Y}$ are two random variables with "x" and "y" being values in their respective discrete alphabets. We restrict the right-hand expression to being a single value-assignment expression while the left-hand side may be a conjunction of such expressions. This is the commonly accepted definition of a rule.

In particular, note that this definition of a rule only involves a single left-hand side proposition, i.e., it only deals with what $\mathbf{Y} = y$ tells us about $\mathbf{X}$ and does not make any reference to the event $\mathbf{Y} \neq y$. This is consistent with the cognitive science definition of a rule where an expert specifies the relation of a particular event to a hypothesis $\mathbf{X}$ but is unwilling to render any information about the complement of this event, or component events in the complement event set, e.g., $\mathbf{Y} = y_2$. Shortliffe and Buchanan ([118], p.239) argue that the expert who volunteers rules in this manner is either irrational or else is not giving probabilistic information. They subsequently develop a pseudo-probabilistic theory of certainty factors to deal with the latter case. Duda et al. [119] define rules using a likelihood ratio defined as

$$\lambda = \frac{p(\mathbf{X} = x | \mathbf{Y} = y)}{p(\mathbf{X} = x | \mathbf{Y} \neq y)}. \tag{2-1}$$

As we shall see later, both of these schemes lead to problems in terms of probabilistic inference. The argument we will propose here is that the simple form of a probabilistic rule defined above is a good model from a cognitive standpoint of an expert's statement about $\mathbf{Y} = y$ and $\mathbf{X}$ in terms of subjective probabilities and, as such, should not be abandoned. We shall see later how to develop models for both probabilistic inference and induction, based on such rules. The rationale for choosing this definition of a rule is fundamental to the entire discussion.

The problem of quantifying rule "goodness" is not trivial. For example if p is near, or equal to, 1, does that mean that the rule is good ? Not if one considers the example

If *the animal is a kangaroo* then *it does not drink beer* with probability = 1

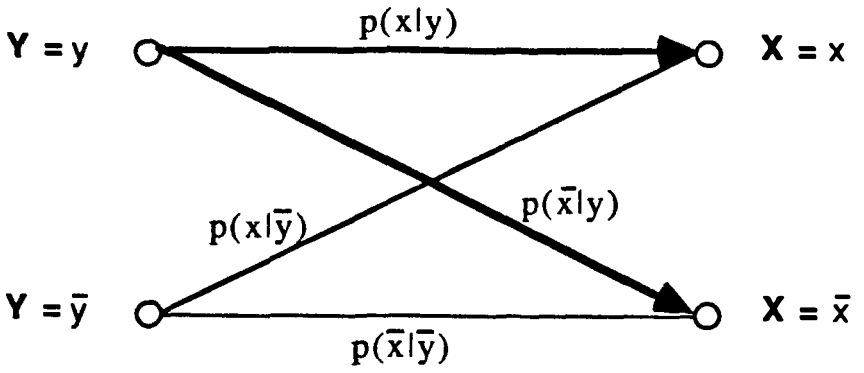


Figure 2.1: A rule can be interpreted as part of a communications channel.

Although $p = 1$ this cannot be viewed as a good rule, at least in most parts of the world. It is reliable but it is simply not useful, given the fact that we know that animals in general are not beer consumers, independent of whether the animal is a kangaroo or not. On the other hand a rule such as

If *the animal is blue* then *it is a bird* with probability $= 0.6$

is quite unreliable but in fact it could be quite useful. Hence, whatever measure we choose to represent the 'goodness' of a rule must at least be a function of the usefulness and the reliability of the rule, although we have not specified how to measure either parameter.

An appropriate starting point in defining the "goodness" of a rule is to define the information content of such a rule. Later we shall see that the information content is indeed an appropriate "goodness" measure, and that it provides the basis for implementing very general rule preference measures. Initially, however, we will focus on the problem of defining a *basic* information measure. Let us consider the variables $\mathbf{Y}$ and $\mathbf{X}$ as the input and output of a discrete memoryless channel, as shown in Figure 2.1. A *rule* corresponds to a particular input event $\mathbf{Y} = y$, rather than the average over all input events as is normally defined, and p, the rule probability, is the transition probability $p(\mathbf{X} = x|\mathbf{Y} = y)$. Let us define $f(\mathbf{X}, \mathbf{Y} = y)$ as the *instantaneous* information that the event $\mathbf{Y} = y$ provides about $\mathbf{X}$, i.e., the information that we receive about $\mathbf{X}$ given that $\mathbf{Y} = y$ has occurred. Similarly we define $F(\mathbf{X}, \mathbf{Y} = y)$ as the average information that the event $\mathbf{Y} = y$ provides about $\mathbf{X}$,

where

$$F(\mathbf{X}, \mathbf{Y} = y) = p(\mathbf{Y} = y).f(\mathbf{X}, \mathbf{Y} = y) \ . \qquad (2\text{–}2)$$

Clearly $F(\mathbf{X}, \mathbf{Y} = y)$ is an *average* in the sense that the information contribution from the other inputs is assumed to be zero. The motivation for this definition will become clearer as we proceed. (The definition expresses the fact that the average information content of a rule is the probability that the left-hand side of the rule is true, multiplied by the instantaneous information).

2.4 Defining the information content of a rule

The instantaneous information is the information content of the rule given that the left-hand side is true. This is the quantity we must define. A reasonable requirement to make (the interested reader can refer to Shannon's original paper [120] for a complete discussion), is that

$$E_y[f(\mathbf{X};\mathbf{Y}=y)] = I(\mathbf{X};\mathbf{Y}) \tag{2-3}$$

where E_y denotes the expectation with respect to the random variable $\mathbf{Y}$. Blachman has shown [121] that $f(\mathbf{X};\mathbf{Y}=y)$ as defined in Equation (2-3) is not unique. In his paper he proposes 2 candidates which satisfy Equation (2-3). By definition, the 2 functions he defined are candidates for our definition of the instantaneous information content. We shall refer to these 2 functions as the *i-measure*, $i(\mathbf{X};\mathbf{Y}=y)$, and the *j-measure*, $j(\mathbf{X};\mathbf{Y}=y)$, where

$$\begin{aligned} i(\mathbf{X};\mathbf{Y}=y) &= H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y}=y) \\ &= \sum_x p(x)\log\left(\frac{1}{p(x)}\right) - \sum_x p(x|y)\log\left(\frac{1}{p(x|y)}\right) \end{aligned} \tag{2-4}$$

and

$$j(\mathbf{X};\mathbf{Y}=y) = \sum_x p(x|y).\log\left(\frac{p(x|y)}{p(x)}\right) . \tag{2-5}$$

These two measures have quite different interpretations. In words, the i-measure is the *change in the average information* required to specify $\mathbf{X}$ given the event $\mathbf{Y}=y$, while the j-measure is the *average change in the information* required to specify $\mathbf{X}$ given $\mathbf{Y}=y$. The difference is subtle, yet significant enough that the j-measure is always non-negative, while the i-measure may be either negative or positive. In fact Blachman [121] has proven that the j-measure is unique as a non-negative information measure which satisfies Equation (2-3), i.e., it is the *only* non-negative measure.

For convenience, from this point onwards we adopt the notation that $p(y)$ stands for $p(\mathbf{Y}=y)$, etc. The property of additivity (Fano, [122]) is a useful property of the mutual average information, $I(\mathbf{X};\mathbf{Y})$, i.e.,

$$I(\mathbf{X};\mathbf{Y},\mathbf{Z}) = I(\mathbf{X};\mathbf{Y}) + I(\mathbf{X};\mathbf{Z}|\mathbf{Y}) . \tag{2-6}$$

The corresponding additivity property of $f(\mathbf{X};y)$ can be defined as

$$f(\mathbf{X};y,z) = f(\mathbf{X};y) + f(\mathbf{X};z|y) . \tag{2-7}$$

The i-measure is additive in this manner since

$$\begin{aligned} i(\mathbf{X};y,z) &= H(\mathbf{X}) - H(\mathbf{X}|y,z) \qquad \text{(by definition)} \\ &= H(\mathbf{X}) - H(\mathbf{X}|y) + H(\mathbf{X}|y) - H(\mathbf{X}|y,z) \\ &= i(\mathbf{X};\mathbf{Y}=y) - i(\mathbf{X};z|y) \ . \end{aligned} \tag{2-8}$$

The j-measure, however, is not additive as can be seen if we choose $\mathbf{Z}$ such that $p(x|y,z) = p(x)$. Hence,

$$j(\mathbf{X};y,z) = 0 \ , \tag{2-9}$$

while

$$j(\mathbf{X};y) + j(\mathbf{X};z|y) \neq 0 \tag{2-10}$$

since the two terms on the left-hand side of 2.10 are non-zero in general.

While additivity is an advantageous property, it is outweighed by the disadvantage of negativity. Blachman showed that the i-measure is the only additive measure which satisfies Equation (2–3). But the i-measure is not non-negative as can be seen from the first line of Equation (2–4), since the *a posteriori* entropy $H(\mathbf{X}|y)$ may be greater than the *a priori* entropy $H(\mathbf{X})$. For instance, if $p(x) = 0.9$ and $p(x|y) = 0.5$, then $i(\mathbf{X};\mathbf{Y}=y) = -0.531$ bits. While negative information measures are awkward to deal with, one can easily envisage schemes which circumvent the problem.

Lemma :

$$\text{If } \ i(\mathbf{X};\mathbf{Y}=y) < 0 \text{ then } I(\mathbf{X};\bar{y}) \geq |I(\mathbf{X};y)| \tag{2-11}$$

where $I(\mathbf{X};\bar{y}) = p(y).i(\mathbf{X};\bar{y})$.

The proof follows directly from the fact that

$$I(\mathbf{X};\bar{y}) + I(\mathbf{X};y) = I(\mathbf{X};\mathbf{Y}) \geq 0 \ . \tag{2-12}$$

This lemma states that if the instantaneous i-measure is negative, then the average information content of the complementary rule, the rule linking $\mathbf{Y} = \bar{y}$ and $\mathbf{X}$, must be positive. Hence, one can argue that, in practice, rules with negative information (either average or instantaneous) will never occur in isolation, since such rules imply the existence of a more informative complementary rule. The two rules can be used together as a regular channel.

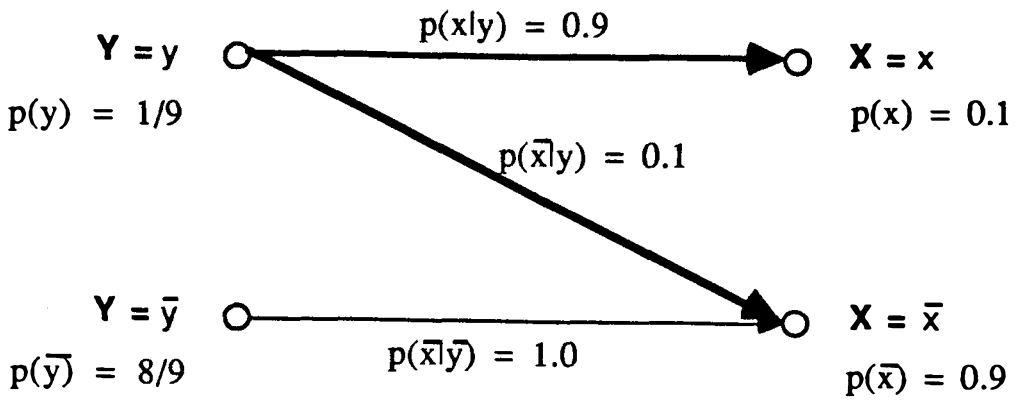


Figure 2.2: Although the occurrence of the event y reverses our "belief" in the state of $\mathbf{X}$, the i-measure, $i(\mathbf{X}; y) = 0$.

However, despite such arguments, there remains a fundamental problem with using measures which are not non-negative. We see that $i(\mathbf{X}; \mathbf{Y} = y)$ can be equal to zero even if $p(x|y) \neq p(x)$, e.g., $p(x|y) = p(\bar{x})$. In other words the i-measure is zero if the transition probabilities in the channel, for a given input, form a permutation of the output probabilities. An appropriate title for this phenomenon is the *information paradox*. Consider the channel in Figure 2.2 (known in communications as a *Z-channel* for obvious reasons). The event $\mathbf{Y} = y$ reverses our belief in the state of the variable $\mathbf{X}$, yet because the i-measure only measures the difference between the *a priori* and *a posteriori* entropies, we get that

$$i(\mathbf{X}; \mathbf{Y} = y) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y} = y) = 0.47 - 0.47 = 0 \text{ bits.}$$

Hence, although it causes our belief in the state of $\mathbf{X}$ to change considerably, the information content of knowing $\mathbf{Y} = y$ is zero as given by the i-measure. This is an example of a fundamental difference between using channel models for cognitive modelling, and using them for standard communication purposes. In the case of the latter, we do not distinguish between individual random events, except in terms of their attached probabilities of occurrence. The entropy of a discrete random variable is the same independent of which probabilities are assigned to which events in the event space of the variable. No significance is attached to particular events other than their probability of occurrence. However, in cognitive modeling this is not true. Individual events may have a semantic "meaning" or importance attached, independent of the event's probability. In a rule-based model, for

example, a particular output proposition $\mathbf{X} = x$ may be irrelevant in a particular context, e.g., of the known rules, none may have $\mathbf{X} = x$ in their left-hand sides. On the other hand, $\mathbf{X} = \bar{x}$ may be particularly useful in the same context. Hence, measures such as the i-measure, which represent the elements in the event space as indistinguishable, are inappropriate for cognitive modelling. On these grounds, the i-measure must be rejected for our purposes.

The j-measure does not suffer from this problem since it is non-negative. Let us now prove this fact, and in addition state and prove some other properties of the j-measure which will provide insight into the more general nature of the measure.

Theorem 2.1:

$$j(\mathbf{X}; \mathbf{Y} = y) \geq 0 \tag{2–13}$$

with equality iff $p(x|y) = p(x)$ for all x.

Proof:

$$\begin{aligned} j(\mathbf{X}; \mathbf{Y} = y) &= -\sum_x p(x|y).\log_2\left(\frac{p(x)}{p(x|y)}\right) \\ &\geq -\sum_x p(x|y).\left(\frac{p(x)}{p(x|y)} - 1\right).\log_2(e) \qquad (2–14) \\ &= -\sum_x (p(x) - p(x|y)).\log_2(e) \\ &= 0\ . \qquad (2–15) \end{aligned}$$

Since $\log(x) = x - 1$ iff $x = 1$ then the inequality in Equation (2–14) is satisfied iff $p(x) = p(x|y)$ for all x.

∎

Theorem 2.2:

$j(\mathbf{X}; \mathbf{Y} = y)$ is convex $\cup$ in the transition probabilities $p(x|y)$

Proof:

The case of interest correponds to a rule, where the only two events of concern are x and $\bar{x}$. For notational convenience, let

$$p = p(x|y), \quad p_1 = p_1(x|y), \quad p_2 = p_2(x|y) \tag{2–16}$$

and let

$$p = \alpha.p_1 + (1-\alpha).p_2, \qquad \text{where } 0 \le \alpha \le 1. \tag{2-17}$$

By definition $j(p) = j(\mathbf{X};\mathbf{Y}=y) = \sum_x p(x|y).\log(\frac{p(x|y)}{p(x)})$

Hence, we have that

$$\begin{aligned} j(p) - \alpha.j(p_1) - (1-\alpha)j(p_2) & \\ &= \sum_x p.\log(\frac{p}{p(x)}) - \alpha\sum_x p_1.\log(\frac{p_1}{p(x)}) - (1-\alpha)\sum_x p_2.\log(\frac{p_2}{p(x)}) && (2\text{–}18)\\ &= \sum_x(\alpha.p_1 + (1-\alpha)p_2).\log(\frac{p}{p(x)}) & \\ &\quad - \alpha\sum_x p_1.\log(\frac{p_1}{p(x)}) - (1-\alpha)\sum_x p_2.\log(\frac{p_2}{p(x)}) && (2\text{–}19)\\ &= \alpha.\sum_x p_1.\log(\frac{p}{p_1}) + (1-\alpha).\sum_x p_2.\log(\frac{p}{p_2}) && (2\text{–}20) \end{aligned}$$

By Jensen's inequality (McEliece [123], p.200),

$$E_x[f(x)] \le f(E[x]) \text{ if } f(x) \text{ is convex } \cup,$$

Since $p \ge p_1$ and $p \ge p_2$, then $\log(\frac{p}{p_1})$ and $\log(\frac{p}{p_2})$ are both convex $\cup$. Hence, from Equation (2–20) we get that

$$\begin{aligned} j(p) - \alpha.j(p_1) - (1-\alpha)j(p_2) &\le \alpha.\log(\sum_x p_1.(\frac{p}{p_1})) + (1-\alpha).\log(\sum_x p_2.(\frac{p}{p_2})) && (2\text{–}21)\\ &= 0. && (2\text{–}22) \end{aligned}$$

By Equation (2–22), $j(p)$ is convex $\cup$ in $p = p(x|y)$.

∎

Corollary 2.1:

$$j(\mathbf{X};\mathbf{Y}=y) \le \max_x\{\log(\frac{1}{p(x)})\} \tag{2-23}$$

Proof: Follows immediately from the convexity of $j(\mathbf{X};\mathbf{Y}=y)$ in $p(x|y)$.

For our purposes, $j(\mathbf{X};\mathbf{Y}=y)$ has the special form,

$$j(\mathbf{X};\mathbf{Y}=y) = p(x|y).\log(\frac{p(x|y)}{p(x)}) + (1-p(x|y)).\log(\frac{(1-p(x|y))}{(1-p(x))}) \tag{2-24}$$

since a rule only gives us information about the event $\mathbf{X} = x$. Because of this form we can plot some typical curves for j(**X**;**Y**=y)as shown in Figure 2.3. From Theorem 2.1 we can

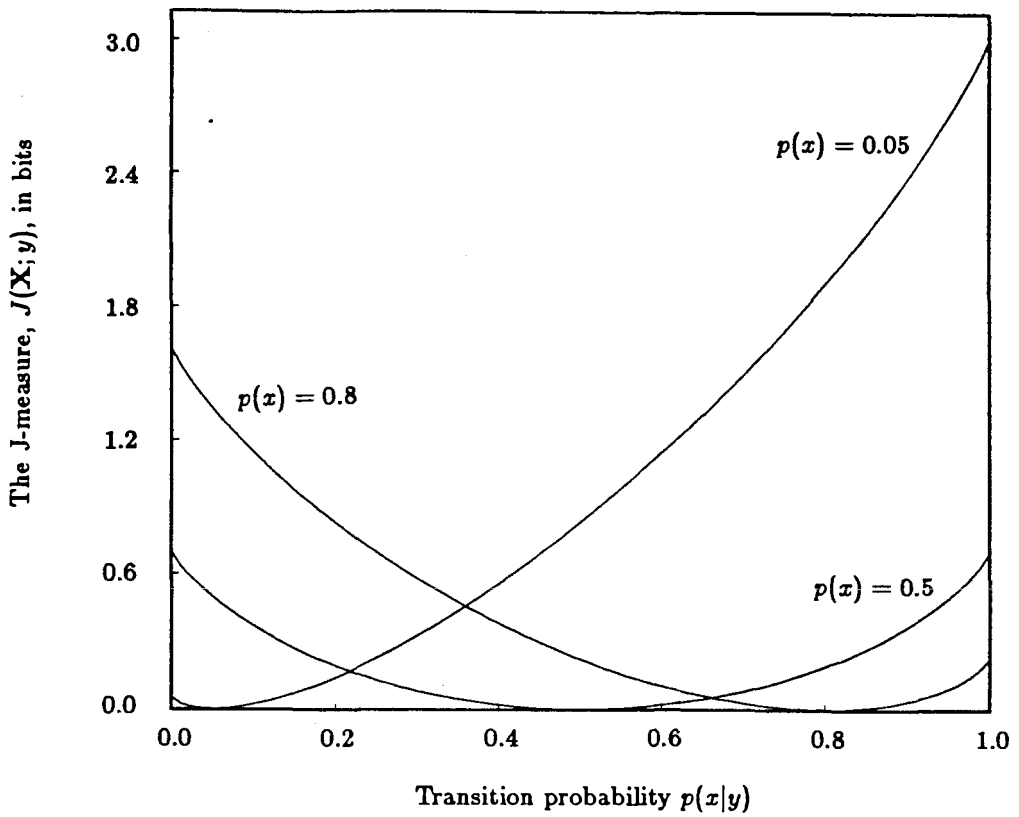


Figure 2.3: A plot of the j-measure against as $p(x|y)$ ranges from 0 to 1, for 3 values of $p(x)$: (i) $p(x) = 0.05$, (ii) $p(x) = 0.5$, and (iii) $p(x) = 0.8$. Note the difference in the curves, reflecting the dependence of $j(\mathbf{X};y)$ on $p(x)$

state that the j-measure is minimised only when $p(x|y) = p(x)$ (in terms of a rule, the *a posteriori* probability for $\mathbf{X} = x$ is the same as the *a priori* probability). As $p(x|y)$ moves away from $p(x)$, either towards 0 or 1, the j-measure is a monotonically increasing function (from Theorem 2.2), ranging from 0 to either $\log(\frac{1}{p(x)})$ or $\log(\frac{1}{(1-p(x))})$. In this sense it is a well-behaved continuous distance measure between the case of zero information and the case where the event $\mathbf{Y} = y$ completely specifies $\mathbf{X}$. Note that in the latter case, where $\mathbf{Y} = y$ completely specifies $\mathbf{X}$, $j(\mathbf{X};\mathbf{Y} = y)$ is equal to the self-information of either $p(x)$ or $p(\bar{x})$. Hence, from an information-theoretic standpoint, the j-measure exhibits the correct behaviour for a rule information measure over all values of $p(x|y)$ and $p(x)$.

A further useful property of the j-measure is the *decomposition* property. Let us say that $\mathbf{X}$ is *not* a binary variable, but m-ary. Our goal is to calculate the average change in information over *every* event in $\mathbf{X}$, given the event $\mathbf{Y} = y$. The j-measure, as we have defined it (for a rule), apparently calculates the average change in information as if $\mathbf{X}$ were binary. Let the m-ary alphabet of $\mathbf{X}$ be defined as $\{x_1, x_2, \ldots, x_m\}$ and let the rule be defined on $\mathbf{X} = x_1$, i.e., we know $p = p(x_1|y)$. Hence, the total probability mass of the other events $(x_2, \ldots, x_m)$ is changed from $1 - p(x_1)$ to $1 - p$, in the light of $\mathbf{Y} = y$. As such, the *a posteriori* probability of each event x_i, $2 \leq i \leq m$, must be equal to $\alpha.p(x_i)$ where $\alpha = \frac{(1-p)}{(1-p(x_1))}$. Hence, we can define

$$m(\mathbf{X}; y) = p.\log(\frac{p}{p(x_1)}) + \sum_{i=2}^{m} \alpha.p(x_i).\log(\frac{\alpha.p(x_i)}{p(x_i)}) \tag{2-25}$$

as the average change in information required to specify $\mathbf{X}$, where the averaging is carried out over all events in $\mathbf{X}$.

Theorem 2.3:

$$m(\mathbf{X}; y) = j(\mathbf{X}; y) \tag{2-26}$$

Proof:

$$m(\mathbf{X}; y) = p.\log\left(\frac{p}{p(x_1)}\right) + \sum_{i=2}^{m} p(x_i).\frac{(1-p)}{(1-p(x_1))}.\log(\frac{(1-p)}{(1-p(x_1))}) \tag{2-27}$$

$$= p.\log\left(\frac{p}{p(x_1)}\right) + \frac{(1-p)}{(1-p(x_1))}.\log\left(\frac{(1-p)}{(1-p(x_1))}\right).\sum_{i=2}^{m} p(x_i) \tag{2-28}$$

$$= p.\log\left(\frac{p}{p(x_1)}\right) + (1-p)\log\left(\frac{(1-p)}{(1-p(x_1))}\right) \tag{2-29}$$

$$= j(\mathbf{X}; y)\ . \tag{2-30}$$

∎

The consequence of this result is that the j-measure is decomposable into component terms, a fact which cannot be fully appreciated until we discuss the issue of defining more general rule "goodness" measures based on the j-measure.

A further point worth making about the j-measure at this juncture is that it appears in the information-theoretic literature under various guises. For instance it can be viewed

as a special case of the *cross-entropy* (Shore and Johnson [124]) or the *divergence* (Kullback [125]), a measure which defines the information-theoretic similarity between two probability distributions. In our case, the two distributions are defined on the binary variables $\mathbf{X}$ and $\mathbf{X}|y$. It is also known as the *binary discrimination* (Blahut [126, p.20]) between two hypotheses H_0 and H_1. More specifically it defines the expected value of the log-likelihood ratio between H_0 and H_1, *with respect to* H_0. For our purposes the log-likelihoods are defined as

$$\Lambda_0 = \log(\frac{p(x|y)}{p(x)}), \qquad \Lambda_1 = \log(\frac{p(\bar{x}|y)}{p(\bar{x})}) \tag{2-31}$$

where H_0 is the hypothesis that $\mathbf{X}$ is dependent on the event $\mathbf{Y} = y$, while H_1 is the hypothesis that $\mathbf{X}$ is independent of $\mathbf{Y} = y$. Hence,

$$j(\mathbf{X}; y) = p(x|y).\Lambda_0 + p(\bar{x}|y).\Lambda_1 \ . \tag{2-32}$$

This particular interpretation of the j-measure as a hypothesis test will prove useful when we discuss the problem of probabilistic inference.

From Equation (2–2) we can define the *average* information content, $J(\mathbf{X}; y)$, as,

$$J(\mathbf{X}; y) = p(y)j(\mathbf{X}; y) \ . \tag{2-33}$$

As we proceed we will find that this *average* information content, to be referred to as the J-measure, is more relevant to our discussion than $j(\mathbf{X}; y)$. Hence, from this point onwards we will focus on $J(\mathbf{X}; y)$ and its properties, referring to $j(\mathbf{X}; y)$ when appropriate. In addition, unless otherwise stated, it can be assumed that binary form of both measures, as defined by Equation (2–24)., is being used. Since $J(\mathbf{X}; y)$ and $j(\mathbf{X}; y)$ are related by the multiplicative factor $p(y)$, we can easily derive equivalent properties of one measure from the other. Based on our earlier findings, we can determine the equivalent information-theoretic properties of $J(\mathbf{X}; y)$ as follows:

Corollary 2.2:

$$J(\mathbf{X}; y) \geq 0 \tag{2-34}$$

Proof: Follows directly from Theorem 2.1.

Corollary 2.3:

$$J(\mathbf{X}; y) \text{ is convex } \cup \text{ in } p(x|y), \text{ for given } p(y).$$

Proof: Follows directly from Theorem 2.2.

Theorem 2.4:

$$J(\mathbf{X};y) \leq p(y).\max\{\log(\frac{1}{\max\{p(x),p(y)\}}),\log(\frac{1}{\max\{p(\bar{x}),p(y)\}})\} \tag{2–35}$$

Proof:

By corollary 2.3, $J(\mathbf{X};y)$ is convex $\cup$ in $p(x|y)$ for given $p(y)$. Hence, the maxima of $J(\mathbf{X};y)$ are located at the two extrema defined by $p(x|y)$. For $j(\mathbf{X};y)$ these extreme points are 0 and 1. However for $J(\mathbf{X};y)$ there are three cases which can occur:

(i) $p(x) > p(y) > p(\bar{x})$: There are no restrictions on $p(x|y)$ and so $J(\mathbf{X};y) \leq p(y)\log(\frac{1}{p(x)})$.

(ii) $p(y) \geq p(x)$: In this case $p(x|y) \leq \frac{p(x)}{p(y)}$.

(iii) $p(y) \geq p(\bar{x})$: In this case $p(x|y) \geq \frac{p(\bar{x})}{p(y)}$.

Hence, in general, the extrema are defined by p_1 and p_2 where,

$$p_1 = \min\{1, \frac{p(x)}{p(y)}\}, \qquad p_2 = \min\{1, \frac{p(\bar{x})}{p(y)}\}\ , \tag{2–36}$$

and the values at these extrema are

$$J_1 = p(y).\log(\frac{\min\{1, \frac{p(x)}{p(y)}\}}{p(x)}), \qquad J_2 = p(y).\log(\frac{\min\{1, \frac{p(\bar{x})}{p(y)}\}}{p(\bar{x})})\ , \tag{2–37}$$

or equivalently

$$J_1 = p(y).\log\Big(\min\{\frac{1}{p(x)}, \frac{1}{p(y)}\}\Big), \qquad J_2 = p(y).\log\Big(\min\{\frac{1}{p(\bar{x})}, \frac{1}{p(y)}\}\Big)\ , \tag{2–38}$$

which in turn is equivalent to Equation (2–35).

∎

In practice we note that since $\mathbf{Y} = y$ is often the conjunction of several events (as the left-hand side of a rule), one is likely to have $p(x) \geq p(y)$. The final property we defined for $j(\mathbf{X};y)$, the decomposition property, also applies to $J(\mathbf{X};y)$ since multiplication by $p(y)$ does not change the underlying nature of the summation involved in establishing this property.

To conclude this section we can say that the both $j(\mathbf{X};y)$ and $J(\mathbf{X};y)$ possess a wide range of desirable properties from an information-theoretic standpoint, and can be interpreted in a number of ways, all of which support the basic notion of quantifying the information content of a probabilistic rule. The next logical step is to analyse the measures from

a cognitive modelling standpoint. It is all very well to define $j(\mathbf{X};y)$ and $J(\mathbf{X};y)$ based on information-theoretic considerations, but do these measures support basic theoretical cognitive principles in terms of modelling human reasoning processes? This point will be addressed in two stages, namely, learning (or induction) and inference. We will be focussing in particular on the J-measure.

2.5. Theoretical induction properties of the J-measure

2.5.1 Properties of the J-measure as a hypothesis preference criterion

Thus far we have concentrated on a very basic rule information measure and its mathematical properties. We will now consider properties of this measure as they relate to previous work in the area of learning. Specifically we will develop theoretical interpretations for the measure both as hypothesis preference criterion for computational learning models, and as a quantitative induction criterion for cognitive learning models. The overall motivation and direction will be to establish the validity of our new measure, the J-measure, with respect to these areas since most of the existing theoretical work in machine learning is based on either or both of these two areas of computational and cognitive learning models.

Firstly, let us examine the properties of $J(\mathbf{X}; y)$ with respect to *computational learning*. First we must define some notation and terminology. Induction can be viewed as a search for hypotheses (restricted to some *hypothesis space*) to account for a set of instances or examples which are often assumed to be restricted to some *instance space*. For our purposes, the hypothesis space is restricted in general to the conjunctive propositions in the discrete space defined by the Cartesian product of the sample spaces of the individual attributes or variables. For a *given concept* (in our terminology, a particular variable) the *hypothesis* space is defined as the Cartesian product of the sample spaces of the other $n-1$ individual attributes or variables (assuming n variables in total), whereas the *instance* space is defined over the entire n-dimensional product space. The general learning problem consists of being given positive and negative instances of some *concept* and trying to find a hypothesis in the hypothesis space which "best" describes this concept. Let v be any positive instance in the instance space for some concept. Symbolic algorithms try to find a deterministic mapping, or a Boolean function F, from the instance space to the hypothesis space, to describe the concept, i.e., we seek an F such that $F(v) = 1$ for all v, where F is in the hypothesis space. The statistical approach, however, tries to find a probabilistic mapping, or a probability distribution, between the two spaces, i.e., $prob(F(v) = 1) \geq 1 - \delta$, where δ is as close to 0 as possible but may be lower bounded by a fundamental parameter of the hypothesis space, such as the Bayes' risk.

Let us consider first the problem of searching for a hypothesis for a *given* concept. Then we will discuss the more general and interesting problem of general concept learning

or generalised rule-induction. We will examine the nature of the J-measure as a basic preference measure among competing hypotheses. There appears to be a general consensus that the two primary criteria for evaluating a hypothesis are the ***simplicity*** of the hypothesis and the ***"goodness-of-fit"*** between the hypothesis and the data (Angluin and Smith [127], Gaines [128] and Michalski [129]). The problem is to combine these two criteria into a single measure such that the hypotheses can be ordered. In terms of the probabilistic rules defined earlier, let us interpret the event $\mathbf{X} = x$ as the concept $F(v) = 1$ to be learned and the event (possibly conjunctive) $\mathbf{Y} = y$ as the hypothesis describing this concept.

The J-measure is the product of two terms. The first, $p(\mathbf{Y} = y)$, is the probability that the hypothesis will occur and, as such, can be interpreted as a measure of hypothesis *simplicity*. Angluin and Smith [127] also mention this idea, that the probability of a hypothesis is an appropriate indication of its simplicity. Theorem 2.4 has an interesting interpretation in this sense. It tells us that if a hypothesis is simpler than a particular concept, then it cannot completely specify that concept. Symbolic algorithms use more ad hoc techniques to determine the simplicity of a hypothesis, such as enumerating the number of basic propositions which make up a conjunctive hypothesis (Angluin and Smith [127]). Such techniques may work in given domains but lack generality. In contrast, the probabilistic criterion for simplicity is perfectly general.

The second term making up $J(\mathbf{X};y)$ is $j(\mathbf{X};y)$. As we have seen in the last section, $j(\mathbf{X};y)$ can be interpreted as the cross-entropy of $\mathbf{X}$ with the variable '$\mathbf{X}$ conditioned on the event $\mathbf{Y} = y$'. Cross-entropy is well known as a goodness of fit measure between two distributions (Shore and Johnson [125]). It can be interpreted as a distance measure where 'distance' corresponds to the amount of information required to specify a random variable. It is frequently used to find the conditional distribution which most closely agrees with the original distribution. The cross-entropy is zero if and only if the two distributions are exactly equal. From corollary 2.1, when the two distributions are as far apart as possible (in an information-theoretic sense), the cross-entropy attains its maximum value of

$$j(\mathbf{X};\mathbf{Y} = y) \;=\; \log \frac{1}{p(\mathbf{X} = x_i)} \;, \tag{2-39}$$

for some x_i. For our purposes we must be very careful to interpret what we mean by goodness-of-fit. In particular we shall see that maximal goodness-of-fit corresponds to maximising, rather than minimising, the cross-entropy. In the probabilistic manner for

which the problem is defined we have an *a priori* value for $p(\mathbf{X} = x) = p(F(v) = 1)$. This represents our best probability estimate as to whether an arbitrary instance v is contained in the concept or not, without any hypothesis. By introducing a hypothesis we now have an *a posteriori* value, $p(F(v) = 1|\mathbf{Y} = y)$. Without loss of generality we can assume that

$$p(F(v) = 1|\mathbf{Y} = y) \geq p(F(v) = 1) \tag{2-40}$$

since otherwise we can define $F(v) = 0$ as the hypothesis of interest. The measure attains its maximum value (for a given F and $\mathbf{Y}$) if and only if

$$p(F(v) = 1|\mathbf{Y} = y) = 1, \tag{2-41}$$

whereas the measure is minimised if and only if

$$p(F(v) = 1|\mathbf{Y} = y) = p(F(v) = 1), \tag{2-42}$$

i.e., the hypothesis is no better than a random guess based on *a priori* probabilities. Intermediate *a posteriori* values, $p(F(v) = 1) < p(F(v) = 1|\mathbf{Y} = y) < 1$, provide a monotonic measure of the information-theoretic distance between the uncertainty of the random guess without the hypothesis, and complete specification. Hence, the "best" goodness-of-fit occurs when the hypothesis specifies $\mathbf{X}$ exactly, or equivalently when the cross-entropy is maximised. Given two hypotheses, $\mathbf{Y} = y$ and $\mathbf{Z} = z$, and assuming without loss of generality that $p(\mathbf{X} = x|\mathbf{Z} = z) > p(\mathbf{X} = x)$, then it can easily be shown that

$$j(\mathbf{X}; \mathbf{Y} = y) > j(\mathbf{X}; \mathbf{Z} = z) \tag{2-43}$$

if and only if

$$p(\mathbf{X} = x|\mathbf{Y} = y) > p(\mathbf{X} = x|\mathbf{Z} = z). \tag{2-44}$$

In this sense $j(\mathbf{X}; \mathbf{Y} = y)$ clearly corresponds to a goodness of fit measure.

Hence, we can conclude that the J-measure trades-off a simplicity component, $p(\mathbf{Y} = y)$, with a goodness-of-fit component, $j(\mathbf{X}; \mathbf{Y} = y)$.

2.5.2 Cognitive aspects of rule induction using the J-measure

Our next step will be to evaluate the J-measure in relation to *cognitive learning* criteria. Holland et al. [130] define four basic operations for the cognitive modelling of the induction of new rules from a set of instances, from a set of existing rules, or more generally, from combinations of both. The operations are *condition-simplifying* generalisation, *instance-based* generalisation, specialization and abduction. We chose this more limited taxonomy (rather than, for example, Dietterich and Michalski's list [131]) since the work of Holland et al. is more directly related to the principles of cognitive science. In this report, *abduction* (generating hypotheses to explain unexpected events) is not dealt with, as it is a sufficiently complex matter to warrant a separate treatment.

Condition-simplifying generalisation:
The basic principle at work here is that rules which have irrelevant conditions on the left-hand side can drop these conditions to become better rules. In a more formal sense, the operation increases the simplicity of the hypothesis. In a good generalisation scheme, this increase in simplicity is traded-off against a change in goodness-of-fit, in order that the overall hypothesis preference measure is increased. Consider that we have a rule with the joint event $\mathbf{Y} = y, \mathbf{Z} = z$ as the left-hand side, where $\mathbf{Y} \neq \mathbf{Z}$. If we drop the condition $\mathbf{Z} = z$ then we have

$$p(\mathbf{Y} = y) \; > \; p(\mathbf{Y} = y, \mathbf{Z} = z) \tag{2–45}$$

so that the simplicity component of the more general rule is always greater. More interestingly consider what happens to the goodness-of-fit $j(\mathbf{X}; \mathbf{Y} = y)$. What is the relationship in general between $j(\mathbf{X}; \mathbf{Y} = y)$ and $j(\mathbf{X}; \mathbf{Y} = y, \mathbf{Z} = z)$? We must look at the fundamental relationship between $p(\mathbf{X} = x | \mathbf{Y} = y)$ and $p(\mathbf{X} = x | \mathbf{Y} = y, \mathbf{Z} = z)$. It can easily be verified that

$$p(\mathbf{X} = x | \mathbf{Y} = y) \; = \; \alpha . p(\mathbf{X} = x | \mathbf{Y} = y, \mathbf{Z} = z) + (1 - \alpha) . p(\mathbf{X} = x | \mathbf{Y} = y, \mathbf{Z} \neq z) \tag{2–46}$$

where

$$\alpha = p(\mathbf{Z} = z | \mathbf{Y} = y). \tag{2–47}$$

Since $0 \leq \alpha \leq 1$, we have that

$$p_1 \leq p(\mathbf{X} = x | \mathbf{Y} = y) \leq p_2 \tag{2–48}$$

where

$$p_1 = \min\{p(\mathbf{X} = x|\mathbf{Y} = y, \mathbf{Z} = z), p(\mathbf{X} = x|\mathbf{Y} = y, \mathbf{Z} \neq z)\} \quad (2\text{–}49)$$

and

$$p_2 = \max\{p(\mathbf{X} = x|\mathbf{Y} = y, \mathbf{Z} = z), p(\mathbf{X} = x|\mathbf{Y} = y, \mathbf{Z} \neq z)\} \; . \quad (2\text{–}50)$$

If $p(\mathbf{X} = x|\mathbf{Y} = y, \mathbf{Z} = z)$ is the maximum of the two, then the transition probability of the more general rule is *less* than that of the original rule, and so by Equations (2–43) and (2–44), the goodness-of-fit of the generalised rule is correspondingly less than that of the original. Note that without loss of generality we are implicitly assuming that $p(x|y) > p(x)$, since otherwise the more general rule implies that $\mathbf{X} = \bar{x}$ is more probable than $\mathbf{X} = x$, which in turn means that we would obtain a better general rule by dropping the $\mathbf{Y} = y$ condition rather than the $\mathbf{Z} = z$ condition, i.e., the same equations would hold but with $\mathbf{Y}$ and $\mathbf{Z}$ reversed. In general, the generalisation step will only increase the J-measure if

$$j(\mathbf{X}; \mathbf{Y} = y) \; > \; \alpha . j(\mathbf{X}; \mathbf{Y} = y, \mathbf{Z} = z) \quad (2\text{–}51)$$

where $\alpha = \frac{p(\mathbf{Y}=y, \mathbf{Z}=z)}{p(\mathbf{Y}=y)}$ as previously defined, i.e., if the fractional increase in simplicity is greater than the fractional decrease in cross-entropy.

Let us consider a very simple example of generalisation. Consider a rule in the reptile domain which says that

If "no.legs" is *true* and "habitat" is *desert* then "is.snake" is *true* with probability 1

where *prob*("is.snake"= *true*) = *prob*("no.legs"= *true*) = 0.3, *prob*("habitat"= *desert*, "no.legs"= *true*) = 0.2. The J-measure of this specialized rule is 0.52 bits. If we generalise this rule to

If "no.legs" is *true* then "is.snake" is *true* with probability 1.

we find that $J_s = 0.35 \; < \; J_g = 0.52$ bits, where J_g and J_s are the information contents of the general and more specialized versions respectively. By dropping the "habitat"= *desert* condition from the specialized rule, we increased the simplicity of the hypothesis by a factor of 1.5, while maintaining the same cross-entropy or goodness-of-fit.

Consider what happens for the same example if we were to drop the "no.legs" condition instead. Let us say that *prob*("is.snake"= *true*, "habitat"= *desert*) = 0.2 and

prob("habitat"= *desert*) = 0.4. For the general rule we then get that p = *prob*("is.snake"= *true*|"habitat"= *desert*) = 0.5 and so the J-measure for this rule becomes

$$J_g = 0.4(\frac{0.2}{0.4}\log(\frac{0.5}{0.3}) + \frac{0.2}{0.4}\log(\frac{0.5}{0.7})) = 0.4(0.3685 - 0.2427) = 0.05 \text{ bits.} \qquad (2\text{–}52)$$

The simplicity increased by a factor of 2, from 0.2 to 0.4, but the goodness-of-fit decreased dramatically, leading to a significant overall decrease in the J-measure for the general rule. The reason for this is the fact that the occurrence of "habitat"= *desert* only increases the probability of "is.snake"= *true* from 0.3 to 0.5, while the inclusion of the "no.legs"= *true* condition completely specifies the "snake" variable. This demonstrates the sensitivity of the J-measure to changes in the transition probability of the rule, particularly when that probability is near an extremum of the convex hull.

Specialisation:
This technique is used to refine hypotheses and is essentially the opposite of generalisation in that a decrease in simplicity is traded-off for an increase in goodness-of-fit in return for a better hypothesis, i.e., a higher preference measure. Specialisation increases the J-measure if and only if

$$j(\mathbf{X}; \mathbf{Y} = y) \; < \; \alpha.j(\mathbf{X}; \mathbf{Y} = y, \mathbf{Z} = z). \qquad (2\text{–}53)$$

We will discuss specialisation at some length in the section to follow since it will be used as the basic mechanism of a proposed induction algorithm. For now we limit our attention to an illustrative example.

Let J_g and J_s be the information contents as defined earlier. Consider an example in the animal domain. The relevant features are "has.wings," "can.fly," and "is.a.penguin," and feature values are restricted to the set {*true*, *false*}. The general rule might be

If "has.wings" is *true* then "can.fly"is *true* with probability 0.9

and the more specialised version might be

If "has.wings" is *true* and "is.penguin" is *false*
then "can.fly" is *true* with probability 1.0.

It is not intuitively obvious which rule is better on the average. If we specify *prob*(*can.fly* = *true*) = 0.27, *prob*(*has.wings* = *true*) = 0.3 and *prob*(*has.wings* = *true*, *is.penguin* =

$false) = 0.27$, then we find that $J_s = 0.51$ bits and $J_g = 0.38$ bits, i.e., the more specialized rule is better in the sense that it provides more information on average. Without a quantitative measure, such as the J-measure, it would be very difficult to rank rules in this fashion.

Instance-based generalisation:

Instance-based generalisation proceeds not from rules but from examples. Most induction algorithms (statistical methods in particular) use this technique as the fundamental rule-generation mechanism. In our discussion so far we have used probabilities without indicating where these probabilities were obtained. Theoretically these probabilities may be assigned to the events in the sample space by any of the relatively well-accepted theories of probability, e.g., subjective estimates (Savage[132]), logical probabilities (Carnap[133]), or based on frequency enumeration. In practice we will prefer the latter if data is available primarily for the reason that it is the most widely trusted technique. When large quantities of sample data *are* available (as for example in vision problems) then we would like to use the maximum-likelihood estimator for p, namely

$$\hat{p} = \frac{r}{N} \tag{2–54}$$

as an estimate for the true p (where r is the number of successes in N random trials). However, if there are very few samples we would prefer to use a more conservative technique such as a maximum entropy estimate. The maximum entropy estimate for a rule transition probability p, in the absence of any constraints, would be 0.5 (assuming that there are only two possible events in the sample space, as we have with $p(x|y)$ and $p(\bar{x}|y)$ for a rule).

One possible approach is to use a point estimator which interpolates between the initial maximum entropy estimate and the maximum likelihood estimate, as the sample size increases. Techniques such as this are an important step in bridging the gap between symbolic and statistical approaches. An example of such an estimator is

$$\hat{p} = \frac{\alpha + r + 1}{\alpha + \beta + N + 2} \tag{2–55}$$

where α and β are parameters of an initial density which can be obtained from initial subjective estimations (Good, [134]). The use of this estimator effectively introduces a sample-size dependent noise term in the channel of Figure 2.1 with the desired effect that the information content of the rule (J-measure) increases with sample-size.

At this point we have clearly shown that the J-measure is a well-defined hypothesis preference criterion and that it satisfies the basic requirements of an induction measure as defined by Holland et al. [130], i.e., it supports the basic inductive mechanisms of condition-simplifying generalisation, instance-based generalisation, and specialization. In the next section we describe a practical implementation of these ideas, namely, the ITRULE algorithm for generalised rule induction.

2.6 ITRULE: an algorithm for generalised rule induction

2.6.1 Motivating and defining the problem of generalised rule induction

Above we have outlined the problem of learning a given concept. While this is an interesting and theoretically challenging problem it can only *partially* model the learning process at best, for how does an agent decide upon *which* particular concepts to learn? C. S. Peirce[135] illustrated the point well by imagining a being from another planet, which in trying to develop a mental model of our planet would spend thousands of years learning useless concepts. This problem of induction being open-ended cannot be resolved by non-quantitative techniques alone. Clearly the agent must have some means at his disposal for *comparing* the relevant concepts for his model. Furthermore, given the set of relevant concepts, and assuming that the agent has only finite computational and storage capacity, he must have some scheme for comparing not only hypotheses for the same concept, but also hypotheses for *different* concepts. In this manner the agent can learn the best hypotheses for the most important concepts. The underlying role of a probabilistic model for the environment is central to the argument. If we view the environment to be modelled as a joint density defined over our n discrete attributes, then in a certain sense, the role of induction is to identify and quantify the most important components of this joint probability distribution, subject to some capacity constraint. Probabilistic rules, which incorporate conditional and prior probabilities, provide a basis for the probabilistic model. The J-measure provides a quantitative measure for the importance of components in the model. In other words, by using the J-measure to rank hypotheses for different concepts (components in the probability model) the agent can obtain the "best" approximation to the environment, given the constraints.

Let us specifically consider the practical problem of obtaining an optimal set of rules, given a set of specific examples, where "optimal" has yet to be precisely defined. What is new is the fact that we do not restrict the consequent clause in our rules to be the classification attribute, i.e., we are interested in attribute-attribute rules as well as attribute-class rules. Previous work in rule-induction has concentrated on classification-type rules where the consequent clause only involves the class variable. However, in data-driven applications a more flexible and general approach is desirable. One wishes to have rules relating attributes. In this sense, by treating the class as 'just another attribute', the *generalised* rule-induction

problem is formulated. The more standard classification approach is seen to be a special case of the general problem.

Consider an example from the animal domain. Say we have a large list of animals (our example set), each animal being described in terms of various attributes such as size (large or small), colour, number of legs, dangerous (yes or no), species, etc. A classification-type of induction algorithm might induce rules such as

If *the animal has no legs* then *it is a snake* with probability p

A more general induction algorithm would also induce rules such as

If *the animal is large* then *it has 4 legs* with probability p.

in addition to the classification rules. The advantages of the more general approach are obvious in terms of providing a more accurate representation of the important concepts about the domain from which the examples are taken. From a cognitive viewpoint the process of constructing a mental model of the environment is directly equivalent to generalised rule induction.

2.6.2 A brief review of earlier work in this area

For the purposes of this thesis we limit our attention to *selective* induction (where the induction algorithm retains the same representation as given by the examples) as opposed to *constructive* induction. In terms of the notation developed in Section 2.5.1 we say that the hypothesis space is contained in the instance space.

The problem of generalised rule induction has not previously been addressed. Prior work has focussed on the problem of learning *classification* rules. While it might be argued that such classification algorithms can solve the general problem by repeated application, an algorithm which is specifically devoted to solving the general problem is clearly preferable. A good taxonomy of automatic induction algorithms is given in Cohen and Feigenbaum [136]. These algorithms can loosely be categorised into two main areas, those which use symbolic manipulation techniques and those which use statistically-oriented techniques. Mitchell's

'version spaces' algorithm [137] is perhaps the best-known symbolic concept learning algorithm. Another example is the AQ11 algorithm of Michalski and Larson [138] which achieved success in the domain of plant disease diagnosis [139]. Typically these algorithms examine the examples *sequentially* and refine what is known as the rule space until a set of general classification rules covering the examples are arrived at. However, noisy examples (as in the case where not *all* but *nearly all* large animals in the domain have 4 legs) are not easily handled by the symbolic approach. In addition the algorithms are computationally unattractive. Consequently their use has been limited to research-oriented endeavours rather than practical applications such as knowledge acquisition for expert systems.

Methods which can be termed as statistical, exploit *average* properties of the example set. As we discussed earlier, *existing* statistical learning algorithms generally lack the flexibility we require, by either imposing a particular statistical model on the environment (Duda and Hart [57]) or, as with tree algorithms, imposing a particular structure on the nature of the solution.

Newer paradigms such as connectionist learning are much more similar in spirit to the approach outlined here. Essentially a mental model of the external environment is constructed by varying connection weights in a neural net, e.g., the generalised Delta rule (Rumelhart et al. [140]) and the Boltzmann algorithm (Sejnowski et al. [141]). However, as we discussed in Section 2.1.1, such techniques, while advantageous in certain respects, are problematic in terms of explicit knowledge representation and speed of learning. Nonetheless, as we shall see later, the general theory developed here shares many characteristics with the connectionist approach.

Mention should also be made of a fuzzy logic approach to the problem of generalised rule induction, which was independently proposed by Gaines and Shaw [142] very recently. This is perhaps the only other work which discusses this problem specifically. Gaines and Shaw define a fuzzy version of cross-entropy which they use to derive fuzzy logic rules from examples. However, their paper does not completely define all the formulae used in deriving these rules. In particular, based on the results given in their paper, the results obtained in the fuzzy domain cannot be extended to the standard probabilistic domain in which we are interested.

2.6.3 The basis for ITRULE: specialising rules to increase the J-measure

Before describing the ITRULE (Information Theoretic RULE induction) algorithm we must first develop some quantitative bounds on the nature of specialisation. The basic premise of the algorithm will revolve around instance-based generalisation from examples to generate an initial set of rules, followed by specialisation of these rules to optimise the rule set. The exact nature of the specialisation is critical to the performance of the algorithm. Hence, we will devote a considerable amount of attention to developing a *theoretical* understanding of specialisation in terms of our information measure.

Specialisation, as we saw in Section 2.5.2, is the process by which we hope to increase a rule's "goodness" by adding an extra condition to, or specialising, the rule's left-hand side. The decrease in simplicity of the hypothesis should be offset by an increase in the goodness-of-fit to the extent that the overall goodness measure is increased. We will examine specialisation, using the J-measure as our definition of rule goodness, with $p(y)$ corresponding to simplicity and $j(\mathbf{X};\mathbf{Y}=y)$ corresponding to goodness-of-fit as discussed in Section 2.5.1.

The question we pose is as follows: given a particular general rule, what quantitative statements can we make about specialising this rule? In particular, if we define J_s and J_g as the J-measures of the specialised and general rules respectively, can we find a bound of the form

$$J_s \leq f(J_g) \tag{2–56}$$

for some $f(.)$ defined on J_g or its component terms? The motivation for bounding J_s in this manner is two-fold. Firstly it produces some theoretical insight into specialisation, while secondly, and more practically, the bound can be used by rule induction algorithms to search the rule space (hypothesis space) efficiently. This section will be devoted to stating, and proving, two particularly useful bounds of the form given by Equation (2–56). The first bound is for the case where we have no information whatsoever about the conditions which are available for specialisation, i.e., the new conditions which we can insert in the left-hand side of the general rule. This bound is primarily of theoretical interest. The second bound is more general and more practical in that it covers the case where we have some probabilistic information (e.g., *a priori* probabilities) about the available conditions, as would be the case in a practical algorithm.

Let us define some simplified notation in order to clarify the upcoming derivations. We are given a general rule whose J-measure, J_g, is defined as

$$J_g = J(\mathbf{X}; \mathbf{Y} = y) \tag{2-57}$$

$$= p(y)\Big(p_g . \log \frac{p_g}{p_x} + (1 - p_g) . \log\Big(\frac{1 - p_g}{1 - p_x}\Big)\Big) \tag{2-58}$$

where $p_g = p(x|y)$ and $p_x = p(x)$. The probability p_g is the transition probability of the general rule. We wish to bound

$$J_s = J(\mathbf{X}; \mathbf{Y} = y, \mathbf{Z} = z) \tag{2-59}$$

$$= p(y, z)\Big(p_s . \log \frac{p_s}{p_x} + (1 - p_s) . \log\Big(\frac{1 - p_s}{1 - p_x}\Big)\Big) \tag{2-60}$$

$$= p(z|y)\ p(y)\ j_s\ , \tag{2-61}$$

where j_s is the specialised j-measure. Without loss of generality we assume that $p_g > p_x$, since if $p_g < p_x$ we can simply reverse the labelling on x and $\bar{x}$, while if $p_g = p_x$ then $J_g = 0$ and the case is not of interest since any condition $z \neq y$ will lead to J_s being greater than J_g. Given no information about $\mathbf{Z}$ whatsoever, we can state the following result:

Theorem 2.5:

$$J_s \leq \max\Big\{p(y)p_g \log \frac{1}{p_x},\ p(y)(1 - p_g) \log \frac{1}{1 - p_x}\Big\} \tag{2-62}$$

$$= \max\Big\{p(x, y) \log \frac{1}{p_x},\ p(\bar{x}, y) \log \frac{1}{p_{\bar{x}}}\Big\}\ .a \tag{2-63}$$

Proof:

We consider 3 distinct cases; (i) $p_s > p_g$, (ii) $p_s < p_x$, and (iii) $p_g \geq p_s \geq p_x$.

Case (i) $p_s > p_g$:

We can write

$$p(x|y) = p(x|y, z)p(z|y) + p(x|y, \bar{z})p(\bar{z}|y) \tag{2-64}$$

as in Equation (2–46), or equivalently,

$$p_g = p_s . p(z|y) + \theta . (1 - p(z|y)) \tag{2-65}$$

where $\theta = p(x|y, \bar{z})$. The left-hand side, p_g, is fixed and represents a constraint which p_s, θ and $p(z|y)$ must satisfy. We want to find a variable $\mathbf{Z}$ which maximises J_s subject to this constraint. First we note that by Equation (2–65),

$$\min\{p_s, \theta\} \leq p_g \leq \max\{p_s, \theta\} \tag{2-66}$$

since $p(z|y) + p(\bar{z}|y) = 1$. Since we have assumed that $p_s > p_g$ initially, we can state that

$$p_s > p_g > \theta \ . \tag{2–67}$$

From Equation (2–65), we have that

$$p(z|y) = \frac{p_g - \theta}{p_s - \theta} \ . \tag{2–68}$$

Hence, our expression for J_s can be written as

$$J_s = p(y).\left(\frac{p_g - \theta}{p_s - \theta}\right).\left(p_s \log\left(\frac{p_s}{p_x}\right) + (1 - p_s)\log\left(\frac{1 - p_s}{1 - p_x}\right)\right) \ . \tag{2–69}$$

■

The only remaining free parameters are p_s and θ, which we will choose so as to maximise J_s. The probabilities p_s and θ are jointly constrained by the fact that

$$0 \leq \left(\frac{p_g - \theta}{p_s - \theta}\right) \leq 1 \ . \tag{2–70}$$

Since this is a multiplicative term in J_s, to maximise J_s we should maximise this term and then check if the value of this maximum constrains p_s in any way. If it does, then we cannot maximise the product terms in Equation (2–69) to find an achievable bound. From Equation (2-67) we know that $0 \leq \theta < p_g$. The following lemma is useful.

Lemma 2.6.1:

$$\max_{0 \leq \theta < p_g} \left\{\frac{p_g - \theta}{p_s - \theta}\right\} = \frac{p_g}{p_s} \tag{2–71}$$

Proof:

Let $\theta_1 < \theta_2$. Therefore,

$$\theta_1(p_s - p_g) < \theta_2(p_s - p_g) \tag{2–72}$$

since $p_s - p_g > 0$, and by adding $p_s p_g + \theta_1\theta_2$ to each side we get

$$p_s p_g + \theta_1\theta_2 + \theta_1 p_s - \theta_1 p_g < p_s p_g + \theta_1\theta_2 + \theta_2 p_s - \theta_2 p_g \tag{2–73}$$

$$\Rightarrow (p_g - \theta_2).(p_s - \theta_1) < (p_s - \theta_2).(p_g - \theta_1) \tag{2–74}$$

and so

$$\frac{p_g - \theta_2}{p_s - \theta_2} < \frac{p_g - \theta_1}{p_s - \theta_1} \tag{2–75}$$

which implies that the maximum occurs for $\theta = 0$.

Accordingly, the choice of $\theta = 0$ minimises the multiplicative term in Equation (2–69) without introducing any extra constraints on p_s. Hence, we can maximise the two terms separately and still obtain an achievable bound. We will refer to this bound as *the product bound*. From Equation (2-69) and the result of the lemma we get that

$$J_s \leq p(y).\frac{p_g}{p_s}.\left(p_s \log\left(\frac{p_s}{p_x}\right) + (1-p_s)\log\left(\frac{1-p_s}{1-p_x}\right)\right) \tag{2–76}$$

$$= p(y).p_g.\left(\log\left(\frac{p_s}{p_x}\right) + \left(\frac{1}{p_s} - 1\right).\log\left(\frac{1-p_s}{1-p_x}\right)\right) \tag{2–77}$$

$$\leq p(y).p_g.\log\left(\frac{p_s}{p_x}\right) \qquad \text{(since the second term is negative)} \tag{2–78}$$

$$\leq p(y).p_g.\log\left(\frac{1}{p_x}\right) \tag{2–79}$$

This proves case(i) of the Theorem 2.5.

Next we consider case (ii) where $p_s < p_x < p_g$. Intuitively what happens here is that the new condition "changes the direction of the rule" so that $\bar{x}$ is being confirmed rather than x. As we shall see, in practice this case is far less likely to occur than case(i). Nonetheless, we must analyse this case to get a general bound. Proceeding as in case(i) we get the equivalent condition to Equation (2–67),

$$p_s < p_x < p_g < \theta \tag{2–80}$$

and so we have that

$$p(z|y) = \frac{\theta - p_g}{\theta - p_s}\ . \tag{2–81}$$

Lemma 2.6.2:

$$\max_{p_g < \theta \leq 1}\left\{\frac{\theta - p_g}{\theta - p_s}\right\} = \frac{1-p_g}{1-p_s} \tag{2–82}$$

Proof:

Let $\theta_1 > \theta_2$. Therefore,

$$\theta_1(p_g - p_s) > \theta_2(p_g - p_s) \tag{2–83}$$

since $p_g - p_s > 0$, and by adding $p_s p_g + \theta_1\theta_2$ as before to each side we get

$$(\theta_2 - p_s).(\theta_1 - p_g) > (\theta_2 - p_g).(\theta_1 - p_s) \tag{2–84}$$

$$\Rightarrow \frac{\theta_1 - p_g}{\theta_1 - p_s} > \frac{\theta_2 - p_g}{\theta_2 - p_s} \tag{2-85}$$

and so, unlike case (i), the maximum occurs for $\theta = 1$.

■

As before, maximising the product term does not constrain p_s in any way since $0 < \frac{(1-p_g)}{(1-p_s)} < 1$ for all allowable values of p_s. Hence, we have that

$$J_s \leq p(y).\left(\frac{1-p_g}{1-p_s}\right).\left(p_s \log\left(\frac{p_s}{p_x}\right) + (1-p_s)\log\left(\frac{1-p_s}{1-p_x}\right)\right) \tag{2-86}$$

$$= p(y).(1-p_g).\left(\frac{p_s}{1-p_s}\log\left(\frac{p_s}{p_x}\right) + \log\left(\frac{1-p_s}{1-p_x}\right)\right) \tag{2-87}$$

Since $\log(\frac{p_s}{p_x}) < 0$, given that $p_s < p_x$, then

$$J_s \leq p(y).(1-p_g).\log\left(\frac{1}{1-p_x}\right) . \tag{2-88}$$

This proves the bound for case (ii).

For case (iii) we can apply the following arguments. If $p_s = p(x)$ then $J_s = 0$ and so the bound holds. If $p(x) < p_s < p_g$ then from case (ii) we see that the simplicity component

$$p(y).\left(\frac{\theta - p_g}{\theta - p_s}\right) \leq p(y) , \tag{2-89}$$

while the goodness of fit component

$$j_s \leq p_g.\log(\frac{p_g}{p_x}) + (1-p_g).\log(\frac{1-p_g}{1-p_x}) \tag{2-90}$$

$$< p_g.\log(\frac{p_g}{p_x}) \tag{2-91}$$

due to the fact that the second term is negative since $(1-p_g) < (1-p_x)$. Hence, we get that

$$J_s < p(y).p_g.\log(\frac{p_g}{p_x}) \tag{2-92}$$

which is less than Equation (2–79), the bound for case (i). Finally if $p_s = p_g$, we can apply the same argument for the goodness of fit, j_s, and noting that the simplicity component must be less than or equal to $p(y)$, we get the same result as Equation (2–92). By combining the results of cases (i), (ii) and (iii), we get the desired result. This proves Theorem 2.5 in its entirety, which we will now restate:

$$J_s \leq p(y).\max\left\{p_g \log\frac{1}{p_x},\ (1-p_g)\log\frac{1}{1-p_x}\right\} \tag{2--62}$$

$$= \max\left\{p(x,y)\log\frac{1}{p_x},\ p(\bar{x},y)\log\frac{1}{p_x}\right\} \tag{2--63}$$

If we recall the original bound we derived in Theorem 2.4, and we make the assumption that $p(y) \leq p(x)$ and $p(y) \leq p(\bar{x})$ then the equivalent bound using that theorem would be

$$J_s \leq p(y).\max\{\log(\frac{1}{p(x)}),\ \log(\frac{1}{p(\bar{x})})\}\ . \tag{2-93}$$

Comparing the two inequalities in Equations (2–93) and (2–62), we see that Theorem 2.5 gives an improvement of a factor of p_g. It is interesting to note that the transition probability of the *general* rule plays such a limiting multiplicative role in the bound. In essence, it tells us the limits imposed by the continued presence of the y term in any more specialised rule.

For the case when $p(y) > p(x)$, or $p(y) > p(\bar{x})$, we note that this introduces an extra constraint into the problem by effectively limiting the achievable value of p_g, and consequently p_s. Clearly the bounds still hold but are no longer achievable. Equivalent achievable bounds can be derived, but are omitted in this presentation, since such pathological cases are not of general interest.

Let us consider the bound for the "non-flying penguins" example given back in Section 2.5.2 on specialisation. From that example we had that $p(y) = 0.3$, $p(x) = 0.3$ and $p_g = 0.9$. In this case the bound works out to be 0.51 bits. It happens that the example given as a specialisation actually achieves this bound. Given the original general rule with $J_g = 0.38$ bits we could have stated that the most information one could get by specialisation is 0.51 bits. Furthermore, we could state that the specialised rule cannot be improved upon by further specialisation.

As a consequence of this theorem we note that since the bound is achievable, then without further information about $\mathbf{Z}$, it cannot be improved upon. In fact if we set $y = z$ then we find that J_g itself also obeys this bound. The logical consequence of this statement is that it precludes using the bound to discontinue specialising based on the value of J_g alone, since unless $p_g = 0$ or $p_g = 1$ then the result of Theorem 2.5 holds as a strict inequality for J_g. Hence, by defining a specialising condition $z = x$, we can always obtain a specialised rule for which the bound can be obtained. In other words, if p_g is not equal to either 1 or 0, then with no information at all available about the other variables, there may always exist a more specialised rule whose information content is strictly greater than that

of the the general rule. Conversely, if $p_g = 0$ or $p_g = 1$ then specialisation can not increase the information content.

Clearly, in order to say more about specialisation, the next step is to consider what the equivalent bounds are for the case when we *have* some information regarding the conditions which are candidates for inclusion in the rule's left-hand side. The information we will have about $\mathbf{Z} = z$ may be in the form of probabilities such as $p(z)$ or $p(y, z)$. If y is a composite proposition, made up of say the conjunction of $y_1, \ldots, y_k$, then we may only have *second-order* probability information available, such as $p(y_1, z), \ldots, p(y_k, z)$. Even more generally, we may only have *bounds* on some lower-order probability information concerning y and z. To motivate the discussion, consider the fact that an agent or an algorithm which is performing generalised rule induction (developing its model of the environment), will, for any particular rule which it is considering, have already stored other rules, or *a priori* probabilities, which may contain relevant probabilistic information. Hence, if we are wondering if some condition like $\mathbf{Z} = \mathbf{z}$ is worth adding to a rule with $\mathbf{X} = x$ as the conclusion, then information like $p(z|x)$ or $p(z)$ may be useful in making the decision whether to specialise or not. This feature has a direct cognitive analogy, as the mental models which humans develop tend to be extended by using whatever information is already present in the model to search for new hypotheses (Holland et al. [130], pp.249-254). Clearly this is a feature which we should incorporate into our (as yet undefined) generalised rule induction algorithm.

From the derivation of Theorem 2.5 we recall that J_s can be written as

$$J_s = p(y).\left(\frac{p_g - \theta}{p_s - \theta}\right).\left(p_s \log\left(\frac{p_s}{p_x}\right) + (1 - p_s)\log\left(\frac{1 - p_s}{1 - p_x}\right)\right) \tag{2-94}$$

where θ and p_s are two free parameters except for the constraint that

$$0 \leq \left(\frac{p_g - \theta}{p_s - \theta}\right) \leq 1 \,. \tag{2-95}$$

Given any probabilistic information about z, of the type discussed above, we can translate such information into constraints on θ, p_s and the quantity in Equation (2-95). For the general case we will have,

$$\theta_1 \leq \theta \leq \theta_2 \tag{2-96}$$

$$p_1 \leq p_s \leq p_2 \tag{2-97}$$

$$\alpha_1 \leq \left(\frac{p_g - \theta}{p_s - \theta}\right) \leq \alpha_2 \ . \tag{2–98}$$

The general problem is to determine an achievable bound given constraints of this form. We can not apply the same arguments as in Theorem 2.5 because given the extra constraints it is not clear that if we maximise one of the product terms, say the first, and find a particular p_s and θ, that these same p_s and θ maximise the overall expression. We can express Equation (2–69) as follows:

$$J_s(p_s, \theta) = p(y).f(p_s, \theta).j_s(p_s) \tag{2–99}$$

where

$$f(p_s, \theta) = \frac{p_g - \theta}{p_s - \theta} \ . \tag{2–100}$$

We know that $j_s(p_s)$ is convex $\cup$ in p_s. It can easily be shown that

$$\frac{\partial^2}{\partial {p_s}^2} f(p_s, \theta) = \frac{2(p_g - \theta)}{(p_s - \theta)^3} \tag{2–101}$$

$$> 0 \ , \tag{2–102}$$

since if we assume that $p_s \neq p_g$ then either

$$p_s < p_g < \theta \ \text{ or } \ \theta < p_g < p_s \ , \tag{2–103}$$

and the inequality holds for either case. Equation (2–102) implies that $f(p_s, \theta)$ is convex $\cup$ in p_s. Consequently, for θ fixed, the maximum value of $f(p_s)$ is at one of the endpoints. For fixed θ, $J_s(p_s)$ is a product of two functions which are convex $\cup$ in p_s, where we note that $J_s(p_s)$ is not just an arbitrary definition of the J-measure, but the *specialised* J-measure, as defined in Equation (2–69). Unfortunately, a product of two convex $\cup$ functions is not necessarily convex $\cup$ itself. If we work through finding the second derivative of $J_s(p_s)$ with respect to p_s we get that

$$\frac{\partial^2}{\partial {p_s}^2} J(p_s, \theta) = \frac{2.p(y).(p_g - \theta)}{(p_s - \theta)^3} \left(\log\left(\frac{(1 - p_s)}{(1 - p_x)}\right) + \theta \log\left(\frac{p_s(1 - p_x)}{p_x(1 - p_s)}\right) + \frac{(p_s - \theta)^2}{2 \log_e(2).p_s.(1 - p_s)} \right) . \tag{2–104}$$

This quantity is not always positive, e.g., it is negative for $p_s = 0.5$, $p_x = 0.4$ and $\theta = 0.4$. Hence, J_s is not convex $\cup$ in p_s and we can not appeal to convexity arguments. However, non-convexity is not a problem as long as the function is monotonic on either side of p_x, i.e., there are no local maxima, just a global minimum at $p_s = p_x$. If it is non-monotonic

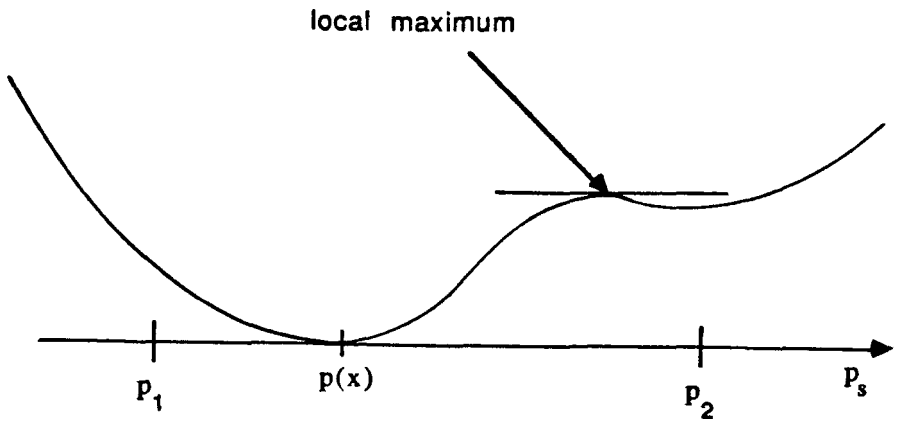


Figure 2.4: The possible case where a local maximum exists between the 2 endpoints p_1 and p_2 of the interval for p_s.

then a situation like that shown in Figure 2.4 can occur where there exists a local maximum between the bounds p_1 and p_2 which we have for p_s. Such non-monotonicity would exclude bounding based on the end-points of the interval alone.

It turns out that in general the function can have local maxima. It can easily be shown that

$$\frac{\partial}{\partial p_s} J(p_s, \theta) = \log\left(\frac{1-p_x}{1-p_s}\right) - \theta\left(\log\frac{(1-p_x).p_s}{p_x.(1-p_s)}\right) , \qquad (2\text{–}105)$$

which, when set to zero, can have more than 1 root depending on the values of θ, p_s, p_x. Hence, the graph in Figure 2.4 is possible. Thus, although we would like to find the possible maxima of the function, given interval ranges for θ and p_s, this would involve either tabulating solutions of the non-linear equation in (2–105) or explicitly solving it each time. A much more practical and realistic approach is to recall that θ is defined as

$$\theta = p(x|y, \bar{z}) . \qquad (2\text{–}106)$$

In practice, the bounds available for θ may well be the trivial bounds of 0 and p_g (from Equation (2–65), assuming that $p_s > p_g$ for the moment). We will adopt the approach of choosing $\theta = 0$, since as we have seen above, even if we *have* information about θ it is relatively difficult to use in the general situation. Choosing $\theta = 0$ may cause our bound to be larger than what is achievable. This is the price we pay for ignoring the extra information.

We must of course show that for all $p_s > p_g$,

$$J_s(p_s, \theta) \leq J_s(p_s, \theta = 0) \ . \tag{2-107}$$

If we refer back to lemma 2.6.1 we see that for any fixed $p_s > p_g$,

$$\left\{\frac{p_g - \theta}{p_s - \theta}\right\} \leq \frac{p_g}{p_s} \ . \tag{2-108}$$

Hence, by definition, Equation (2–107) holds. Now, with $\theta = 0$, the first derivative of J_s (2.99) is always positive for $p_s > p_x$ (and always negative for $p_s < p_x$). Hence, given the initial information that $p_1 \leq p_s \leq p_2$, and assuming that $p_2 > p_x$ we can state the following bound on J_s:

$$J_s \leq p(y).\left(\frac{p_g}{p_2}\right).j_s(p_2) \tag{2-109}$$

where j_s is of course the j-measure.

Similarly, if $p_1 < p_x$ we can work through an exactly equivalent analysis, this time choosing $\theta = 1$ and using lemma 2.6.2, with the result that

$$J_s \leq p(y).(\frac{1 - p_g}{1 - p_1}).j_s(p_1) \ . \tag{2-110}$$

Depending on the relative values of p_1, p_x and p_2 we can obviously derive lower bounds also if we so desire, e.g., if $p_x < p_1 < p_2$ then

$$J_s \geq p(y).(\frac{p_g}{p_1}).j_s(p_1) \ . \tag{2-111}$$

The most likely case to occur in practice will probably be $p_1 < p_x < p_2$ and so the bound of most practical interest and use is

$$J_s \leq p(y).\max\left\{(\frac{p_g}{p_2}).j_s(p_2),\ (\frac{1 - p_g}{1 - p_1}).j_s(p_1)\right\} \ . \tag{2-112}$$

Two points remain to be clarified. We did not mention the case when $p_x \leq p_s \leq p_g$ in our analysis because, as before, when p_s is in this range, $J_s \leq J_g$. Hence, the upper bounds cannot be affected. In addition we have not mentioned how to use the bounds α_1 and α_2 (Equation (2–98)), if available. Since the alpha's depend on θ we cannot use these bounds directly. Rather, the bounds should be re-written so as to provide bounds on p_s which can be used directly.

In case one is wondering exactly where bounds like p_1 and p_2 appear from, here is a typical example. Let us say that y is the conjunction of y_1, y_2 and y_3. We want to bound

$$p_s = p(x|y,z) = \frac{p(x,y,z)}{p(z,y)} \tag{2-113}$$

We can find an upper bound (for example) by upper bounding $p(x,y,z)$ as follows

$$p(x,y,z) \leq \min\{p(x,y), p(z,y_1), p(z,y_2), p(z,y_3)\} \ , \tag{2-114}$$

(this bound is just one of several possible), and lower bounding $p(y,z)$ with

$$p(z,y) \geq \max\{0,\ p(y)+p(z)-1\} \tag{2-115}$$

to get

$$p(x|y,z) \leq \min\{1, \frac{\min\{p(x,y), p(z,y_1), p(z,y_2), p(z,y_3)\}}{\max\{0,\ p(y)+p(z)-1\}}\} \ . \tag{2-116}$$

In general the terms within the bounds may themselves only be bounded.

Finally, after a rather long, but necessary, discussion on bounding J_s, we are ready to discuss the induction algorithm.

2.6.4 A definition of the ITRULE algorithm

We will now define the ITRULE algorithm and then discuss the basic ideas which lead to this particular definition. The ITRULE algorithm takes sample data in the form of discrete attribute vectors and generates a set of K rules, where K is a user-defined parameter. The set of generated rules are the K most informative rules from the data as defined by the J-measure. In this sense the algorithm can be described as optimal. The probabilities required for calculating the J-measures are estimated directly from the data, based on the techniques outlined in Section 2.5.2.

A flowchart description of the algorithm is given in Figure 2.5. The algorithm proceeds by first finding K rules, calculating their J-measures, and then placing these K rules in an ordered list. The smallest J-measure, that of the Kth element of the list, is then defined as the running minimum J_{min}. From that point onwards, new rules which are candidates

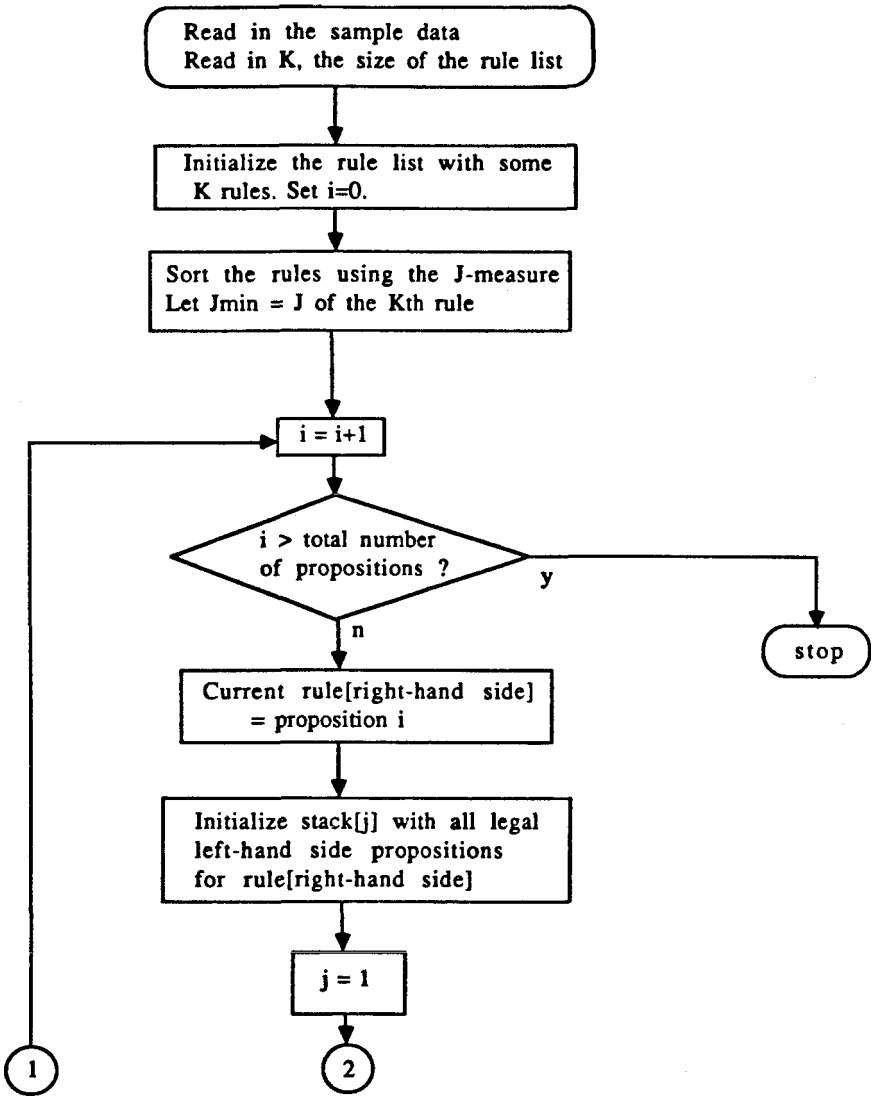


Figure 2.5(a): A flowchart description of the ITRULE algorithm.

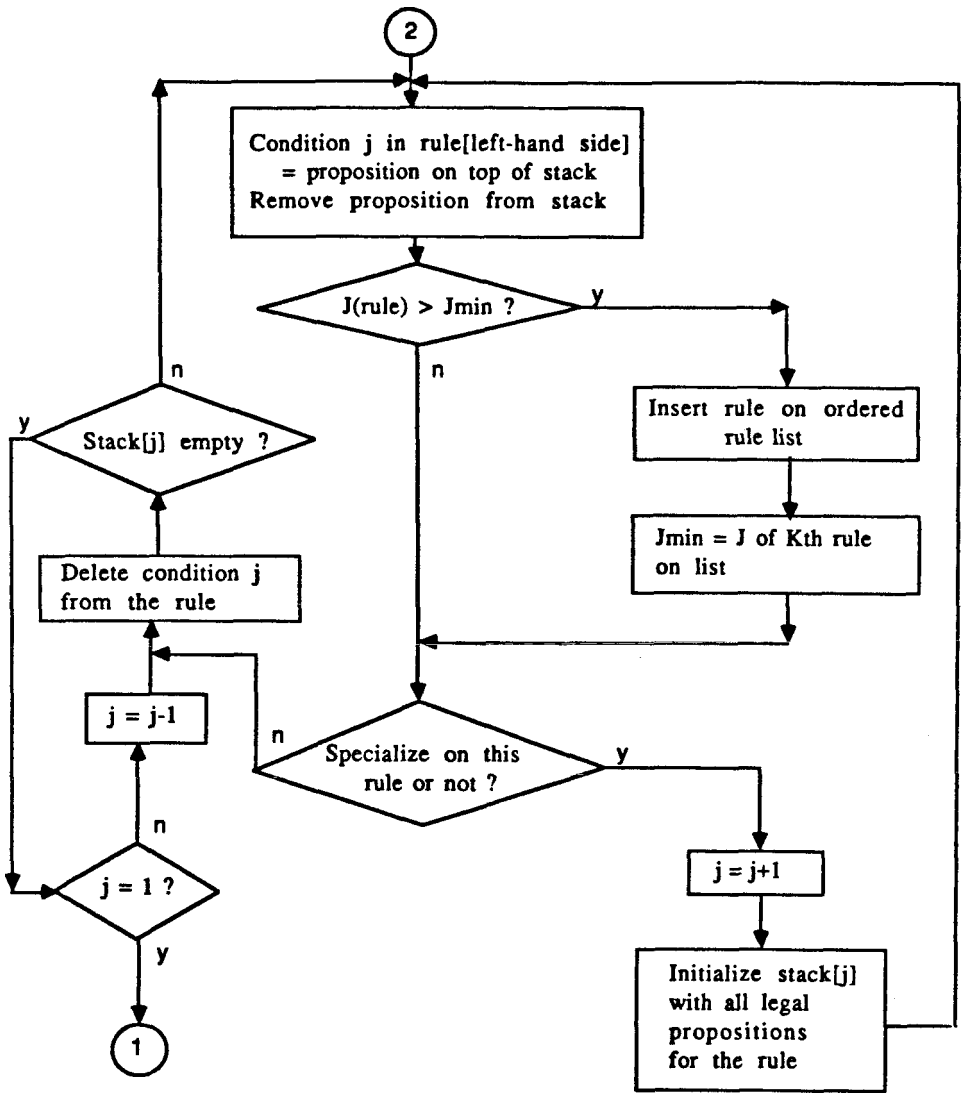


Figure 2.5(b): Flowchart description of ITRULE (continued).

for inclusion in the rule set have their J-measure compared with J_{min}. If greater than J_{min} they are inserted in the list, the Kth rule is deleted, and J_{min} is updated with the value of the J-measure of whatever rule is now Kth on the list. The critical part of the algorithm, the specialisation criterion, is not explicitly described by the flow-chart since it may vary from application to application. We will later describe a general specialisation strategy.

The order in which we search for rules is of course quite critical from a computational viewpoint. If we have n m-ary attributes, the number of possible rules in the data is R where,

$$R = nm\Big((2m+1)^{n-1} - 1\Big) \tag{2-117}$$

since for each of the nm possible right-hand sides, the other $n-1$ attributes have $2m+1$ possible states, namely, a truth statement and its negation for each of the m propositions and a "don't care" state for the attribute as a whole (for the case of *binary* attributes $m = 1$ because the negation of a proposition is also a basic proposition). As an example, if we have 10 binary attributes, there are $N = 10.3^9 - 10 = 196,820$ possible rules. From a practical point of view of course, we are likely to have neither the data to support so many inductive assertions nor the computational resources to manage them. Hence, in order to define a tractable algorithm we will need to "prune" the set of possible rule candidates considerably. Let us define a kth order rule as a rule with k basic propositions in its left-hand side. Let a_k be the number of possible kth order rules, so that we have

$$r_k = (2m)^k.\binom{n-1}{k}.nm \qquad 1 \leq k \leq n-1\ , \tag{2-118}$$

since there are $\binom{n-1}{k}$ sets of propositions of size k and $(2m)^k$ rules for each set. (That $\sum_k r_k = R$ holds can be verified by using binomial identities such as those given on page 63 in Feller[143]). The ratio

$$\alpha = \frac{r_k}{r_{k-1}} = \frac{2m.(n-k)}{k} \tag{2-119}$$

gives the ratio of the number of rules of order k to the number of order $k-1$. When α becomes less than 1 we have the condition that the number of rules is falling off rather than increasing, i.e., when

$$k > \Big(\frac{2m}{2m+1}\Big)\ n\ . \tag{2-120}$$

The significance of this result is that for all practical purposes, as we increase the order of the rules from $k = 1$ upwards, the size of the search space *increases*, and for k relatively

small compared to n it increases geometrically (by Equation (2-119)). We can write R as

$$R = nm \sum_{k=1}^{n-1} \alpha_k . r_{k-1} \tag{2–121}$$

$$= nm \sum_{k=1}^{n-1} \left(\prod_{i=1}^{k} \alpha_i \right), \tag{2–122}$$

where

$$\alpha_i = 2m . \left(\frac{n-i}{i} \right) \quad \text{and} \quad r_0 = 1 \, . \tag{2–123}$$

If we imagine implementing an algorithm which begins with first-order rules and specialises to higher orders (in order to find rules with higher J-measures) then an algorithm using blind search would have complexity $O(R) = O(n(2m)^n)$, as defined above. On the other hand, an algorithm which "prunes" the search space will have complexity $O(R')$ where

$$R' = nm \sum_{k=1}^{n-1} \left(\prod_{i=1}^{k} \beta_i \right) , \tag{2–124}$$

where the $\beta_k < \alpha_k$. A tractable algorithm will have $\beta_k < 1$ for (at least) k greater than some small fraction of n. We would like to have an algorithm for which R' is some computationally feasible polynomial expression in n and m, rather than exponential in either. There are some obvious possibilities. We could choose $\beta_k < 1$ and simply consider the most promising $\beta_k . r_{k-1}$ rules from level $k-1$ for specialisation at level k. However, this strategy would not be optimal. Similarly, if we were to implement an algorithm which only considered rules less than some pre-defined order i, then this would also be non-optimal. However, the latter case is more realistic since in practice we are estimating probabilities from frequency data, and inevitably there will not be enough data to support reasonable confidence intervals on higher order probability estimates. This strategy in itself is not sufficiently powerful to use as the basis for the algorithm.

The approach we take with ITRULE is to let the algorithm prune the search space using the bounds derived in Section 2.6.2, in particular using Equation (2–62), which we now restate:

$$J_s \leq p(y) . \max \left\{ p_g \log \frac{1}{p_x}, \; (1-p_g) \log \frac{1}{1-p_x} \right\} . \tag{2–125}$$

The algorithm begins by finding all $nm.(n-1).2m$ possible first-order rules. This allows us to begin initialising the rule set and also allows us to collect all available first and second-order statistics. This initial *breadth-first* step has a computational burden of

$O(m^2n^2)$ in terms of rules processed and imposes a storage requirement also of $O(m^2n^2)$. We choose to pay this fixed cost in order to use the collected statistics for bounding later in the algorithm. From this point onwards, for each of the $n.m$ possible right-hand sides, the algorithm employs *depth-first* search over possible left-hand sides, starting with the second-order conditions and specialising from there. Specialisation on a given left-hand side only occurs if the bound of Equation (2–125) is *less* than J_{min}. If the transition probability of a given general rule is equal to 1 or 0, then as we have seen earlier, we can also cease specialising. The algorithm systematically tries to specialise all $nm.(n-1).2m$ first-order rules and terminates when it has determined that no more first-order rules exist which can be specialised to achieve a higher J-measure than J_{min}.

After the first step, for each of the $m.n$ propositions in the system, we have stored all $2m(n-1)$ associated second-order joint probabilities, either directly (e.g., $p(z,y)$), or in the form of conditional ($p(z|y)$) and *a priori* probabilities ($p(y)$). The exact nature of the storage depends on the particular implementation of the algorithm. A factor of 2 (asymptotically, for large $m.n$) can be saved by storing only $p(z|y)$, $p(z)$ and $p(y)$ and using Bayes' rule to calculate $p(y|z)$, but this requires extra overhead and computation.

The general situation occurs when we have a right-hand side $\mathbf{X} = x$ and a left-hand side $y_1, \ldots, y_k$, where we have just evaluated J_g and inserted the rule in the list if $J_g > J_{min}$. In practical terms, in order to calculate J_g, we have sorted the original data into a sub-table conditioned on $y_1, \ldots, y_k$. We now wish to decide (using the bounds) whether further specialisation, and consequent sorting, is worthwhile. The decision whether to continue specialising or to back-up on the depth-first search is determined by the following sequence:

(i) If $p_g = 1$ or $p_g = 0$ then back-up the search, else

(ii) If $j(\mathbf{X};y)_g \leq J_{min}$ then check if for *any* z we can hope to find $J_s > J_{min}$. Calculate

$$J_1 = \max\left\{p(y)p_g \log\frac{1}{p_x}, p(y)(1-p_g)\log\frac{1}{1-p_x}\right\}$$

and (by Theorem 2.5) if $J_1 \leq J_{min}$ then back-up the search.

Note that, in practice, we do not use the more complicated bounds on J_s based on the additional information of bounding p_s for a particular z condition. The rationale behind this simpler algorithm is that the bounding approach requires significant extra overhead,

while the simpler approach, of step (ii) above, will detect "bad" z conditions at one level further in the search. Hence, the benefit of saving one level of sorting is not significant enough to outweigh the computational disadvantages of calculating probability bounds.

The complexity of the ITRULE algorithm cannot be determined exactly since it is highly dependent on the nature of the input data (in referring to "the algorithm", we mean the general version, where, the exact nature of the specialisation may vary). Probabilistic analysis, based on average performance over all possible input data sets, is too difficult to carry out directly without invoking unrealistic assumptions concerning the nature of the inputs. On the other hand, worst-case analyses will not necessarily give tight upper bounds, since such worst cases may be detectable by appropriate specialisation bounds. We know that the complexity is lower bounded by $O(m^2n^2)$, but determining an upper bound is problematic since it depends both on the data and on the value of K, the size of the output rule set. The best we can do is to invoke the argument that as specialisation (or rule order) increases, the simplicity of the hypotheses decrease to the extent that their probability of occurrence is very small. Hence, our bounds should eliminate the majority of the higher-order rules from consideration. In effect the β_k (in Equation (2–124)) should become negligible as k increases. Worst-case analyses do not necessarily give tight upper bounds, since, such worst cases may be detectable by appropriate specialisation bounds and, hence, a good algorithm would perform considerably better than the bound would suggest. The general question of upper bounding the complexity of the ITRULE algorithm remains an open and challenging problem.

The choice of K, the number of rules which the algorithm keeps in the list, obviously affects the computational complexity, since the value of the J-measure of the Kth rule has a considerable impact on the effectiveness of the bounding. For example, K may be chosen so large that J_{min} is zero or near zero at all times. However, there is normally no reason to choose such large K. If we are just interested in data analysis then very often some integer value of K between say 20 and 100 is sufficient for our purposes. However, if we wish to use the rules for probabilistic *inference* then we generally require more rules. In particular for each proposition in the system, we would like to have at least r rules with that proposition in their conclusions, or in terms of a graph where each proposition is a node, r is the number of rules entering a node or the "fan-in" of the node. In order for the system to perform useful inference (for example, multiple pieces of evidence supporting the same hypothesis)

we require that r be some integer greater than 1. Yet r should not be too large in order that the inference itself is computationally feasible as we shall see later. Hence, we can say that for inference purposes, $K = O(nm)$.

The general description of ITRULE given above is not intended as a definitive statement of how the algorithm should be implemented. Particular implementations may depend heavily on the nature of the particular problem. For instance, in data analysis we may only want to look at rules which conclude certain propositions of interest. The algorithm simply restricts the right-hand side propositions to that subset of interest (the limiting form of this approach is of course a classifier where we are only interested in propositions in the event space of a single variable, the "class"). The *order* of the search may also improve the efficiency of the algorithm, since if K relatively informative rules can be determined initially, J_{min} is initially high and the bounding is more effective. However, it is not clear what the gain would be, given the extra overhead involved in generating heuristics for searching. Simple heuristics such as ordering the right-hand propositions based on those which were most informative in terms of first-order rules, before beginning the depth-first search, would probably be effective. As mentioned earlier, a statistical confidence interval test may be incorporated into the algorithm. In exactly the same manner as we discussed with decision trees, the user can specify initially a desired interval at a given confidence level. If the specifications are not met at some level (order) for a conditional rule probability, then the algorithm can cease specialising and back-up on the depth-first search.

Let us note in conclusion that the lack of quantitative results on the complexity of the ITRULE algorithm reflects an inherent difficulty in quantifying the complexity of 'open-ended' induction problems. Essentially, the notion of identification in the limit as proposed by Gold [144] views induction as an infinite process, and as such, we can only hope to determine an induction algorithm's limiting behaviour. The consequence is that a learning agent *must* continue learning at all times, since it can never know itself whether it has converged to a correct hypothesis or not in the sense that future data may contradict the current hypothesis. Recent results (Valiant [145], Haussler [146], Kearns et al. [147]) have established the complexity of basic deterministic learning problems (i.e., where no noise is present) by modelling the *learning itself* in a probabilistic manner. Unfortunately, learning a deterministic function probabilistically and learning a probabilistic function are two very different problems, despite indications to the contrary in [146]. Hence, the complexity

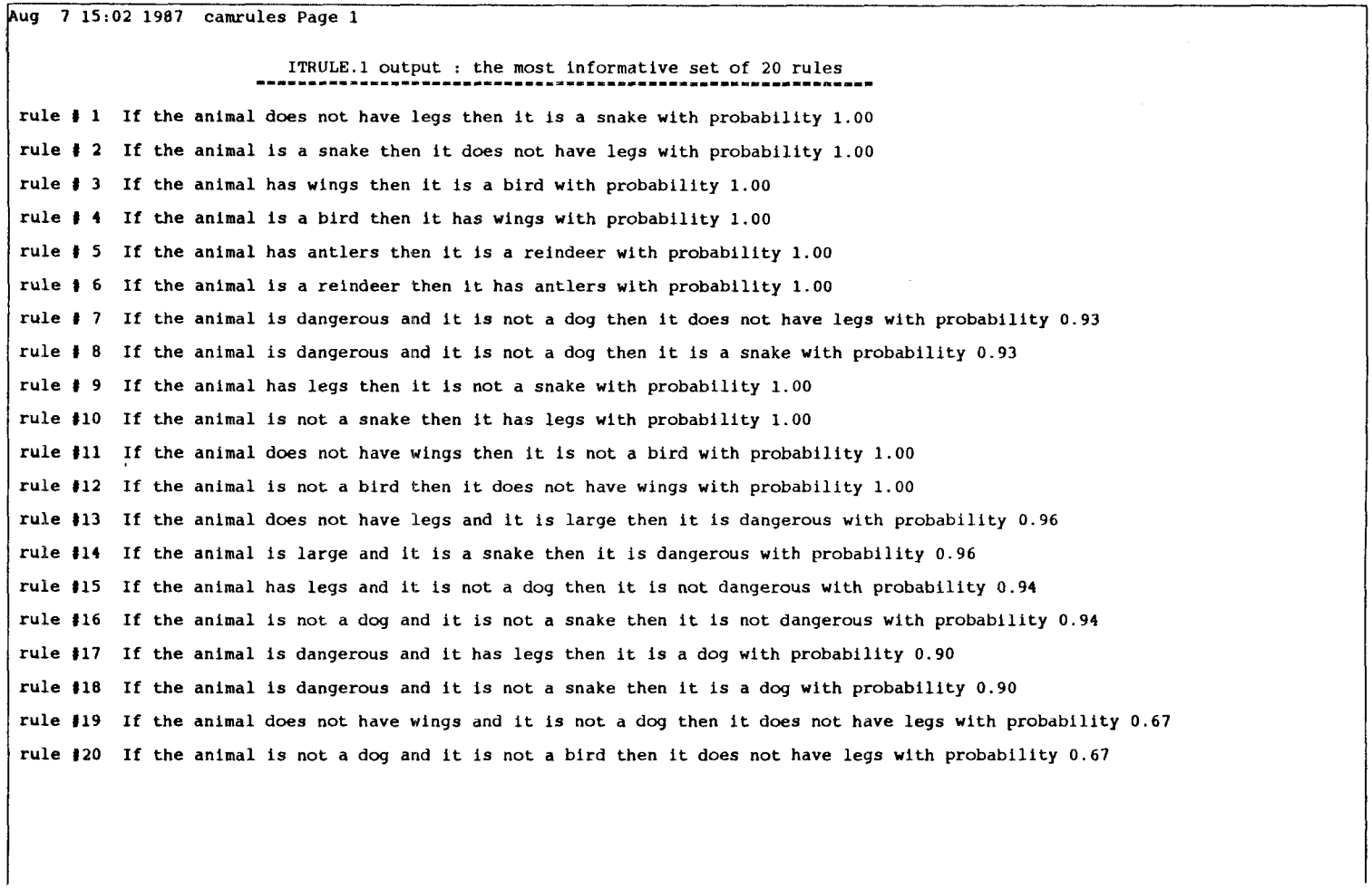
results for deterministic learning problems can not be extended directly to the probabilistic domain. We note also that in the framework presented here we have decoupled learning and inference, i.e., we have separated the two problems of finding the right model and then using it. As we shall see, unlike most approaches in the field, our underlying quantitative approach gives a bridge to link learning and inference together. It is worth noting that the decoupling of these two problems is entirely unnatural in cognitive terms. One of the primary advantages of some of the symbolic and connectionist approaches, over the purely statistical approach, is the ability to learn *incrementally*. An interesting future research direction would be to model learning and inference in a more unified manner using the J-measure.

2.6.5 Experimental results with the ITRULE algorithm

In this section we will discuss the output generated by ITRULE for two particular sets of data. On both sets the algorithm only specialised as far as second-order rules. The first data set was artificially generated to test out the basic ideas of the algorithm. It consists of 14 prototype "animals" which were weighted (as shown in Figure 2.6(a)) to generate an artificial sample of 1,000 such prototypes (obviously the weightings are only for illustrative purposes and bear no direct correlation with actual frequencies of occurrence). The samples contain a large amount of implicit higher order statistical information. The object of the exercise was simply to test the performance of the J-measure in terms of ranking relevant concepts. The output for $K = 20$ is shown in Figure 2.6(b). Clearly the resulting rules tally well with our own subjective judgement of which concepts in the initial data are most important, e.g., the highest ranked rules being that "snakes don't have legs" and "birds have wings". In this sense the J-measure seems to exhibit performance which is similar to that of our own learning skills. We see for example that the top four rules about snakes and birds are ranked higher than the rules about reindeers (ranked fifth and sixth) since reindeers only have an *a priori* probability of 0.14 of occurrence, while snakes and birds have higher *a priori* probabilities of 0.28 and 0.22 respectively. On the other hand, dogs, which are the most likely animal in the set with an *a priori* probability of 0.36, are ranked relatively lowly in terms of rules since the attributes used in this example are not sufficient

antlers	wings	dangerous	legs	size	dog	reindeer	snake	bird	Frequency of occurence
yes	no	no	yes	large	no	yes	no	no	0.10
yes	no	yes	yes	large	no	yes	no	no	0.01
yes	no	no	yes	small	no	yes	no	no	0.03
no	no	yes	yes	large	yes	no	no	no	0.16
no	no	yes	yes	small	yes	no	no	no	0.03
no	no	no	yes	large	yes	no	no	no	0.09
no	no	no	yes	small	yes	no	no	no	0.08
no	no	yes	no	large	no	no	yes	no	0.18
no	no	yes	no	small	no	no	yes	no	0.07
no	no	no	no	large	no	no	yes	no	0.01
no	no	no	no	small	no	no	yes	no	0.02
no	yes	no	yes	large	no	no	no	yes	0.01
no	yes	yes	yes	large	no	no	no	yes	0.06
no	yes	no	yes	small	no	no	no	yes	0.15

Figure 2.6: (a) Frequency data for the "prototype" animals.



```
Aug  7 15:02 1987  camrules Page 1

                    ITRULE.1 output : the most informative set of 20 rules

rule # 1  If the animal does not have legs then it is a snake with probability 1.00
rule # 2  If the animal is a snake then it does not have legs with probability 1.00
rule # 3  If the animal has wings then it is a bird with probability 1.00
rule # 4  If the animal is a bird then it has wings with probability 1.00
rule # 5  If the animal has antlers then it is a reindeer with probability 1.00
rule # 6  If the animal is a reindeer then it has antlers with probability 1.00
rule # 7  If the animal is dangerous and it is not a dog then it does not have legs with probability 0.93
rule # 8  If the animal is dangerous and it is not a dog then it is a snake with probability 0.93
rule # 9  If the animal has legs then it is not a snake with probability 1.00
rule #10  If the animal is not a snake then it has legs with probability 1.00
rule #11  If the animal does not have wings then it is not a bird with probability 1.00
rule #12  If the animal is not a bird then it does not have wings with probability 1.00
rule #13  If the animal does not have legs and it is large then it is dangerous with probability 0.96
rule #14  If the animal is large and it is a snake then it is dangerous with probability 0.96
rule #15  If the animal has legs and it is not a dog then it is not dangerous with probability 0.94
rule #16  If the animal is not a dog and it is not a snake then it is not dangerous with probability 0.94
rule #17  If the animal is dangerous and it has legs then it is a dog with probability 0.90
rule #18  If the animal is dangerous and it is not a snake then it is a dog with probability 0.90
rule #19  If the animal does not have wings and it is not a dog then it does not have legs with probability 0.67
rule #20  If the animal is not a dog and it is not a bird then it does not have legs with probability 0.67
```

Figure 2.6: (b) The best 20 rules as derived by the ITRULE algorithm based on 1,000 prototypes.

to distinguish dogs from the other animals. This simple experiment gives us an immediate appreciation of the potential benefits of ITRULE as a general purpose learning tool, as it provides information about the hypothesis space as a *whole* in contrast with classifier systems.

For this particular set of data we have some general first-order rules at the the top of the list, followed by more specialised second-order rules. In running the algorithm on different data sets, one finds typically a few good general rules which are ranked quite highly, with the majority of the rules being more specialised.

The second data set we will look at is based on actual data obtained from published information on the performance of mutual fund investment companies over a period of five years [148]. Figure 2.7(a) shows a typical set of samples. The attributes are largely self-explanatory. *5-year return* describes the percentage return over the 5-year period. *Diversity* is a rating measure of the investment strategies and *Beta* is a rating of the risk associated with these strategies. The *Type* indicates the diversity of the strategies, "A" being the most diverse and "E" being the least diverse. Since much of the data is continuous in nature it was necessary to quantise it. The approach taken was to use expert-supplied information as far as possible to determine "natural" quanta, e.g., the attribute *5-year return* was quantised into two values: above or below the return of the Standard and Poor index. Attributes for which there existed no such obvious groupings were quantised into bins of nearly equal probability in order to maximise the entropy of the quantised variable. This technique results in the minimum loss of information in the attribute quantisation process for a given number of bins and has been advocated as a useful quantisation technique (Wong and Chiu, [149]). Figure 2.7(b) shows some of the samples after quantisation. Figures 2.8(a) and 2.8(b) show the quantisation boundaries for different attributes.

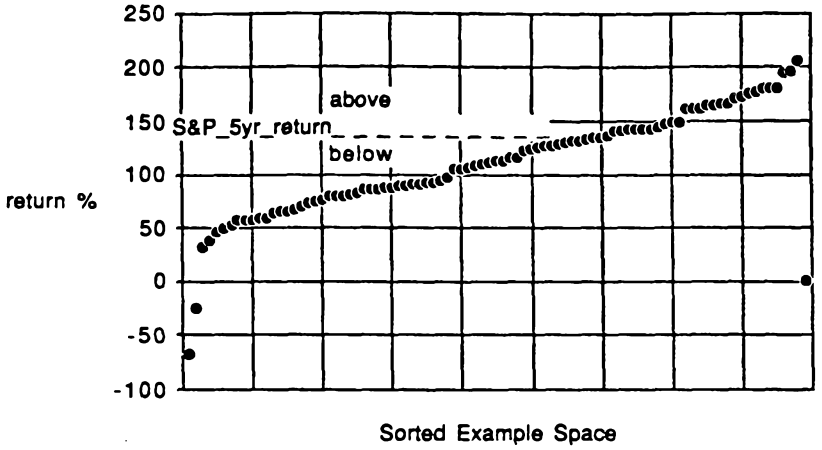
Figure 2.8(c) gives the resulting output for the top 45 rules [150]. The rules are listed in ranked order in terms of the J-measure. Note that the wording of the consequent phrase of the rule reflects the "stronger" direction of the rule, i.e., whichever of x or $\bar{x}$ has the greater transition probability. However, the actual numbers $p(x|y)$ and $p(x)$ given for that rule are not switched and, hence, may need to be subtracted from 1, in certain cases, to yield the correct interpretation. As with the animal data, many of the rules are typically what one might expect. For example, rule 1 says that if the overall 5-year return of a fund was above

Fund Type	5 Year Return%	diver-sity	Beta (Risk)	Bull Market Perfom.	Bear Market Perfom.	Stocks %	Largest Holding	Distributed Dividends (per share)	Distributed Cap. Gains (per share)	Net Asset Value $	Distri-butions as % of NAV	Portfolio Turnover Rate %	Total Assets ($M)
Balanced	135.6	C	0.8	B	D	87	information	0.5	6.07	37.27	17.63	34	414.6
Growth	32.5	C	1.05	E	B	81	manufacture	0.11	0	12.49	0.88	200	16.2
Growth&Income	88.3	A	0.96	C	D	82	financial	0.16	0.41	11.92	4.78	127	26.8
Agressive	-24.4	A	1.23	E	E	95	oil	0	0.6	6.45	9.30	161	64.2
Growth&Income	172.2	E	0.59	A	B	73	officeequip	0.52	0.84	13.64	9.97	31	112.5
Growth&Income	80.4	A	0.58	D	B	98	computers	0.45	2.2	13.7	19.34	64	76.4
Growth	52.7	B	0.86	E	D	95	airlines	0.24	0	8.72	2.75	54	11.7
Balanced	143.8	C	0.71	B	B	51	materials	0.66	0.7	13.03	10.44	239	190
Agressive	91.7	B	1.24	C	E	100	technology	0.04	1.8	8.11	22.69	218	96.2
Growth	105.4	A	0.99	B	D	98	energy	0.55	3.21	13.62	27.61	20	253.8
Growth	93	C	0.91	C	B	94	drugs	0	0	27.44	0.00	8	3.9
Growth	148.9	A	0.9	A	E	87	financial	0.5	5.8	32.4	19.44	37	452.9

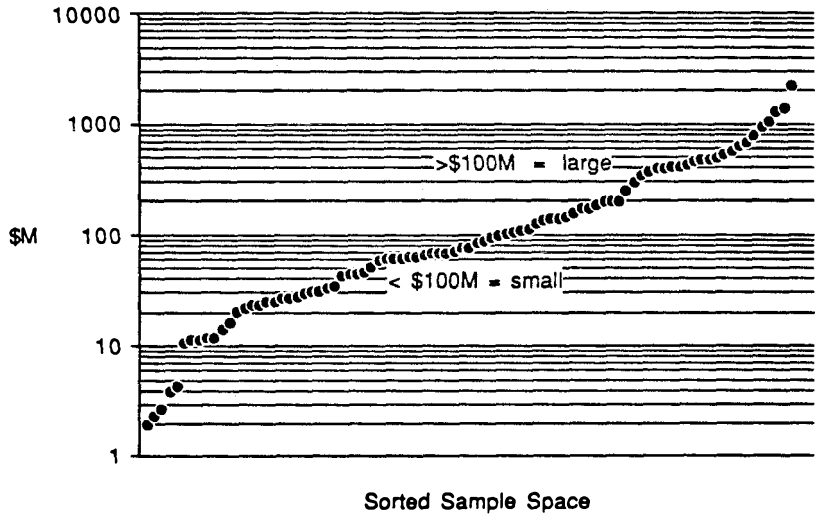
Fund Type	5 Year Return%	diver-sity	Beta (Risk)	Bull Market Perfom.	Bear Market Perfom.	Common Stocks >90%	Distributions	Portfolio Turnover Rate	Total Assets
Balanced	below	average	under1	good	poor	no	high	low	large
Growth	below	average	over1	poor	good	no	low	high	small
Growth&Income	below	high	under1	average	poor	no	low	high	small
Agressive	below	high	over1	poor	poor	yes	low	high	small
Growth&Income	above	low	under1	good	good	no	low	low	large
Growth&Income	below	high	under1	poor	good	yes	high	low	small
Growth	below	high	under1	poor	poor	yes	low	low	small
Balanced	above	average	under1	good	good	no	low	high	large
Agressive	below	high	over1	average	poor	yes	high	high	small
Growth	below	high	under1	good	poor	yes	high	low	large
Growth	below	average	under1	average	good	yes	low	low	small
Growth	above	high	under1	good	poor	no	high	low	large

Figure 2.7: (a) Typical data describing mutual funds,

(b) typical data after quantisation.



(a) 5-year return



(b) Size of assets

Figure 2.8: Quantisation levels for (a) "5-year return" attribute and (b) "Assets" attribute.

	antecedent #1	antecedent #2	consequent	p(x|y)	p(y)	p(x)	j(X;y)	J(X;y)
1	5yrRet>S&P? above		Bull perf good	0.966	0.311	0.511	0.75411	0.23461
2	Bull_perf good	Bear_perf NOT poor	5yrRet>S&P? above	0.242	0.356	0.689	0.60743	0.21597
3	Bull perf NOT good		5yrRet>S&P? below	0.978	0.489	0.689	0.40940	0.20015
4	5yrRet>S&P? below	Assets small	Bull perf NOT good	0.159	0.478	0.511	0.39009	0.18638
5	Bull_perf good	Bear_perf good	5yrRet>S&P? above	0.167	0.189	0.689	0.84334	0.15930
6	type NOT A	Bull perf good	Assets large	0.786	0.456	0.456	0.32960	0.15015
7	diversity NOT low	Assets large	Bull perf good	0.875	0.344	0.511	0.43274	0.14906
8	Bear_perf NOT poor	Assets large	5yrRet>S&P? above	0.269	0.278	0.689	0.53537	0.14872
9	Beta over1	Bear perf poor	stks>90%? yes	0.111	0.189	0.611	0.78686	0.14863
10	type NOT G	Bull perf good	5yrRet>S&P? above	0.286	0.300	0.689	0.49371	0.14811
11	Beta over1	Bear_perf poor	type A	0.722	0.189	0.244	0.72779	0.13747
12	type NOT A	Assets large	Bull perf NOT poor	0.025	0.433	0.267	0.31528	0.13662
13	type NOT A	Assets large	Bull perf good	0.825	0.433	0.511	0.31050	0.13455
14	Bull_perf poor		Assets small	0.080	0.267	0.456	0.49554	0.13214
15	type NOT A	Bull perf good	5yrRet>S&P? above	0.381	0.456	0.689	0.28889	0.13161
16	stks>90%? no	Assets large	Bull perf good	0.848	0.356	0.511	0.36439	0.12956
17	type NOT GI	Bull_perf NOT good	Assets small	0.139	0.389	0.456	0.33154	0.12893
18	Bear perf NOT poor	Assets large	Bull perf good	0.885	0.278	0.511	0.45974	0.12771
19	Bull perf good	stks>90%? no	5yrRet>S&P? above	0.361	0.389	0.689	0.32676	0.12707
20	Assets large		Bull_perf good	0.810	0.456	0.511	0.27804	0.12666
21	Bull perf good		5yrRet>S&P? above	0.404	0.511	0.689	0.24750	0.12650
22	Bull perf good	stks>90%? no	Assets large	0.778	0.389	0.456	0.31295	0.12170
23	Bull_perf NOT good		Assets small	0.178	0.489	0.456	0.24767	0.12108
24	Beta over1	Bear perf NOT good	stks>90%? yes	0.231	0.278	0.611	0.43274	0.12021
25	distributions low	Assets small	Bull perf poor	0.594	0.344	0.267	0.33951	0.11694
26	Beta over1	Bear_perf NOT average	type A	0.636	0.233	0.244	0.49475	0.11544
27	type NOT A	5yrRet>S&P? above	Assets large	0.815	0.289	0.456	0.39539	0.11422
28	Bull perf NOT average	Bear perf NOT poor	5yrRet>S&P? above	0.405	0.456	0.689	0.24663	0.11235
29	Bull_perf NOT poor	Bear_perf NOT poor	5yrRet>S&P? above	0.405	0.456	0.689	0.24663	0.11235
30	Beta over1	Bull perf NOT good	type A	0.650	0.211	0.244	0.52854	0.11158
31	Assets large		Bull perf NOT poor	0.048	0.456	0.267	0.24076	0.10968
32	Bear_perf poor	Assets small	type A	0.600	0.267	0.244	0.41026	0.10940
33	Bull perf good		Assets large	0.723	0.511	0.456	0.21240	0.10856
34	Bull perf NOT average	Assets large	5yrRet>S&P? above	0.389	0.389	0.689	0.27443	0.10672
35	Bear_perf NOT average	Assets small	Bull_perf NOT good	0.216	0.400	0.511	0.26536	0.10614
36	type NOT B	Bear perf poor	5yrRet>S&P? below	0.925	0.433	0.689	0.23936	0.10372
37	Bear perf poor	Assets small	Bull perf NOT good	0.160	0.267	0.511	0.38785	0.10343
38	Bear_perf NOT poor	turnover high	stks>90%? no	0.952	0.222	0.611	0.46535	0.10341
39	type B	Assets large	5yrRet>S&P? above	0.167	0.122	0.689	0.84334	0.10307
40	Assets small		Bull perf NOT good	0.260	0.544	0.511	0.18900	0.10290

Figure 2.8: (c) Best rule set as derived by ITRULE on the quantised data.

average, then, the bull market performance was good with probability 0.966. Some of the more interesting rules (numbers 4, 7, 8, 12, 13, 20, etc.) suggest that funds with large assets perform better than funds with small assets, and that *not* being of type A is a contributory factor to good performance.

The next step will be to use these rules to perform automated reasoning. In order to do this, however, we must first consider the two problems of *inference* and *control.*

2.7 Probabilistic inference: theoretical foundations and practical techniques

2.7.1 Background and motivation

In this section we extend our theory to the problem of probabilistic inference. Because this is such a vast topic, a caveat is in order. The main goal of this section is to define the theoretical requirements for inference, identify the practical difficulties involved, examine some justifiable approaches and propose a feasible inference scheme using information theoretic ideas. The word "feasible" is important. We will concentrate on the main theoretical aspects of the problem, and hence, necessarily, some of the practical implementation details are omitted.

So far we have been mainly concerned with the problem of *learning*. We have developed a learning mechanism (ITRULE) by which our rational agent can build up a probabilistic model of its external environment, the quality of which is limited only by its resources. The next problem of course is in *using* this model. It is true to say (as we have seen from the experimental results of the previous section) that the learned model (the set of rules) is useful for analysing the nature of the domain, and brings to light new insights. However the real potential of the model lies in harnessing its inherent ability to model *reasoning* processes about the domain. In other words it should be possible to use this model to *infer* the states of certain propositions in the environment, based on partial or uncertain information about other propositions. In particular, if the reasoning agent is to appear competent, then it must display the ability to perform sequential reasoning, in that it can interact with its environment and obtain more *relevant* information about the proposition which it is trying to infer.

The notions of *relevance* and *context* are essential. *Context* means that at any point in the reasoning process the probabilistic model reflects the agent's best *a posteriori* beliefs as to the current state of the environment based on information obtained so far. Clearly this involves some form of probabilistic updating, but since we only have a partial model and incomplete information, the standard Bayesian equations for updating are not sufficient. We will be looking at this problem in detail later in this section. *Relevance* involves more than just having an accurate model, it is the process by which the agent can rank its possible courses of action and consequently make the decision as to which is the most relevant

in the current context. For our purposes we define the problem as that of *goal-directed* inference, where the goal proposition is defined by some external authority such as a user or a higher agent. The only actions which the agent can initiate are essentially of two forms. It can query the external environment concerning the states of certain attributes (*observable* attributes), and of course it can perform *updating* based on the information it obtains. Note that global updating is not necessarily automatic, but when viewed as a possible course of action it will only be carried out when it is relevant enough compared with other possible courses of action.

The computational trade-off between whether one updates the state of the entire model or queries for more new information is quite interesting. In some applications updating may be much cheaper than obtaining information externally. For example, in reasoning about a chemical process plant, external tests may be expensive enough to outweigh, by a considerable margin, the computational costs of probabilistic updating. On the other hand, if one had a real-time vision system where large quantities of information are available at virtually no cost, then one might prefer a greedier approach and obtain as much data as possible until one could directly infer the goal proposition without reasoning about intermediate hypotheses. This trade-off between obtaining information and then rationalising about it is prevalent in human reasoning. In some problem-solving domains where data can be obtained cheaply, humans don't seem to update their model at all. For example in determining whether or not the sun is shining we look out the window and see if the sky is blue, whereas in fact we could probably have inferred this fact by any number of other clues which did not require us to change our field of vision. On the other hand if we want to make some long-distance, complicated, travel arrangements we will think long and hard about the possible inferences we can make about our schedule (based on previous experience, particular constraints and perhaps current information about currency fluctuations), *before* we pick up the telephone and call a travel agent, given that conversations with travel agents tend to be rather taxing.

The point of all this is simply that in order to display rational behaviour an agent must have the following capabilities:

(i) the ability to model context, i.e., a scheme for updating its (probabilistic) model, and

(ii) the ability to determine relevance, i.e., a quantitative method of ranking the possible

courses of action.

Current approaches in expert systems which purport to model reasoning rarely display the ability to take context or relevance into account. Most systems either ignore probabilistic considerations or else, as we shall see later, use questionable approximations to the Bayesian approach. As such, it can be argued that these systems do not model context at all. The standard approach in many commercial expert systems today is to make wide-ranging global independence assumptions to the extent that context is irrelevant since it never changes. However, as we shall see, global independence assumptions are rarely justified. In terms of relevance, systems either use pre-programmed algorithmic techniques to impose an ordering on the sequence of actions (and consequently lack generality), or more commonly, use rather ad hoc techniques to determine the ordering. This ordering scheme is termed *conflict resolution* and consists of techniques to determine which rule, of several candidates, is the best to follow. Commonly used techniques include choosing the rule whose left-hand side has the most propositions (specificity), or using the most recently updated rule (recency). Apart from the heuristic nature of such schemes, they suffer the disadvantage of not easily allowing the mixing of forward and backward chaining strategies. *Forward chaining* consists of being given some facts about certain propositions, and then *firing* rules which have these propositions in their left-hand side. *Backward chaining* consists of choosing a right-hand side of a rule and then trying to determine if the propositions in the left-hand side of the rule are true, using further backward chaining if necessary. In a sequential machine, it is necessary to make decisions as to which rules to choose for either form of chaining. In our model, forward chaining consists of probabilistic updating via rules while backward chaining is equivalent to querying for a particular left-hand side proposition in order to infer a given right-hand side. The ability to mix both modes of reasoning is a necessary component for a good model.

The next two sections will be devoted to analysing the problems of probabilistic inference and relevant decision strategies. Since both of these topics constitute large areas of research in their own right, the treatment presented will necessarily be focussed on the particular problem of developing a computational model of probabilistic reasoning using our probabilistic rule representation. In particular, we will decouple the two problems, and discuss decision making with the implicit assumption that some form of updating is in place. In each section we will discuss the implementation of a relatively simple, plausible

(lower-bound technique) inference scheme. It should be noted, however, that this is primarily for illustrative purposes and that the *general* ideas discussed are not limited to any one particular inference scheme. The final section will discuss some experimental results obtained using the preceding theory as implemented with the particular plausible inference scheme.

2.7.2 Why probabilistic inference is not as hard as people think it is

Probabilistic inference in the face of incomplete information is a very difficult and relatively unsolved problem. Generating a potential solution is not the difficult part. The difficulty lies in justifying the solution. The general statement of the problem involves the probabilities of some propositions ($\{e_1, \ldots, e_n\}$, the evidence) being specified and we are required to *infer* the *a posteriori* probability of some related proposition (h, the hypothesis). Unfortunately we only have a partial model of the overall joint probability density, and in particular we do not generally have the higher-order conditional probability $p(h|e_1, \ldots, e_n)$, although we may have lower-order information such as $p(h|e_1)$, etc. Since previous work on this problem has been diverse, we will discuss all the major contributions in the field at appropriate points throughout the section, rather than in one summary review. For the present it suffices to know that most inference techniques generally use independence assumptions, and/or interval (rather than point) estimates for the probabilities, to simplify the problem. We will look at the problem from a very fundamental level initially, develop upwards and determine exactly where the Bayesian updating equations break down, and what alternative approaches, if any, exist. By doing so, we will carefully analyse and justify any assumptions which we make and identify the inconsistencies which may arise if other, more arbitrary, assumptions are made. A very common approach for example, is to make wide-ranging independence assumptions *independently* of the data. While, as we shall see, independence assumptions of some form are generally necessary, they should be made only if the data supports them. The general philosophy of ITRULE allows us to make *data-dependent* independence assumptions based on our rule set and the properties of the J-measure.

Before the actual discussion, a few points are worth noting which are often overlooked. The first point is that humans perform probabilistic reasoning based on an incomplete probability model, i.e., given the number of variables which we deal with in the environment it is unreasonable to expect that we have an exponentially large probability model. Leaving implementation details such as parallelism aside, we clearly structure our model in such a way as to render the number of probabilistic relationships tractable, yet accurate enough to perform useful inference. This should motivate our search for a computationally tractable model.

Second, it should be noted that while probabilistic information helps us a great deal, there is no strict requirement for it to be exact. By definition probabilities reflect an inherent uncertainty in our model of the environment. The philosophical question, of what probabilities actually mean, is relevant. One interpretation is that *all* probabilities, in essence, are subjective, since given any set of mutually exclusive and exhaustive events, they reflect *our* belief as to what the probability assignment over that set should be. Whether the assignment is derived from a physical model, from logical arguments, from random samples, or from purely subjective estimates, is immaterial in the sense that in the end, we are simply prescribing a 1-1 mapping between probabilities and events, a mapping that we believe to be correct. Hence different agents may have different assignments which reflect different beliefs. As measures of each agent's belief, or willingness to gamble, they are correct. As absolute measures of belief, corresponding to the real environment, there exists an ordering in terms of correctness. Yet it is likely that while none are absolutely correct, all are useful in some respect. While we would like our agent or model to be as correct as as possible, it is important to realise that even if our inference strategy cannot always be guaranteed to be correct, it may still be very *useful.* Cheeseman [92] and Pearl [93] discuss similar ideas in more detail.

A third related point is to refute the widely-held belief that Bayesian probabilistic inference is inherently doomed because of its combinatorial intractability. This theory has led to the abandonment, by many, of rigorous Bayesian inference techniques, with little or no justification (as tabulated for example in the overview paper by Stephanou and Sage [151]). The attitude was fostered by a number of influential papers in the late 1970's and early 1980's. What has been largely ignored is that these arguments were mainly proposed by researchers in the field of *medical diagnosis*, e.g., particularly influential

papers by Szolovits and Pauker [152] and Adams [153]. Medical diagnosis is an extremely difficult domain for probabilistic reasoning, even for human experts. It can certainly be argued that Bayesian techniques will be difficult to apply due to the extreme difficulty in identifying mutually exclusive events (symptoms and diseases) and assigning probabilities to these events. The fallacy in the general argument lies in extrapolating this result to general domains where such difficulties do not exist, particularly domains which exhibit considerable structure and are relatively well-understood, e.g., fault diagnosis in man-made systems such as telecommunication networks. It is in applications such as this that we must begin applying our models of probabilistic reasoning, and then work towards solving the more inherently difficult problems. Arguments based on the difficulties of modelling reasoning in medical diagnosis can not necessarily be extended to other domains.

2.7.3 Basic inference results

We begin by changing our notation slightly both to increase the clarity of the discussion and to be consistent with other treatments in this area. It is convenient to think of propositions as nodes in a network, and rules as links which connect nodes together. The nodal propositions may be conjunctions of basic propositions (events in the sample space of an attribute) and no requirement of exhaustivity or exclusivity is made initially (in terms of the nodes). We will adopt the shorthand of using $p(h)$ and $p(e_1)$ to represent $p(\mathbf{H} = h)$ and $p(\mathbf{E_1} = e_1)$ respectively. In addition $\sum_{\mathbf{H}} p(h)$ is taken to mean the sum over some mutually exclusive set of events in the sample space of $\mathbf{H}$, e.g., $p(h) + p(\bar{h})$. The notation is generally self-explanatory and unambiguous given the context. We can categorise our propositions into three groups: directly observable evidence, intermediate hypotheses, and the goal hypothesis. While we will generally talk about particular evidence and hypotheses for a given problem it is important to remember that our analysis is perfectly general in the sense in that *any* proposition in the system can be in any one of the three categories above for a given problem. Hence our system is extremely flexible in this sense, a direct consequence of the notion of generalised rule-induction.

A notational point is worth mentioning at this juncture. Others (primarily, Duda et al.

[119] and Pearl [154]) have defined inference networks based on propositional nodes which are linked by *channels* rather than *rules*. By default such networks have become known as *Bayesian inference networks* because of their adherence to probabilistic theory and Bayes' rule. Although our approach here is also Bayesian, reference to *Bayesian networks* from this point onwards may be understood as referring to prior work on these non-rule inference networks. We will refer to the approach presented here as *rule-based networks*.

Typically we are tracing the effects of evidence on intermediate hypotheses, which in turn bear on the goal hypothesis. Only the probabilities of the evidence propositions can be obtained directly from the external environment. The probabilities of all other propositions must be inferred. In order to apply sequential decision procedures, we need to be able to take each piece of evidence separately and then update the probabilities in the network based on that evidence alone. To facilitate the discussion we introduce the notion of a *source*, similar to the notion of a source in communication theory, and as has been used before in discussions on inference nets by Pearl [155]. A source is defined as a proposition whose probability has been determined via the external environment, a probability which can only be changed by contradictory evidence provided by other sources. For our purposes we can assume the simple case of consistent evidence, i.e., for the duration of the inference, the probability of a source remains constant. The advantage of defining a source in the system is that it provides a reference point form which to update. Lest we imagine that the definition is too restrictive, it is worth noting that *non-monotonic* reasoning, or reasoning where we can "change our minds" is quite a challenging field of research in its own right [156]. Monotonic reasoning has enough unsolved problems to keep us occupied.

For ease of notation, we use $p_s(x)$ ($p(x)$ with an s subscript) to denote the probability of the proposition x, *after* we have taken into account the evidence provided by the source s. Probabilities without subscripts refer to probabilities *before* s was taken into account. Since s is only the current source, probabilities without subscripts may have been updated based on earlier evidence. Reference to prior and posterior probabilities will be made relative to s. We will find it useful to establish some basic results regarding source updating.

Lemma 2.7.1:

$$p_s(h|s) = p(h|s) \tag{2-126}$$

Proof:

Both $p_s(h|s)$ and $p(h|s)$ are conditioned on s being true. Hence they are not affected by changes in the value of $p_s(s)$.

In a recent paper by Kyburg [157, section 4, p.276] a similar statement is made, with further supporting arguments from a philosophical viewpoint.

Corollary

$$p_s(h_1,\ldots,h_n|s) = p(h_1,\ldots,h_n|s) \tag{2–127}$$

Lemma 2.7.2:

$$p_s(h,s) = \alpha p(h,s) \qquad \text{where } \alpha = \frac{p_s(s)}{p(s)} \tag{2–128}$$

Proof:

$$\begin{aligned} p_s(h,s) &= p_s(h|s)p_s(s) \\ &= p(h|s)p_s(s) \qquad \text{(by lemma 2.7.2)} \\ &= \alpha p(h|s)p(s) \\ &= \alpha p(h,s) \end{aligned} \tag{2–129}$$

∎

Corollary

$$p_s(h_1,\ldots,h_n,s) = \alpha p(h_1,\ldots,h_n,s) \tag{2–130}$$

From these basic identities we can prove some useful results as follows.

Theorem 2.7.1:

$$p_s(h_1,\ldots,h_n|e_1,\ldots,e_m,s) = p(h_1,\ldots,h_n|e_1,\ldots,e_m,s) \tag{2–131}$$

Proof:

$$\begin{aligned} p_s(h_1,\ldots,h_n|e_1,\ldots,e_m,s) &= \frac{p_s(h_1,\ldots,h_n,e_1,\ldots,e_m,s)}{p_s(e_1,\ldots,e_m,s)} \\ &= \frac{\alpha.p(h_1,\ldots,h_n,e_1,\ldots,e_m,s)}{\alpha.p(e_1,\ldots,e_m,s)} \qquad \text{(by lemma 2.7.2)} \\ &= p(h_1,\ldots,h_n|e_1,\ldots,e_m,s) \end{aligned} \tag{2–132}$$

In words this result tells us that any *conditional* probabilities in the system, which include s in the conditional term, are invariant to changes in $p_s(s)$ (the same results hold trivially for probabilities with $p(\bar{s})$ in the conditional term since a change in $p_s(s)$ is also a change in $p_s(\bar{s})$). We note however that terms which do *not* have s in the conditional term, may change if $p_s(s)$ changes, i.e.,

$$p_s(h_1,\ldots,h_n|e_1,\ldots,e_m) \neq p(h_1,\ldots,h_n|e_1,\ldots,e_m) \qquad \text{in general} \qquad (2\text{–}133)$$

and

$$p_s(h_1,\ldots,h_n,s|e_1,\ldots,e_m) \neq p(h_1,\ldots,h_n,s|e_1,\ldots,e_m) \qquad \text{in general.}\ , \qquad (2\text{–}134)$$

provided that none of these probabilities are equal to either 1 or 0, in which case they are invariant. In particular we note that terms like $p_s(h|e)$ and $p_s(s|e)$ will be different in general from their *a priori* values of $p(h|e)$ and $p(s|e)$. In terms of an inference net, this means that links or rules which do not have s in their left-hand sides are liable to have their transition probabilities changed if $p_s(s)$ changes. The consequences for rule-based expert systems are serious. If the probability of any proposition in the system is observed and then changes from its *a priori* value, not only are the *a priori* probabilities of the other propositions in the system changed also, but the rule *transition* probabilities linking these propositions may change. This point has not previously been mentioned in earlier work on probabilistic inference, despite the fact that it has extremely important ramifications. For example, the issue of the computational complexity involved in global updating becomes relevant in this light (we shall return to this point later).

Of course rule bases should be designed in a modular manner to the extent that rule transition probabilities do not in fact change. Static rule transition probabilities are implicitly assumed in current expert system applications. We must focus on determining if such assumptions are feasible. Can we design rule-bases such that the rule transition probabilities do not change unless there is a "local" source, i.e., $p_s(h|e) = p(h|e)$ unless $h = s$. It transpires that only in certain cases can we make such assumptions.

Theorem 2.7.2:

$$p_s(h|e) = \delta.\frac{p_s(s)}{p_s(e)} + \bar{\delta}.\frac{1-p_s(s)}{p_s(e)} \qquad (2\text{–}135)$$

where

$$\delta = p(h,e|s) = p_s(h,e|s)\ , \tag{2–136}$$

$$\bar{\delta} = p(h,e|\bar{s}) = p_s(h,e|\bar{s})\ , \tag{2–137}$$

i.e., δ and $\bar{\delta}$ do not depend on $p_s(s)$, they only depend on *a priori* probabilities.

Proof:

$$p_s(h|e) = p_s(h|e,s)p_s(s|e) + p_s(h|e,\bar{s})p_s(\bar{s}|e) \tag{2–138}$$

$$= p(h|e,s)p_s(s|e) + p(h|e,\bar{s})p_s(\bar{s}|e) \qquad \text{(by Theorem 2.7.1)} \tag{2–139}$$

$$= p(h|e,s)p_s(e|s)\frac{p_s(s)}{p_s(e)} + p(h|e,\bar{s})p_s(e|\bar{s})\frac{p_s(\bar{s})}{p_s(e)} \qquad \text{(by Bayes' rule)} \tag{2–140}$$

$$= p(h|e,s)p(e|s)\frac{p_s(s)}{p_s(e)} + p(h|e,\bar{s})p(e|\bar{s})\frac{p_s(\bar{s})}{p_s(e)} \qquad \text{(by Theorem 2.7.1)} \tag{2–141}$$

$$= \frac{p(h,e,s)}{p(e,s)}\frac{p(e,s)}{p(s)}\frac{p_s(s)}{p_s(e)} + \frac{p(h,e,\bar{s})}{p(e,\bar{s})}\frac{p(e,\bar{s})}{p(\bar{s})}\frac{p_s(\bar{s})}{p_s(e)} \tag{2–142}$$

$$= p(h,e|s)\frac{p_s(s)}{p_s(e)} + p(h,e|\bar{s})\frac{1-p_s(s)}{p_s(e)} \tag{2–143}$$

∎

Theorem 2.7.2 tells us exactly how to calculate the *a posteriori* transition probability of any rule in the system, given a change in $p_s(s)$. This is fine except that in practice we may not know all the quantities in the above equations. For example,

$$p(h|e,s)p(e|s) = p(h,e|s) \tag{2–144}$$

may not be known in Equations (2–141) and (2–143), since these probabilities are not necessarily the original prior probabilities, but are the probabilities calculated based on all the prior evidence. Hence to have updated these probabilities previously, we would have had an equation similar to (2–143) except that all probabilities would be an order higher. Let us imagine the cumulative effect of calculating $p_{s_k}(h|e)$ where s_k is the kth piece of evidence considered. First we need $p(h,e|s_k)$, a third order piece of information. This in turn depends on the $(k-1)$th piece of evidence, s_{k-1}, and will require terms like $p(h,e,s_k|s_{k-1})$, and so on, until finally we require $p(h,e,s_k,s_{k-1},\ldots,s_2|s_1)$. In other words, in order to calculate $p(h|e)$ accurately (a second order term), based on k pieces of evidence, we need information

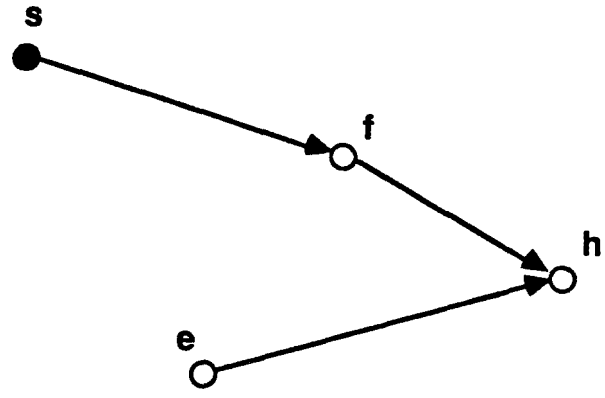


Figure 2.9: An example of the effect of context

up to order $k+2$ (we could have stated this directly since we are obviously estimating a joint distribution of order $k+2$, but is more insightful to see how the computations force the same requirement).

The difficulty of updating based on lower-order information alone is clear. Consider the consequences of not updating, using the simple example depicted in Figure 2.9. Let us say that e is a very expensive proposition to test, which, if true, confirms h to a certain extent, e.g., $p(h|e) = 0.8$. Consider that s is an initial piece of evidence which we have observed. We do not have any link directly between s and e. *Without* updating, if s makes h uncertain (say $p_s(h) = 0.5$) then we might still try to determine e if $p(e)$ were high and based on $p(h|e)$ being 1. However as we know from our theorems, both $p(e)$ and $p(h|e)$ may have changed to $p_s(e)$ and $p_s(h|e)$. In particular, if either $p(e)$ or $p(h|e)$ is near zero, then the test e has become *irrelevant* in the current *context*. With updating, we would have recognised this fact. The importance of updating and the consequences of ignoring context are apparent.

We have seen clearly that updating is necessary, but difficult. We must look at the basic updating equations and determine if we can reduce the complexity and in particular see if we can reduce the dependence on higher-order terms. Clearly we can express any updating in the network, using the key equations from Theorem 2.7.2, simply by choosing

e and h appropriately, where

$$p_s(h|e) = p(h,e|s)\frac{p_s(s)}{p_s(e)} + p(h.e|\bar{s})\frac{1-p_s(s)}{p_s(e)}$$
$$= p(h|e,s)p(e|s)\frac{p_s(s)}{p_s(e)} + p(h|e,\bar{s})p(e|\bar{s})\frac{p_s(\bar{s})}{p_s(e)} \; . \qquad (2\text{–}145)$$

The complexity arises because to update a kth order term requires $(k+1)$th order information. An obvious approach towards simplification is to replace higher order terms in the updating equations with lower order equations. Our treatment falls into two separate cases, namely, singly-connected networks and multiply-connected networks. The first covers the case where e or h is connected to s only via a single path, as shown for example in Figure 2.9. The second case, multiply-connected networks, is the more general and difficult case where e or h may be connected to s by multiple paths, as shown for example in Figure 2.10.

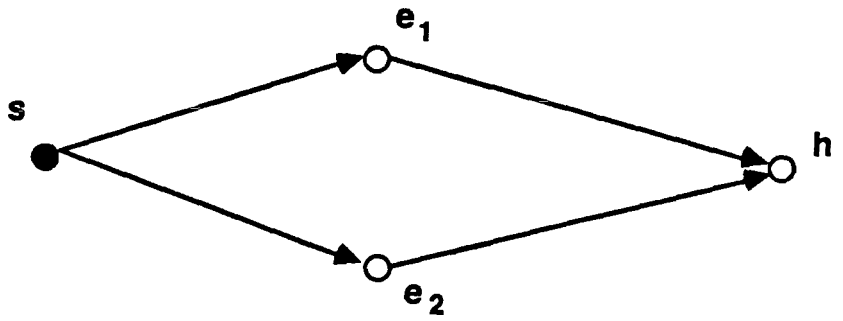


Figure 2.10: An example of a multiply-connected network

2.7.4 Inference in singly-connected networks

It will be useful to keep the examples of Figure 2.9 and 2.10 in mind as we proceed. In Equation (2–145) we cannot directly replace $p(h|e,s)$ by either $p(h|e)$ or $p(h|s)$ since h clearly depends on both s and e. Since we are calculating probabilities relating to e, given

s, it is more appropriate to rewrite

$$\begin{aligned} p(h,e|s) &= \frac{p(h,e,s)}{p(s)} \\ &= \frac{p(e|h,s)p(h,s)}{(p(s)} \\ &= p(e|h,s)p(h|s) \end{aligned} \tag{2–146}$$

and to focus on the $p(e|h,s)$ term. An immediately intuitive assumption is that

$$p(e|h,s) = p(e|h)\ . \tag{2–147}$$

In words, the probability of e given the fact that both h and s are true is equal to the probability of e given the fact that h is true. It does *not* mean that e depends only on h — it depends on s indirectly through h. As we shall discuss later, this is directly equivalent to the Markov independence assumption made in communication theory when $\mathbf{Z}$ depends on $\mathbf{X}$ only through an intermediate variable $\mathbf{Y}$. In terms of channels, the channel between $\mathbf{X}$ and $\mathbf{Y}$ and the channel between $\mathbf{Y}$ and $\mathbf{Z}$ are sufficient to describe the dependence of $\mathbf{X}$ and $\mathbf{Z}$. There is no invisible third channel directly between $\mathbf{X}$ and $\mathbf{Z}$. Our assumptions here are necessarily weaker than the Markov chain assumption in general, since in Equation (2–147) we are assuming conditional independence only on propositions, not on variables.

At this point we must address the problem of how the assumption in 2.140 relates to similar assumptions made in previous work. The reason for doing this is both to set the context of this presentation, and to check if the commonly accepted interpretation (Snow [158], Cheng and Kashyap [159], Stephanou and Sage [151], Winter and Girse [160]) of a result obtained by Pednault et al. [161] is correct. The claim in [161] essentially states that conditional independence assumptions lead to such inconsistencies that no probabilistic updating can take place. We have that

$$p(e,s|h) = \frac{p(e,s,h)}{p(h)}$$

$$= \frac{p(e|s,h)p(s,h)}{p(h)} \tag{2–148}$$

$$= \frac{p(e|h)p(s,h)}{p(h)} \qquad \text{(by assumption)} \tag{2–149}$$

$$= p(e|h).p(s|h) \tag{2–150}$$

Hence, the assumption made in Equation (2–147)is directly equivalent to the assumption that $p(e,s|h) = p(e|h)p(s|h)$. This latter form is known as the *weak* form of the conditional

independence assumption (Pearl [162]), and was originally introduced by Duda et al. [119] for the purposes of calculating likelihood ratios in inference nets. It has also been termed "CIA," or Conditional Independence on Atomic hypotheses (Cheng and Kashyap [159]). Duda et al. [119] also assumed in their initial paper that

$$p(e,s|\bar{h}) = p(e|\bar{h})p(s|\bar{h}) \ , \tag{2–151}$$

an additional assumption that has since caused much controversy. This is termed "CIN" or Conditional Independence on the Negation of hypotheses [159]. Considerable importance has been attributed to the findings of Pednault et al. (and, later, Cheng and Kashyap [159]) who have shown that if there are n hypotheses, $h_1,\ldots,h_n$ which are both mutually exclusive and exhaustive ($\sum_{i=1}^{n} h_i = 1$), and if $n > 2$, then the assumption of both CIN and CIA for each of the n hypotheses leads to the result that s and e are independent. This result has been misused to discredit *any* form of conditional independence assumptions. What is omitted is that there is no need or motivation for making the CIN assumption. As an example consider CIA and CIN for $n = 3$:

$$\text{CIA} \left\{ \begin{array}{l} p(e,s|h_1) = p(e|h_1).p(s|h_1) \\ p(e,s|h_2) = p(e|h_2).p(s|h_2) \\ p(e,s|h_3) = p(e|h_3).p(s|h_3) \end{array} \right\} \qquad \text{CIN} \left\{ \begin{array}{l} p(e,s|\bar{h_1}) = p(e|\bar{h_1}).p(s|\bar{h_1}) \\ p(e,s|\bar{h_2}) = p(e|\bar{h_2}).p(s|\bar{h_2}) \\ p(e,s|\bar{h_3}) = p(e|\bar{h_3}).p(s|\bar{h_3}) \end{array} \right\} .$$

If we restate the CIA assumptions using our original form (Equation (2–147)),

$$\text{CIA} \left\{ \begin{array}{l} p(e|s,h_1) = p(e|h_1) \\ p(e|s,h_2) = p(e|h_2) \\ p(e|s,h_3) = p(e|h_3) \end{array} \right\} ,$$

then it is even clearer that the CIN assumptions are unnecessary. It appears that while Duda et al. implicitly assumed that $n = 2$ (in which case CIA and CIN are equivalent) in their original paper, it has been erroneously interpreted that they advocated using CIA and CIN for $n > 2$. Hence, the negative results of Pednault et al. are not generally relevant.

It is interesting to note that Cheng and Kashyap [159] analysed the implications of assuming CIA and simultaneously assuming that e and s are independent, as is done in the well-known MYCIN expert system (Buchanan and Shortliffe [118], Adams [153]). It turns out that such assumptions are non-realistic in the sense that in order for them to be true, there are severe restrictions imposed on the possible nature of the underlying probability

distributions in the system. Purely from a probabilistic standpoint, without even considering the advantages (for updating) of doing so, Cheng and Kashyap conclude that of all the assumptions, CIA, or the weak form of conditional independence, is the least restrictive assumption to make. Further discussion of independence assumptions can be found in Charniak [163], Spiegelhalter [164], Wise and Henrion [165], Horvitz and Heckerman [166] and Johnson [167].

As we shall see when we discuss multiply-connected networks, we can make an even weaker assumption than CIA. Given our representation in terms of *rules* rather than *channels* it is not necessary to assume that $p(e|h_i, s) = p(e|h_i)$ for *every* h_i in the event set of $\mathbf{H}$. We merely need to make the assumption for those propositions for which links exist. In this sense we can refer to our assumption as a *partial Markov* assumption, where the complete Markov assumption corresponds to the channels which form a Markov chain as discussed earlier.

So far we have only justified the partial Markov assumption on the grounds that existing negative arguments against independence assumptions do not apply, and the fact that it corresponds to a common assumption made in analysing communication channels. However, given what we know about the J-measure and the origins of the rule-set, we can construct a more powerful argument in its favour. One particular interpretation of the instantaneous j-measure is that of a discrimination measure between two hypotheses, H_0 and H_1 (Section 2.4, originally in Blahut [126]). H_0 is the hypothesis that $\mathbf{X}$ is dependent on the event $\mathbf{Y} = y$, while H_1 is the hypothesis that $\mathbf{X}$ is independent of $\mathbf{Y} = y$. Since the j-measure, or binary discrimination, is the expected value of the log-likelihood between H_0 and H_1 with respect to H_0, then this expected value is large if H_0 is true, and small otherwise (H_1 is true). If ITRULE generated a rule-set based on the j-measure alone, to a certain degree of confidence, we could believe that the absence of a rule (or a link) implies independence. However ITRULE generates the links based on the j-measure multiplied by the probability of the event y. Hence we can only have a certain degree of confidence *on average* that conditional independence holds, since there may be rare events (when $p(y)$ is very small) when the j-measure is high (independence does not hold) but for which there is no link from s to e (the product, $p(y)$ times the j-measure is too small). Quantification of the exact level of confidence and the frequency of "rare" events are not addressed here. Instead we note that this is consistent with the notion we discussed earlier of trading-off the correctness of

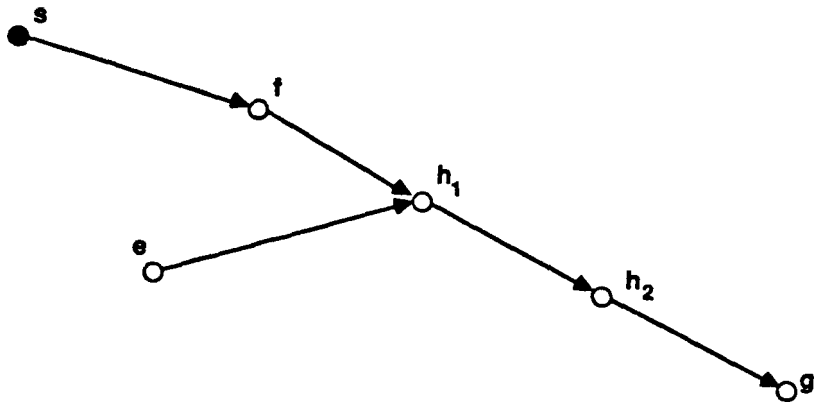


Figure 2.11: Singly-connected inference paths

the system with the resources available.

Let us examine the practical consequences of our partial Markov independence assumption, using the example shown in Figure 2.11. Here s is a source, h_1 and h_2 are intermediate hypotheses, e is an evidence proposition, and g is the goal hypothesis. The paths shown are assumed to be the only paths between the variables and so the propositions of interest are singly-connected. By our partial Markov assumption,

$$p_s(h_2|h_1,s) = p(h_2|h_1,s) = p(h_2|h_1) \tag{2-152}$$

so that to find $p_s(h_2)$, we find $p_s(h_1)$, and then calculate $p_s(h_2)$ based on $p_s(h_1)$ and $p(h_2|h_1)$ (Note that we are conveniently ignoring the problem of calculating the proposition probabilities *exactly*, since, as in this instance, $p(h_2)$ depends on $p(h_2|\bar{h_1})$. We will address this shortly, but for now we are mainly concerned with changes in *transition* probabilities). Hence, by Equation (2–152), we see that "forward propagation" of probabilities from the source does not cause the "forward" transition probabilities to change, based on our assumptions. Forward transition probabilities (for a singly connected network) are those which are "directed" towards the goal state (e.g., $p(h_1|e)$, $p(h_2|h_1)$ above). They correspond directly to the rule transition probabilities of interest in evaluating the J-measure.

Now consider updating $p_s(h_1|e)$. As before we have that $p_s(e|h_1,s) = p(e|h_1)$. Hence

we get that

$$p_s(h_1|e) = p(e|h_1)\frac{p_s(h_1)}{p_s(e)} \tag{2-153}$$

$$= \frac{p(e|h_1)}{p_s(e)}(p(h|s)p_s(s) + p(h|\bar{s})p(\bar{s})) \ . \tag{2-154}$$

From this we see that "backward" propagation of probabilities causes *forward* transition probabilities to change, based on our assumptions. Clearly this has important consequence on the *relevance* of e as determined by the J-measure. Next we consider the more difficult, yet potentially much more powerful, case of multiply-connected networks.

2.7.5 Inference in multiply-connected networks

A multiply-connected network is more general than a singly-connected network since we allow for multiple paths from evidence to hypotheses, or to use our earlier notation, e may be connected to s by parallel paths, as shown in Figure 2.10(b). In logical reasoning (essentially a special case of probabilistic reasoning where all the probabilities are either 1 or 0), single chains of inference are sufficient for determining conclusions. In probabilistic reasoning however, multiple chains (or paths) are inherently more powerful than single chains because they allow us to aggregate the effect of multiple intermediate hypotheses to conclude the probability of the goal hypothesis more accurately. Yet, because of the difficulties involved, the multiple path case has received far less attention than single path inference. Let us imagine treating each path separately as if it were singly-connected. Based on our previous analysis we know that updating on each path separately will in general involve some approximations or assumptions. Hence, different paths to a node, will, in general, involve different probabilities for that node. In essence this is the crux of the problem in multiple-path inference, namely, enforcing consistent updating.

Previous work in this area has circumvented the problem by various means which are not appropriate for our purposes. For example, Pearl [168] proposed an approach called *conditionalisation* where multiply-connected networks are effectively transformed to singly-connected networks by instantiating propositions to either *true* or *false*, and then linearly combining the results. We will not consider this approach primarily because Pearl uses

variables rather than propositions at the nodes and hence his model requires far more conditional probability information. In this sense his approach is not rule-based. The PROSPECTOR system of Duda et al. [119] used likelihood ratios rather than actual probabilities and hence allows one only to rank competing hypotheses rather than evaluate their probabilities explicitly. For our model we require exact probabilities in order to use the J-measure to evaluate relevance.

A simple example will illustrate the nature of the problem. Consider Figure 2.12(a) where we have a source connected to a hypothesis h via two intermediate pieces of evidence, e_1 and e_2. The joint probability distribution of h, e_1 and e_2 is tabulated in Figure 2.12(b). We can write $p_s(h)$ as

$$p_s(h) = \sum_{e_1,e_2,s} p_s(h|e_1,e_2,s)p_s(e_1,e_2,s) \tag{2-155}$$

$$= \sum_{e_1,e_2,s} p(h|e_1,e_2,s)p_s(e_1,e_2,s) \qquad \text{(by Theorem 2.7.1)} \tag{2-156}$$

$$= \sum_{e_1,e_2} p(h|e_1,e_2)\Big(\sum_s p_s(e_1,e_2|s)p(s)\Big) \qquad \text{(Markov independence)} \tag{2-157}$$

$$= \sum_{e_1,e_2} p(h|e_1,e_2)p_s(e_1,e_2) \tag{2-158}$$

$$= p(h|0,0)p_s(0,0) + p(h|0,1)p_s(0,1) + p(h|1,0)p_s(1,0) + p(h|1,1)p_s(1,1) \tag{2-159}$$

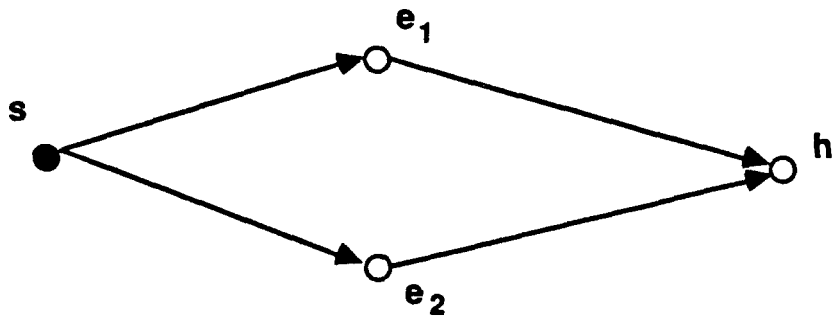
$$= 1.p_s(0,0) + \frac{3}{4}.p_s(0,1) + \frac{2}{3}.p_s(1,0) + 0.p_s(1,1) \ . \tag{2-160}$$

Hence, depending on $p_s(e_1,e_2)$, $p_s(h)$ can range from 0 to 1. For instance, if $e_1 = 0$ and $e_2 = 0$ then $p_s(h) = 1$, while if $e_1 = 1$ and $e_2 = 1$ then $p_s(h) = 0$. If we only have first-order rules in the system (i.e., the only links we know about are those shown in Figure 2.11(a)), then we only know that

$$p(h|e_1) = 0.6 \qquad \text{and} \qquad p(h|e_2) = 0.6 \ .$$

Given these 2 pieces of information alone does not in general allow us to find $p_s(h)$. If we learn that $e_1 = 1$ and $e_2 = 1$ then we may update $p(h)$ to 0.6, whereas in reality $p(h) = 0$ in this case.

Situations like this are bound to occur in practice, where our approximations do not hold. Based on our earlier discussions, the approach to take is to ensure that such occurrences are as rare as possible given the resource constraints. In the example above, we note that



(a) Multiply-connected

e_1	e_2	h	$p(h,e_1,e_2)$
0	0	0	0.00
0	0	1	0.30
0	1	0	0.15
0	1	1	0.30
1	0	0	0.05
1	0	1	0.15
1	1	0	0.05
1	1	1	0.00

(b) Joint distribution

Figure 2.12: (a) An example of a multiply connected inference network, (b) Joint probabilities of e_1, e_2 and h.

(i) $p(\bar{h}|e_1, e_2) = 1$, i.e., there exists a good rule which disconfirms h

(ii) $p(e_1, e_2) = 0.05$, which is quite small.

The first point which arises is that higher-order conjunctive propositions clearly play a useful role in multiple path inference. In this example, if the conjunctive proposition $\{e_1, e_2\}$ were present, then, as argued previously, the presence or absence of a link between this node and the node h indicates the dependence or independence of h on $\{e_1, e_2\}$ on average. Depending on the resource constraints, the link would be present based on the trade-off between $p(e_1, e_2) = 0.05$ and $p(\bar{h}|e_1, e_2) = 1$. Clearly the inclusion of kth order probabilistic information in the form of links in a connected graph requires the inclusion of nodes of order $k - 1$. Including higher-order propositions improves the quality of the underlying model. There is an interesting analogy between previous work in cognitive modelling and the fact that higher-order information improves inference capabilities in our model. Higher than second order statistical relationships are known to be required by humans for perceptual tasks such as texture discrimination (Julesz [169]) and learning the Boolean exclusive-or function (Minsky and Papert [170]). Hence for certain types of learning and inference problems, higher-order nodes may be necessary. Sejnowski et al. [141] introduced a similar concept in connectionist models. Their introduction of *hidden units* was motivated by the desire "to capture higher-order statistical relationships." It is worth noting that their model allows for the direct expression of disjunctive concepts, rather than purely conjunctive concepts as we have here. It should be noted at this point that we will not explicitly address the problem of updating conjunctive nodes given component information, e.g., determining $p(e_1, e_2)$ given $p(e_1)$ and $p(e_2)$, although we will make reference to this aspect of the problem later. One possible scheme would be to use the presence or absence of a rule between e_1 and e_2 to aid the updating by making data-dependent independence assumptions, e.g., if there is no rule then assume that $p(e_1, e_2) = p(e_1).p(e_2)$.

Probabilistic inference can be viewed as something of a gamble. A completely cautious inference scheme would only infer facts from other facts, i.e., use only rules and propositions with probabilities of 1 or 0. Cheeseman [171], and others (Konolige [172], Lemmer and Barth [173], Shastri and Feldman [174] Hunter [175] and Shore [176]) have proposed maximum entropy techniques for inference, where the probabilistic information as represented by the evidence and the rules is treated as a set of constraints on the underlying joint distribution of

the evidence and hypotheses. The entropy of the distribution is then maximised as originally proposed by Jaynes [177]. This technique has the advantage that it results in the *a posteriori* density which assumes no more information than that which is given. In addition it has been shown that conditional independence assumptions are subsumed by the maximum entropy approach (Konolige [172], Cheeseman [171]). However in this thesis we will not investigate the use of this inference scheme. One of the reasons for this decision is that it is computationally awkward to implement, i.e., it does not fit into the standard production rule system paradigm. The other reason is that the alternative, plausible reasoning, *is* easy to implement while not abandoning the basic tenets of probability theory. Plausible reasoning involves *bounding* the probabilities rather than calculating exact estimates. In a sense this interval-based approach is a generalisation of the point-valued approach, since the latter is a special case of the former when the upper and lower bounds are equal. The interval-valued approach is also more cautious in that the probabilistic conclusions may be weaker. It seems appropriate that we should imbue our artificial inference scheme with a conservative updating algorithm initially in that it has the option of being less committal when the supporting evidence is less certain. The arguments proposed by Cheseman [92] for using maximum entropy are certainly correct *if* one chooses the point-valued approach. Here we choose interval-values instead. It should be noted however that all of the results we obtain for using the J-measure in plausible reasoning are equally applicable to the maximum-entropy approach, if one were to implement that scheme.

The argument for bounding probabilities can be summarised in the following manner. For singly-connected inference we have seen that probabilistic updating is based on lower-order probabilistic approximations and conditional independence assumptions. With multiply-connected networks, inconsistencies at the nodes will inevitably result in practice, because of these approximations and assumptions. However, multiple inference paths offer potentially powerful inference capabilities over single paths (particularly for *probabilistic* inference). Hence, we would like to devise a practical updating scheme to handle multiple paths. To remove the requirement for probabilistic consistency at the nodes we relax the requirement that the probability values are point-valued.

2.7.6 Plausible inference: rationale and techniques

For each proposition of the form $\mathbf{X} = x$ we define a lower bound $t(\mathbf{X} = x)$ on the probability of the proposition being true and define $f(\mathbf{X} = x)$ as a lower bound on the probability of it being false, i.e.,

$$p(\mathbf{X} = x) \geq t(\mathbf{X} = x) \quad , \quad p(\mathbf{X} \neq x) \geq f(\mathbf{X} = x) \; . \tag{2-161}$$

The basic motivation is that we do not have sufficient information for updating *exactly*, since essentially we have only partial channels (rules) defined between the variables. Hence we adopt a more cautious approach where we draw inferences only if the data supports them. Among the advantages of an interval approach is the fact that the precision of our knowledge about the probability of a proposition x is represented by the difference between the two bounds. This bounding approach is termed *plausible* inference (Quinlan [178]). While the INFERNO [178] scheme of Quinlan also uses bounds, the scheme proposed here is quite different, in particular given that we have taken context into account. Quinlan's scheme ignores context since he does not allow the transition probabilities to change as a result of new evidence.

The basic mode of probability updating is via rules. For singly-connected networks if we have a rule of the form "If $\mathbf{Y} = y$ then $\mathbf{X} = x$ with probability p_1," and if we learn that $p(\mathbf{Y} = y) \geq p_2$ then

$$p(\mathbf{X} = x) \geq t(\mathbf{X} = x) = p_1.p_2 \tag{2-162}$$

$$p(\mathbf{X} \neq x) \geq f(\mathbf{X} = x) = (1 - p_1).p_2 \tag{2-163}$$

The propagation of probabilities in this manner is directly equivalent to forward chaining in expert systems. We begin to see the potential of this approach. High probabilities for hypotheses can only be inferred if they are supported by a highly probable piece of evidence which is linked to the hypothesis by a rule with a high transition probability. In particular, as the probabilities go to 1 or 0, the scheme allows for factual inference or deduction.

For multiply-connected networks, the updating is not so simple. Consider again the example of Figure 2.12. Even with interval-valued probabilities it is not clear how to update $p(h)$ in the light of s. Let us define the *maximum likelihood* updating equations as follows:

$$t(h) = \max_{1 \leq i \leq n} \{p(h|e_i).p(e_i)\} \tag{2-164}$$

$$f(h) = \max_{1 \leq i \leq n} \{p(\bar{h}|e_i).p(e_i)\} \quad , \tag{2–165}$$

where the e_i, $1 \leq i \leq n$, are the propositions (possibly conjunctive and by no means mutually exclusive) which are directly connected to h. These equations (to be referred to as the ML equations) require careful justification. An intuitive interpretation runs as follows. Because of the Markov principle, $p(h)$ can only be affected by changes in $p(e_1), \ldots, p(e_n)$. All we have available are the $p(h|e_i)$ and the $p(e_i)$, or bounds on the $p(e_i)$. For instance we might have $p(h|a,b)$ and $p(h|a)$ available, where $p(a) > p(a,b)$ but $p(h|a) < p(h|a,b)$. $p(a)$ is the more likely piece of evidence but $p(a,b)$ supports h more strongly. The ML equations tell us to choose $t(h)$ such that

$$t(h) = \max\{p(h,a), p(h,a,b)\} \ .$$

Of course theoretically we know that $p(h,a) \geq p(h,a,b)$, but in practice our beliefs assigned to these joint propositions do not necessarily obey this inequality, particularly when we we may be lower bounding the probabilities. In a sense we are choosing the most likely event, of the events we know, in the probability space defined by the set $\{h, e_1, \ldots, e_n\}$, where the propositions which are not mutually exclusive are appropriately redefined so as to render the space a proper event space. If this most likely event does not have a very high probability then the inference is cautious. On the other hand if the event has a high probability then both $p(e_j)$ and $p(h|e_j)$ must be near 1. In particular, if $p(e_j)$ and $p(h|e_j)$ are both 1, the inference is *certain* and facts are being propagated.

Inference involves a trade-off. Stronger inference statements allow us to be more decisive, but these statements may not always be correct. The maximum likelihood equations seem an appropriate compromise. If $p(h|e_j) = 1$ then the updating rule is provably correct, since the transition probability corresponds to a fact and is invariant to any changes to other propositions. However when $p(h|e_j) < 1$ then the statement that

$$p(h) > p(h|e_j)p(e_j) \tag{2–166}$$

may not be true. Returning to our earlier notation, we see that the correct statement to make is that

$$p(h) > p_{\{e_1,\ldots,e_n,\ i \neq j\}}(h|e_j)p(e_j) \ . \tag{2–167}$$

Of course $p_{e_1,\ldots,e_n}(h|e_j)$ is our old friend, the unobtainable updated conditional probability. The transition probability from e_j to h may be affected by the probabilities of the other

e_i. For instance, referring back to the example of Figure 2.12, we see that if $p(e_2) = 1$ then $p(h|e_1)$ changes from 0.6 to 0.0, which is quite a change. However, the basis of our argument will be that this type of change is rather unusual, and that on average the ML equations are likely to be correct. We introduce the following notation. We have a proposition h, and n evidential propositions $e_1, \ldots, e_n$. For any proposition e_j, let $\mathbf{O_j}$ be a variable whose distribution is defined as the joint probability distribution of $\mathbf{E_1}, \ldots, \mathbf{E_i}, \ldots, \mathbf{E_n}$, $j \neq 1, i, n$, where each $\mathbf{E_i}$ is defined as a variable taking values in the alphabet $\{e_i, \bar{e_i}\}$, and where it is assumed that the conjunctive propositions have been broken down into first-order propositions so as to render the space a proper event space, i.e., the events comprising of the alphabet of $\mathbf{O_j}$ are mutually exclusive and exhaustive. In words, $\mathbf{O_j}$ is defined as the joint variable of all the evidential propositions except for e_j. As before we denote $\sum_{e_j}$ or $\sum_{o_j}$ as the sum over all elements of the alphabet of $\mathbf{E_j}$ or $\mathbf{O_j}$ respectively. Using this notation, the correct updating equations are

$$p(h) = p_{o_j}(h|e_j)p(e_j) \tag{2–168}$$

$$= \Big(\sum_{o_j} p(h|e_j, o_j)p_{o_j}(o_j|e_j)\Big)p(e_j) \tag{2–169}$$

rather than the ML equation given by Equation (2–163). The ML equation will be in error whenever

$$p(h|e_j, o_j) < p(h|e_j) \tag{2–170}$$

given that o_j and e_j are both true. This occurs on average with probability $p(o_j, e_j)$. Hence we define the average error T as the size of the error multiplied by the probability of its occurrence:

$$T = \Big(\sum_{\{o_j : p(h|e_j,o_j) < p(h|e_j)\}} (p(h|e_j) - p(h|e_j, o_j)).p(o_j|e_j)\Big)p(e_j) \quad . \tag{2–171}$$

T is a measure of the average error in using the ML equations for updating. Obviously we would like T to be small if possible.

Theorem 2.7.3

$$T \leq \Big(p(h|e_j)\big(1 - p(h|e_j)\big)\Big).p(e_j) \tag{2–172}$$

$$\leq 0.25p(e_j) \tag{2–173}$$

Proof:

$$\text{Let } t = \sum_{o_j | p(h|e_j,o_j) < p(h|e_j)} (p(h|e_j) - p(h|e_j, o_j)).p(o_j|e_j) \tag{2-174}$$

$$\Rightarrow \quad T = p(e_j).t \tag{2-175}$$

Let S denote the set of propositions in the event space of $\mathbf{O_j}$ such that $p(h|e_j, o_j) < p(h|e_j)$, and $\bar{S}$ is the complement of this set.

$$\text{Let } \quad \alpha = p(h|e_j) \ , \tag{2-176}$$

$$q_i = p(o_i|e_j) \ , \tag{2-177}$$

$$\text{and } \quad p_i = p(h|e_j, o_i) \ . \tag{2-178}$$

where i is an index over the event space of $\mathbf{O_j}$. We seek to bound

$$t = \sum_S (\alpha - p_i) q_i \tag{2-179}$$

given

$$\sum_{o_i} p_i q_i = \alpha \ , \ \sum_{o_i} q_i = 1 \ , \ 0 \leq p_i, q_i \leq 1 \ . \tag{2-180}$$

$$\text{Let } q = \sum_S q_i \tag{2-181}$$

$$\Rightarrow \quad t = \alpha q - \sum_S p_i q_i \tag{2-182}$$

$$\text{Let } \quad \hat{p} = \frac{\sum_S p_i q_i}{q} \tag{2-183}$$

$$\Rightarrow \quad t = q(\alpha - \hat{p}) \tag{2-184}$$

Clearly $t \leq \alpha q$, where equality is achieved iff $\hat{p} = 0$.

$$\text{If } \ \hat{p} = \frac{\sum_S p_i q_i}{q} = 0$$

then either

(i) $\sum_S q_i = 0$, i.e., all $q_i \in S$ are zero. But this implies that $q = 0$ and hence $t = 0$ and so this case is of no use as a bound. Hence

$$\sum_S q_i \neq 0$$

(ii) Some $q_i \in S = 0$ and some $p_k \in S = 0$ where $i \neq k$ and i and k cover all elements of S. But if $q_i = 0, \Rightarrow p(o_j|e_j) = 0$ by definition. Hence $p(h|o_j, e_j) = p_i$ is not defined and so without loss of generality we can set all $p_i \in S = 0$.

From (i) and (ii), without loss of generality, all $p_i \in S = 0$.

$$\Rightarrow \quad \alpha = \sum_{S} p_i q_i + \sum_{\bar{S}} p_i q_i \quad \text{(by definition)} \tag{2-185}$$

$$\leq \sum_{\bar{S}} q_i \tag{2-186}$$

$$= 1 - q \tag{2-187}$$

$$\Rightarrow \quad q \leq 1 - \alpha \tag{2-188}$$

$$\Rightarrow \quad \alpha q \leq \alpha(1 - \alpha) \tag{2-189}$$

$$\Rightarrow \quad t \leq \alpha q \leq \alpha(1 - \alpha)$$

$$= p(h|e_j)\Big(1 - p(h|e_j)\Big) \tag{2-190}$$

∎

From this theorem we see that as $p(h|e_j)$, the initial rule transition probability (not updated), approaches 1, the bound on the error goes to zero. This is a consequence of the fact that the closer $p(h|e_j)$ is to 1 initially, the less susceptible it is to changes in the probabilities of other propositions. As mentioned earlier, for the special case of $p(h|e_j) = 1$, it is invariant to change. The bound is attained if and only if all of the $p(h|e_j, o_j) \in S$ are equal to zero. Such zero terms, or even near-zero terms, have the interpretation that $p(\bar{h}|e_j, o_j) = 1 - p(h|e_j, o_j)$ must be near 1 and hence, such terms will either be represented explicitly as rules, or if not, their overall probability of occurrence, $p(e_j, o_j, \bar{h})$ must be quite small. Using ITRULE to generate the rules in the network helps to keep T small, for if no rules exist for terms like $p(h|e_j, o_j)$ in the definition of T, then either $p(h|e_j, o_j) \approx p(h)$ or else $p(e_j, o_j)$ is very small. In either case the contribution to the error T is very small. Hence in the context of using rule-sets as generated by ITRULE, we can argue that the derived bound on T is considerably larger than the actual value of T in practice.

Another aspect of updating with the ML equations is the fact that conflicts may occur. A conflict is said to occur for a given proposition h if $t(h) + f(h) > 1$ simultaneously. As an example consider what happens if we have $p(h|e_j) = 0.8$, $p(e_j) = 0.9$ and $p(\bar{h}|e_k) = 0.7$,

$p(e_k) = 0.9$. Clearly we need to know $p(h|e_j, e_k)$ but in general we may not have this information. If we update using the ML equations we may get that $t(h) = 0.72$ and $f(h) = 0.63$. We will not dwell on this particular problem in this thesis except to acknowledge that any practical inference algorithm needs to have a scheme for resolving such conflicts. A well-known example of this is the Quaker Republican dilemma, where we have the following four rules:

rule 1: Dick is a Quaker

rule 2: Dick is a Republican

rule 3: Republicans are not pacifists with probability p_1

rule 4: Quakers are pacifists with probability p_2.

The problem of estimating the probability of Dick being a pacifist is awkward since there is conflicting evidence. Shastri and Feldman [174] argue that *both* the evidence of rule 3 and rule 4 are directly relevant, and they use a maximum entropy approach to find a value for the probability. Clearly we need to alter $t(h)$ and/or $f(h)$ in order to resolve the conflict, and while we can *detect* such conflicts using our scheme, the actual conflict resolution is left as as an open question.

In principle, with the maximum likelihood updating equations, we have now defined a scheme for probabilistic inference in a multiply-connected rule-based network. Later we will consider the *implementation* of such a scheme in practice, work through an actual example and discuss some of the remaining problems in more detail. Before we do that however we need to define our "executive" or control scheme, since, as we have seen earlier, inferencing or probabilistic updating is only part of the problem. In a sense we have built an engine but we need a control scheme to utilise it fully. *Controlling* the inference in a rational manner is the topic we now address.

2.8 Controlling the reasoning using decision theory

2.8.1 Background on statistical decision theory

Von Neumann and Morgenstern defined a quantity in 1947 called "utility" [179]. Utility is defined as a real-valued quantity which allows one to determine a preference ordering of possible decisions in the face of a set of possible outcomes, based on trading off the benefits and costs of each decision. The exact form of the utility function is determined by the particular problem at hand. Von Neumann and Morgenstern's original motivation in defining utility was to model subjective decision preferences in an effort to model economic behaviour of markets in a mathematical sense. Indeed most of their treatment of utility is relegated to an appendix of [179]. However, the importance of the concept of utility was quickly recognised, and was further refined and developed in the fields of statistical decision theory and decision analysis, e.g., Marschak [180], Wald [181], Savage [182], Blackwell and Girshick [183], Chernoff and Moses [184], Fishburn [185], De Groot [186], Luce and Krantz [187] and Von Winterfeldt and Edwards [188] to name but a few.

Although the theory would seem to be well-suited to the problem of building a rational machine (as we shall shortly see), almost all of the applications appear to have been in the fields of statistical experiment design and decision analysis. One wonders why the theory was not applied to the automated reasoning problem. Perhaps, in artificial intelligence circles at least, the reason is the lack of an underlying probabilistic model (in artificial intelligence) of the type we have proposed in this thesis. As we have stated before, our purpose is to design a machine that, using its *a priori* model of its external environment, obtains information from that environment in a rational manner and, hence, arrives at a conclusion about a particular situation. This is a sequential decision problem in the sense that the machine must decide what to query for, and given some information, whether to continue or to terminate (and make some conclusion). In a sense the problem is similar to that of decision tree design except that the agent must make decisions in "real-time," rather than fixing the decision sequence *a priori* and "off-line" as we would in tree-design.

One of the fundamental concepts of utility theory is that of *rational behaviour*, whereby if a_1 and a_2 are two acts with utilities $u(a_1)$ and $u(a_2)$, and if $u(a_1) > u(a_2)$, a rational decision-maker will always choose act a_1 over a_2. A consequence of this fact, and some

other axioms of rational behaviour as defined in [179], is the principle of maximum expected utility, or MEU as it is more commonly referred to. In words this means that a rational decision-maker will always choose the act which maximises his/her expected utility, where the expectation is defined over some suitable event space. It seems entirely appropriate to imbue an artificial reasoning machine with this well-defined theoretical scheme which models rational behaviour. We note of course that it is well known that humans do not exhibit simple MEU-type decision behaviour if utilities are measured in monetary terms, since otherwise insurance companies could not possibly make a profit (cf. the St.Petersburg paradox (Bernoulli [189]), the Allais paradox [190], the Ellsberg paradox [191]). Theoretical extensions to basic utility theory to encompass behaviour such as human aversion to ruinous losses have been developed by Kahneman and Tversky [192], among others (see Von Winterfeldt and Edwards [188], chapter 10, for a detailed discussion). Of course we are primarily interested in implementing a computational model rather than an accurate cognitive model, and hence the basic MEU theory is quite sufficient for our purposes. Future work might be to investigate using some of the other utility schemes which have been proposed [188].

More formally we define a set of m possible acts, $\{a_1, \ldots, a_m\}$, one of which must be carried out. In addition we define a set of n mutually exclusive and exhaustive events, $\{e_1, \ldots, e_m\}$, and define a probability measure on these events such that $\sum_{i=1}^{m} p(e_i) = 1$ in the usual manner. These events are intended to represent all possible states of the external environment. Hence in our case, the probability measure defined on these events is the joint distribution over all (or some subset of) the variables in our model. We define a real-valued utility measure over tuples of events and acts, i.e.,

$$u(e_i, a_j) \ , \ 1 \leq i \leq n \ , \ 1 \leq j \leq m \tag{2–191}$$

such that an ordering is defined in terms of preference of event-act tuples, to the extent that if the decision-maker prefers the tuple (e_i, a_j) over (e_k, a_l), then

$$u(e_i, a_j) \ > \ u(e_k, a_l) \ . \tag{2–192}$$

For our purposes we will define a real-valued utility *function* for any e_i and a_j. However it is worth noting that very often (e.g., in decision analysis [ref VW]), the utilities are determined indirectly from an expert's subjective preferences.

Furthermore we define probability measures on the events conditioned on particular acts such that,

$$\sum_{i=1}^{n} p(e_i|a_k) = 1 \ , \ 1 \leq k \leq m \ . \tag{2–193}$$

Given these basic definitions we can define the *expected utility* of an act as

$$u(a_k) = \sum_{i=1}^{n} p(e_i|a_k).u(e_i, a_k) \ , \ 1 \leq k \leq m \tag{2–194}$$

and hence the principle of maximum expected utility (MEU) is to choose the act such that

$$u(a_k) \geq u(a_j) \ , \ 1 \leq j \leq n, \ k \neq j \ . \tag{2–195}$$

Raiffa and Schlaifer [193] have further generalised the definition in Equation (2–195), but we shall use the simplified form. Our goal here is not to investigate the many possible avenues which exist in applying decision theory to artificial intelligence, but rather to *demonstrate* that it can be done and pave the way for future work in this area.

The maximisation of expected utility is directly equivalent to a Bayes decision which minimises the risk if we define loss functions which are the negatives of the utilities in Equation (2–195) (DeGroot [186], pp.122–123, Raiffa and Schlaifer [193], p.83). In practice the event space may not represent all possible states of the environment for obvious reasons. For example, to carry the principle of MEU to its fullest extent would imply that we analyse every possible future state and its utility. This is impractical both from the computational point of view and because the probability and utility assignments to many of the states would necessarily be quite inaccurate. Obviously humans do not analyse decisions in this manner. We make decisions based on near-term, simplified analyses, rather than analysing the consequences of a decision in terms of every possible state describing the rest of our lives. As argued on page 16 in Raiffa and Schlaifer [193], since the decision-maker *must* act, then even if he/she has only a limited model, the validity of using MEU based on this model is in no way diminished. One must make the best decision based on the information available. For our application we shall see that, given the nature of our model, decisions will be made based on local analyses. The advantages of using a *dynamic* scheme which invokes a more complicated analysis (larger event space) for a more important decision (higher utility, greater risk) is certainly apparent, but will not be dealt with in this thesis. The analogy with human behaviour is obvious if we think of the number of possible outcomes

we consider for the decisions to drink/not drink a cup of tea versus to buy/not buy a new car.

To illustrate the concepts of MEU let us consider a simple example. A technology advisor to a government in some country is faced with the following problem. A comet is due to fly over the country at some future date. The government would like to investigate the possibility of sending up an autonomous spacecraft with on-board telemetry for scientific purposes. The technology advisor is asked to recommend one of three possible decisions: launch from site A, launch from site B, or not launch at all (acts a_1, a_2, and a_3 respectively). The situation is such that if the weather is bad on the appointed day at the launch site, the launch cannot take place (and vice versa). The cost of not launching is put at 10 million rallod (local currency units), the estimated monetary effects of the negative political ramifications of missing the launch. The expected utility of act a_3, not to launch, is clearly unaffected by the events good and bad weather, and so the advisor defines $u(a_3) = 10$, in million rallod. The two possible events of concern are good and bad weather, e_1 and e_2 respectively. Based on historical data the advisor estimates the joint probabilities shown in table 2.1(a). The next step is to determine the 4 utilities, $u(e_1, a_1), \ldots, u(e_2, a_2)$. As is common practice (cf. [193], p.80), the advisor decides to sum costs and benefits linearly. Although the costs and benefits may not be immediately expressible in the same units, one can define a utility by normalising them to some common unit of measurement. For our purposes this is a natural role for the expert in the design stage of the expert system, since there exists a large body of theory and techniques in decision analysis for transforming expert's preferences into utilities (cf. Von Winterfeldt and Edwards [188]). Henrion and Cooley [194] provide an interesting comparison of decision-analytic techniques and expert system knowledge-acquisition ideas both being applied to a given problem.

Returning to the example, the advisor determines that the estimated benefits of a successful launch are worth 60 million rallod. The cost of building at sites A and B is 30 and 20 million rallod respectively. In addition, there is an extra 5 million rallod cost at site B in the event of no launch (bad weather) due to a local union contract for employee renumeration. Hence, for example, the utility of the tuple (site B, bad weather) is

$$\begin{aligned} u(e_2, a_2) &= u(\text{benefit}) + u(\text{construction}) + u(\text{no launch}) + u(\text{union}) \\ &= 0 + (-20) + (-10) + (-5) \end{aligned}$$

Probabilities	Good weather	Bad weather
Site A	0.9	0.1
Site B	0.8	0.2

Utilities	Good weather	Bad weather
Site A	30	-40
Site B	40	-25

Table 2.1: (a) Joint probabilities for the events in the spacecraft launch-site example, (b) event/act utilities for the same example.

$= -35$

The complete listing of utilities is given in table 2.1(b). Hence the advisor finds that

$$u(\text{site A}) = 23$$
$$u(\text{site B}) = 25$$
$$u(\text{no launch}) = -10 \quad .$$

Based on MEU the advisor recommends site B to the government. It is interesting to note what might have happened if the launch site was actually built at B and the weather was bad on the proposed launch day. If the government was made of up of rational politicians (this would severly limit the possible countries), it would accept the outcome (bad weather, no launch) in the knowledge that it had made the correct decision. More likely, not knowing any decision theory, it would fire the advisor even though he acted correctly. The implication for expert system technology is immediate. Based on MEU principles we know that the system may make decisions which, in retrospect, are not favourable. While we, as system designers, know that such an event has a definite (and hopefully small) probability of occurrence, it is extremely important that the *users* of the system understand this also. This is why explicit knowledge representation, and explanation and audit facilities, are extremely important for gaining the acceptance of automated reasoning technologies in any user community (the further implication being that the system should have memory of some form). The expert system may be "wrong" just like a human expert, but it is important to remember that the user understands that the system is still performing correctly.

2.8.2 Applying statistical decision theory to rule-based expert systems

Having outlined the basic principles of utility theory the next step is to see where it fits into a rule-based reasoning scheme. It is useful to remember our original goal of defining an implicit, and theoretically correct, control scheme for a rule-based system, while still retaining the characteristics of the basic match-and-fire architecture of production systems. We have seen that, in principle, MEU is a theoretically justifiable and appropriate decision scheme. The purpose of this section will be to define how it can be used to control the

reasoning in rule-based inference, i.e., how one might implement MEU theory to perform optimal backward and forward chaining (or combinations of both). The discussion here will focus on a general theory rather than a detailed implementation. In the section to follow we will discuss an application of the theory using the plausible inference schemes outlined earlier.

As we discussed earlier, rule-based systems use forward and backward chaining techniques to *focus* the search through the rule-base while trying to prove or disprove a particular hypothesis. Most commonly, heuristic techniques are used to order the rules on a *rule agenda*. The rule which has the greatest "score" based on some heuristic measure is selected for chaining. Such techniques include the use of *metarules* which encode domain-specific heuristics about "what to do next" in the form of rules (cf. Lenat and Harris [195], Davis [196], Lenat et al. [197]), and *conflict resolution* which establishes an ordering based on measures such as specificity (the number of propositions in the left-hand side which are true) and recency (higher ranking to more recently activated rules) (cf. McDermott and Forgy [198] and Hayes-Roth et al. [199]). These, and other more recent schemes such as that of Friedman [200], are primarily *qualitative* in nature, and although motivated by cognitive models, are not sufficiently quantitative to form the basis for a robust computational model for rule-based reasoning. More relevantly, for *probabilistic* reasoning, they are completely insufficient as mechanisms for controlling the reasoning, due to the additional computational requirements imposed by using probabilities. As we have seen earlier, rule-based systems using uncertainty have primarily adapted non-Bayesian techniques such as certainty factors [118]. There have been few contributions to the arena of controlling the inference in a rule-based system in a Bayesian model. The PROSPECTOR system (Duda et al. [201]) used log-likelihood measures to dynamically order the rules. While similar in spirit to the approach outlined here, the "rules" in the PROSPECTOR are really full channels. Ginsberg [202] discusses the general problem of controlling inference based on probabilistic rules, but his proposed techniques are rather simple and ad hoc, being based primarily on thresholding, which, as we know from Chapter 1 is difficult to implement in practice. Marques [203] uses the transition probabilities of the rules themselves for ranking. The disadvantages of this approach are immediately apparent if we consider the advantages of using the J-measure, rather than rule transition probabilities, to evaluate rule "goodness" as outlined in Section 2.5.

Let us first consider backward chaining. We know that the general idea is to focus recursively on the left-hand sides of rules in order to establish the validity of a particular rule or hypothesis. In our case we can assume that a goal hypothesis has been defined, and we wish to determine a probability statement for this goal proposition by backward chaining in some rational manner. If we can define the utility of a rule then we can use the MEU principle to make locally optimal backward chaining decisions. Consider the two rules $y_1 \Rightarrow x$ and $y_2 \Rightarrow x$. Let us define the possible events or outcomes as x and $\bar{x}$. The possible acts available to the decision-maker correspond to determining either $\mathbf{Y_1}$ or $\mathbf{Y_2}$, the random variables with alphabets $\{y_1, \bar{y_1}\}$ and $\{y_2, \bar{y_2}\}$. Since the outcomes of the acts are non-deterministic, the utility associated with the event-act tuples is an expectation over the possible act outcomes, i.e.,

$$u(x, \mathbf{Y_1}) = p(y_1).u(x, y_1) + p(\bar{y_1}).u(x, \bar{y_1}) \quad . \tag{2-196}$$

Since we are using rules rather than channels, then by definition,

$$u(x, \bar{y_1}) = 0 \tag{2-197}$$

and so

$$u(x, \mathbf{Y_1}) = p(y_1).u(x, y_1) \quad . \tag{2-198}$$

Hence we get the expected utility of the act of finding $\mathbf{Y_1}$ as

$$u(\mathbf{Y_1}) = \sum_x p(x|y_1).u(x, \mathbf{Y_1}) \tag{2-199}$$

$$= \sum_x p(x|y_1).p(y_1).u(x, y_1) \tag{2-200}$$

$$= p(y_1).\sum_x p(x|y_1).u(x, y_1) \tag{2-201}$$

The choice of $u(x, y)$ determines the nature of the expected utility. For example, if we define $u(x, y)$ as the change in the number of bits required on average to specify the event x, given y, we have

$$u(x, y) = \log_2 \frac{1}{p(x)} - \log_2 \frac{1}{p(x|y)} \tag{2-202}$$

$$= \log_2 \frac{p(x|y)}{p(x)} \tag{2-203}$$

and so

$$u(\mathbf{Y}) = p(y).\sum_x p(x|y).\log_2 \frac{p(x|y)}{p(x)} \tag{2-204}$$

$$= J(\mathbf{X}; y) \quad . \tag{2-205}$$

Defining the utility in terms of the expected information gain means that the agent's measure of benefit is in terms of information or uncertainty reduction. Since the agent's goal is to reduce its uncertainty about the goal proposition, such a definition is entirely appropriate. We may wish to include in $u(x, y)$ a measure of the cost $C(y)$ which will be incurred in determining y, e.g., we may determine y directly by observation, perhaps by performing an expensive or time-consuming test. In this case we define

$$u(x, y) = \log_2 \frac{p(x|y)}{p(x)} - C(y) \tag{2-206}$$

so that

$$u(\mathbf{Y}) = J(\mathbf{X} : y) - C(y) \tag{2-207}$$

where we follow the approach in [193] in defining additive costs, and where "cost" is suitably transformed from monetary units to bits (perhaps by expressing the expert's bits/dollar preferences using decision-analytic techniques). Cost could also incorporate a measure of the statistical risk associated with the rule, say in terms of a confidence interval based on the sample size from which the rule transition probability was estimated. The advantage of incorporating cost in this manner is that costs may be dynamic, subject to change even during a given inference situation. Hence the agent can easily adapt to the current state of the environment. If costs change after the inference net is designed, one need not redesign the system, as would be the case with a pre-defined decision structure such as decision trees.

The utilities defined above are very simple local measures. More complicated definitions are possible. If for example y is not directly observable then $C(y)$ will depend on other variables, the inference paths that are used to determine y, and so on. In turn, if $\mathbf{X}$ is not the goal variable, but an intermediate one, then the event space we consider may need to be extended. One approach would be to incorporate in the utilities $u(x, y)$ and $u(\bar{x}, y)$ some measure of how relevant each is in relation to determining the goal state. For example, one could weight them in terms of the utilities of the rules which have x and $\bar{x}$ in their left-hand sides, using the decomposition property of $j(\mathbf{X}; \mathbf{Y} = y)$ which we derived earlier. Hence, even if $\log \frac{p(x|y)}{p(x)}$ is very high, $u(x, y) = 0$ if x is not in the left-hand sides of any goal-relevant rule in the system (cognitive aspects of determining the support for rules in reasoning problems are discussed in Holland et al. [130]). More extensive utility measures, however, may lead to a greater computational burden on the system and a less robust the model. Hence, for the purposes of this thesis, we acknowledge the potential advantages

of more complicated utility measures while restricting our attention here to local utilities. This is in line with our overall goal of demonstrating the basic components of a probabilistic rule-based model.

By dynamically ordering the rules via expected utilities, the agent can implement the MEU principle, based on the J-measure, to locally optimise its backward chaining strategy. The strategy implicitly models *context* by using the latest probabilities available for calculating the utilities and J-measures, while the use of the MEU principle ensures that the agent is always following the most directly *relevant* line of reasoning.

The next step is to model forward chaining in a similar fashion. Forward chaining using probabilistic rules amounts to probabilistic updating, or inferencing, via rules, a topic we discussed extensively in Section 2.7. One of the main ideas from that section was that probabilistic updating itself can be classified as a possible act, i.e., the agent must choose whether to update based on available information, or whether to ignore it and pursue a different line of reasoning. In this sense if we can define a utility measure for forward chaining we can then mix forward and backward chaining, and combine inference and control within the single framework of MEU.

For forward chaining the possible acts can be identified as of the form: "updating $\mathbf{X}$ based on $\mathbf{Y}$," where we may consider such acts defined for $\mathbf{X_1}, \ldots, \mathbf{X_k}$ for a particular $\mathbf{Y}$, or we may even consider such acts defined for several $\mathbf{Y}$'s each with its own set of conclusions. In this way we can consider forward chaining based on either a single piece of evidence, or more generally, based on multiple, competing, pieces of evidence. The event space corresponds to the events y and $\bar{y}$ but since $u(x, \bar{y}) = 0$ by definition, the expected utility is defined as

$$u(\mathbf{Y}, \mathbf{X}) = p(y)u(\mathbf{X}, y) \tag{2-208}$$

for a given tuple, or rule, $(\mathbf{Y}, \mathbf{X})$. If we define the utility of $(\mathbf{X}, y)$ to be $u(\mathbf{X}, y) = j(\mathbf{X}; y)$, the instantaneous information, or the average change in information required to specify $\mathbf{X}$ given $\mathbf{Y} = y$, then we have that

$$u(\mathbf{Y}, \mathbf{X}) = J(\mathbf{X}; y) \quad . \tag{2-209}$$

Since forward propagation changes the probability of the propositions x and $\bar{x}$, all related J-measures must be recalculated. Hence there is an associated computational cost $C(\mathbf{X}, \mathbf{Y})$

associated with a particular rule so that

$$u(\mathbf{Y},\mathbf{X}) = J(\mathbf{X};y) - C(\mathbf{X},\mathbf{Y}) \quad , \tag{2-210}$$

where again $C(\mathbf{X},\mathbf{Y})$ is suitably normalised in terms of bits. As with the definition of utilities for backward chaining, more complicated measures which are less local are possibly worthy of attention but will not be considered here.

Expressing both forward and backward chaining decisions in the form of utilities which depend on the J-measure allows the agent to implement a decision strategy which mixes both schemes. This is a distinct advantage over existing techniques. Furthermore, the decision strategy inherently embodies the principle of rational behaviour by following the MEU principle. To quote from a paper [204] at a recent major artificial intelligence conference:

> One of the major weaknesses of current automated reasoning systems is that they lack the ability to control inference in a sophisticated, context-directed fashion.

Incorporation of the MEU theory in our model as described in this section is clearly a step forward. The next step, to be described in the next section, is to combine this theory of how to control the inference with a *particular* inference scheme, i.e., to examine the implementation issues in somewhat more detail.

2.9 Combining inference and control: implementation issues and problems

2.9.1 Simplifying the model

The problem of implementing the MEU theory with a particular inference scheme, namely lower-bounded probabilities, is now considered. The control (MEU) and inference (lower-bounding) together form the basis for a practical system. However it must be pointed out that the general theory is in principle applicable to a wide variety of control strategies and inference algorithms, not just the specific case presented here. For example, inference schemes based on maximum entropy could equally well be implemented using the MEU approach for control. However, as we have seen earlier, lower bounding of probabilities is particularly well-suited (as an inference scheme) to a *rule*-based representation.

We are going to make a number of simplifying assumptions in the example to be presented. The motivation for this simplification is to emphasise the contribution of our approach in its own right, without getting swamped by the practical difficulties and problems which are inherent to the general problem of probabilistic inference. We will identify the limitations which are imposed by these unsolved problems, and propose directions for future research.

From Section 2.7 we recall a scheme where we proposed using upper and lower bounds, $f(x)$ and $t(x)$ respectively, on the probability of a proposition x. To simplify the analysis we dispense with the upper bound $f(x)$ entirely and treat propositions and their negations separately. Hence a rule of the form $y \Rightarrow x$ can only be used to update $t(x)$ and $t(\bar{x})$. The logical implications of this technique are apparent. In essence, the inference only allows probabilistic *confirmation* of propositional statements. The evolution over time of any probability in the system must be non-decreasing. Problems will arise if the evidence allows for multiple interpretations depending on the order in which it is received, as for example with the case given in Figure 2.12 in Section 2.7. If e_1 and e_2 are observed sequentially, rather than together, the evolution of our belief in h (its probability) is decreasing. We acknowledge the monotonic restrictions of our simple scheme and defer discussions of non-monotonicity until a later point.

Another important issue which arises is the problem of utility estimates. Our prob-

ability estimates are lower bounded, which introduces a problem in terms of defining the expected utility. Loui [205] defined expected utility *intervals* by bounding the set of possible expected utilities. Unfortunately one can not in general define an ordering of the acts based on indeterminate utilities, and hence the MEU approach can not be directly applied by using interval estimates. We adopt the approach of using the *a priori* point estimate of a proposition's probability initially, replacing this by the lower bound, or $t(x)$, once this proposition is updated. The advantage of the scheme is its simplicity. The disadvantage lies in the fact that although the J-measure $J(\mathbf{X};y)$ (and consequently the utility) is convex in $p(x)$, using a lower bound on $p(x)$ does not guarantee a lower bound on $J(\mathbf{X};y)$. In other words, the number $t(x)$ does not reflect our belief in x, but rather a lower bound on our belief in x. Nonetheless, since this only affects the *order* in which we make decisions and has no effect on the correctness of the probabilities obtained, we use the simple approach even though it may be less optimal than some interval-based decision technique. It should be noted that such alternative approaches, based on bounding $J(\mathbf{X};y)$, are far from trivial. In particular for many cases of interest, when $p(y).p(x|y) > t(x)$ (i.e., application of a rule will confirm x), the lower bound on $J(\mathbf{X};y)$ is zero, which is of no use in making a decision. Hence we define for any rule (whether for backward or forward chaining) of the form $y \Rightarrow x$,

$$u(\mathbf{X},\mathbf{Y}) = t(y)\Big(p(x|y).\log\Big(\frac{p(x|y)}{t(x)}\Big) + p(\bar{x}|y).\log\Big(\frac{p(\bar{x}|y)}{1-t(x)}\Big)\Big) \qquad (2\text{–}211)$$

once x and y are updated, otherwise we use prior probabilities instead of $t(x)$ or $t(y)$.

We will not deal with costs in this example. However it should be apparent to the reader that variable costs require no theoretical extensions to the scheme, but simply a change in the definition of utility. For clarity we use an example based on first-order rules only and discuss later how to incorporate higher-order rules.

2.9.2 A worked example of rule-based reasoning

As a rule base for our inference we use the rules derived by the ITRULE algorithm using the mutual funds data as described in Section 2.6. The rules are listed in Figure 2.13, ranked in terms of the J-measure and defined with their *a priori* probability parameters. For computational purposes the parameters of interest for each rule are $p(x|y)$, $p(y)$, and

$p(x)$. The attributes are based on both general characteristics of mutual fund companies and their performance over a 5-year time span. For instance, *type A* (aggressive) and *type B* (balanced) are categories describing the type of investment strategy used, while *Bear perf* (good/average/poor) describes the company's performance in Bear markets over the 5-year time span. Figure 2.14 provides a diagrammatic representation of the inference chain, which we will work through. The attributes *type A* and *type B* are considered to be directly observable (*yes/no*) while the other attributes are not.

The inference begins when the user specifies a goal proposition whose current state is unknown. A fundamental difference between this rule-based approach and that of Bayesian nets (inference nets based on *channels*) is that we begin with an *empty* graph (just a set of vertices (nodes), no edges (rules)), and depending on the specified goal and subsequent reasoning decisions, add rules in a dynamic manner. Let us say that we choose *5-year return* as the goal attribute. This attribute says whether a company's performance over the 5 years was above (*a*) or below (*b*) the Standard and Poor index of performance. In Figure 2.13(a), the rule agenda is initialised with rules whose conclusions contain propositions about *5-year return*.

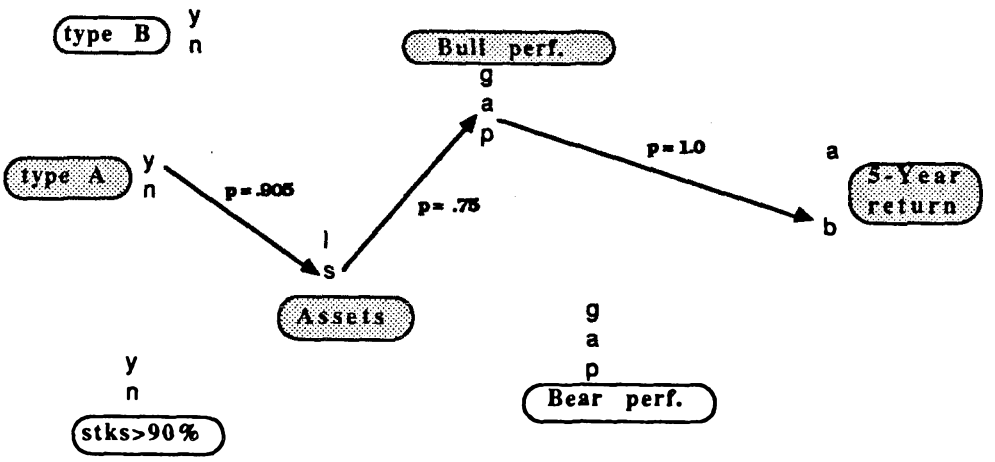
Since there is no initial data, the system begins by backward chaining from the goal proposition. The rule agenda (a dynamic list of ranked rules) is initialised with rules whose conclusions contain propositions about *5-year return* (Figure 2.14(a)). Backward chaining occurs initially using J-measures calculated from *a priori* probability values. The rule with the highest J-measure initially is "*Bull perf* $\neq$ *good* $\rightarrow$ *5-year return* = *below*". Since *Bull perf* $\neq$ *good* is not directly observable, further backward chaining must occur.

Here we choose to implement a depth-first backward search (as is most common in practice), where the system now makes up an agenda of rules with right-hand sides of *Bull perf* $\neq$ *good*. An alternative (breadth-first) strategy would be to continue backward-chaining on the initial proposition simultaneously. The utility approach allows the system to implement easily such different search strategies, perhaps varying strategies depending on the situation. Backward chaining on the new agenda continues in a depth-first manner as shown in Figure 2.14(a), via *Assets* = *small*, until the system arrives at proposition *type A* = *yes* which *is* observable. Let us assume it is observed to be true, i.e., the system is told that $p(type\ A = yes) = 1$.

ITRULE: first order rules for mutual funds

IF (antecedent)	THEN (consequent)	p(x/y)	p(y)	p(x)	J(X;y)
5yrRet>S&P? = above	Bull_perf = good	1	0.307	0.511	0.2968
Bull_perf = poor	Assets = small	0.957	0.261	0.545	0.1641
Assets = large	Bull_perf = good	0.825	0.455	0.511	0.1409
Assets = large	Bull_perf=NOT poor	0.975	0.455	0.739	0.139
Bull_perf = poor	5yrRet>S&P? = b	1	0.261	0.693	0.1381
Bull_perf = good	5yrRet>S&P? =above	0.6	0.511	0.307	0.1346
5yrRet>S&P? = above	Bull_perf=NOT poor	1	0.307	0.739	0.1341
Bull_perf = average	5yrRet>S&P? = b	1	0.227	0.693	0.1201
Bull_perf = good	Assets = large	0.733	0.511	0.455	0.118
5yrRet>S&P? = above	Bear_perf=NOT poor	0.889	0.307	0.557	0.116
5yrRet>S&P? = above	Bull_perf=NOTaverag	1	0.307	0.773	0.1141
Assets = small	Bull_perf=NOT good	0.75	0.545	0.489	0.1121
type = A	Assets = small	0.905	0.239	0.545	0.1064
type = B	stks>90%? = no	1	0.148	0.614	0.1041
Bear_perf = poor	5yrRet>S&P? = below	0.923	0.443	0.693	0.101
5yrRet>S&P?= above	Assets = large	0.778	0.307	0.455	0.0966
type = B	5yrRet>S&P? = above	0.769	0.148	0.307	0.0966
5yrRet>S&P? = below	Bull_perf =NOT good	0.705	0.693	0.489	0.0961
stks>90%? =yes	type = NOT B	1	0.386	0.852	0.0891
Assets = large	type = NOT A	0.95	0.455	0.761	0.0867
type = B	Beta = under1	1	0.148	0.682	0.0816
Beta = over1	stks>90%? = yes	0.679	0.318	0.386	0.08
type = A	stks>90%? = yes	0.714	0.239	0.386	0.0759
Beta = over1	type = NOT B	1	0.318	0.852	0.0734
type = A	Bear_perf = poor	0.762	0.239	0.443	0.0725
Assets = small	5yrRet>S&P?= below	0.875	0.545	0.693	0.0721
type = B	Bear_perf = good	0.692	0.148	0.295	0.0713
type = B	Bear_perf =NOT poor	0.923	0.148	0.557	0.0707
Assets = small	Bull_perf= NOT poor	0.542	0.545	0.739	0.0704
stks>90%? = yes	Beta = over1	0.559	0.386	0.318	0.0684
Assets = large	5yrRet>S&P? =above	0.525	0.455	0.307	0.0672
5yrRet>S&P? = above	type = NOT B	0.63	0.307	0.852	0.0663
type = A	Beta = over1	0.619	0.239	0.318	0.0655
stks>90%? = yes	Bear_perf = poor	0.676	0.386	0.443	0.0616
Bear_perf = good	stks>90%? = no	0.846	0.295	0.614	0.0555
Beta = over1	type = NOT A	0.536	0.318	0.761	0.0554
stks>90%? = yes	type = NOT A	0.559	0.386	0.761	0.0548
Bear_perf = poor	type = NOT B	0.974	0.443	0.852	0.0547
Bull_perf = poor	stks>90%? = yes	0.652	0.261	0.386	0.0543
type = A	Bull_perf=NOT good	0.762	0.239	0.489	0.0539
Bear_perf = poor	stks>90%? = yes	0.59	0.443	0.386	0.0539
type = A	Bull_perf = poor	0.524	0.239	0.261	0.0534
type = B	Bull_perf = good	0.846	0.148	0.511	0.0529
stks>90%? = no	Beta = under1	0.833	0.614	0.682	0.0526
Bear_perf = good	type = NOT B	0.654	0.295	0.852	0.0518
Bull_perf = poor	type = NOT A	0.522	0.261	0.761	0.051
stks>90%? = yes	Bear_per =NOT good	0.882	0.386	0.705	0.0503
5yrRet>S&P? = below	type = NOT B	0.951	0.693	0.852	0.0499
Bear_perf = good	5yrRet>S&P?=above	0.538	0.295	0.307	0.0491
5yrRet>S&P? = above	Bear_perf = good	0.519	0.307	0.295	0.048
Assets = small	type = NOT A	0.604	0.545	0.761	0.0477
stks>90%? = no	type = NOT A	0.889	0.614	0.761	0.0467

Figure 2.13: A list of the first order rules for the mutual funds inference problem, as ranked by the ITRULE algorithm

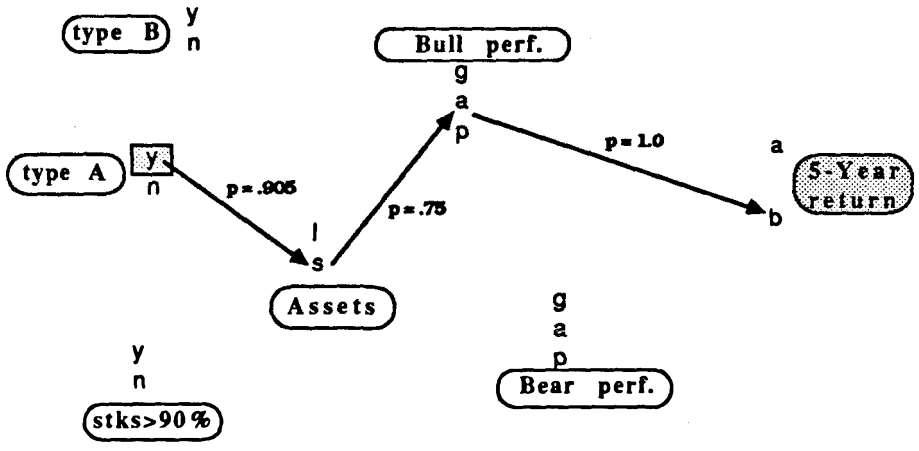


[1] Goal attribute = 5-Year return
Jmax = 0.138 bits with "Bull perf. = not good"

[2] Backward chain on "Bull perf = not good"
Jmax = 0.054bits with "Assets = small"

[3] Backward chain on "Assets = small"
Jmax = 0.112 bits with "type A = yes"

Figure 2.14: Representation of the inference flow: (a) initial backward chaining.



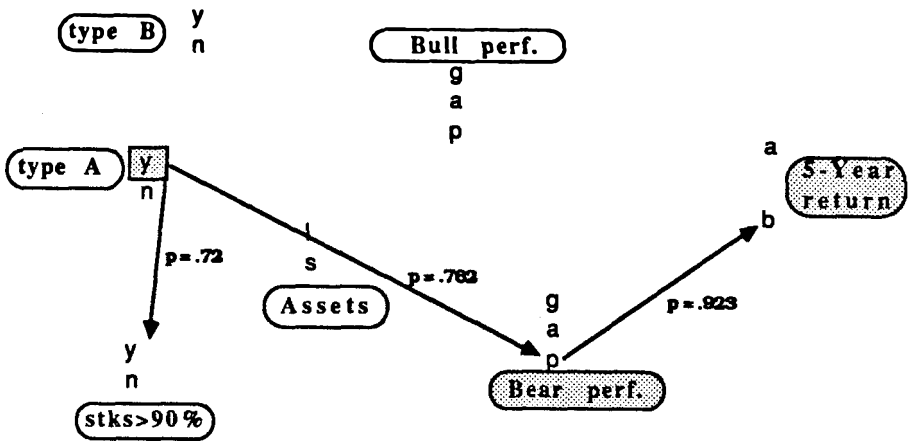
[4] Determine type A by direct observation :
type A = yes

[5] Forward chain from "type A = yes" to
"assets = small."
t(assets = small) = 0.905

[6] Forward chain from "assets = small" to
"Bull perf. = not good"
t(Bull perf. = not good) = 0.678

[7] Forward chain from "Bull perf. = not good" to
"5-Year return = below"
t(5-Year return = below) = 0.678

Figure 2.14: Representation of the inference flow (continued):
(b) forward chaining from evidence to goal.



[8] Jmax = 0.318 bits
Forward chain from "type A = yes" to "stocks > 90% = yes"

[9] Jmax = 0.304 bits
Forward chain from "type A = yes" to "Bear perf. = poor"
t(Bear perf = poor) = 0.762

[10] Jmax = 0.173 bits
Forward chain from "Bear perf=poor" to "5-Year return = below"
t(5-Year return=below) = 0.703

Figure 2.14: Representation of the inference flow (continued):

(c) mixed mode of chaining.

Forward chaining rules from *type A = yes* are put on the agenda. From this point onwards we can either forward chain based on the available evidence ($p(\textit{type } A = \textit{yes}) = 1$) or backward chain from the goal proposition along some other reasoning path. The latter might be preferable if the obtained evidence was not much different from our *a priori* probability estimate, i.e., forward chaining would yield little information. The utility measure determines which decision is chosen. In this case, since the evidence is definite ($p = 1$), the J-measures of the rules emanating from *type A = yes* increase to become equal to the instantaneous information of those rules. It turns out that the utilities of these rules are significantly higher than the utilities of any backward chaining rules. The best forward chaining rule has a utility of 0.455 bits while the best backward chainer's utility is only 0.138 bits. However it is worth noting that as forward chaining occurs, probabilities of propositions are changed to the extent that the utilities of backward chaining rules may increase to be ranked highest, given the new *context*.

As shown in Figure 2.14(b), forward chaining occurs from *type A = yes* to *Assets = small*. $t(\textit{Assets} = \textit{small})$ is updated to 0.905 using the updating rule defined in Section 2.7, i.e., for a rule of the form $y \Rightarrow x$,

$$t(x) = p(x|y).t(y) \quad . \tag{2-212}$$

where $t(y) = p(y) = 1$ in this case. Depth-first forward chaining continues along the same path as the initial backward chaining, until $t(\textit{5-year return} = \textit{below}) = 0.68$. Already the greedy J-measure strategy has paid off by finding a path of plausible inference from a direct observation, through intermediate hypotheses, to the goal attribute.

Continuing on in Figure 2.14(c), based on the greedy strategy, it continues to be more informative to forward chain from *type A = yes*, rather than backward chain from the goal attribute *5-year return*. However, after we forward chain from *type A = yes* to *Bear perf = poor*, $t(\textit{Bear perf} = \textit{poor}) = 0.762$. Since the *a priori* probability of this proposition was 0.443, the effect is to increase the $p(Y = y)$ term in the J-measures of rules emanating from this node. Thus, the J-measure for the rule "*Bear perf = poor* $\rightarrow$ *5-year return = below*" is increased by a factor of $\frac{0.762}{0.443}$. As a consequence, this rule which had been a backward chaining candidate lower on the agenda, has its J-measure increased sufficiently (from 0.101 bits to 0.173 bits) to be ranked highest. The important point is to note how the notion of *context* is taken into account implicitly. In the context of *type A = yes* being true, *Bear*

perf = *poor* becomes a more likely piece of supporting evidence for the goal proposition. This rule at the top of the agenda may be interpreted as either a forward-chaining rule from *Bear perf* = *poor* to the goal proposition, or else, a backward chaining rule in the reverse direction, i.e., the system could decide to try to further increase its belief in the proposition *Bear perf* = *poor* via further depth-first backward chaining. Had we included cost in our definition of the utility for this example, the two modes of chaining would have different costs in general, even for the same rule, and the system could have decided on the direction of inference based on the lowest cost (highest utility).

The inference continues until some strategy-dependent termination criterion is reached. We note that using a purely greedy technique can cause the inference to focus on irrelevant paths, i.e., become temporarily trapped at local optima. For instance in the above example the system greedily forward chained to *stocks* > 90%, even though this proposition is not relevant to the goal. In large inference networks such behaviour could be quite inefficient. The trade-off between local and global strategies is an interesting and complex issue. For example, we could use lookahead techniques that weight the terms in the J-measure of a rule, in proportion to the J-measures of rules associated with those terms. Other interesting questions arise. Would it be beneficial to *vary* the control strategy as the inference progresses, perhaps based on information-theoretic considerations such as the entropy of the goal variable? Are some strategies suited to particular applications? Could the strategy evolve with experience? The problem of defining more sophisticated control strategies and investigating the trade-off between computational control complexity and increased inference performance, certainly merit further investigation in future work.

The significance of our simple example should be emphasised. We have implemented a small rule-based probabilistic expert system in a completely *automated* manner, in terms of both learning the rules ("the learning phase") and controlling the inference ("the run-time phase"). The set of sample data, on which the model is based, is statistically accurate in that it represents the entire universe of possible funds. In addition it contains "expert" knowledge in the definitions and categorisations of the different attributes. We have achieved the goal of retaining explicit knowledge representation and implicit control based on a theoretical model. Of course we have only defined a very basic framework here. Some of the work which remains to be done, both theoretical and practical, is outlined in the next section.

2.9.3 Open issues and tentative solutions

There are a number of unresolved issues which we conveniently ignored in the example of the previous section. The first of these which we will consider is that of higher-order rules, and the representation of higher-order information in general in the rule network. As we have seen from the section on learning, more specialised higher-order rules may be very informative for some attribute sets depending on the inherent complexity of the underlying probability distribution. One approach, which we discussed earlier, is that of representing higher-order conjunctive propositions directly as nodes in the network. In a sense, these nodes are analogous to the intermediate or "hidden" units as defined in connectionist models [141]. A problem arises, however, in updating the probabilities of these nodes, for we have not defined anywhere in our theory the notion of rules whose conclusion statements are other than simple propositions.

One possible approach is to define "dummy" rules which update the probability of a conjunction based on bounds calculated on its component terms. For example, if $a.b$ is a conjunctive node, dummy rules are defined to it from node a and node b. We could upper bound $t(a,b) < \min\{t(a), t(b)\}$ (and define rules appropriately), but lower bounding $p(a,b)$ based on treating a and b separately is problematic. A better approach may be to change the representation at a more fundamental level so that at the *learning* stage we can identify important conjunctive nodes and include them as right-hand sides of rules. A natural extension of this is to allow the inclusion of arbitrary Boolean expressions of simple propositions. Clearly this is a much more difficult learning problem than the conjunctive rule learning problem we dealt with in this thesis. Generalising the ITRULE algorithm by defining information measures for rules relating arbitrary Boolean concepts may not be sufficient in itself given the combinatorial nature of the problem. One approach might be to use the expert to identify, in a heuristic manner, important intermediate concepts as a starting point for the algorithm. Another approach may be to use a neural network to learn the hidden units and then somehow identify them for inclusion as nodes in our inference network.

Another fundamental issue which occurs in our bounding scheme is that of upper and lower bound conflicts, or non-monotonicity. Clearly the simple scheme outlined in the previous section will not hold up in practice. Consider again the example given in Figure

2.12, Section 2.7.5. If we first learn that e_1 is true then we may update $t(h)$ to be 0.6, given that $p(h|e_1) = 0.6$. If we then learn that e_2 is also true, then we know that $t(\bar{h}) = 1.0$, since $p(h|e_1, e_2) = 1$. Clearly there is a conflict of belief since $t(h) + t(\bar{h}) > 1$. We know that in this case $t(h)$ is incorrect and should be zero. But this is only because we are certain that $t(\bar{h}) = 1$, i.e., this is based on factual information. If the probabilities did not correspond to facts, but were known with less than certainty, then we would not be sure which of $t(h)$ or $t(\bar{h})$ was incorrect. The general problem is to define a scheme to resolve this type of conflict. One approach is to use the ML updating scheme proposed in Section 2.7.6. Essentially the ML equation identifies the most likely set of supporting evidence, or the most likely overall joint probability among the variables present. We may recall that in fact the ML decision scheme has quite a low probability of error. Consequently we could use the ML equation to identify which of $t(h)$ or $t(\bar{h})$ is most likely to be correct. The other term can then be adjusted to $1 - t(\bar{h})$ or $1 - t(h)$ as appropriate. This scheme is a simple way to enforce consistency of probabilities between propositions and their negations. It allows us to implement a form of belief revision or non-monotonic reasoning.

Another level at which to implement conflict resolution is as follows. Essentially in the example with e_1, e_2 and h, what has occurred is that once e_2 becomes known we have that

$$p_{e_2}(h|e_1) = 0 \quad , \tag{2–213}$$

using the notation and ideas outlined in Section 2.7.3. Recalling the result of Theorem 2.7.2, i.e.,

$$p_s(h|e) = \delta.\frac{p_s(s)}{p_s(e)} + \bar{\delta}.\frac{1 - p_s(s)}{p_s(e)} \tag{2–214}$$

where

$$\delta = p(h, e|s) = p_s(h, e|s) \ ,$$
$$\bar{\delta} = p(h, e|\bar{s}) = p_s(h, e|\bar{s}) \ ,$$

we see that transition probability of the rule from e_1 to h has changed in the light of the new evidence e_2. Hence the earlier inference based on this rule is no longer valid. Ideally if we had a perfect probability model we could detect this immediately and recalculate the inference. In practice however, the change in transition probability will manifest itself under the guise of a probability conflict on h once e_2 becomes known. Given the conflict, if we could trace the source of the error to the transition probability at fault, i.e., $p(h|e_1)$, we need not use

the indirect ML approach but identify the exact problem instead. In some cases it may be possible to detect such errors using bounding techniques, since there are a variety of upper and lower bounds which can be defined on $p_s(h|e)$ using the definition of Theorem 2.7.2. If our value of $p(h|e)$ lies outside these bounds then it is definitely a source of error. In general, tracing possible inconsistencies in this manner is a complex and computationally expensive task. Quinlan's INFERNO system [206] uses a heuristic algorithm to detect and correct probabilistic inconsistencies in his rule-based network. However, he primarily attributes sources of error to being *input* data errors and does not explicitly acknowledge the fact that transition probabilities themselves are subject to change and cause errors. Attributing errors to external sources, rather than to a necessarily incomplete internal model, has been justly criticised by Cheeseman [92]. The topic of consistency and non-monotonicity in probabilistic rule-based systems certainly requires further investigation. For example, in order to detect sources of inconsistency, it is necessary for each node to keep track of how it is updated, i.e., memory is required. A significant side-effect of using memory in this manner would be to facilitate greatly explanation and audit capabilities of the system. We note in passing, that Pearl [155] has successfully defined algorithms for such non-monotonic probabilistic inference in *Bayesian* inference networks.

2.10 Future directions and current conclusions

2.10.1 Interesting research problems

In some respects this thesis has opened up more new problems than it has solved old ones. Undoubtedly, however, these new problems are quite interesting and merit continued consideration.

One of the immediate areas in which improvement can occur is with the ITRULE algorithm. The ITRULE algorithm is quite new and is based on a very simple premise, namely, maximise the J-measure. There are many "obvious" extensions. The knowledge representation scheme or model as used by ITRULE at present is extremely simple, being limited to conjunctive statements. A more general learning algorithm should be able to learn arbitrary Boolean expressions in order to improve its model of its environment. A practical approach might be to begin by quantifying the information content of rules such as,

If A or B then C.

We know in theory that identifying important disjunctive concepts is a very difficult problem. For example, neural networks (such as those based on Boltzmann learning algorithms) explore combinatorially explosive search spaces in order to discover *hidden units* representing higher order concepts derived from the input representation. In a sense, the same problem is known in artificial intelligence as *constructive* induction, where the algorithm discovers new representations in order solve a learning problem, i.e., improve the quality of its learned model. However the AI approach has little to contribute (except this extra terminology), as very little work has been done on this problem, indicating its degree of difficulty.

Yet another extension to ITRULE is the "re-coupling" of learning and inference. At present the algorithm learns rules independently of each other and has no overall criteria for evaluating a *set* of rules. For instance, there is nothing to prevent it from learning 20 rules relating to one proposition and none for another, if the J-measures turn out that way. A better approach obviously is not to have learning in isolation, but learning to solve specific problems. In other words learning and inference should occur together, as in a neural network or stochastic learning automaton. From a theoretical statistical viewpoint this type

of incremental learning is non-trivial in that we must make assumptions regarding the stationarity of the external environment. The potential advantages, however, are considerable. Expert systems, for example, could begin as non-committal novices and learn from experience (just like neural networks). Furthermore these systems would be truly adaptable to their environment, representing a significant advance over the current state of affairs where many of the expert systems being deployed are inevitably doomed to short life-spans.

The obvious question which arises in a discussion of this sort is "what exactly is the significance of the analogies between inference nets and connectionist models?" After all, both are trying to solve the same problem, both learn internal models of external worlds, and both use these models to effect conclusions concerning partial input statements. Yet, formally the two approaches are far removed from each other. The conjecture proposed here is that the differences between the two approaches are to a certain extent cosmetic and that at a fundamental level they have many inherent similarities. For instance, the recent work of Geffner and Pearl [64] has established a *plausible* interpretation of neural weights in terms of log-likelihood ratios in a Bayesian inference net. While this interpretation is not necessarily exact, it appears that there may well be a link between the weights of feed-forward type neural networks and rule transition probabilities, possibly via some logarithmic transformation. Could the threshold biases on the neurons in such feed-forward networks be related to *a priori* probabilities? Are the activation functions in the neurons related to probabilistic updating schemes complete with independence assumptions? Answers to these questions require considerable insight into the fundamental statistical nature of connectionist learning. The potential benefit of identifying such a mapping from neural to Bayesian nets is considerable. However, it must be stressed that there are fundamental theoretical issues to be resolved in bridging the gap, so that recent claims of progress in this area may be somewhat over-optimistic (Baum and Wilcek [207], Gallant [208]).

To close the discussion on this neural/Bayesian net analogy, we note that perhaps the most convincing argument arises at the implementation level, when one actually considers implementing a probabilistic inference system. Given the computational complexity of updating probabilities and utilities, it becomes obvious very quickly that a sequential machine will run into trouble. The ideal solution is to implement each proposition as an autonomous process which communicates with its neighbours, resulting in a distributed parallel updating system. Credit for the origination of this idea must go to Pearl [162] who has developed

algorithms precisely for parallel updating in Bayesian networks under certain assumptions, although no practical applications of this technique have been reported. One of the particular problems we would have to solve to implement our model in a distributed manner is the problem of distributed decision-making, since our scheme incorporates both inference *and* control. The conclusion that parallelism is necessary leaves one with a multiple path inference structure which, apart from the terminology, looks extraordinarily similar to a feed-forward neural network.

Another avenue of research concerns human expertise. The role of the expert is changed considerably with automated knowledge acquisition techniques. This leads to interesting questions in terms of using the expert to identify independence relations and to structure and quantise the data, and the statistical modelling of expert-supplied small samples of case studies. One could use decision-analytic techniques to elicit preferences from the expert in order to define domain-dependent utility functions. Another idea is to develop the role of the expert in working interactively with an algorithm like ITRULE, in order to reduce the learning complexity by effectively identifying significant intermediate concepts. These techniques are potentially far more effective in identifying domain expertise than existing techniques which try to elicit quantitative information from the expert directly.

There are interesting problems concerning the role of statistical decision theory and the control of probabilistic inference in expert systems. What are the trade-offs between greedy local decision strategies and more complex global strategies? Are such strategies application-dependent, i.e., should expert systems in some domains be risk-averters or risk-takers? Is MEU sufficient to model complex decision processes? What are the viable alternatives?

Of course many opportunities exist in terms of *applying* the theory in different domains. It seems likely that particular domains will exhibit sufficient prominent peculiarities to merit more detailed domain-dependent research. The implementation of a general machine vision system seems particularly interesting, where we could incorporate, in a very flexible manner, both pixel-derived information (attributes based on edge, texture and statistical information) with contextual domain-dependent information (e.g., the likelihood of certain objects given certain other objects). This combination of a higher-level inference process with lower level perceptual algorithms (such as edge detectors) is very interesting.

Problems such as speech recognition could be modelled in a similar manner. Naturally, the wide variety of "typical" expert systems applications are also amenable to this approach, with the general note of caution that 'structured' domains (where there is a large amount of inherent independence, e.g., man-made systems) are much easier problems to solve than "unstructured" domains (e.g., medical diagnosis). Less obvious applications include the use of ITRULE purely as an informative data analysis tool. For example, one application might be credit history analysis for a financial services companies. Even in a standard classification problem the rules generated by ITRULE yield a wealth of information concerning the relative importance of the attributes and their relation to each other. In general, the possible applications are limited by only our imagination but must be tempered by a degree of realism in assessing if the problem really fits the general theory we have outlined.

2.10.2 Conclusions

We have established that statistical and artificial intelligence ideas can be combined to yield interesting, new, computational models of intelligent behaviour. Our information-theoretic model for rule-based expert systems solved several problems (automated knowledge acquisition, implicit control) and introduced some new ones (the complexity of learning, probabilistic inference). The key points in the development included the definition of the J-measure to quantify the information content of a rule, the use of the ITRULE algorithm to identify the key components in a probabilistic model of the environment, the implicit data-dependent independence assumptions involved in using ITRULE rule sets to create an inference net, the use of bounding rather than exact calculations to perform probabilistic inference in a practical manner, and the adaptation of the statistical decision theory notion of utility to implement an implicit model of inference control. The outcome of these theoretical developments is that we are able to build systems which can learn from historical data, interact with their environment and exhibit rational decision-making behaviour. While the systems and algorithms we have described here may not be sophisticated enough to pass the "machine in a room" test, it seems clear that further work on the marriage of theoretical statistical principles, artificial intelligence insights, and parallel implementation techniques, will result in machines which could indeed pass the test.

Appendix 1

Proof of Theorems 1.1 and 1.2:

Let **Q** be a discrete random variable which can assume values from 1 to m, each with probability q_k, $1 \leq k \leq m$, $\sum_{k=1}^{m} q_k = 1$. Consider a discrete memoryless channel as shown in Figure A.1.

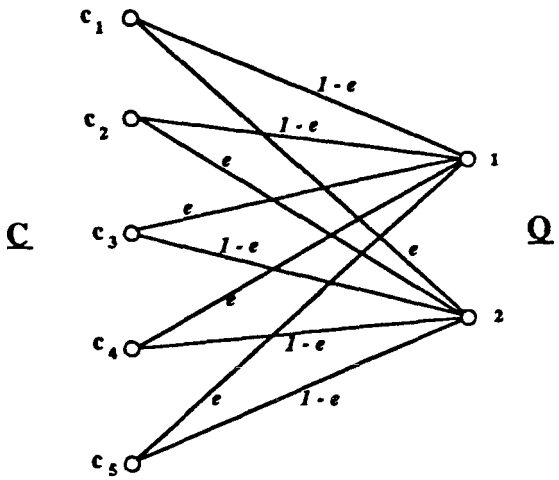


Figure A.1: Example of a node modelled as a discrete memoryless channel.

The input is **C**, the class variable, which has an alphabet $\{c_1, c_2, \ldots, c_K\}$ as previously defined. The received output is **Q**, the partition random variable, which has an alphabet of m letters corresponding to the m-ary partition of **C**. In the example in Figure A.1, $K = 5$, $m = 2$ and the partitioned subsets are $\{c_1, c_2\}$ and $\{c_3, c_4, c_5\}$. In other words, evaluating a feature at a node is equivalent to observing the output of a noisy channel where the possible outputs correspond to the different branches emanating from the node. One can thus write

$$I(\mathbf{C}; node_j) = I(\mathbf{Q}; \mathbf{C}) \qquad \text{(A-1)}$$

$$= H(\mathbf{Q}) - \mathrm{H}(\mathbf{Q}|\mathbf{C}) \; . \qquad \text{(A-2)}$$

In particular for no noise, $H(\mathbf{Q}|\mathbf{C}) = 0$ and this proves Theorem 1.1.

For the case with noise, one can define an m-ary symmetric channel with transition

probabilities such that

$$p(k|c_i) = 1 - \epsilon \quad \text{if } c_i \in k \tag{A-3}$$

$$= \frac{\epsilon}{m-1} \quad \text{if } c_i \notin k\ , \tag{A-4}$$

where $1 \leq k \leq m$ and $1 \leq i \leq K$

Then we have

$$\begin{aligned} H(\mathbf{Q}|\mathbf{C}) &= \sum_{k=1}^{m} \sum_{i=1}^{N_C} p(k, c_i) \log(\frac{1}{p(k|c_i)}) \\ &= \sum_{k=1}^{m} \sum_{i=1}^{N_C} p(k|c_i) p(c_i) \log(\frac{1}{p(k|c_i)}) \\ &= \sum_{k=1}^{m} \sum_{c_i \in k} p(c_i)(1-\epsilon) \log(\frac{1}{1-\epsilon}) + \sum_{k=1}^{m} \sum_{c_i \notin k} p(c_i) \frac{\epsilon}{m-1} \log(\frac{m-1}{\epsilon})\ . \end{aligned} \tag{A-5}$$

Since the partitions define disjoint subsets of the c_i's then

$$\sum_{c_i \notin k} p(c_i) = 1 - q_k \tag{A-6}$$

and so

$$\sum_{k=1}^{m} \sum_{c_i \notin k} p(c_i) \ = \ \sum_{k=1}^{m} 1 - q_k \ = \ m - 1\ . \tag{A-7}$$

This yields

$$H(\mathbf{Q}|\mathbf{C}) = (1-\epsilon) \log(\frac{1}{1-\epsilon}) \ + \ \epsilon \log(\frac{m-1}{\epsilon})\ . \tag{A-8}$$

Appendix 2

Proof of Theorem 1.3:

From Theorem 1.1, $I(\mathbf{C};\mathbf{Q}) = H_2(p)$, where p is the probability of one of the 2 subsets (and $1-p$ is the probability of the other subset). Therefore, it suffices to prove that

$$H_2(p) \geq H_2(\max\{p_{max}, \frac{1}{3}\}) . \tag{A-9}$$

There are three cases to consider:

(i) $p_{max} > \frac{1}{2}$:

Choose the partition probability $p = p_{max}$, i.e., partition the maximum component of the distribution on its own. Hence, in this case, the inequality is actually an equality condition. This choice is, in fact, the optimal choice for the partition since the function $H_2(p)$ is convex about the point $p = 0.5$. However, this is incidental to the proof, so it is sufficient to state that

$$H_2(p) = H_2(p_{max}) = H_2(\max\{p_{max}, \frac{1}{3}\}) .$$

(ii) $\frac{1}{3} \leq p_{max} \leq \frac{1}{2}$:

Now choose $p = p_{max} + \sum_i p_i$ so that

$$\left|\frac{1}{2} - p\right| \leq \left|\frac{1}{2} - p_{max}\right| \tag{A-10}$$

if such p_i's exist, i.e., p is closer to $\frac{1}{2}$ than $p_{\max}$. If such p_i's do not exist then choose $p = p_{\max}$. In any case, because of the convexity of $H_2(p)$,

$$H_2(p) \geq H_2(p_{max}) = H_2(\max\{p_{max}, \frac{1}{3}\}) . \tag{A-11}$$

(iii) $p_{max} \leq \frac{1}{3}$:

There exist at least 3 other $p_i < p_{\max}$ since

$$\sum_{i=2}^{3} p_i + p_{max} < \frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1 . \tag{A-12}$$

Consequently, it must be possible to construct an appropriate p by adding p_i to $p_{\max}$. More specifically, since all of the $p_i < \frac{1}{3}$ then there must exist some combination of k p_i's such that

$$\frac{1}{3} \leq \sum_{j}^{k} p_j < \frac{2}{3} . \tag{A-13}$$

Let $p = \sum_j^k p_j$, so that

$$\frac{1}{3} \leq p < \frac{2}{3} . \tag{A-14}$$

Now $\max\{p_{max}, \frac{1}{3}\} = \frac{1}{3}$ by definition so that by the convexity argument one gets

$$H_2(p) \geq H_2(\frac{1}{3}) = H_2(\max\{p_{max}, \frac{1}{3}\}) .$$

From cases (i), (ii) and (iii) the statement is true in general.

$$\text{If } p_{\max} < \frac{2}{3} \text{ , then } I(\mathbf{C};\mathbf{Q}) > H_2(\frac{1}{3}) = 0.918 \text{ bits.} \tag{A-15}$$

Proof of Theorem 1.3 extended:

Theorem 1.2 established that

$$I(\mathbf{C};\mathbf{Q}) = H_2(p) - H_2(\epsilon) . \tag{A-16}$$

Hence, it suffices to prove that

$$H_2(p) \;\geq\; H_2(\max\{p_{\max} + \epsilon(1 - 2p_{\max}), \frac{1+\epsilon}{3}\}) . \tag{A-17}$$

We have already seen in the noiseless case that

$$|p' - 0.5| \leq |\max\{p_{\max}, \frac{1}{3}\} - 0.5| , \tag{A-18}$$

i.e., we choose p' so that it is at least as close to 0.5 as the greater of $p_{\max}$ or $\frac{1}{3}$.

By introducing noise, can the basic inequality by changed, i.e., can $p_{\max}$ or $\frac{1}{3}$ be closer to 0.5 than p' when noise is accounted for? If not, then Equation (A-18) remains true with the inclusion of noise and, hence, Equation (A-17) holds.

It can easily be shown that

$$p + \epsilon(1 - 2p) = \max\{p_{\max} + \epsilon(1 - 2p_{\max}), \frac{1}{3} + \epsilon\frac{1}{3}\} \tag{A–19}$$

by using the definition $p = \max\{p_{\max}, \frac{1}{3}\}$. Hence, it remains to show that

$$|0.5 - (p' + \epsilon(1 - 2p'))| \leq |0.5 - (p + \epsilon(1 - 2p))| \tag{A–20}$$

given that

$$|0.5 - p'| \leq |0.5 - p| \ . \tag{A–21}$$

Equation (A–21) can be manipulated to yield

$$\begin{aligned}
\Big(0.5 - (p + \epsilon(1 - 2p))\Big)^2 &= (0.5 - \epsilon)^2 + 2p(0.5 - \epsilon)(2\epsilon - 1) + p^2(2\epsilon - 1)^2 \\
&= (0.5 - \epsilon)^2 + (2\epsilon - 1)\Big((1 - 2\epsilon)p + p^2(2\epsilon - 1)\Big) \\
&= (0.5 - \epsilon)^2 + (2\epsilon - 1)^2(p^2 - p) \\
&\geq (0.5 - \epsilon)^2 + (2\epsilon - 1)^2(p'^2 - p') \\
&= \Big(0.5 - (p' + \epsilon(1 - 2p'))\Big)^2 \ .
\end{aligned} \tag{A–22}$$

Appendix 3

Proof of Theorem 4

First we consider the following problem. We have N indistinguishable balls and k distinguishable cells. We can place the N balls in the k cells in any manner. Let us say that cell i contains c_i balls, $1 \leq i \leq k$, $0 \leq c_i \leq N$. In this sense the situation is equivalent to non-negative integer solutions of the equation

$$c_1 + c_2 + \ldots + c_k = N \ , \tag{A-23}$$

where N is fixed and the c_i are variable. It is easy to show (Feller [143]) that the *total* number of distinguishable solutions, T, is

$$T = \binom{N+k-1}{k-1} \ . \tag{A-24}$$

How many solutions exist which have more than l balls in any cell, where we restrict l such that $l \geq \frac{N}{2}$? For any given cell, say c_1 without loss of generality, we can rewrite an equivalent equation,

$$\sum_{i=2}^{k} c_i = N - c_1 \ . \tag{A-25}$$

If we assume that c_1 is *fixed*, i.e., c_1 is some fixed number $l_j > l$, then there are s_j distinguishable solutions to this equation and so, as before,

$$s_j = \binom{(N-l_j)+(k-1)-1}{(k-1)-1} \ . \tag{A-26}$$

Thus the total number of solutions of this form for which there are more than l balls in cell 1 is

$$\sum_{j=l+1}^{N} s_j = \sum_{j=l+1}^{N} \binom{(N-l_j)+(k-1)-1}{(k-1)-1} \tag{A-27}$$

$$= \sum_{v=0}^{v=N-l-1} \binom{v+k-2}{k-2} \tag{A-28}$$

$$= \binom{N-l+k-2}{k-1} \ . \tag{A-29}$$

This is only the number of solutions which have more than l balls in cell 1. The total number of such solutions, t_l, is simply k times this number by virtue of the fact that all such solutions are distinguishable given that $l \geq \frac{N}{2}$. Hence

$$t_l = k\binom{N-l+k-2}{k-1} \ . \tag{A-30}$$

The *proportion* p of such solutions is

$$p = \frac{t_l}{T} \tag{A-31}$$

$$= k.\frac{\binom{N-l+k-2}{k-1}}{\binom{N+k-1}{k-1}} \tag{A-32}$$

$$= k.\frac{(N-l+k-2).(N-l+k-3)\ldots(N-l)}{(N+k-1)(N+k-2)\ldots(N+1)} \; . \tag{A-33}$$

Let us now imagine a sample space consisting of k mutually exclusive and exhaustive events. We define a 'quantum' of probability mass to have a value 1, where a total of N such units are assigned to the complete space. To any event we can assign n_i quanta of probability mass (where n_i is an integer between 0 and N) such that $p_i = \frac{n_i}{N}$ where p_i can be interpreted as the probability of event i. We assume that the n_i are appropriately defined such that the basic axioms of probability are obeyed. By taking the limit as N goes to infinity we can model any general probability distribution in this manner. In this sense, $\lim_{N\to\infty} p$, where p is defined in the equation above, can be interpreted as the probability that any event in the sample space has a probability above p^*, where $p^* = \frac{l}{N} \geq 0.5$, based on the assumption that all probability distributions are equally likely as in Maxwell-Boltzmann statistics. Therefore,

$$\text{prob}(p_{max} > p*) = \lim_{N\to\infty} k.\frac{(N-l+k-2).(N-l+k-3)\ldots(N-l)}{(N+k-1)(N+k-2)\ldots(N+1)} \tag{A-34}$$

$$= k.\left(\frac{N-l}{N}\right)^{k-1} \tag{A-35}$$

$$= k.(1-p*)^{k-1} \; . \tag{A-36}$$

This concludes the proof.

Appendix 4

Proof of Theorem 1.5:

$$H(\mathbf{C}) - H(\mathbf{C}|\mathbf{T}) = I(\mathbf{C};\mathbf{T})$$
$$= \sum_{j \in S} p(node_j) I(\mathbf{C}; node_j) \ . \qquad \text{(A–37)}$$

In addition, we have that

$$\bar{d} = \sum_{j \in S} p(node_j) \ . \qquad \text{(A–38)}$$

But $I(\mathbf{C}; node_j) \leq 1 - H_2(\epsilon)$ at any node by Theorem 1.2, and under the assumption made for part (b) of this theorem one can use Equation (1–10) in Section 1.1.4,

$$I(\mathbf{C}; node_j) \geq H_2(p + \epsilon(1-2p)) - H_2(\epsilon) \ . \qquad \text{(A–39)}$$

Hence, by Equations (A–38) and (A–39) one gets,

$$\frac{I(\mathbf{C};\mathbf{T})}{1 - H_2(\epsilon)} \ \leq \ \bar{d} \ \leq \ \frac{I(\mathbf{C};\mathbf{T})}{H_2(p + \epsilon(1-2p)) - H_2(\epsilon)} \ . \qquad \text{(A–40)}$$

where $I(\mathbf{C};\mathbf{T}) = H(\mathbf{C}) - H(\mathbf{C}|\mathbf{T})$

Appendix 5

Derivation of a confidence interval for p_{max}

We consider the problem of deriving a confidence interval for $\hat{p}$ which is our estimate of p_{max} (henceforth to be referred to as p), the 'true' probability of the most likely class at a given node. We just consider the local problem, i.e., *given* the situation that we are at a given node in the tree design algorithm. p_{max} (or p) is an appropriate parameter to derive a confidence interval on, since it is the probability of correct classification at that node *if* we decide to declare it a leaf.

Let n be the number of samples at the node. For each class i, where $1 \leq i \leq K$, there are n_i samples such that $\sum_{i=1}^{K} n_i = n$ and $\hat{p}_i = \frac{n_i}{n}$ is the maximum likelihood estimate of p_i. However, from the point of view of misclassification error, we can reduce the effective number of classes to 2: 'the most likely class' and 'not the most likely class.' Let $\hat{p}_k = \hat{p}_{max}$ be our estimate of the most likely class, i.e.,

$$\hat{p_k} \geq \hat{p_i}, \qquad 1 \leq i \leq K, i \neq k \quad . \tag{A-41}$$

To simplify the notation we will simply refer to $\hat{p}_k$ as $\hat{p}$ and p_k as p.

We can think of our n samples as n independent experiments where p is the probability of success for each experiment. It follows that $\hat{p}$ follows a binomial distribution defined by

$$P\left(\hat{p} = \frac{k}{n}\right) = \frac{1}{n}.\binom{n}{k}.p_k(1-p)^{n-k}, \qquad 0 \leq k \leq n. \tag{A-42}$$

Assuming n is reasonably large, we can invoke the usual arguments (Cramér [209], p.515) and approximate this distribution by a normal distribution of the form

$$\hat{p} \sim N\left(p, \sqrt{\frac{p(1-p)}{n}}\right) . \tag{A-43}$$

Hence, we can write a confidence interval for $\hat{p}$, so that for a $(100-\alpha)\%$ confidence interval we have

$$\hat{p} = p \pm \lambda\sqrt{\frac{p(1-p)}{n}} , \tag{A-44}$$

where λ is the $\alpha\%$ level of the standard distribution $N(0,1)$.

What we actually need is a confidence interval for p in terms of $\hat{p}$, which we can estimate. We have that

$$\hat{p} - p = \pm\lambda\sqrt{\frac{p(1-p)}{n}} , \tag{A-45}$$

which if we square both sides yields

$$\hat{p} - p^2 = \frac{\lambda^2}{n}(p - p^2) \ , \tag{A-46}$$

from which we can get that

$$p^2\left(1 + \frac{\lambda^2}{n}\right) + p\left(-2\hat{p} - \frac{\lambda^2}{n}\right) - \frac{\lambda^2}{n}p = 0 \ . \tag{A-47}$$

The solutions of this quadratic equation yield the equivalent $(100 - \alpha)\%$ confidence interval for p in terms of $\hat{p}$, i.e.,

$$p = \frac{n}{n + \lambda^2}\left(\hat{p} + \frac{\lambda^2}{2n} \pm \lambda^2\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{\lambda^2}{4n^2}}\right) \ . \tag{A-48}$$

However, since n is large we can assume that

$$\lambda << n \ , \tag{A-49}$$

and, hence, we have

$$p \approx \hat{p} \pm \Delta \tag{A-50}$$

where the standard error Δ is

$$\Delta = \lambda\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \ . \tag{A-51}$$

As an example, consider a 99% confidence interval. We find that $\lambda = 2.576$. If $\hat{p} = 0.7$ and $n = 100$ we would have a confidence interval for p of

$$p = 0.7 \pm 0.18.$$

Derivation of an estimate of the total number of samples required

We would like to have a rough estimate of the total number of samples required for an algorithm which is using a particular W_α design parameter (for a definition of W_α see Section 1.1.6 on termination rules). Let M represent our estimate of the number of samples required.

$$M \sim \sum_{leaves} n_{critical} \tag{A-52}$$

$$= \sum_{leaves} \frac{4\lambda^2}{{W_\alpha}^2}\hat{p}(1-\hat{p}) \tag{A-53}$$

$$\sim \frac{4\lambda^2}{{W_\alpha}^2}(0.25)\sum_{leaves} 1 \tag{A-54}$$

$$\sim \frac{\lambda^2}{{W_\alpha}^2} n_l \ , \tag{A-55}$$

where n_l is the number of leaves. n_l is difficult to estimate since it depends on the noise, the particular termination rules in use and the particular data set. However, from our earlier results, we know that the average depth $\bar{d}$ grows as a function of H(C). To a very rough approximation we can estimate

$$2^{H(C)} \leq n_l \leq 2^{H(C)+2} , \tag{A–56}$$

based on empirical observations from using this algorithm. A reasonable estimate for M would appear to be

$$O(M) \sim \left(\frac{\lambda^2}{{W_\alpha}^2}\right) 2^{H(C)+1}. \tag{A–57}$$

As an example, consider the alpha-numeric problem discussed in Section 1.1.7, where $H(C) = 5$, $W_\alpha = 0.4$, $\lambda = 2.0$, which represents a 95% confidence interval of ± 0.2. We get that

$$O(M) \sim 1,600 , \tag{A–58}$$

so that the sample size of 5,000 which was used in the experiments appears to be quite sufficient based on these parameters and this approximate analysis.

Appendix 6

The form of the Delta-entropy rule given in Section 1.1.6 will only work when the original class distribution at the root node is uniform. In general, however, the distribution is non-uniform and a more general form of the rule is required. The general form works with normalised probability components $p(c_i)$ relative to the root node, i.e., before applying the Delta-entropy rule divide each non-zero probability component $p(c_i)$ at the node by its original value $p_0(c_i)$ at the root node. Then renormalise the distribution. Effectively,

$$p(c_i) \text{ is replaced by } \frac{p(c_i)}{p_0(c_i)} \frac{1}{\sum_{k=1}^{K} \frac{p(c_k)}{p_0(c_k)}} \tag{A–59}$$

Then proceed to apply the simple Delta-entropy rule as previously given to the normalised class distribution at the node. The reason for the normalization is to measure the *relative* decrease in uncertainty for each class.

References

1.1 The mutual information decision tree design algorithm

1. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.

2. G. F. Hughes, 'On the mean accuracy of statistical pattern recognizers,' *IEEE Transactions on Information Theory*, **IT–14** (1), 55–63, 1968.

3. A. Wald, *Sequential Analysis*, New York: Wiley, 1947.

4. P. M. Lewis, 'The characteristic selection problem in recognition systems,' *IRE Transactions on Information Theory*, **IT–8** (2), 171–178, 1962.

5. K. S. Fu, *Sequential Methods in Pattern Recognition and Machine Learning*, New York: Academic, 1968.

6. L. N. Kanal, 'Problem-solving models and search strategies for pattern recognition,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI–1** (4), 193–201, 1979.

7. P. H. Swain and H. Hauska, 'The decision tree classifier : design and potential,' *IEEE Transactions on Geoscience Electronics*, **GE–15** (7), 142–147, 1977.

8. S. Ganapathy and V. Rajaraman, 'Information theory applied to the conversion of decision tables to computer programs,' *Communications of the ACM*, **16** (9), 532–539, 1973.

9. C. R. P. Hartmann, P. K. Varshney, K. G. Mehrotra and C. L. Gerberich, 'Application of information theory to the construction of efficient decision trees,' *IEEE Transactions on Information Theory*, **IT–28** (4), 565–577, 1982.

10. B. M. E. Moret, 'Decision trees and diagrams,' *ACM Computing Surveys*, **14** (4), 593–623, 1982.

11. P. A. Chou, 'A procedure to design efficient probabilistic decision trees,' Master's thesis, Electrical Engineering and Computer Science Department, University of California (Berkeley), 1983.

12. P. A. Chou and R. M. Gray, 'On decision trees for pattern recognition,' presented at the 1986 IEEE International Symposium on Information Theory, Ann Arbor, Michigan, October 1986.

13. L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and regression trees*, Belmont, CA: Wadsworth, 1984.

14. R. G. Casey and G. Nagy, 'Decision tree design using a probabilistic model,' *IEEE Transactions on Information Theory*, **IT–30** (1), 93–99, 1984.

15. I. K. Sethi and G. P. R. Sarvarayudu, 'Hierarchical classifier design using mutual information,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI–4** (4), 441–445, 1982.

16. Q. R. Wang and C. Y. Suen, 'Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI–4** (4), 406–417, 1984.

17. E. Black, 'An experiment in computational discrimination of English word senses,' *IBM Journal of Research and Development*, **32** (2), 185–194, 1988.

18. P. Argentiero, R. Chin and P. Beaudet, 'An automated approach to the design of decision tree classifiers,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-4** (1), 51-57, 1982.

19. H. M. Dante and V. V. S. Sarma, 'Automated speaker identification for a large population,' *IEEE Transactions on Acoustics Speech, and Signal Processing*, **ASSP-27** (3), 255-263, 1979.

20. J. R. Quinlan, 'Discovering rules by induction from large collections of examples,' *Expert Systems in the Micro-electronic Age*, D. Michie (editor), Edinburgh: Edinburgh University Press, 168–201, 1979.

21. J. R. Quinlan, 'Learning efficient classification procedures and their application to chess endgames,' *Machine Learning: an Artificial Intelligence Approach*, R . S. Michalski, J. G. Carbonell and T. M. Mitchell (editors), Palo Alto, CA: Tioga Publishing Company, 463–482, 1983.

22. A. Patterson and T. Niblett, *ACLS User Manual*, Glasgow: Intelligent Terminals Ltd., 1983.

23. I. Kononenko, I. Bratko and E. Roskar, 'Experiments in the automatic learning of medical diagnostic rules,' Technical Report, Jozef Stefan Institute, Ljubljana, Yugoslavia, 1984.

24. C. E. Riese, 'RULEMASTER: Control strategies,' Radian Technical Report RI–RS–00296, Austin, Texas: Radian Corporation, 1985.

25. A. D. Shapiro, *Structured Induction in Expert Systems*, Reading, MA: Addison-Wesley, 1987.

26. D. Michie, 'Current developments in expert systems,' in *Applications of Expert Systems*, J. R. Quinlan (editor), Sydney: Addison-Wesley, 137–156, 1987.

27. D. Michie, S. Muggleton, C. Riese and S. Zubrick, 'RULEMASTER: a second-generation knowledge engineering facility,' *Proceedings of the First Conference on Artificial Intelligence Applications*, IEEE Computer Society, 1984.

28. J. D. Stuart, S. D. Pardue, L. S. Carr and D. A. Feldcamp, 'TITAN: an expert system to assist in troubleshooting the Texas Instruments 990 minicomputer system,' Radian Technical Report ST–RS–00974, Austin, Texas: Radian Corporation, 1984.

29. K. A. Horn, P. J. Compton, L. Lazarus and J. R. Quinlan, 'An expert system for the interpretation of thyroid assays in a clinical laboratory,' *Australian Computer Journal*, **17** (1), 1985.

30. R. M. Fano, *Transmission of Information*, Cambridge, MA: MIT Press, p 187, 1961.

31. R. G. Gallagher, *Information Theory and Reliable Communication*, New York: Wiley, p 43, 1968.

32. D. A. Huffman, 'A method for the construction of minimum redundancy codes,' *Proceedings of the IRE*, **40** (9), 1098–1101, 1952.

33. L. Hyafil and R. L. Rivest, 'Constructing optimal binary decision trees is NP-complete,' *Information Processing Letters*, **5** (6), 15–17, 1976.

34. J. K. Wolf and J. Ziv, 'Transmission of noisy information to a noisy receiver with minimum distortion,' *IEEE Transactions on Information Theory*, **IT–16** (4), 406–411, 1970.

35. R. W. Conners and C. A. Harlow, 'A theoretical comparison of texture algorithms,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-2** (3), 204–222, 1980.

36. V. A. Kovalevsky, *Image Pattern Recognition*, translated from Russian by A. Brown, New York: Springer-Verlag, p79, 1980.

37. M. R. Garey and R. L. Graham, 'Performance bounds for the splitting algorithm for binary testing,' *Acta Informatica*, 347-355, 1974.

38. J. R. Quinlan, 'The effect of noise on concept learning,' *Machine Learning (Volume 2)*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (editors), Los Altos: CA, Morgan Kaufmann Publishers, 149–166, 1986.

39. P. A. Chou, T. Lookabaugh and R. M. Gray, 'Optimal pruning with applications to tree-structured source coding and modelling,' preprint, Department of Electrical Engineering, Stanford University, CA, 1987.

40. A. G. Konheim, *Cryptography, a Primer*, New York: Wiley, p.16, 1981.

41. W. H. Highleyman, 'The design and analysis of pattern recognition experiments,' *Bell System Technical Journal*, **41** (3), 723–744, 1962.

1.2 The application of decision trees to edge detection

42. R. L. Andersson, 'Real-time gray-scale video processing using a moment-generating chip,' *IEEE Journal of Robotics and Automation*, **RA-1** (2), 79–85, 1985.

43. D. Marr and E. Hildreth, 'Theory of edge detection,' *Proceedings of the Royal Society (London) B*, **B–207**, 187–217, 1980.

44. J. F. Canny, 'Finding edges and lines in images,' M.I.T. Artificial Intelligence Laboratory, Report AI–TR–720, June 1983.

45. F. M. Wahl, *Digital Image Signal Processing*, Norwood, MA: Artech House, 1987.

46. A. M. Waxman, J. J. Le Moigne, L. S. Davis, B. Srinivasan, T. R. Kushner, E. Liang and T. Siddalingaiah, 'A visual navigation sytem for autonomous land vehicles,' *IEEE Journal of Robotics and Automation*, **RA-3** (2), 124–142, 1985.

47. D. Nitzan, 'Development of intelligent robots: achievements and issues,' *IEEE Journal of Robotics and Automation*, **RA-1** (1), 3–13, 1985.

48. I. Pitas and A. N. Venetsanopoulos, 'Edge detectors based on nonlinear filters,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-8** (4), 538–550, 1986.

49. I. E. Abdou and W. K. Pratt, 'Quantitative design and evaluation of enhancement/thresholding edge detectors,' *Proceedings of the IEEE* , **67** (5), 753–763, 1979.

50. D. C. Stoller, 'Univariate two-population distribution-free discrimination,' *Journal of the American Statistical Association*, **49**, 770–775, 1954.

51. J. H. Friedman, 'A recursive partitioning decision rule for nonparametric classification,' *IEEE Transactions on Computers*, **C-26** (4), 404–408, 1977.

52. D. E. Knuth, *The Art of Computer Programming (volume 3)*, Reading, MA: Addison-Wesley, p231, 1973.

1.3 Discussion and conclusions

53. A. P. White, 'Predictor: an alternative approach to uncertain inference in expert systems,' *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, 328–330, 1985.

54. J. R. Quinlan, 'Decision trees as probabilistic classifiers,' *Proceedings of the Fourth International Workshop on Machine Learning*, Los Altos: CA, Morgan Kaufmann, 31-37.

55. B. Arbab and D. Michie, 'Generating rules from examples,' *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, 631–633, 1985.

56. J. R. Quinlan, 'Generating production rules from examples,' *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, 304–307, 1987.

2.1 Building intelligent machines

57. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.

58. A. L. Samuel, 'Some studies in machine learning using the game of checkers,' *IBM Journal of Research and Development*, **3**, 211–229, 1959.

59. P. Winston, 'Learning structural descriptions from examples,' in *The Psychology of Computer Vision*, P. Winston (editor), New York: McGraw-Hill, 1975, 157–209.

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

61. J. A. Feldman and D. H. Ballard, 'Connectionist models and their properties,' *Cognitive Science*, **6**, 205–254, 1982.

62. J. J. Hopfield, 'Neural networks and physical systems with emergent collective abilities,' *Proceedings of the National Academy of Sciences, USA*, **79**, 2554–2558, 1982.

63. D. E. Rumelhart and J. J. McClelland, 'PDP models and general issues in cognitive science,' in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, D. E. Rumelhart and J. J. McClelland (editors), Cambridge, MA: The MIT press, 110–146, 1986.

64. H. Geffner and J. Pearl, 'On the probabilistic semantics of connectionist networks,' *Proceedings of the 1987 IEEE International Conference on Neural Networks (San Diego)*, **2**, 187–195, 1987.

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

66. R. Duda, P. E. Hart, N. J. Nilsson, P. Barrett, J. Gaschnig, K. Konolige, R. Reboh and J. Slocum, 'Development of the PROSPECTOR consultation system for mineral exploration,' SRI report, Stanford Research Institute, Menlo Park, CA, October 1978.

67. A. N. Campbell, V. F. Hollister, R. O. Duda and P. E. Hart, 'Recognition of a hidden mineral deposit by an artificial intelligence program,' *Science*, **217**, 927–929, 1982.

68. E. H. Shortliffe, S. G. Axline, B. G. Buchanan, T. C. Merigan, and S. N. Cohen, 'An artificial intelligence program to advise physicians regarding antimicrobial therapy,' *Computers and Biomedical Research*, **6**, 544–560, 1973.

69. V. L. Yu, L. M. Fagan, S. W. Bennett, W. J. Clancey, A. C. Scott, J. F. Hannigan, B. G. Buchanan and S. N. Cohen, 'An evaluation of MYCIN's advice,' *Journal of the American Medical Association*, **242**, 1279–1282, 1979.

70. R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum and J. Lederberg, 'Applications of artificial intelligence for organic chemistry: the DENDRAL project,' New York: McGraw-Hill, 1980.

71. D. H. Smith, B. G. Buchanan, R. S. Engelmore, J. Adlercreutz and C. Djerassi, 'Applications of artificial intelligence for chemical inference IX. Analysis of mixtures without prior separation as illustrated for estrogens,' *Journal of the American Chemical Society*, **95**, p.6078, 1973.

72. J. McDermott, 'R1: an expert in the computer systems domain,' *Proceedings of AAAI 1*, 269–271, 1980.

73. J. S. Bennett and C. R. Hollander, 'DART: an expert system for computer fault diagnosis,' *Proceedings of the 1981 International Joint Conference on Artificial Intelligence*, 843–845, 1981.

74. G. T. Vesonder, S. J. Stolfo, J. E. Zielinski, F. D. Miller and D. H. Copp, 'ACE: an expert system for telephone cable maintenance,' *Proceedings of the 1983 International Joint Conference on Artificial Intelligence*, 116–121, 1983.

75. S. K. Goyal, D. S. Prerau, A. V. Lemmon, A. S. Gunderson and R. E. Geinke, 'COMPASS: An expert system for telephone cable maintenance,' Computer Science Laboratory Report, GTE Laboratories, Waltham, MA, 1985.

76. H. Brown, C. Tong and G. Foyster, 'PALLADIO: an exploratory environment for circuit design,' *IEEE Computer*, 41–55, December 1983.

77. P. P. Bonissone and H. E. Johnson, 'Expert system for diesel electric locomotive repair,' *Knowledge-based Systems Report*, General Electric Co. , Schenectady, NY, 1983.

78. R. G. Smith and J. D. Baker, 'The DIPMETER advisor system,' *Proceedings of the 1983 International Joint Conference on Artificial Intelligence*, 122–129, 1983.

79. D. A. Waterman and M. Peterson, 'Models of legal decision-making,' Report R–2717–ICJ, The Rand Corporation, Santa Monica, CA, 1981.

80. W. A. Martin and R. J. Fateman, 'The MACSYMA system,' *Proceedings of the Second Symposium on Symbolic and Algebraic manipulation*, 59–75, 1971.

81. D. Michie, S. Muggleton, C. Riese and S. Zubrick, 'RULEMASTER: a second-generation knowledge engineering facility,' *Proceedings of the First Conference on Artificial Intelligence Applications*, IEEE Computer Society, 1984.

82. G. F. Luger, 'Mathematical model building in the solution of mechanics problems: human protocols and the MECHO trace,' *Cognitive Science*, 5, 55–77, 1981.

83. E. Scarl, J. Jamieson and C. Delaune, 'Knowledge-based fault monitoring and diagnosis in space shuttle propellant loading,' *Proceedings of the National Aeronautics and Electronics Conference*, Dayton, Ohio, 1984.

84. L. A. Zadeh, 'Fuzzy sets,' *Information and Control*, 8, 338–353, 1965.

85. B. R. Gaines, 'Fuzzy and probability uncertainty logics,' *Information and Control*, 38, 154–169, 1978.

86. H. Prade, 'A computational approach to approximate and plausible reasoning with applications to expert systems,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI–7**, 260–283, 1985.

87. L. A. Zadeh, 'Is probability sufficient for dealing with uncertainty in AI: a negative view,' in *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer (editors), New York: Elsevier Science Publishers B.V., 103–116, 1986.

88. P. Cheeseman, 'Probabilistic v. fuzzy reasoning,' in *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer (editors), New York: Elsevier Science Publishers B.V., 85–102, 1986.

89. A. P. Dempster, 'Upper and lower probabilities induced by a multivalued mapping,' *Annals of Mathematical Statistics*, **38**, 325–329, 1967.

90. G. Shafer, *A Mathematical Theory of Evidence*, Princeton: Princeton University Press, 1976.

91. H. E. Kyburg, 'Bayesian and non-Bayesian evidential updating,' *Artificial Intelligence*, **31**, 271–293, 1987.

92. P. Cheeseman, 'In defense of probability,' *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (Los Angeles)*, **2**, Los Altos, CA: Morgan Kaufmann Publishers, 1002–1009, 1985.

93. J. Pearl, 'How to do with probabilities what people say you can't,' *Proceedings of the Second Conference on Artificial Intelligence Applications*, Washington DC: IEEE Computer Press, 6–12, 1985.

94. D. V. Lindley, 'Scoring rules and the inevitability of probability,' *International Statistical Review*, **50**, 1–26, 1986.

1.2 Rule-based systems: principles and problems

95. R. Davis and J. J. King, 'The origins of rule-based systems in AI,' in *Rule-based Expert Systems: the MYCIN projects of the Stanford Heuristic Programming Project*, B. G. Buchanan and E. H. Shortliffe, Reading, MA: Addison-Wesley, 20–54, 1984.

96. E. L. Post, 'Formal reductions of the general combinatorial decision problem,' *American Journal of Mathematics*, **65**, 197–268, 1943.

97. A. A. Markov, *Theory of Algorithms*, National Academy of Sciences, USSR, 1954.

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

99. A. Newell and H. A. Simon, 'An example of human chess play in the light of chess playing programs,' in *Progess in Biocybernetics*, N. Wiener and J. P. Schade (editors), Amsterdam: Elsevier Press, 1965.

100. A. Newell, 'On the analysis of human problem solving protocols,' in *Calcul et Formalisation dans les Sciences de l'Homme*, J. C. Gardin and B. Jaulin (editors), Paris: CNRS, 146–185, 1968.

101. A. Newell, 'A theoretical exploration of mechanisms for coding the stimulus,' in *Coding Processes in Human Memory*, A. W. Melton and E. Martin (editors), Washington, DC: Winston Press, 373–434, 1972.

102. A. Newell, 'Production systems: models of control structures,' in *Visual Information Processing*, W. G. Chase (editor), New York: Academic Press, 463–526, 1973.

103. P. W. Thorndyke, 'Pattern-directed processing of knowledge from texts,' in *Pattern-Directed Inference Systems*, D. A. Waterman and F. Hayes-Roth (editors), Orlando, FL: Academic Press, 347–360, 1978.

104. J. R. Anderson, *The Architecture of Cognition*, Cambridge, MA: Harvard University Press, 1983.

105. J. R. Anderson, *Language, memory and thought*, Hillsdale, NJ: Lawrence Erlbaum and Associates, 1976.

106. N. E. Johnson, 'Mediating represenations in knowledge elicitation,' *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Reading, England, 1987.

107. P. E. Johnson, 'What kind of expert system should a system be?,' *The Journal of Medicine and Philosophy*, **8**, 77–97, 1983.

108. A. Hart, *Knowledge Acquisition for Expert Systems*, New York: McGraw Hill, 1986.

109. M. Welbank, 'A review of knowledge acquisition techniques for expert systems,' *British Telecommunications Laboratories Research Report*, Martlesham Heath, Ipswich, England, 1983.

110. P. Johnson, 'The expert mind: a new challenge for the information scientist,' in *Beyond Productivity*, T. M. A. Bemelmans (editor), Amsterdam: North Holland, 1983.

111. J. Boose, 'Personal construct theory and the transfer of expertise,' *Proceedings of AAAI 1984*, 27–33, 1984.

112. D. Kahneman, P. Slovic and A. Tversky, *Judgement under Uncertainty: Heuristics and Biases*, Cambridge, England: Cambridge University Press, 1982.

113. G. Kahn, S. Nowlan and J. McDermott, 'MORE: an intelligent knowledge acquisition tool,' *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (Los Angeles)*, **1**, Los Altos, CA: Morgan Kaufmann Publishers, 581–584, 1985.

114. S. Muggleton, 'Duce, an oracle–based approach to constructive induction,' *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (Milan, Italy)*, **1**, Los Altos, CA: Morgan Kaufmann Publishers, 287–292, 1987.

115. A. Patterson and T. Niblett, *ACLS User's manual*, Glasgow, Scotland: Intelligent Terminals Ltd., 1983.

116. A. D. Shapiro, *Structured Induction in Expert Systems*, Reading, MA: Addison-Wesley, 1987.

117. C. E. Riese, 'RULEMASTER: Control strategies,' Radian Technical Report RI-RS-00296, Austin, Texas: Radian Corporation, 1985.

2.3 Quantitative modelling of rules

118. B. G. Buchanan and E. H. Shortliffe (editors), *Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Reading MA: Addison-Wesley, 1984.

119. R. O. Duda, P. E. Hart and N. J. Nilsson, 'Subjective Bayesian methods for rule-based inference systems,' Technical note 124, Artificial Intelligence Center, SRI International, Menlo Park, CA, 1976.

2.4 Defining the information content of a rule

120. C. E. Shannon, 'A mathematical theory of communication,' *Bell System Technical Journal*, **27**, 379–423, 1948.

121. N. M. Blachman, 'The amount of information that y gives about X,' *IEEE Transactions on Information Theory*, **IT–14** (1), 27–31, 1968.

122. R. M. Fano, *Transmission of Information*, Cambridge, MA: MIT press, 1961.

123. R. J. McEliece, *The Theory of Information and Coding*, Reading, MA: Addison Wesley, 1977.

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

125. S. Kullback, *Information Theory and Statistics*, New York: Wiley, 1959.

126. R. E. Blahut, *Principles and Practice of Information Theory*, Addison-Wesley : Reading, MA, 1987.

2.5 The J-measure and its application to induction

127. D. Angluin and C. Smith, 'Inductive inference: theory and methods,' *ACM Computing Surveys*, **15** (3), 237-270, 1984.

128. B. R. Gaines, 'Behaviour/structure transformations under uncertainty,' *International Journal of Man-Machine Studies*, **8**, 337-365, 1976.

129. R. S. Michalski, 'Pattern recognition as rule-guided inference,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI–2**, 349–361, 1980.

130. J. H. Holland, K. J. Holyoak, R. E. Nisbett, P. R. Thagard, *Induction: Processes of Inference, Learning and Discovery*, Cambridge, MA : MIT Press, 1986.

131. T. Dietterich and R. Michalski, 'A comparative review of selected methods for learning from examples,' *Machine Learning: an Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (editors), Palo Alto, CA: Tioga Publishing Company, 1983.

132. L. J. Savage, *The Foundations of Statistics*, New York: Wiley, 1954.

133. R. Carnap, *Logical Foundations of Probability*, Chicago: University of Chicago Press, 1950.

134. I. J. Good, *The estimation of probabilities: an essay on modern Bayesian methods*, Research monograph no. 30, Cambridge, MA: M.I.T. Press, 1965.

2.6 ITRULE: an algorithm for generalised rule induction

135. C. S. Peirce, *Collected Papers*, C. Hartshorne, P. Weiss and A. Burks (editors), Cambridge, MA: Harvard University Press, 1931–1958.

136. P. R. Cohen and E. A. Feigenbaum, *The Handbook of Artificial Intelligence (Volume 3)* William Kaufmann Inc., 1982.

137. T. M. Mitchell, 'Generalization as search,' *Artificial Intelligence*, **18**, 203-226.

138. R. S. Michalski and J. B. Larson, 'Selection of most representative training examples and incremental generation of VL1 hypotheses,' Report No. 867, Computer Science Department, University of Illinois, 1978.

139. R. S. Michalski and R. L. Chilausky, 'Learning by being told and learning from examples,' *International Journal of Policy Analysis and Information Systems* , 4, 125–161, 1980.

140. D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning internal representations by error propagation,' in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, D. E. Rumelhart and J. J. McClelland (editors), Cambridge, MA: The MIT press, 318–363, 1986.

141. T. J. Sejnowski, P. K. Kienker and G. E. Hinton, 'Learning symmetry groups with hidden units: beyond the perceptron,' *Physica*, **22D** (1986), 260–275, 1986.

142. B. R. Gaines and M. L. G. Shaw, 'Induction of inference rules for expert systems,' *Fuzzy Sets and Systems*, **18** (3), 315–328, 1986.

143. W. Feller, *An Introduction to Probability Theory and its Applications*, Volume 1, New York: Wiley, 1968.

144. E. M. Gold, 'Language identification in the limit,' *Information and Control*, **10**, 447–474, 1967.

145. L. G. Valiant, 'A theory of the learnable,' *Communications of the ACM*, **27** (11), 1134–42, 1984.

146. D. Haussler, 'Bias, version spaces and Valiant's learning framework,' *Proceedings of the Fourth International Workshop on Machine Learning*, Los Altos: CA, Morgan Kaufmann, 324–336, 1987.

147. M. Kearns, M. Li, L. Pitt and L. G. Valiant, 'Recent results on Boolean concept learning,' *Proceedings of the Fourth International Workshop on Machine Learning*, Los Altos: CA, Morgan Kaufmann, 337–352, 1987.

148. The American Association of Indivdual Investors, *The Individual Investor's Guide to No-load Mutual Funds*, International Publishing Corporation: Chicago, 1987.

149. A. K. C. Wong and D. K. Y. Chiu, 'Synthesizing statistical knowledge from incomplete mixed-mode data,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI–9** (6), pp.796–805, 1987.

150. R. M. F. Goodman and P. Smyth, 'Information-theoretic rule induction,' *Proceedings of the 1988 European Conference on Artificial Intelligence*, in press, Munich, West Germany, 1988.

2.7 Probabilistic inference: theoretical foundations and practical techniques

151. H. E. Stephanou and A. P. Sage, 'Perspectives on Information Processing,' *IEEE Transactions on Systems, Man and Cybernetics*, **SMC–17** (5), 780–798, 1987.

152. P. Szolovits and S. G. Pauker, 'Categorical and probabilistic reasoning in medical diagnosis,' *Artificial Intelligence*, **11**, 115–144, 1978.

153. J. B. Adams, 'Probabilistic reasoning and certainty factors,' *Mathematical Biosciences*, **32**, 177–186, 1976.

154. J. Pearl, 'Fusion, propagation and structuring in belief networks,' *Artificial Intelligence*, **29**, 241–288, 1986.

155. J. Pearl, 'Distributed revision of composite beliefs,' *Artificial Intelligence*, **33**, 173–215, 1987.

156. M. L. Ginsberg, 'Does probability have a place in non-monotonic reasoning,' *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (Los Angeles)*, **1**, Los Altos, CA: Morgan Kaufmann Publishers, 107–111, 1985.

157. H. Y. Kyburg, 'Bayesian and non-Bayesian evidential updating,' *Artificial Intelligence*, **31**, 271–293, 1987.

158. P. Snow, 'Tatting inference nets with Bayes' theorem,' *Proceedings of the Second Conference on Artificial Intelligence Applications*, Washington DC: IEEE Computer Press, 635–640, 1985.

159. Y. Cheng and R. G. Kashyap, 'Irrelevancy of evidence caused by independence assumptions,' *Technical Report TR–EE 86–17*, Purdue University IN, also presented at the IEEE International Symposium on Information Theory, Ann Arbor: MI, October 1986.

160. C. L. Winter and R. D. Girse, 'Evolution and modification of probability in knowledge bases,' *Proceedings of the Second Conference on Artificial Intelligence Applications*, Washington DC: IEEE Computer Press, 27–31, 1985.

161. E. P. D. Pednault, S. W. Zucker and L. V. Muresan, 'On the independence assumption underlying subjective Bayesian updating,' *Artificial Intelligence*, **16**, 213–222, 1981.

162. J. Pearl, 'Reverend Bayes on inference engines: a distributed hierarchical approach,' *Proceedings of the National Conference on Artificial Intelligence*, 133–136, 1982.

163. E. Charniak, 'The Bayesian basis of common sense medical diagnosis,' *Proceedings of the National Conference on Artificial Intelligence (Washington, DC)*, 70–73, 1983.

164. D. J. Spiegelhalter, 'A statistical view of uncertainty in expert systems,' in *Artificial Intelligence and Statistics*, W. Gale (editor), Reading, MA: Addison Wesley, 17–55, 1985.

165. B. P. Wise and M. Henrion, 'A framework for comparing uncertain inference systems to probability,' *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer (editors), New York: Elsevier Science Publishers B.V., 69–84, 1986.

166. E. Horvitz and D. Heckerman, 'The inconsistent use of probabilities in artificial intelligence research,' *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer (editors), New York: Elsevier Science Publishers B.V., 137–152, 1986.

167. R. W. Johnson, 'Independence and Bayesian updating methods,' *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer (editors), New York: Elsevier Science Publishers B.V., 197–202, 1986.

168. J. Pearl, 'A constraint-propagation approach to probabilistic reasoning,' *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer (editors), New York: Elsevier Science Publishers B.V., 357–370, 1986.

169. B. Julesz, 'Visual pattern discrimination,' *IRE Transactions on Information Theory*, **IT–18**, 84–92, 1962.

170. M. Minsky and S. Papert, *Perceptrons*, Cambridge: MIT press, 1969.

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

172. K. Konolige, 'An information-thoeretic approach to subjective Bayesian inference in rule-based systems,' Artificial Intelligence Center, Stanford Research Institute, Menlo Park: CA, 1982.

173. J. F. Lemmer and S. W. Barth, 'Efficient minimum information updating for Bayesian inferencing in expert systems,' *Proceedings of the National Conference on Artificial Intelligence (Pittsburgh)*, 424–427, 1982.

174. L. Shastri and J. A. Feldman, 'Evidential reasoning in semantic networks: a formal theory,' *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (Los Angeles)*, **1**, Los Altos, CA: Morgan Kaufmann Publishers, 465–474, 1985.

175. D. Hunter, 'Uncertain reasoning using maximum entropy inference,' *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer (editors), New York: Elsevier Science Publishers B.V., 203–210, 1986.

176. J. E. Shore, 'Relative entropy, probabilistic inference and AI,' *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer (editors), New York: Elsevier Science Publishers B. V. , 211–216, 1986.

177. E. T. Jaynes, 'Information theory and statistical mechanics I,' *Physical Review*, **106**, 620–630, 1957.

178. J. R. Quinlan, 'Inferno: a cautious aproach to uncertain inference,' *Computer Journal*, **26**, 255–266, 1983.

2.8 Controlling the reasoning using decision theory

179. J. Von Neumann and O. Morgenstern, *Theory of Games and Economic Behaviour*, Princeton, NJ: Princeton University Press, 1947.

180. J. Marschak, 'Rational behaviour, uncertain prospects, and measurable utility,' *Econometrica*, **18**, 111–141, 1950.

181. A. Wald, *Statistical Decision Functions*, New York: John Wiley and Sons, 1950.

182. L. J. Savage, *The Foundations of Statistics*, New York: John Wiley and Sons, 1954.

183. D. Blackwell and M. A. Girschick, *Theory of Games and Statistical Decisions*, New York: John Wiley and Sons, 1954.

184. H. Chernoff and L. E. Moses, *Elementary Decision Theory*, New York: John Wiley and Sons, 1959.

185. P. C. Fishburn, 'Utility theory,' *Management Science*, **14**, 335–378, 1968.

186. M. H. DeGroot, *Optimal Statistical Decisions*, New York: McGraw Hill, 1970.

187. R. D. Luce and D. H. Krantz, 'Conditional expected utility,' *Econometrica*, **39**, 253–271.

188. D. Von Winterfeldt and W. Edwards, *Decision Analysis and Behavioral Research*, Cambridge: Cambridge University Press, 1986.

189. D. Bernoulli, 'Specimen theoriae novae de mensura sortis,' *Comentarii Academiae Scientiarum Imperiales Petropolitanae*, **5**, 175–192, 1738.

190. M. Allais, 'Le comportement de l'homme rationnel devant le risque: Critique des postulats et axiomes de l'ecole americaine,' *Econometrica*, **21**, 503–546, 1953.

191. D. Ellsberg, 'Risk, ambiguity and the Savage axioms,' *Quarterly Journal of Economics*, **75**, 643–669, 1961.

192. D. Kahneman and A. Tversky, 'Prospect theory: An analysis of decision under risk,' *Econometrica*, **47**, 263–291, 1979.

193. H. Raiffa and R. Schlaifer, *Applied Statistical Decision Theory*, Division of Research, Graduate School of Business Administration, Harvard University, 1961.

194. M. Henrion and D. R. Cooley, 'An experimental comparison of knowledge engineering for expert systems and decision analysis,' *Proceedings of the AAAI Conference 1987*, Seattle, 471–476, 1987.

195. D. B. Lenat and G. Harris, 'Designing a rule system that searches for scientific discoveries,' in *Pattern-Directed Inference Systems*, D. A. Waterman and F. Hayes-Roth (editors), Orlando, FL: Academic Press, 25–52, 1978.

196. R. Davis, 'Meta-rules: reasoning about control,' *Artificial Intelligence*, **15**, 179–222, 1980.

197. D. Lenat, R. Davis, J. Doyle, M. Genesereth, I. Goldstein, H. Schrobe, 'Reasoning about reasoning,' in *Building Expert Systems*, F. Hayes-Roth, D. Waterman and D. B. Lenat (editors), Reading, MA: Addison-Wesley Inc. , 219–240, 1983.

198. J. McDermott and C. Forgy, 'Production system conflict resolution strategies,' in *Pattern-Directed Inference Systems*, D. A. Waterman and F. Hayes-Roth (editors), Orlando, FL: Academic Press, 177–202, 1978.

199. F. Hayes-Roth, D. Waterman and D. B. Lenat, 'Principles of pattern-directed inference systems,' in *Pattern-Directed Inference Systems*, D. A. Waterman and F. Hayes-Roth (editors), Orlando, FL: Academic Press, 577–602, 1978.

200. L. Friedman, 'Controlling production firing: the FCL language,' *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, 359–366, 1985.

201. R. O. Duda, P. E. Hart, N. J. Nilsson and G. L. Sutherland, 'Semantic network representations in rule-based inference systems,' in *Pattern-Directed Inference Systems*, D. A. Waterman and F. Hayes-Roth (editors), Orlando, FL: Academic Press, 203–222, 1978.

202. M. L. Ginsberg, 'Implementing probabilistic reasoning,' in *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer (editors), New York: Elsevier Science Publishers B.V., 331–338, 1986.

203. T. E. Marques, 'A symptom-driven expert system for isolating and correcting network faults,' *IEEE Communications magazine*, **26** (3), 6–13, 1988.

204. D. P. Benjamin, 'Learning strategies by reasoning about rules,' *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, 256–259, 1987.

2.9 Combining inference and control: implementation issues and problems

205. R. P. Loui, 'Interval-based decisions for reasoning systems,' in *Uncertainty in Artificial Intelligence*, L. N. Kanal and J. F. Lemmer (editors), New York: Elsevier Science Publishers B.V., 459–472, 1986.

206. J. R. Quinlan, 'Consistency and plausible reasoning,' *Proceedings of the Eigth International Joint Conference on Artificial Intelligence (Karlsruhe, West Germany)*, 137–144, 1983.

2.10 Future directions and current conclusions

207. E. B. Baum and F. Wilcek, 'Supervised learning of probability distributions by neural networks,' presented at the IEEE Conference on Neural Information Processing Systems, Denver, Colorado, 1987.

208. S. I. Gallant, 'Connectionist expert systems,' *Communications of the ACM*, **31** (2), 152-169, 1988.

Appendices

209. H. Cramér, *Mathematical Methods of Statistics*, Princeton, NJ: Princeton University Press, 1945.