Knowledge Acquisition (1990) **2**, 1–19

Decision tree design using information theory

RODNEY M. GOODMAN AND PADHRAIC SMYTH

Department of Electrical Engineering 116-81, California Institute of Technology, Pasadena CA 91125, USA

(Based on a paper presented at the AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, November 1988)

In this paper we examine the greedy mutual information algorithm for decision tree design, analysing both theoretical and practical applications (namely, edge detection). We review our earlier theoretical results on tree design algorithms such as ID3, which confirm that the greedy mutual information heuristic is well-founded. The theoretical models based on rate-distortion theory and prefix-coding analogies explain previously observed experimental phenomena reported in the literature. An application to edge detection is described where we primarily emphasize the inductive methodology rather than the domain application (image processing) *per se*. We conclude that inductive learning paradigms based on information-theoretic models are both theoretically well-behaved and useful in practical problems.

1. Introduction

This paper develops and applies ideas from information theory for use in learning algorithms. We concentrate on the area of decision trees and review our recent theoretical work on the mutual information decision tree design algorithm. A new application of this algorithm is reported, where we formulated edge detection for image processing as a classification problem. We show how the algorithm learned to detect edges and yielded new domain-specific information about the problem.

1.1. BACKGROUND

Rule induction from large data sets is currently receiving attention in the areas of machine learning and expert systems. Classifier design from labelled training samples is a problem which shares many characteristics with the rule induction problem. A recent paper by Bundy, Silver & Plummer (1985) provides a useful discussion of how the two problems relate to each other.

The basic premise of many rule induction mechanisms, when the data is probabilistic rather than deterministic, is to induce a hierarchy or decision tree as a representation of the relationships between the attributes (evidence) and the classes (hypotheses). Hence general relationships between classes and attributes are induced or learned by the induction mechanism. Note that the terms *attributes* and *classes* occur more often in pattern recognition literature than the terms *evidence* and *hypotheses* which tend to be used in the artificial intelligence domain. For the purposes of this paper we adopt the former.

When the attribute-class relationships are probabilistic rather than deterministic, the induction mechanisms which work best appear to be those based on statistical

This work was supported in part by Pacific Bell.

1042-8143/90/010001 + 19$03.00/0

and/or information theoretic techniques, e.g. the ID3 algorithm (Quinlan, 1983) based on information theory, the CART system (Breiman, Friedman, Olshen & Stone, 1984) for medical diagnosis, or PREDICTOR (White, 1985), a rudimentary inference tool, the latter two based on statistical techniques. Perhaps the best known inference mechanism of this type is the ID3 algorithm since it has already been implemented as part of an expert system design tool (Michie, Muggleton, Riese & Zubrick, 1984) which in turn has been used to design systems for weather forecasting (Riese, 1984) and fault diagnosis (Stuart, Pardue, Carr & Feldcamp, 1984). Many more recent applications are reported by Michie (1987).

1.2. OUTLINE OF THE PAPER

In this paper we concentrate on the average mutual information heuristic for rule induction from large data sets, of which the ID3 algorithm is an example. After defining the general mutual information induction algorithm we briefly review our earlier work in this field. Primarily we have defined a general information theoretic model for this induction problem. From this model we develop several general principles. The model enables us to understand why algorithms like ID3 work so well in practice and from it we can confirm conjectures such as those of Quinlan (1986) and others. In the second part of the paper we apply the algorithm to the problem of edge detection in images. The resulting induced edge detectors are both computationally faster than 'manually' designed operators and, perhaps more significantly, yield new knowledge for this problem domain. The overall conclusion reached from both theoretical and practical considerations is that the mutual information induction mechanism is indeed a powerful tool and merits further research and applications.

2. The mutual information decision tree algorithm

The algorithm we are about to describe is a general algorithm which derives a decision tree from a large data set of labelled samples using the average mutual information function in a 'greedy' fashion to find the most informative attributes relative to the class. Instances of this algorithm may vary considerably depending on the termination rules used, where a termination rule is essentially a mechanism for deciding that no more attributes are relevant or need not be tested. Hence Quinlan's ID3 algorithm with the Chi-square termination test (Quinlan, 1986) is a specific case, as are several others in the pattern recognition literature (Sethi & Sarvarayudu (1982), Casey & Nagy (1984), and Wang & Suen (1984)). Recent work by Smyth (1988) and Chou (1988) provide more extensive bibliographies on such 'top-down' tree design algorithms. It is worth noting that there are extensions to 'top-down' algorithms, such as the pruning algorithms of Breiman *et al.* (1984) and Chou, Lookabaugh & Gray (1988), which have recently received attention. In addition to being computationally costly, these pruning algorithms begin by growing an initial tree, typically using the greedy algorithm we are about to describe. Hence the performance of this algorithm is fundamental to the quality of most existing tree design techniques.

Consider that one is given a table of data as in Fig. 1. The table consists of M samples drawn from the population at large. Each sample is described in terms of N

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

FIG. 1. Typical table of sample data.

attribute values and a class label. 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-attribute conditional probability distributions.

If all known attributes are given then the algorithm will inherently select the most relevant attributes and ignore the irrelevant ones. This is an additional advantage of the mutual information approach over conventional classification techniques since it yields valuable information on the relative importance of the various attributes. In applications such as medical diagnosis this can be quite useful (Breiman *et al.*, 1984). We assume that the sample size is sufficiently large to yield reliable estimates of the class distribution conditioned on the values of the N attributes.

$\mathbf{C}$ is defined as the class random variable with a discrete alphabet of K components such that $p(\mathbf{C} = c_i) = p_i$, $\sum_{i=1}^{K} p_i = 1$, $1 \leq i \leq K$ where c_i is the ith class. Then $H(\mathbf{C}) = \sum_{i=1}^{K} p_i \log 1/p_i$ is the entropy of $\mathbf{C}$.

There are N attributes each denoted by $\mathbf{A}_i$, $1 \leq i \leq N$. In turn, each attribute $\mathbf{A}_i$ has n_i values which it can assume; n_i is finite, i.e. the attribute values are quantised. $I(\mathbf{C}; \mathbf{A}_i)$ is the mutual information (as defined by Shannon) between $\mathbf{C}$ and $\mathbf{A}_i$. In practice the probabilities are estimated from the data using standard statistical estimation techniques.

Now let us consider the algorithm itself. Essentially it just deals with one tree-node at a time. The initial node (root node) consists of the original table of data samples, i.e., unconditioned on any attribute values. Susbequent nodes result from evaluating certain attributes and obtaining sub-tables conditioned on the outcome of all prior evaluations. The algorithm continues to process nodes until there remain no candidate nodes, 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 a leaf. The algorithm is defined in pseudocode in Fig. 2.

```
tree( )
  begin
  if(node_list is not empty)
    {
      node = node_list[topnode]
      if (node is a leaf)
          {
            increment leaf_list
          tree( ) }
      else
        {
              A = maximum information attribute
              for (each value of A) {
                  sort data table
                  increment node_list}
    }
  }
  else return( )
end
```

FIG. 2. Pseudo-code description of the tree classifier design algorithm.

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 characterize any 'top-down' tree derivation algorithm, i.e.,

(i) determine if the node is a leaf

(ii) if it is not a leaf then determine the feature or attribute which yields the most information at that node, i.e., choose $\mathbf{A_k}$ 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,

$$H(\mathbf{C} \mid \mathbf{A_k}) \leq H(\mathbf{C} \mid \mathbf{A_i}) \qquad 1 \leq i \leq N, \quad i \neq k.$$

In the next section we summarise our recent work on this algorithm.

3. Theoretical results for the algorithm

Prior work on this algorithm has been applications motivated, as for example the various applications reported in the field of optical character recognition (Casey & Nagy, 1984; Wang & Suen, 1984). The work of Quinlan (1983, 1986) has suggested that the algorithm can be used in the more general domain of rule induction from examples. Hence our work in this field (Goodman & Smyth 1986; 1988*b*) has been directed towards a better understanding of the algorithm with a view to applications in rule induction for expert systems. In particular we examined such issues as average path length/misclassification rate trade-offs, the effect of increasing the noise on the termination rules and the overall effectiveness of the mutual information heuristic. In this section we present a brief summary of these results.

We begin by modelling the problem in a communication theory setting. This communication theory model was recently suggested by both ourselves and Chou & Gray (1986) who proposed an equivalent rate-distortion model. Essentially the classification problem is equivalent to a communications problem where one is communicating through a noisy channel at a variable rate. The noisy channel

represents the overlap in the class-attribute conditional probability densities, and is equivalent to the noise which Quinlan discussed (1986) (note that this noise is due to the inherent ambiguities imposed by a particular set of features in a given problem). The general idea is to transmit the most informative bits (or attributes) to give the decoder (decision tree scheme) the best chance of decoding the message (class). Hence we can immediately grasp the fundamental distortion-rate trade-off, more bits meaning less distortion.

This trade-off is formalised in communication theory by the rate-distortion curve for a given noisy channel which provides a fundamental lower bound on the achievable distortion for a given rate. By modelling the induction problem in this setting the rate becomes the average tree depth, $\bar{d}$, while the distortion is the average probability of misclassification, P_e, in using the tree. Figure 3 shows a typical distortion-rate curve. The horizontal asymptote represents the minimum achievable misclassification rate, P_{min}, for a given set of features and a given classification problem. In pattern recognition literature this is termed the Bayes' rate and is a result of the inherent noise or overlap between the class-attribute conditional probability densities. Hence the distortion-rate curve provides a lower bound for what might be termed the 'operating point' of the tree classifier. A good algorithm will induce classification rules which have lower distortion, P_e, as the rate, $\bar{d}$, increases. We have shown in previous work that, for certain termination rules, P_e is indeed bounded above by a function which decreases as $\bar{d}$ increases (Goodman & Smyth, 1988*b*).

We can also use the distortion-rate model to confirm and explain an earlier conjecture by Quinlan (1986), namely that it is better to use noisy data than 'non-noisy' good data in the design phase, if the induced classification rules are to be used on noisy data. Figure 4 shows clearly that this is true. Consider that we wish to design a tree to achieve some specified P_e. Using 'non-noisy' data is like designing

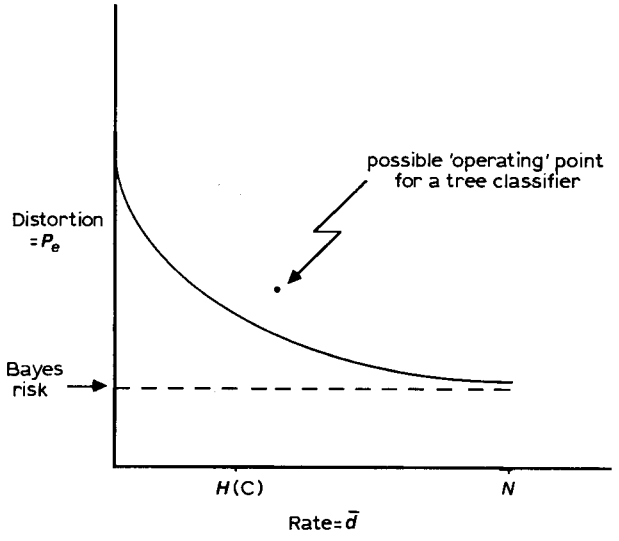


FIG. 3. General distortion-rate characteristic.

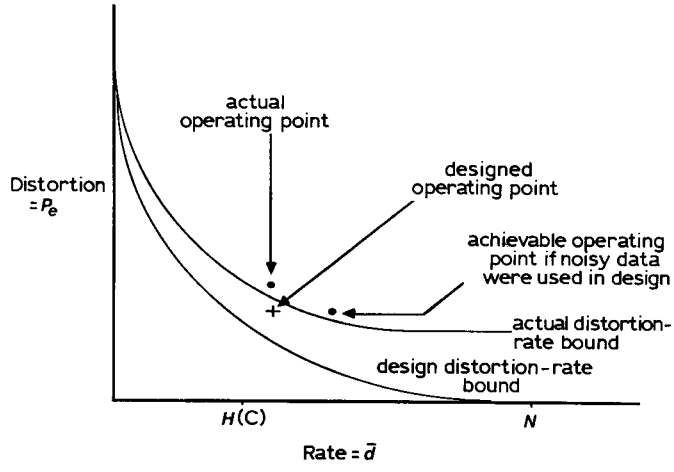


FIG. 4. How 'good' data can lead to the wrong result.

the classifier with an artificially low distortion-rate bound; $\bar{d}$ is fixed at an artificially low value of P_e. In using the tree P_e will be larger, perhaps much larger, than the specified P_e. However if noisy data had been used, the designed value of P_e and the true value would be the same, with perhaps only a slight increase in $\bar{d}$.

Continuing the communications analogy we have also shown that the top-down mutual information algorithm is equivalent in a certain sense to Shannon–Fano prefix-coding (Fano, 1961). Choosing the attribute which yields the most information is equivalent to choosing the attribute which most closely partitions the classes into subsets of equal probability. The Shannon-Fano prefix-coding scheme is known to be near-optimal compared to the optimal scheme by Huffman (1952) but has computational advantages. This equivalence to prefix-coding is a very positive result with respect to the mutual information induction heuristic. We have established upper and lower bounds on $\bar{d}$ (in terms of the entropy of the class distribution $H(\mathbf{C})$ and the noise level) using the coding ideas. The tightness of the bounds obtained (Goodman & Smyth, 1988*b*) are further proof that the algorithm is, on average, near-optimal. For example with no noise, and with some constraints on the class probability distribution, we get

$$H(\mathbf{C}) \leq \bar{d} \leq 1.09H(\mathbf{C})$$

The rate-distortion model and the prefix coding equivalence provided the basis for some new results regarding termination rules. From Fig. 3 we see that the form of the termination rule will critically influence the position of the 'operating point' of the classifier.

We have also shown (Goodman & Smyth, 1988*b*) that the information gained at any node is simply the entropy of the branch probabilities less a noise term. The size of the noise term limits the available information. Prior work in this area had indicated that threshold-type termination rules did not perform well in practice (Breiman *et al.*, 1984; Sethi & Sarvarayudu, 1982). In terms of our model it is clear why this happens due to the basic phenomenon of limited information is the presence of noise. Termination rules based on statistical tests such as the Chi-square test (Quinlan, 1986) are an improvement but may not be sufficient due to limitations

on the sample size, i.e., the rule is most critically needed at nodes deep in the tree—but at such nodes the number of samples may be very small, precluding the use of the Chi-square test. Consequently we have proposed a new termination rule called the Delta-entropy rule which basically compares the normalised entropies of the class distribution at a node, with and without p_{max}, and decides to terminate or not based on whichever is larger. The rule has been found to work well in practice and is robust in the presence of noise.

In summary for this section, the main findings of our earlier work has been:

(1) to confirm that the mutual information induction algorithms (such as ID3) are indeed near optimal;

(2) provide quantitative bounds on the performance of such algorithms; and

(3) gain a general understanding of such issues as termination rules and choice of design data, and consequently develop a new robust termination procedure.

Having gained considerable *theoretical* insight into the performance of the mutual information mechanism we turned our attention to a *practical* application. The remainder of this paper reports on the results of our practical work.

4. A practical application: edge detection

4.1. MOTIVATION AND BACKGROUND

Edge detection remains one of the fundamental issues in the field of computer vision. For the purposes of this paper 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. Recent advances in edge detection theory (Canny, 1983; Marr & Hildreth, 1980) 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 not realisable for many applications such as real-time automated inspection.

Instead the smaller window size operators such as the Sobel (Duda & Hart, 1973) continue to be used in the applications environment (Waxman *et al.*, 1985; Wahl, 1987). However even for small size windows (e.g. 3×3) the bottleneck remains that two-dimensional convolution is computationally slow. There is considerable motivation then to investigate any techniques which can reduce the fundamental computation requirements in edge detection (Nitzan, 1985).

4.2. OUTLINE OF THE APPROACH

We introduce here a new approach (from a rule induction viewpoint) to edge filtering with a resulting significant decrease in computation cost. Essentially we formulate the edge detection problem 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.

After defining a general procedure for designing hierarchical edge operators we proceed to derive such operators on particular training images and evaluate their performance on these and other images. For comparison purposes we use two other operators, namely, the afore-mentioned Sobel operator, and the dispersion operator based on order statistics, recently introduced by Pitas & Venetsanopoulos (1986). The reasons for choosing precisely these operators are explained later but it is sufficient to note that of the local (3×3 window) edge detectors the Sobel is adjudged to perform best (Abdou & Pratt, 1979) while the dispersion operator is in principle the fastest. The next section gives a brief description of these operators. It is important to note that the 3×3 size window was chosen for convenience only and the method outlined is perfectly general for any arbitrary window size.

4.3. THE SOBEL AND DISPERSION EDGE OPERATORS

For evaluation purposes we have chosen to use the Sobel operator and the less well known dispersion operator to compare with the hierarchical approach. The pixel notation is given in Fig. 5 and the two masks for the Sobel operator are defined in Fig. 6.

The procedure is to obtain two 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 $\mathrm{EI}(i, j)$ to be the grey-scale value of the enhanced image at pixel (i, j) then,

$$\mathrm{EI}_1(i, j) = \left| \sum_{k=1}^{3} (x_k - x_{k+6}) \right|$$

$$\mathrm{EI}_2(i, j) = \left| \sum_{k=1}^{3} (x_{3k-2} - x_{3k}) \right|$$

(where x_k is the grey-scale value of the kth pixel (Fig. 5), so that

$$\mathrm{EI}(i, j) = \mathrm{EI}_1(i, j) + \mathrm{EI}_2(i, j).$$

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

$$\mathrm{EI}(i, j) > t$$

otherwise it is classified as a non-edge.

1	2	3
4	5	6
7	8	9

FIG. 5. Pixel notation.

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

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

FIG. 6. The vertical and horizontal Sobel edge masks.

The dispersion operator of Pitas & Venetsanopoulos (1986) has the advantage of being in principle computationally faster than any other known 3×3 operator. It is based on order statistics. To obtain the order statistics it is first necessary to sort the N pixels in the window in order of increasing magnitude so that

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

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

$$w_{(1)} = x_{(N)} - x_{(1)}$$

Quasi-ranges are defined by

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

One can define the range and quasi-ranges as primitive operators; they 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. The dispersion operator was defined as an average over the quasi-ranges to combat the effect of noise, i.e.

$$W = \sum_{i=1}^{\lfloor N/2 \rfloor} w_i$$

The W value for each pixel (i, j) is thresholded to determine if or not it contains an edge. It can be shown (Pitas & Venetsanopoulos, 1986) that this operator performs almost as well as the Sobel operator over a variety of comparison measures and can be implemented much faster.

4.4. EDGE DETECTION AS A CLASSIFICATION 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 attributes. 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 attributes, e.g., the probability that the class is 'edge' given that the ith attribute has value j. From the set of training samples relating attributes and classes we infer some general rules for classifying pixels based on their attribute description, using the mutual information induction algorithm.

First however the problem of attribute selection must be addressed. 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 attributes we choose must be computationally simple to implement—the term 'primitive' provides a convenient description. Window operators can be considered as 'composite' binary attributes in the sense that they are composed of combinations of primitive attributes. They are binary-valued because the pixels are essentially classified as 'edge' or 'non-edge' according to whether the composite attribute is greater than or less than a certain threshold. What might be the appropriate primitive attributes? We claim that 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 attributes can be defined by supplying pairs of arguments to the difference operator. Consider the standard 3×3 window as shown in Fig. 5 with associated notation. We could for example define the following primitive attributes,

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

where the i ranges over the window as described in Fig 5 and x_i is the grey level at the ith pixel. It is easy to see that the Sobel operator can be defined in terms of primitive attributes like the ones above. We use F_i to represent the ith attribute 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 attributes to choose from (e.g., we are not necessarily constrained to using pixels within a 3×3 window)—which ones should be chosen? As in any classification-type problem attribute selection is almost an art form (however one of the advantages of the mutual information algorithm is to delete irrelevant attributes from consideration, making the attribute selection process easier). For the purposes of this paper we have restricted our attributes to be the primitive operators which make up the Sobel and dispersion operators—this facilitates comparison on both a performance and computational level. It is important to realise however, that throughout the rest

of the discussion, a primitive attribute can be thought of as *any* primitive pairwise operator defined on pixels. We note again that the composite operators or attributes are essentially binary-valued, i.e., greater than or less than some threshold. In computation terms this amounts to a simple comparison. Hence we must also implement the primitive attributes in this 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. There are a variety of schemes (e.g. Abdou & Pratt, 1979), for calculating threshold parameters for edge detectors subject to some optimisation criterion. However we cannot strictly apply any of those techniques for *hierarchical* edge detection, the reasoning being as follows. Consider the conditional probability of the class 'edge' being present given that some binary primitive attribute, F_i, is greater than some threshold, i.e. $p(edge \mid f_i > t)$, where f_i is the random variable denoting the value of the attribute F_i, and t is the threshold. Since the pixels are correlated it is obvious that the various class and attribute probabilities are not independent, i.e. if attribute x has some value then the probability that 'there is an edge given attribute y greater than some other value' must be conditioned on attribute x.

So for example the Bayesian minimum error threshold t_{opt} is chosen to minimise the quantity

$$p(error) = p(edge, f_i \leq t) + p(non\ edge, f_i > t)$$
$$= p(edge \mid f_i \leq t) \times p(f_i \leq t) + p(non\ edge \mid f_i > t) \times p(f_i > t)$$

In a hierarchical classifier the above probabilities must be conditioned on all the previously evaluated attributes and so t_{opt} depends on these attributes. Hence one must calculate all the possible $N(N-1)$ optimal thresholds before designing the classifier (N is the number of attributes). This is not practical given the inherent difficulties in estimating the probabilities in the equation for $p(error)$.

Instead we propose an approach which circumvents the problem by using the tree derivation algorithm. As we have seen the algorithm works by choosing the best attribute (in an information-theoretic sense) conditioned on whatever attributes have already been chosen. At any given node in the tree the best threshold is found for each attribute. This is achieved by use of the Kolmogorov–Smirnov test, which has certain desirable properties for this problem (Friedman, 1977; Smyth, 1988). Given the thresholds, the attribute which then yields the most information about the class variable is selected at that node.

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 combine segments of different images in order to synthesise a training image; since the individual samples are pixels then it is quite easy to obtain a large sample so that the only constraints imposed may be by the computing facilities available to run the classifier design algorithm, e.g., memory constraints. The typical image is then classified via standard edge-detection techniques using the best edge-detection algorithm available and the 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.

4.5. POTENTIAL ADVANTAGES OF THIS APPROACH

We can now begin to appreciate the potential advantages of the hierarchical operator. If for example the typical image contains low edge content, as may be 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 and for those pixels which are not immediately so classified apply other attributes to determine if an edge is present. Since, in machine-vision problems for example, many images have at least 90% of their pixels as 'non-edge,' the computational advantages are obvious.

For the hierarchical operator to be used for comparison purposes we chose to define the primitive attributes 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 Fig. 7. The rationale for choosing the latter four attributes (as defined in Fig. 7) was to include some primitive attributes which had spatial dependence and could indicate edges in various orientations. The attributes chosen may not be the set of optimal primitive attributes but rather are a set of easily computable operations which are easy to interpret. However it is important to remember that any redundant attributes which might be chosen, or equivalently attributes which yield no relevant information, will by definition be suppressed by the tree derivation algorithm. The goal here is not necessarily to derive *optimal* hierarchical operators but rather to demonstrate the feasibility of hierarchical operator design and investigate the resulting performance.

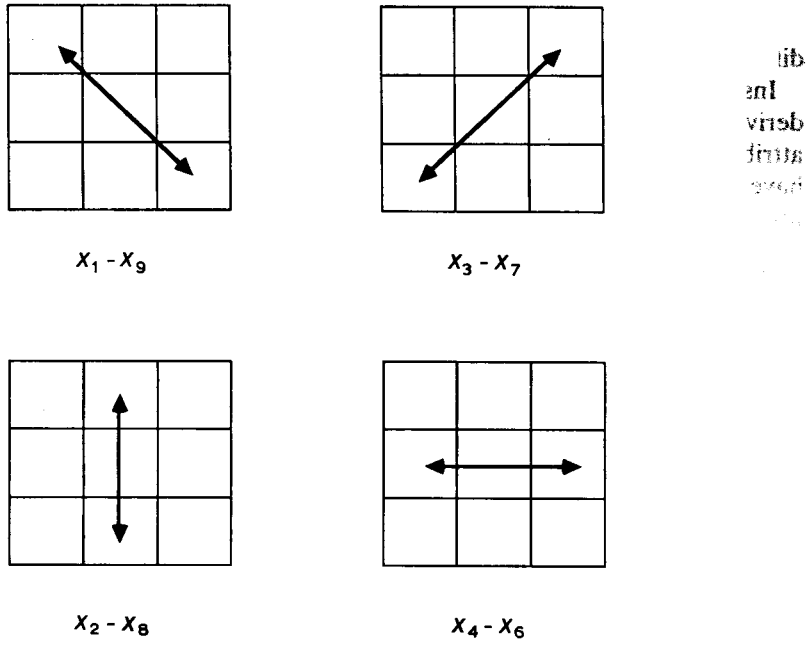


FIG. 7. Spatial primitive features.

Having defined the attributes the next step is to run the algorithm given in Section 2 on the training samples. 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 and;

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

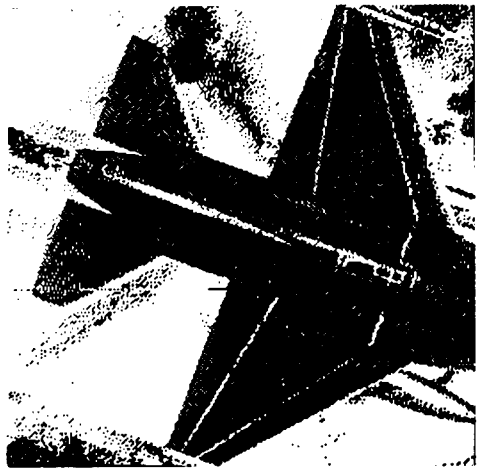
5. Experimental results

In this section we compare the performance of the Sobel, dispersion and hierarchical operators on two different images. The images were generated via a scanning device using 32 grey levels. Each image is 256 pixels by 256 pixels.

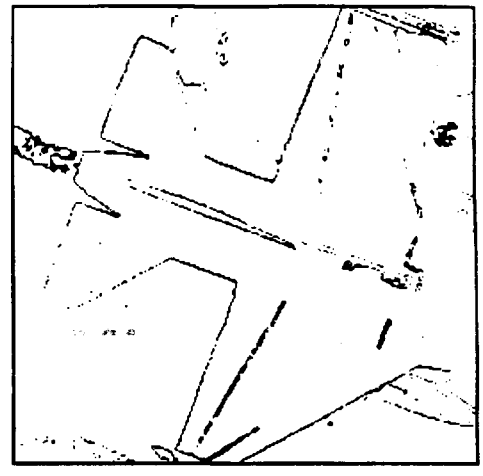
Before considering the results it is worth mentioning some details. The operator used to classify the samples was the Sobel operator since we found that it yielded slightly higher quality edge maps than the dispersion operator. The threshold t for both operators was chosen heuristically based on visual evaluation of the effects on the edge maps of varying t. Because of memory limitations in our software the entire image could not be used for training purposes. Consequently 80 pixel by 80 pixel size 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 the sub-image was located within the overall image. In each of Figs 8 and 9, the 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 Fig. 8(a). The output of the Sobel operator in Fig. 8(c) is clearly better than that of the dispersion operator in Fig. 8(b). The hierarchical output in Fig. 8(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 Fig. 8(d) can be explained by the actual tree structure of the operator as shown in Fig. 10. It uses mainly as primitive components the quasi-range which as we can see in Fig. 8(b) tends to merge edges which are close together (this difference can be seen by considering the effect of passing the two operators over alternating black and white pixel-wide vertical lines). The average number of additions/subtractions performed by the tree is much less that of the other two methods, i.e. only 1·34 as compared to 13 for the Sobel and 7 for the dispersion.

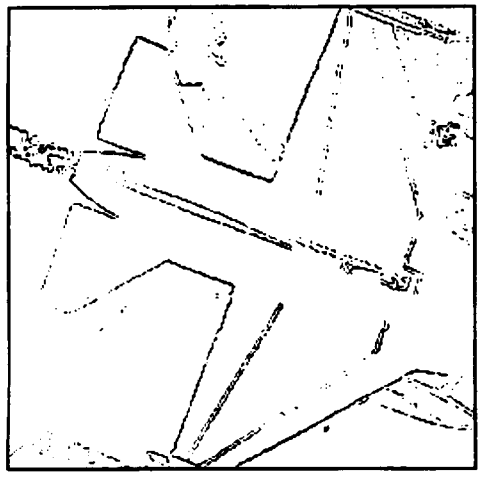
Figure 9 contains equivalent information to Fig. 8 but for another image—a bin of tools. Interestingly enough the hierarchical operator derived for this image was quite similar to that derived from the airplane image. All three operators perform similarly here with perhaps the order of merit being dispersion, 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 hierarchical operator can only resolve edges as well as its primitive attributes will allow—better performance could be achieved by using a higher quality classifier than the Sobel operator and defining more primitive attributes.



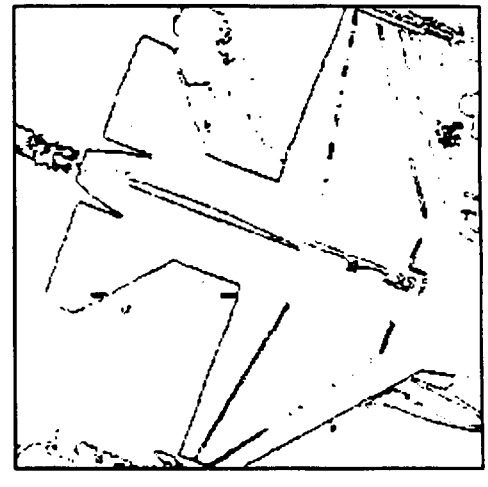
(a) *Half-toned original*



(b) *Dispersion*



(c) *Sobel*



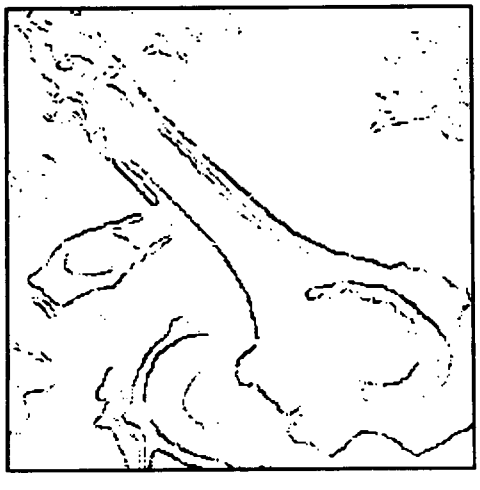
(d) *Hierarchical*

FIG. 8. Edge maps for 'airplane' image.

Now consider the relative *computational* performance of the three operators. The results are tabulated in Table 1. Since the hierarchical operators for both images were 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 eight comparisons for a parallel sorting structure (Knuth, 1973). For sorting in serial form the number in the dispersion column corresponds to the minimum number of comparisons required to sort nine numbers (Knuth, 1973). Because we do not need to sort all the numbers

(a) Half-toned original

(b) Dispersion



(c) Sobel

(d) Hierarchical

FIG. 9. Edge maps for 'bin of tools' image.

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 implementation, the hierarchical operator will always be considerably faster than the dispersion operator particularly for images of low edge content, which is the usual case. 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 in, Pitas &

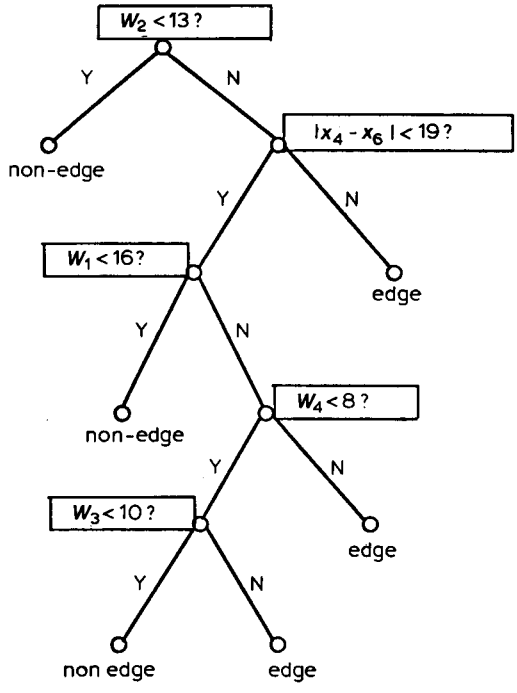


FIG. 10. Hierarchical operator for airplane image.

Venetsanopoulos, 1986) 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 s with the dispersion operator, 23·7 s with the Sobel, and 16·4 s 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.

TABLE 1
Comparison of edge detectors in terms of computation

	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	9
Serial	0	19	15·8

6. Conclusion

The question we addressed in this paper was basically the following: given that induction algorithms which use mutual information (such as ID3) are intuitively appealing, can we then prove their utility both theoretically and practically? (The answer to the latter part of the question has already been proven true to some extent elsewhere—Michie, 1987; Riese, 1984; Wang & Suen, 1984; Sethi & Sarvarayudu, 1982). The results we referred to in the earlier section of this paper certainly provide a sound theoretical basis for the algorithm and justify its use. From a practical viewpoint we have also met with success, i.e., the edge detection experiment. It is important to realise the general implications we can draw from these *practical* results.

(1) The algorithm yields efficient and accurate 'classifiers' or 'rule-hierarchies.'

(2) Attribute 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 attributes, then one can over-specify the number of attributes. Hence, as in this problem, image-processing 'novices' such as ourselves can achieve relatively 'expert' results, i.e., see Figs. 8 and 9.

(3) We noted that in all the hierarchical operators derived by the algorithm (for the two 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 induction algorithm.

We conclude that the mutual information algorithm is well-founded and has practical potential as an induction tool. The immediate goal in our research is to generalise the algorithm for applications in expert systems design. At present the induced tree structure may not be flexible enough in itself for an expert system type of environment. For instance if data is unavailable at run time the hierarchy should be such as to allow for this (White, 1985 also mentions this idea). The existing algorithm can easily accommodate this in principle simply by deleting the appropriate attribute data at each node as if it did not exist and so obtaining an extra 'don't know' branch. However there are practical problems associated with this approach, such as exponential order tree growth, which must be solved. More general approaches such as induction algorithms which derive *sets* of rules or general *graph* structures, based on information-theoretic measures, also appear promising (Goodman & Smyth, 1988*a*; Smyth, 1988).

References

ABDOU, L. E. & PRATT, W. K. (1979). Quantitative design and evaluation of enhancement/thresholding edge detection. *Proceedings IEEE*, **67**(5), 753–763.

BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A. & STONE, C. J. (1984). *Classification and Regression Trees*. Belmont, CA.: Wadsworth.

BUNDY, A., SILVER, B. & PLUMMER, D. (1985). An analytical comparison of some rule-learning programs. *Artificial Intelligence,* **27,** 137–181.

CANNY, J. F. (1983). Finding edges and lines in images. *M.I.T. Artificial Intelligence Lab., Rep. AI-TR-720.*

CASEY, R. G. & NAGY, G. (1984). Decision tree design using a probabilistic model. *IEEE Transactions on Information Theory,* **30,** 93–99.

CHOU, P. A. (1988). *Applications of Information Theory to Pattern Recognition and the Design of Decision Trees and Trellises,* Department of Electrical Engineering, Stanford University.

CHOU, P. A. & GRAY, R. M. (1986). *On Decision Trees for Pattern Recognition, Presented at the* 1986 *IEEE International Symposium on Information Theory,* Ann Arbor, Michigan.

CHOU, P. A., LOOKABAUGH, T. & GRAY, R. M. (1989). Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Transactions on Information Theory* **35,** 299–315.

DUDA, R. O. & HART, P. E. (1973). *Pattern Classification and Scene Analysis,* p. 271. New York: Wiley.

FANO, R. M. (1961). *Transmission of Information,* p. 187. Cambridge, MA.: MIT Press.

FRIEDMAN, J. H. (1977). A recursive partitioning decision rule for non-parametric classification. *IEEE Trans. on Computers,* **26,** 404–408.

GOODMAN, R. M. & SMYTH, P. (1986). *An Information Theoretic Approach to Decision Tree Design, Presented at the* 1986 *IEEE, International Symposium of Information Theory.* Ann Arbor, Michigan.

GOODMAN, R. M. & SMYTH, P. (1988*a*). Information-theoretic rule induction. In *Proceedings of the* 1988 *European Conference on Artificial Intelligence.* Pitman Publishing.

GOODMAN, R. M. & SMYTH, P. (1988*b*). Decision tree design from a communication theory standpoint. *IEEE Transactions on Information Theory,* **34,** 979–994.

HUFFMAN, D. A. (1952). A method for the construction of minimum redundancy codes. In *Proceedings IRE,* **40,** 1098–1101.

KNUTH, D. E. (1973). *The Art of Computer Programming,* vol. 3, p. 231. Reading MA.: Addison-Wesley.

MARR, D. & HILDRETH, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London B,* **207,** 187–217.

MICHIE, D. (1987). Current developments in expert systems. In *Applications of Expert Systems,* J. R. QUINLAN, Ed., pp. 137–156. Sydney: Addison-Wesley.

MICHIE, D., MUGGLETON, S., RIESE, C. & ZUBRICK, S. (1984). RuleMaster—a second generation knowledge engineering facility. In *Proceedings of the First Conference on Artificial Intelligence Applications,* Denver, CO.

NITZAN, D. (1985). Development of intelligent robots: achievements and issues. *IEEE Journal of Robotics and Automation,* **1**(1), 3–13.

PITAS, I. & VENETSANOPOULOS, A. N. (1986). Edge detectors based on nonlinear filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **8,** 538–550.

QUINLAN, J. R. (1983). Learning efficient classification procedures and their application to chess endgames. In *Machine Learning: An Artificial Intelligence Approach,* R. S. MICHALSKI, J. G. CARBONELL & T. M. MITCHELL, eds. Palo Alto, CA.: Tioga.

QUINLAN, J. R. (1986). The effect of noise on concept learning. In *Machine Learning,* R. S. MICHALSKI, J. G. CARBONELL & T. M. MITCHELL, Eds., vol. 2, pp. 149–166. Morgan Kaufmann Publishers.

RIESE, C. (1984). RuleMaster: control strategies. In *Radian Technical Report RI-RS-*00296, Radian Corporation, Austin TX.

SETHI, I. K. & SARVARAYUDU, G. P. R. (1982). Hierarchical classifier design using mutual information. In *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **4,** 441–445.

SMYTH, P. (1988). *The Application of Information Theory to Problems in Decision Tree Design and Rule-Based Expert Systems,* PhD Thesis, Department of Electrical Engineering, California Institute of Technology.

STUART, J. D., PARDUE, S. D., CARR, L. S. & FELDCAMP, D. A. (1984). TITAN: an expert system to assist in troubleshooting the Texas Instruments 990 minicomputer system, *Radian Technical Report ST-RS-*00974, Radian Corporation, Austin TX.

WAHL, F. M. (1987). *Digital Image Signal Processing,* Norwood, MA: Artech House.

WANG, Q. R. & SUEN, C. Y. (1984). 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,* **4,** 406–417.

WAXMAN, A. M. *et al.* (1985). A visual navigation system for autonomous land vehicles. *IEEE Journal of Robotics and Automation,* **3**(2), 124–142.

WHITE, A. P. (1985). PREDICTOR: an alternative approach to uncertain inference in expert systems. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* pp. 328–330. Los Angeles CA.