and V_n with value in $\{1,\cdots,M\}$. The local performance of δ_n is measured by $L_n(x) = P\{\delta_n(X) \neq Y | V_n, X = x\}$, while the global performance by $L_n = E(L_n(X)|V_n)$. $L^*(x)$ and L^* denote the local Bayes' probability of error at xR^d and the Bayes' probability of error, respectively.

Let $p_i(x) = E(I_{\{Y=i\}}|X = x)$ and let $p_{in}(x)$ be its estimate on the data $(X_1, I_{\{Y_1=i\}}),\cdots,(X_n, I_{\{Y_n=i\}})$ derived from m_n. The recursive computation of p_{in} may be carried out by

$$p_{i0}(x) = r_{i0}(x) = 0$$

$$r_{in}(x) = r_{in-1}(x) + K\left(\frac{x - X_n}{h(n)}\right)$$

$$p_{in}(x) = p_{in-1}(x) + r_{in-1}^{-1} \cdot \left(I_{\{Y_n=i\}} - p_{in-1}(x)\right) K\left(\frac{x - X_n}{h(n)}\right).$$

Using p_{in} we obtain a classification rule which classifies every X as coming from any class which maximizes $p_{in}(X)$. Reasoning similarly as in [7] we conclude from Theorem 1 and 2 the next two theorems concerning asymptotic optimality of the rule.

Theorem 3: If (1), (2), (3) are satisfied then

$$L_n(x) \to L^*(x) \quad \text{in probability as } n \to \infty \text{ for almost all } x(\mu).$$

If in addition (4), (5) hold then

$$L_n(x) \to L^*(x) \quad \text{almost surely as } n \to \infty \text{ for almost all } x(\mu).$$

Theorem 4: Under the same assumptions as in Theorem 3 we obtain, respectively,

$$L_n \to L^* \quad \text{in probability}$$

and

$$L_n \to L^* \quad \text{almost surely as } n \to \infty.$$

Appendix

Proof of Lemma 2: Let us consider the quotient

$$\sum_{i=1}^{n} h^d(i) \Big/ \sum_{i=1}^{n} EK\left(\frac{x - X_i}{h(i)}\right) \leq \max_{1\leq i\leq n}\left(h^d(i)/EK\left(\frac{x - X_i}{h(i)}\right)\right) \leq \beta^{-1} \max_{1\leq i\leq n}\left(h^d(i)/\mu\left(S_{x,rh(i)}\right)\right).$$

This lemma follows from (2) and the fact that $h^d/\mu(S_{x,rh})$ possesses a finite limit as $h \to 0$ for almost all $x(\mu)$ (see Devroye [3, lemma 2.2]). □

Proof of Lemma 3: Let us transform expression (7) as follows:

$$c_2 H(0) \sum_{n=1}^{\infty} \gamma n \left(\sum_{i=1}^{n} h^d(i)_i \Big/ \sum_{i=1}^{n} h^d(i)\right)^{-1} \cdot \left(h^d(n) \Big/ \left(\sum_{i=1}^{n} h^d(i)\right)^2\right), \quad (10)$$

where

$$\gamma_n = h^{-d}(n) EK\left(\frac{x - X_n}{h(n)}\right).$$

If μ is absolutely continuous with density g then $\gamma_n \to g(x)$ as $n \to \infty$ for almost all $x(\mu)$ (see Wheeden and Zygmund [9, th. 9.13]). By Kronecker's lemma and by (9) the series in (10) is convergent for almost all $x(\mu)$. Thus Theorem 2 follows.

Next, reasoning similarly as in the proof of Theorem 2, (7) may be transformed for almost all $x(\mu)$ to the form

$$c(x) \sum_{n=1}^{\infty} \left(n\mu\left(S_{x,r_2h(n)}\right) \Big/ \sum_{i=1}^{n} \mu\left(S_{x,r_1h(n)}\right)\right)\left(1/n \sum_{i=1}^{n} h^d(i)\right),$$

where $c(x)$ is a positive constant independent of n, because by (3) there exist positive numbers c_3, c_4, r_1, r_2 such that

$$c_3 I_{\{\|x\|\leq r_1\}}(x) \leq K(x) \leq c_4 I_{\{\|x\|\leq r_2\}}(x).$$

If μ is purely atomic the by assumption (1) the first term in brackets converges to 1 for every $x \in \{x: \mu(\{x\}) > 0\}$ as $n \to \infty$. By successive application of (8) and (9) Theorem 2 follows. □

Acknowledgment

The authors wish to thank an anonymous referee for suggesting improvements in this correspondence.

References

[1] R. Bojanic and E. Seneta, "A unified theory of regularly varying sequences," *Math. Zeitschrift*, vol. 134, pp. 91–105, 1973.
[2] L. Devroye and T. J. Wagner, "On the $L1$ convergence of kernel estimators of regression function with application in discrimination," *Zeitschrift für Wahrscheinlichkeitstheorie und Vervandte Gebiete*, vol. 51, pp. 15–25, 1980.
[3] L. Devroye, "On the almost everywhere convergence of nonparametric regression function estimates," *Ann. Statist.*, vol. 9, pp. 1310–1319, 1981.
[4] W. Greblicki and A. Krzyżak, "Asymptotic properties of kernel estimates of a regression function," *J. Statist. Planning and Inference*, vol. 4, pp. 81–90, 1980.
[5] L. Györfi, "Recent results on nonparametric regression estimate and multiple classification," *Prob. Contr. Inform. Theory*, vol. 10, pp. 43–52, 1981.
[6] A. Krzyżak and M. Pawlak, "Universal consistency results for the Wolverton-Wagner regression estimate with application in discrimination," *Prob. Contr. Inform. Theory*, vol. 12, pp. 33–42, 1983.
[7] ——, "Distribution-free consistency of nonparmetric kernel regression estimate and classification," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 78–81, Jan. 1984.
[8] M. Loeve, *Probability Theory I*, 4th ed. New York: Springer-Verlag, 1977.
[9] R. L. Wheeden and A. Zygmund, *Measure and Integral*. New York: Marcel Dekker, 1977.

Decision Tree Design Using a Probabilistic Model

RICHARD G. CASEY AND GEORGE NAGY, SENIOR MEMBER, IEEE

***Abstract*—A sequential optical character recognition algorithm, ideally suited for implementation by means of microprocessors with limited storage capabilities, is formulated in terms of a binary decision tree. Upper bounds on the recognition performance are derived in terms of the stability of the digitized picture elements. The design process is described in detail. The algorithm is tested on single-font typewritten characters and the experimental and theoretical results are compared.**

I. Introduction

When first introduced in the 1960's, optical character recognition (OCR) was pursued primarily as a high-throughput means of

Manuscript received January 11, 1982; revised April 13, 1983.
R. G. Casey is with the IBM Research Laboratory, San Jose, CA 95193.
G. Nagy is with the Department of Computer Science, University of Nebraska, Lincoln, NE 68588.

0018-9448/84/0100-0093$01.00

data entry using costly, special purpose hardware. Speeds on the order of thousands of characters per second were needed in order to justify the expense of an OCR system. Indeed, a rule of thumb [1] was that an automatic entry device had to displace five to ten key-punches to be economically favorable.

The economic criteria have changed with the marked reduction in the costs of electronic technology. Documents can now be scanned by a solid-state sensor priced several orders of magnitude below a CRT-based transducer. Inexpensive logic chips can perform the necessary image-processing functions. The theme of simplicity and low cost can be maintained throughout the device if the classification algorithms are implemented in a microprocessor. These elements are realized in several hand-held character recognition units for restricted symbol sets that have appeared on the market bearing price tags in the low thousands of dollars. OCR, it seems, may become an interactive tool, perhaps finding its way even into the home terminal environment.

The most frequently used commercial recognition techniques do not seem readily adaptable to the construction of a CPU-based OCR device. Template comparison [2] or feature detection [3] schemes require numerous special logic circuits operating in parallel in order to achieve even moderate speeds. In a high-speed data entry system the expense of this parallelism can be justified, and indeed it does not constitute a disproportionate part of the total cost of such a machine. In a low-cost device, particularly if a serial processor is to implement the logic functions, less complex recognition techniques are called for.

A classification method that is well suited to serial implementation is the decision tree. Usually based on pattern features, it was one of the earliest techniques used for the assignment of identities in a recognition system [4] and remains a subject of continuing interest [5].

In the following sections, the application of decision trees to character recognition will be explored. The discussion will be restricted to the use of trees operating directly on the individual picture elements (or pixels) of a scanned character, rather than on feature inputs defined on combinations of pixels. However, the methods developed are also applicable to the latter case.

The plan of exposition is as follows. The principles and implementation of a decision tree are presented. The automatic design of decision tree logic from sample scanned characters is then described. A design technique based on a probability model for the frequency of black occurrences in the various pixel positions is developed. An analytical model is developed in order to obtain bounds on the performance of a tree as a function of its size and the reliability of the character pixels. Such a model indicates the nature of the data environment in which decision trees can be expected to yield low error rates. Recognition experiments with single-font typewritten characters serve to validate the design scheme.

II. A Decision Tree Using Individual Pixel Values

The techniques for scanning a line of text into a binary image, for segmenting it into individual character frames, and for registering the character patterns in standardized positions in preparation for a classification decision, have been implemented in the course of the present investigation, but will not be elaborated here. The reader is referred to the literature [6], [7], [8] for a general understanding of the OCR process.

It is presumed, then, that a succession of two-dimensional binary patterns each representing a single registered character, is presented to the OCR decision logic for identification. The procedure for classifying a pattern may be represented by a tree graph, as in the example of Fig. 1.

The decision tree calls for determination of the values of a sequence of picture elements in the scanned character array. The first pixel to be tested is predetermined. It corresponds to the root node of Fig. 1. After the first pixel and in all succeeding steps, the next pixel to be examined depends on the values of the pixels previously queried. Thus, in Fig. 1, each node represents a pixel test, and from any node one of two branches may be taken, depending on whether the pixel is observed to be *white* or *black*. (In this correspondence pixel values may also be indicated by "W" or "B".)

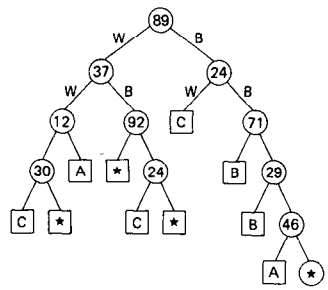

Fig. 1. Decision tree. Interior nodes, shown as circles, indicate which pixel is being tested. Leaves are shown as squares, with the asterisk indicating no decision (reject).

TABLE I
Tree of Fig. 1 Represented as a Table[1]

Address	Pel No.	Branch on 'W'	Branch on 'B'
1	89	2	3
2	37	4	5
3	24	-3	6
4	12	7	-1
5	92	0	8
6	71	-2	9
7	30	-3	0
8	24	-3	0
9	29	-2	10-
10	46	-1	0

[1]Negative integers indicate leaves: for instance, -2 in the sixth row means that if pixel 71 is white, the character is classified as "B". Zeros stand for reject nodes.

The branch from a node may also lead to a "leaf," or decision node. When a leaf node is reached the pixel tests cease and the input pattern is given the label associated with the leaf. Permissible labels include not only the symbols in the alphabet to be read, but also a special "reject" code indicating that the pattern did not appear to fit any category well enough to be identified. Rejects will ordinarily be recognized either visually, or else by a more complex default decision procedure.

The classification of a pattern by a decision tree thus consists of following a path through the tree from the root to a leaf. A given pixel may appear more than once in the tree, but logically should occur at most once along any path. The tree is not necessarily balanced, and indeed will ordinarily be highly unbalanced, as will be seen. This is because a short path may permit the reliable identification of certain patterns, whereas with other patterns a much greater number of pixels must be examined in order to classify them with the same expected accuracy. In general, however, a decision will be made after testing only a small fraction of the total number of pixels in a character.

The implementation of a decision tree in a general purpose processor may be achieved by means of a table (Table I) containing three columns. Each row of Table I corresponds to a node of the decision tree. The first row defines the root node. Column number 1 contains the position (converted to a single integer) of the pixel to be examined at the associated node. Columns 2 and 3 contain the branch addresses to other rows in the event that the pixel tested is black or white, respectively. A leaf, or decision, node is denoted by a special convention, such as the negative of the class number (e.g., Table I).

The algorithm for recognizing a pattern by means of a decision tree has a simple iterative structure, as the preceding description indicates. If the table is located in main storage, and if the

processor has the appropriate indexing capability and instruction set, then very few processor cycles will be needed in order to implement the inner loop. Since only a small subset of the pattern field is examined prior to a decision, the tree method is ordinarily much faster than, say, a template matching algorithm against idealized reference patterns, in which *every* pixel of a scanned character would be compared to the corresponding pixel of *each* stored template.

Background—Decision Tree Design

There are two schools of thought on what may perhaps be called the "philosophy" of designing OCR decision logic. Proponents of interactive design hold that it is the human who knows the features and rules that identify character classes, and that a means must be provided to express and make use of this understanding. Typically, interactive specification is carried out by successive modification of an initial design and optimization of performance on a large collection of sample characters. The designer attempts to understand and correct the causes of error while maintaining accuracy on correctly identified inputs [9].

Other researchers have maintained that recognition is a statistical matter to be solved by abstracting information from sample patterns. Visual analysis of the data base is deemed inefficient and unappetizing; the task of the designer is to specify the information to be collected. A computer is programmed to process the sample patterns and is further instructed how to use the data obtained in order to construct an appropriate classification scheme. An analytical model often governs the design process [10], [11].

In this correspondence the *automatic* construction of decision trees is described. Since a decision tree is an example of a sequential process, a book by Fu [12] is pertinent to the general problem area. The presentation here, however, is intended to be self-contained and particular to the OCR application.

The automatic design technique is based upon a probabilistic model of pattern variation. The statistical parameters of the model are estimated from observations on identified sample characters. Using these parameters, the variables of the decision logic are specified so as to minimize an analytical expression for the uncertainty in the recognition process.

The risk in reliance on a probabilistic model is that it often requires simplifying assumptions that can fail to hold in practice. In the model developed here, for example, error rate is calculated on the supposition that pixel values observed while recognizing a sample character are mutually independent.

An alternative, deterministic approach to design consists of operating on a set of scanned patterns in order to produce a system that recognizes them with minimal error rate. Such an approach has also been investigated but appears to lack the flexibility of the probabilistic method.

III. An Algorithm for Decision Tree Design

Since a decision tree defines a sequential classification process, the technique for specifying a tree will be sequential also: at any level of the tree the pixels selected for observation depend on those selected at higher levels of the tree. Thus, following the estimation of statistical parameters from sample character patterns, the actual tree construction is begun with the specification of the root node. The design process then adds nodes one-by-one below the root.

The design loop consists of choosing a tree node just below one that has already been specified, and of evaluating all pixels as candidates for this node in the light of the statistics measured for the various classes. The desirability of any candidate pixel depends on the relative probabilities of the classes at the node, and on the pixels previously examined. After a pixel has been selected, a decision must be made whether either of the outcomes possible upon observing the pixel permit an identity to be assigned to the input pattern. If so, the branch corresponding to the outcome is routed to a leaf node. This node specification sequence is repeated until all tree paths terminate, or until some other constraint condition is reached (for example, a limit on tree size or a required level of estimated performance).

The sequence in which new nodes are attached to the tree is an important consideration. For example, the tree may be designed level-by-level, or by pursuing particular paths in depth. A performance-related selection mechanism will be discussed later in the paper.

It is clear that because of the interrelation among the nodes, decision tree design is inherently a complicated multivariable optimization problem, perhaps best stated in dynamic programming terms. The problem of optimal design cannot be resolved neatly by a node-by-node progression. The iterative design flow is, however, capable of producing large design trees for many thousands of sample patterns with an efficient use of computational resources.

A general procedural version of the algorithm follows.

procedure TREEDESIGN
Estimate *a priori* class probabilities for current application.
Estimate pixel state probabilities for each class from training samples
Node list = (root node)
  while number of nodes $< N$ **and** $P_c < T$ **do**
    Select node with highest entropy from node list.
      Evaluate information gain for each untested pixel and assign pixel with highest gain to this node.
      Create black and white successor nodes to current node and add to node list.
  end while
assign identity to each leaf with $P_c > T$, designate all other leaves as "reject."
end TREEDESIGN

Details of these steps are discussed in succeeding sections.

A. Estimation of State Probabilities

The first step in the design process is to estimate the *state probabilities* (i.e., the probabilities of white and black for different pixel positions) for each character class. The estimate for a given pixel is formed on the basis of the frequency distribution of white and black in that position for a large collection of identified sample character patterns.

An error in the value assigned to a given pixel state probability can result in degraded performance, particularly for probabilities in the neighborhood of 0 or 1. Suppose, for example, that a given pixel is observed to be black for 100 out of 100 samples of class A. The actual state probability may be, say, 0.99 and it is only the luck of the draw that no white observations resulted from the 100 trials.

If we assume *a priori* that the probability θ that a pixel is black is uniformly distributed over $[0, 1]$, then the Bayes' estimate (i.e., the expected value of the *a posteriori* distribution) of θ is

$$\hat{p} = \frac{\int_0^1 \theta^{n+1}(1-\theta)^{N-n}\,d\theta}{\int_0^1 \theta^{n}(1-\theta)^{N-n}\,d\theta} = \frac{n+1}{N+2},$$

where n is the number of "black" values observed for the pixel in N samples.

The estimate $\hat{p}$ is always nearer to 0.5 than is the observed frequency of black. In addition, it approaches the observed frequency n/N as a large number of observations are made.

To obtain a more flexible estimation rule, the expression for $\hat{p}$ can be modified to

$$\hat{p} = \frac{n+\alpha}{N+2\alpha}.$$

Fig. 2. Composite digitized characters. Integers indicate the probability that a given pixel is black in fifty design samples of that class. Asterisks show positions which are black in all of the samples, blanks correspond to "all white," zeros stand for probabilities in range of 0 to 0.10.

This more general formula results from the assumption that the *a priori* distribution is a beta-distribution with parameter α rather than a uniform distribution [3]. Note, however, that the specification $\alpha = 1$ corresponds to the latter assumption. In practice, α is chosen for the particular application and is not critical for large sample sizes. In the experiments reported here, the value $\alpha = 1$ was employed routinely. For very small sample sizes, however, a value of α much larger than 1 yielded superior results, probably because the pixel distribution tends to be concave rather than flat.

If we assume, further (and this is a critical assumption for simplifying the design process), that the state probabilities $p_k^i = \Pr(\text{pixel } j = 1 | \text{ class } k)$ are statistically independent, then the initial estimates can be used at every node in the tree. If the pixel correlation were included in the analysis then the state probabilities would vary from node to node. The state probability at a given node would then depend on the outcomes of pixel observations along the path from the root to this node.

Thus, not only are the calculations simplified if pixel independence is assumed, but in addition, the relevant statistics, namely, the pixel state probabilities per class, can be held fixed during the tree design phase. The patterns on which the design is based are not referenced again after the pixel probabilities have been estimated. Since in practice a design may be conducted using hundreds of thousands of scanned characters, the reduction in computational expense due to neglecting correlation is considerable.

The validity of the independence assumption depends largely on the uniformity of printing. Scanned samples in a single-font style printed by one machine are found to vary mainly in edge pixels in a random way (see Fig. 2). Pixel probabilities tend to be independent in such cases. As more printers, additional font styles, etc., are included in the sample population, the assumption of independence becomes less realistic. Variations in line thickness among characters, for example, imply that pixels along the periphery of the patterns are correlated. In such cases, the independence assumption becomes an *ad hoc* measure, to be justified by the performance attained in competition with alternative procedures.

B. Probability Distribution of Classes at a Node

Before it can assign a pixel to a tree node, the design process must calculate the class probability distribution at the node. These can be computed by traversing the path to the node as follows. Suppose pixel j is tested at a given node. Let the probability that a character belonging to the kth class follows the path to this node be $P(k)$. The two branches out of this node correspond to the color values possible for pixel j and lead to a "black" successor node and a "white" successor node, respectively. The probability that the sample from class k arrives at the black node is $P_B'(k) = P(k)P(\text{pixel } j = B|k) = P(k)p_k^j$. Likewise, the probability that it arrives at the white successor node is $P_W'(k) = P(k)(1 - p_k^j)$.

At the root of the tree, the class probabilities are the *a priori* probabilities of the classes, which can be estimated from analyzing sample documents. The above formulas permit calculation of the class distribution at successive nodes along any path from the root.

C. Criteria for Extension and Termination

The candidate nodes for extension at any step in the design process are the leaf nodes in the current tree. One of these has to be chosen, assigned a pixel test, and linked to two new leaves. The principle adopted here is to choose the candidate node whose class distribution has the greatest mathematical uncertainty [13]. (The concept of uncertainty in a decision tree will be discussed in a later section.)

Several criteria for termination of the design process have been considered. One convenient rule is to stop when an overall estimated recognition rate has been attained. The probability of correct recognition C may be calculated by summing the majority class populations over all leaf nodes in the current tree. That is, let $C_n = \max_k P_n(k)$, where $P_n(k)$ is the probability that a sample belonging to the kth class arrives at node n. Then,

$$C = \sum_{n \in L} C_n$$

where L is the set of leaf node indices.

As an alternative rule, design may be stopped when a prespecified tree size (total number of nodes) has been attained. The size specification can be made on the basis of implementational considerations such as memory limitations or maximum number of computations (path length).

When design is terminated due to either criterion the leaf nodes are assigned decision labels. Those leaves having C_n exceeding a threshold T are labeled with the majority class, while the remainder are classified as reject nodes. By varying the reject threshold T the quantities C_n may be used to plot a curve of estimated reject rate versus error rate. The designer can then choose the operating point best suited to his requirements. Note that the expected overall substitution error rate cannot exceed $1 - T$.

D. Pixel Selection—Entropy

The effectiveness of a node-by-node design scheme is highly dependent on the rule by which pixels are evaluated for assignment to a given node. A pixel must be selected on its ability to contribute discrimination to the classification process. a measure based on the entropy notions of information theory possesses desirable properties as a pixel selection criterion.

The application of the entropy measure presumes that the state probability for each class

$$p_k^j = \Pr\{\text{pixel } j = B | \text{ class } k\}$$

has been estimated for every pixel. Also assumed known is $P(k)$, the *a priori* probability that a member of the kth class arrives at the node being considered.

The mathematical uncertainty in the identity of an input pattern presented to the node is represented by the entropy

$$H_0 = -\sum_k P(k) \log P(k).$$

Similarly, the uncertainty of identification after observing "B" or "W", respectively, if the jth pixel is tested, is expressed by the two quantities:

$$H_B^j = -\sum_k P^j(k|B) \log P^j(k|B),$$

$$H_W^j = -\sum_k P^j(k|W) \log P^j(k|W),$$

where $P^j(k|B)$ is the probability that the sample belongs to the kth class given that pixel j is black, and is calculated from the pixel state probabilities p_k^j by Bayes' rule:

$$P^j(k|B) = \frac{P(k)p_k^j}{\sum_i P(i)p_i^j}.$$

In like manner, the probability of class k given that the jth pixel is white is

$$P^j(k|W) = \frac{P(k)(1-p_k^j)}{\sum_i P(i)(1-p_i^j)},$$

where $P(k)$ is the probability of arrival of class k at the node being considered.

The probability that pixel j is actually observed to be black is calculable from the two sets of probabilities already estimated

$$P_B^j = \sum_k P(k)p_k^j,$$

and in similar fashion

$$P_W^j = \sum_k P(k)(1-p_k^j).$$

Note that P_B^j, P_W^j, and $P(k)$ will take on different values from node to node, but for simplicity we omit the node index here.

The expected uncertainty after testing pixel j is then $\bar{H}^j = P_B^j H_B^j + P_W^j H_W^j$ and the net information gained in examining the pixel is measured by the average reduction in uncertainty $I^j = H_0 - \bar{H}^j$.

The rule for pixel selection employed in our design procedure is to choose the pixel j that minimizes $\bar{H}^j$, i.e., the one that maximizes the information gain.

In the development above we have used probabilities conditioned on the particular node to be extended. Alternatively, it is possible to define the overall entropy of the tree as

$$H_t = \sum_n \sum_k P(k,n) \log P(k,n),$$

where $P(k,n)$ is the joint probability that a sample arrives at node n and belongs to class k. Selecting a pixel for a given node so as to minimize H_t is equivalent to maximizing the conditional node information gain as described above, but the formulas contain an extra normalizing factor. However, it is worth noting that in selecting the node to be extended during a given design loop it is the contribution of the node to H_t that must be calculated, and not the conditional entropy for each node, since the normalizing factors vary from node to node.

The entropy criterion for selecting pixels tends to promote short path lengths by approximate balancing of the decision tree. It favors the splitting of an input population into two subpopulations of equal weight while not splitting the component class populations, to the extent that these objectives are attainable. These and several additional properties of I^j are discussed in the Appendix, where a reformulation of the expression above is given.

IV. Bounds on Recognition Performance

A practical decision tree represents a compromise between storage space and classification accuracy. If the recognition processor were capable of unlimited storage, then characters scanned into n-pixel arrays could be optimally recognized by a decision tree having at most 2^n nodes. But with existing or foreseeable storage capacities, the optimal tree is impractically large for the typical OCR situation in which an unknown character is represented by hundreds of pixels. For example, a balanced tree having only 20 nodes along each path defines a table of the form of Table I having more than three million entries.

The decision tree method of recognition must therefore rely on examining only a subset of the character pixels in order to make an identification. The pixels must be both highly reliable and discriminatory in order to permit rapid accurate decisions. A simple probabilistic model is helpful in estimating what performance can be achieved in a given environment.

Let us define the *reliability* of a pixel test as the greater of the state probability p_k^j and its complement (which is the probability that the pixel is "W" for the given class). As above, we shall assume that pixel states along any path are mutually independent.

Consider the performance of a balanced tree containing L levels, and such that every pixel has a reliability equal to P, regardless of the identity of the input pattern. A particular path through the tree consists of a sequence of L branches, of which, say, i correspond to the more likely of the two possible outcomes for samples of a given class. Then the probability that the specified path is followed by samples belonging to this class is $\beta_i = P^i(1-P)^{L-i}$.

Within the balanced tree there are $\binom{L}{i}$ paths having this same value of β_i, where $\binom{L}{i}$ is the usual notation for the number of combinations possible for L distinct items taken i at a time. The quantity i is a variable that may take on any integer value from 0 to L.

Suppose that there are N equally likely classes. Let us partition the leaves of the tree into N groups of equal size, and associate each group with a different class. That is, there are $(2^L \div N)$ leaves within each group. If N is not a factor of 2^L, then the leaves are partitioned into N groups such that no two groups differ in size by more than one leaf. To the leaves in each group, we assign the maximum values of β_i for the respective class, beginning with β_L and proceeding downwards. The probability that a sample pattern is correctly identified is equal to the probability that it arrives at one of the leaves in its class group. The assignment described assures that the 2^L most probable outcomes, i.e., the 2^L largest values of β_i over all classes, yield correct classifications. No alteration in the assignments to the groupings can improve on this performance. Therefore, the tree just described sets an upper limit on average recognition rate under the prescribed conditions. The average recognition rate P_c is given as follows.

Let a be the integer such that

$$\sum_{i=0}^{a} \binom{L}{i} \leq \frac{2^L}{N} < \sum_{i=0}^{a+1} \binom{L}{i}.$$

Then

$$P_c = \sum_{i=0}^{a} \binom{L}{i} P^{L-i}(1-P)^i + \left[\frac{2^L}{N} - \sum_{i=0}^{a} \binom{L}{i}\right] P^{L-a-1}(1-P)^{a+1}.$$

We observe that P_c cannot be increased by pruning the balanced tree in some manner. Pruning results in accumulating the occupancy probabilities of several nodes into a single node, and thus cannot increase discrimination. Furthermore, a reduction in the reliability of any node decreases the peak leaf occupancy probabilities for the affected class and thus cannot yield a higher

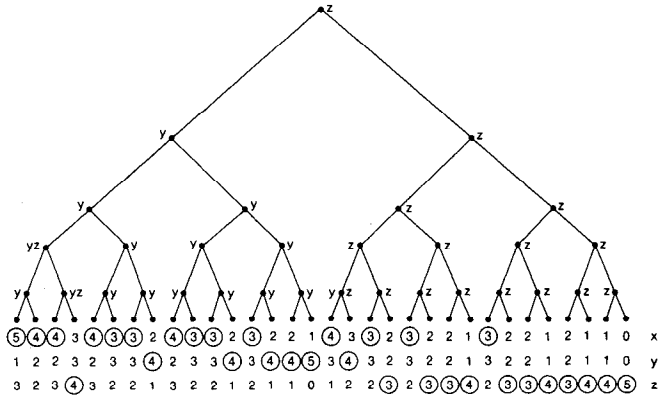

Fig. 3. Realizability of the bound on classification accuracy. There are three classes, x, y, and z, and each interior node of the tree is assumed to specify a pixel of reliability P. Each node is labeled with classes for which the most probable exit branch is to the right. Otherwise the most probable branch is to the left. Below leaves are tabulated for each class the quantity i (see text), the number of high probability branches along the path to the leaf. Circled number is the maximum i for the leaf, and indicates the class assigned to the leaf. Among the 96 values of i listed, the 32 largest values (three 5's, fifteen 4's, and fourteen 3's) are each assigned to a different leaf. Therefore probability of correct classification for this tree equals the upper bound given in text with L = 5 and a = 1.

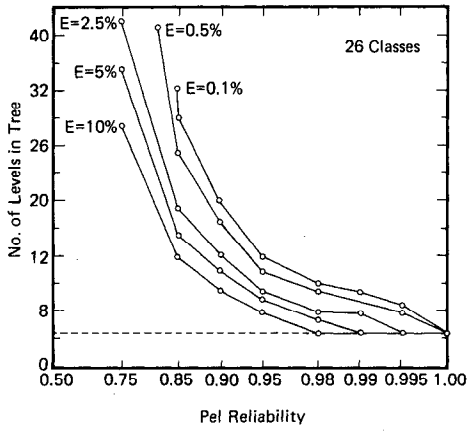

Fig. 4. Lower bound on error rate as a function of pixel reliability and tree size. Calculations are based on balanced trees of depth (path length or number of levels) as shown on the ordinate. Minimum number of levels necessary to discriminate between 26 classes is 6. Pixel reliability is assumed to be the same for all pixels in the tree.

probability of correct recognition no matter how the leaves are redesignated.

These observations may be formalized to arrive at the following conclusion: the quantity, P_c is an upper bound on the recognition rate of N equally likely classes by a decision tree whose longest path comprises L nodes, and none of whose pixel tests have a reliability greater than P. The bound is achieved if every pixel test has reliability P for each class, and if an "appropriate" distribution of the state probabilities for each class exists within the tree, i.e., if it is actually possible to distribute the class probabilities over the leaves in the manner indicated above. If it is not possible, then P_c is still a bound, but can not be realized by any tree of the given size.

Fig. 3 shows an example with three classes and five levels in which P_c is realizable. As can easily be shown, however, for the same number of classes, but with only four levels in the tree, the bound is unrealizable.

Fig. 4 is a plot of the quantity $1 - P_c$ (the lower bound on the rate of unrecognized characters) as a function of maximum path length for several values of pixel reliability, and assuming 26 classes.

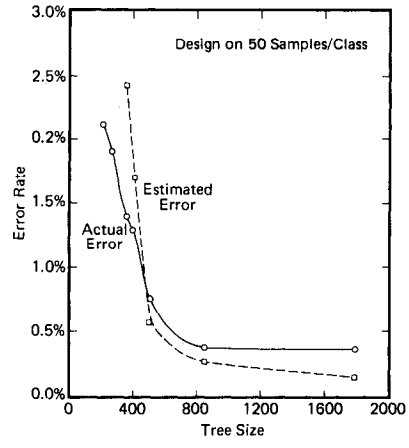

Fig. 5. Actual and estimated error rate as a function of the number of nodes in tree. Estimated error for each of seven trees is computed as part of the design process on the basis of the pixel probabilities in design sample. New samples were used to determine actual classification error for each tree.

V. Experiments

A. Scanned Data

The bounds on recognition performance summarized in Fig. 4 indicate that to obtain low error rates from small trees, high reliability is required of the pixels selected for inclusion in the tree. Such a condition is best provided by well-registered machine printed characters in a single-font style.

Fig. 2 is derived from the superposition of 500 sample characters in a Courier type font. The characters were scanned at a resolution of 4 mil. Each scanned character is registered by translating it so as to center its bounding rectangle in the pattern field. The figure illustrates what one would expect for well-printed characters. Most of the pixel positions are stable; noise is restricted to the contours of the characters and is due to slight errors in registration, to discretization by the scanner, and to small differences in the printed characters.

The data of Fig. 2 is typical of the scanned characters used in the experiments. Upper case characters were scanned at several different resolutions (4 to 6 mil), and stored on magnetic tape. One set of scanned patterns was reserved for tree specification; the remainder were set aside for recognition performance tests. For design experiments a portion of the training set was read from tape, registered by the bounding rectangle method, and used to estimate pixel frequencies. Characters used for recognition tests were also registered by the same method.

B. Estimated Versus Actual Performance

Fig. 5 shows a comparison of estimated performance against that obtained by actual recognition of scanned characters. Trees of several different sizes (as measured by the number of test nodes) were constructed by the design algorithm. The cross-over point in Fig. 5 is a consequence of the probabilistic design approach. Because of the small number of design samples used (50 per class) and the rule which reduces the estimated pixel reliability below its observed value, the estimated error tends to be higher than the actual error rate for small trees (where only a few pixels are used in classification). Performance improves as larger trees are constructed, but not as quickly as estimated due to correlation among the selected pixels. That is, the additional pixels selected for inclusion in the tree do not supply independent information as postulated by the model.

C. Reject Criterion

By designating selected leaf-nodes of the trees to be reject nodes (according to the criterion described earlier in the paper),

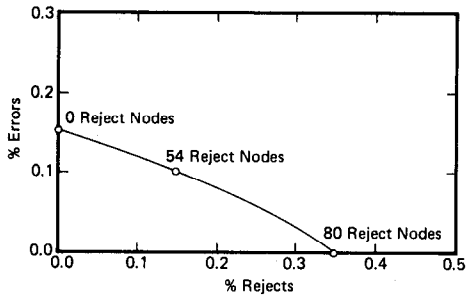

Fig. 6. Error-reject curve. Error rate can be reduced significantly by converting the leaves with the highest estimated error rates to reject nodes.

the curve in Fig. 6 was obtained. The original tree in this sample contained 300 test nodes (and therefore 301 decision nodes). When every leaf was associated with a character label, the error rate was 0.15 percent. By successively setting to reject nodes the leaves whose estimated error rate was highest, the substitution rate was gradually reduced to zero as shown. The penalty was a 2.3 to 1 ratio of rejected characters to substitution errors avoided.

VI. Conclusion

A sequential character recognition algorithm based on binary decision trees was described and the method tested on single-font typewritten characters. The design approach is simple and computationally efficient. The only information required for design is the *a priori* class probabilities and the number of black bits in each pixel position in each class in the design sample. The following advantages are observed:

a) The classification is rapid and ideally suited for microprocessor implementation.
b) A tree can be designed with just enough nodes to yield a prescribed expected error probability (within the limitations imposed by the assumption of statistical independence between pixels).
c) A rational method of rejecting samples at certain leaves is provided. This method minimizes the expected error rate for the prescribed reject rate.
d) Compensation for the uncertainties in the estimation of pixel probabilities from small numbers of design samples can be achieved by extending the recognition tree.
e) The scheme is analytically tractable and upper bounds on the expected recognition performance for a tree with a given number of nodes can be readily calculated. This model can assist in estimating the tree size required in a given application.

Appendix

An Alternative Formulation of the Expression for Information Gain

The entropy expression defined in the text can be rewritten as follows:

$$P_B^j H_B^j = -\sum_k P_B^j P^j(k|B)\log P^j(k|B)$$

$$= -\sum_k P(k)p_k^j \log P(k)p_k^j + \sum_k P(k)p_k^j \log p_B^j$$

but

$$\sum_k P(k)p_k^j = P_B^j,$$

and so

$$P_B^j H_B^j = P_B^j \log P_B^j - \sum_k P(k)p_k^j \log P(k)p_k^j.$$

Likewise,

$$P_W^j H_W^j = P_W^j \log P_W^j - \sum_k P(k)(1-p_k^j)\log P(k)(1-p_k^j).$$

Combining these two expessions yields

$$\bar{H}^j = P_B^j H_B^j + P_W^j H_W^j$$

$$= \left[P_B^j \log P_B^j + P_W^j H_W^j\right] - \sum_k \left[P(k)p_k^j \log p_k^j + P(k)(1-p_k^j)\log(1-p_k^j)\right] - \sum_k P(k)\log P(k).$$

Noting that the last term on the right-hand side is H_O, and defining several new entropy expressions, the average information gained by observing the state of the jth pixel is

$$I^j = H_0 - \bar{H}^j = H^j(B|W) - \sum_k P(k)H_k^j(B|W)$$

where

$$H^j(B|W) = -P_B^j \log P_B^j - P_W^j \log P_W^j$$

$$H_k^j(B|W) = -p_k^j \log p_k^j - (1-p_k^j)\log(1-p_k^j).$$

The latter pair of expressions can be interpreted directly. $H^j(B/W)/$ is the uncertainty of the state of the jth pixel, averaged over all classes. $H_k^j(B/W)$ is the uncertainty of the state of the jth pixel for the kth class in particular.

Several properties result from this reformulation.

1) Since both $H^j(B/W)$ and $H_k^j(B/W)$ are numbers between 0 and 1, then the maximum possible information to be gained by testing a pixel is unity.
2) This maximum value occurs if $H^j(B/W)$ is unity, and if the individual terms $P(k)H_k^j(B/W)$ vanish. That is, each class having a nonzero probability of occurrence should have the jth pixel either always "black" or always "white", and the sum probability of occurrence of the classes associated with either state should be 50 percent.

In addition, the reformulation has certain computational advantages over the original expression for information gain. For instance, all the uncertainties $H_k^j(B/W)$ can be calculated and stored prior to starting the tree design procedure. Only one entropy per pixel, namely, $H^j(B/W)$ has to be recalculated for each node specified.

References

[1] Datapro Research Corporation, *All About Optical Readers*. New Jersey: Delaran, 1975.
[2] J. Balm, "An introduction to optical character reader considerations," *Patt. Recogn.*, vol. 2, pp. 151–166, 1970.
[3] L. A. Kamentsky and C. N. Liu, "Computer-automated design of multifont print recognition logic," *IBM J. Res. Dev.*, vol. 7, no. 1, pp. 2–13, 1963.
[4] M. A. Stevens, "Automatic character recognition," U.S. Department of Commerce Tech. Note 112, PB 161613, 1961.
[5] I. K. Sethi and G. P. R. Sarvarayudu, "Hierarchical Classifier Design Using Mutual Information," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-4, no. 4, pp. 441–445, July 1981.
[6] G. L. Fischer, D. K. Pollock, B. Raddack, and M. E. Stevens, *Optical Character Recognition*. Washington, DC: Spartan Books, 1962.
[7] British Computer Society, *Character Recognition*. London: BCS, 1967.
[8] R. Shillman, C. Cox, T. Kuklinski, J. Ventura, B. Blesser, and M. Eden, "A bibliography in character recognition: Techniques for describing characters," *Visible Lang.*, vol. 7, no. 2, pp. 151–166, 1974.
[9] W. C. Naylor, "Some studies in the interactive design of character recognition systems," *IEEE Comput.*, vol. C-20, pp. 1075–1086, Spet. 1971.
[10] A. Kulkarni and L. Kanal, "An optimization approach to hierarchical classifier design," in *Proc. 3rd IJCPR*, Coronado, CA, 1976.
[11] P. Argentiero, R. Chin, and P. Beaudet, "An automated approach to the design of decision tree classifiers," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-4, no. 1, pp. 51–57, Jan. 1982.
[12] K. S. Fu, *Sequential Methods in Pattern Recognition and Machine Learning*. New York: Academic 1968.
[13] H. Portig, private communication.