STANFORD ARTIFICIAL INTELLIGENCE PROJECT
MEMO AIM-136
COMPUTER SCIENCE DEPARTMENT
REPORT NO. STAN-CS-70-183

MACHINE LEARNING THROUGH SIGNATURE TREES. APPLICATIONS TO HUMAN SPEECH

BY

GEORGE M. WHITE

OCTOBER 1970

COMPUTER SCIENCE DEPARTMENT
STANFORD UNIVERSITY

STANFORD ARTIFICIAL INTELLIGENCE PROJECT OCTOBER 1970
MEMO NO. AIM-136

COMPUTER SCIENCE DEPARTMENT
REPORT NO. CS183

MACHINE LEARNING THROUGH SIGNATURE TREES. APPLICATION TO HUMAN SPEECH.

by

George M. White

ABSTRACT: Signature tree "machine learning", pattern recognition heuristics are investigated for the specific problem of computer recognition of human speech. When the data base of given utterances is insufficient to establish trends with confidence, a large number of feature extractors must be employed and "recognition" of an unknown pattern made by comparing its feature values with those of known patterns. When the data base is replete, a "signature" tree can be constructed and recognition can be achieved by the evaluation of a select few features. Learning results from selecting an optimal minimal set of features to achieve recognition. Properties of signature trees and the heuristics for this type of learning are of primary interest in this exposition.

The research reported here was supported in part by the Advanced Research Projects Agency of the Office of the Department of Defense (**SD-183**), and in part by a Public Health Service Post Doctoral Research Fellowship.

Reproduced in the USA. Available from the Clearinghouse for Federal Scientific and **Technical Information**, Springfield, Virginia 22151. Price: Full size copy $3.00; microfiche copy $ .65.

PREFACE

The purpose of the work reported here has been to develop a computer program for manipulating signature trees as a general research tool for exploring machine learning and pattern recognition. Application of the program to speech recognition was done simply to test its effectiveness for a specific problem. Other areas of potential utility are visual pattern identification and time series analysis.

A signature tree is a binary decision tree used to classify unknown patterns. At any node in the tree, the decision to take a particular branch is determined by a single feature in the unknown pattern. The tree is automatically generated during the Yearning" phase of the program; and during the "identification" phase, the tree completely controls feature extraction procedures.

The signature tree method was devised in an attempt to generalize and extend the signature table machine learning technique developed by A. L. Samuel for checker playing.

The program may be viewed as a means of **testing** of hypotheses about characteristic features of patterns. This is accomplished by program selection of a small set of features from a user supplied list. In a manner of speaking, the user "suggests" a set of features that may be sufficient for recognition and then the program selects the "most useful" suggestions and applies them in the proper order to achieve the recognition in the fewest number of steps. Just how the program determines which user supplied suggestions are, in fact, the "most useful", is determined by the heuristics associated with the signature trees. The most useful trees in terms of speed and accuracy are trees with the smallest number of nodes as is explained in the main test.

Several notions from my personal philosophy of artificial intelligence have guided the development of the program,

(1) Hyperplane selection, data base analysis, and tree generation are all accomplished through a single <u>recursive</u> procedure.

(2) The program accepts "<u>advice</u>" through the user supplied hypothesis list.

(3) Generality and flexibility are innate since the program can process any data bases stored in reasonable formats and since the program forces the user to supply his own feature extractors.

(4) Although time has not permitted the implementation of a signature tree language, this is the intended next step.

The author has found that the system can be quite helpful in identifying functional sets of feature extractors. It effectively eliminates relatively useless feature extractors while preserving the useful ones. It also circumvents the need to explicitly consider the order of application of the extractors since the signature tree heuristics determine the ordering.

INTRODUCTION

Limited speech recognition by adaptive signature trees is being investigated. Motivation for this work comes from the success of the signature table method as employed by Samuel[1] in his research on machine learning using the game of checkers.

The signature tree heuristic requires a data base of known **"template"** patterns stored in an array which we shall hereafter refer to as the lexicon. By applying feature extractors to all the patterns, the lexicon can be reordered to place patterns with similar features in the same portion of the lexicon. This reordering reduces search time for finding an entry in the lexicon when used with an appropriate indexing scheme.

On one extreme, the existence of a single feature value (or lack of it) can be used to order the utterances in the lexicon so that all utterances in a specified portion of the lexicon have (or do not have) that feature value. Existence of the feature in an unknown utterance reduces the length of the **lexicon** search but does not eliminate the need for a search. On the other extreme, the one of primary interest here, a **sufficient** number of features can be utilized to eliminate the need for a search entirely.

Significant increases in speed of lexicon searches can be obtained by utilizing the signature tree indexing heuristic. This increase in speed results from eliminating redundancy as will be explained shortly. However, since redundancy is often useful to correct identification, there can be a corresponding decrease in accuracy.

The method has been applied to computer recognition of human speech. It has achieved an identification rate of 5 utterances per second (not including preprocessing time) with 90% correct identification for short lists on a DEC PDP-10. This is quite fast but the accuracy is not especially good. However, machine learning is the subject of greatest interest here and the method does very well in this respect. For example, training on a list of twenty-two different utterances, each spoken once, enables the machine to correctly predict only 50% of the words spoken again by the same speaker. But when the training data consists of four examples of each utterance (for a total of 88 utterances), the percentage correct rises to 90%. When an exhaustive search is made of the-lexicon rather than using signature tree indexing, the score rises to between 95% and 100%. However, the exhaustive search is about 30 times slower in this example.

While it was originally intended that Samuel's form of the signature tables be preserved in the speech work, preliminary studies carried out by Samuel and Astrahan and independently by the author indicated that significant modifications in the method were needed for speech.

The author made at least four"(4) major modifications in the method: (1) The size of an individual signature table was reduced to two entries indexed by a single input parameter. (2) The selection of the single input parameter to a table became dynamically adjustable during program execution (which means that the interconnections between tables became dynamically determined). (3) The terminal nodes in the signature table tree **pointed** to entries in a data base. (4) The generation of signature tables was done by a recursively defined function. These changes and others led to basic departures from signature table approach to machine learning. In fact, the changes culminated in new heuristic methods which are collectively known as the signature <u>tree</u> heuristics.

Special attention will be paid to Samuel's approach after a description is given of the present technique.

1.1 <u>Feature Space</u>: Let the lexicon contain M different pattern classes, e.g., M different words. Let there be N feature extractors. The N feature extractors return feature values that allow the pattern classes to be represented in an N dimensional feature space. An example of a feature extractor is a subprogram that measures the area under a curve or counts the number of maxima in a 'curve.

Let f_{ij} be the integer value returned by feature extractor j when applied to a pattern of class i. For example, f_{18} might be the integer returned for the utterance number 1, e.g., "Hello", by subprogram 8 that perhaps counts the number of syllables. (Note well that "feature extractor" is used here to mean a subprogram that returns an integer, not real, value for the measured feature.) $\widetilde{F}_i = f_{i1}\widetilde{V}_1 + f_{i2}\widetilde{V}_2 + \ldots f_{iN}\widetilde{V}_N$ where $1 \leq i \leq M$. The $\widetilde{V}_i$'s are mutually orthogonal unit vectors, i.e., $\widetilde{V}_i \cdot \widetilde{V}_j = \delta_{ij}$. f_{0ij} stands for the ensemble average of f_{ij}, that is:

$$\left\langle f_{ij} \right\rangle = \frac{1}{k_i} \sum_{n=1}^{k_i} f_{ij}(n)$$

where k_i is the number of times a pattern belonging to class i appears in the lexicon and $f_{ij}(n)$ is the integer value returned by feature extractor j for the nth occurrence of a pattern in class i.

1.2 Pattern Classification by Minimum Absolute Difference Error: The absolute difference error, $e_i(\widetilde{F})$, for any vector $\widetilde{F}$, is defined by

$$e_i(\widetilde{F}) = \sum_{k=1}^{N} \left| \left(f_{ik} - f_k \right) \right| = \left| \left(\widetilde{F}_i \right) - \widetilde{F} \right|$$

Suppose that the array $_0f_{ij}$ has been filled for all i and j values. Then the class of an unknown pattern G can be identified as that value of k for which $e_k(\widetilde{G})$ is minimized where $1 \leq k \leq M$. For a point of reference, this is what would be done in an exhaustive lexicon search if the lexicon contained only average feature values for each class. Since the expectation of an exact match between an unknown pattern and any of the templates is nearly zero, the entire lexicon would be searched for the best, though not perfect, fit. The best match would be defined to be the one that. produces the smallest absolute error.

The minimum absolute error criterion for identifying patterns has a geometrical interpretation which goes as follows:

The cluster center of a pattern i in the N dimensional feature space is located by $\widetilde{F}_i$ where

$$\langle \widetilde{F}_i \rangle = \sum_{j=1}^{N} \langle f_{ij} \rangle \widetilde{V}_j$$

The problem of identifying an unknown pattern is solved by finding the cluster center closest to the feature space point of the unknown. The nearest cluster center to the unknown is that value of k that minimizes $e_k\ (\widetilde{G})$.

This method, which is based on the utilization of average feature values, improves its performance as the number of samples increases, since the mean feature values are improved. This provides an alternative to an exhaustive search of a lexicon containing every sample as a unique point in feature space. Surprisingly the large increases in lookup speed and savings in core storage obtained by averaging are accompanied by little loss of accuracy. (The conditions under which a lexicon of average feature values is just as accurate as the full parent lexicon are not well defined. This should be investigated. However, we do not attempt to do so here.)

So far, no explanation has been made of signature tree heuristics, but we are now in a position to consider them.

2.1 Signature Trees: Signature trees are binary decision trees used to partially or totally identify patterns. Distinctive features of an unknown pattern are used one at a time to index the sequence of nodes which compose the tree. The terminal nodes of the tree point to lexicon entries that either identify the pattern or restrict the number of candidates. Automatic generation of the "**best**" tree, the tree with the greatest expectation of correctly identifying new patterns, is a primary goal of this research.

Before tree generation can start, a number of representative patterns, appropriately identified, must be available in the lexicon. Algorithmic feature extractors can then be applied to all patterns in the lexicon to produce measurements of features.

2.2 Introduction to Tree Generation: Let S_m be a subset of m integers selected without replacement from the set S of the first M positive integers 1, 2,...M, 1 < m < M. A simplified version of the signature tree method is based on the following hypothesis: Given the array f_{ij}, $1 \leq i \leq M$, $1 \leq j \leq N$, for some 'specific value of j, call it k, the following is true:

$$\langle f_{gk} \rangle < x < \langle f_{hk} \rangle$$

where-y is any element of the set S_m and h is any one of the remaining elements in $S-S_m$. Restated, there exists a feature k that serves to partition the set of all pattern classes into two nonempty subsets where one subset is composed of all pattern classes that have average feature values less than x and the other subset is the compliment of the first.

Restated again, all pattern classes do not return the same average value for feature k and so we may choose a number x such that some average feature values will be larger than x and some will be smaller. This fact can be used to reorder the entries in the lexicon of average feature values. For instance, all patterns in S_m could be placed below those in $\bar{S}_m$, the compliment of S_m. Of the pattern classes in Sm created by feature extractor k, there will be some that can be separated further by another feature extractor. This process can be repeated until there is only one pattern class left in the specified lexicon range at which time the process ends. The process is recursive and can be easily programmed for computer execution. The end result is a tree structure, a signature tree, the nodes of which contain the numbers x and k to identify the appropriate feature extractor and test value. The terminal nodes point to locations in the lexicon where a particular pattern class is stored.

The generation of a signature tree can be a good deal more sophisticated than suggested above. As it stands, the resulting tree is needlessly liable to produce errors in recognition. In particular, an unknown pattern which does not yield feature values exactly equal to its average feature values can take the wrong branches in the signature tree. In order to minimize the taking of wrong branches, the tree generator could consider **probablistic** distribution functions of patterns in the lexicon. It would search for test feature values that separate different pattern classes and in so doing affect the other classes so as to separate a minimal number of elements from other elements of the same class. This heuristic which perhaps seems intuitively clear has an information theoretic justification which is given below.

2.3 Trees that Extremize Entropy: A definition of an "optimal" set of hyperplanes can be given in terms of minimizing and/or maximizing entropy (or the information theoretic H). Maximizing H, the information content, is the same as minimizing entropy, S.

Let the information in the lexicon serve to determine $p(k,f_{ij})$, the probability density of f_{ij} for ail i and j. The integer k is a dummy index that runs over the range of values allowed for f_{ij}. (Feature values, the f_{ij}'s, are integers that typically range from 0 to 15.) If we-restrict the value of a particular feature and allow all the other feature values to remain free, we have specified a hyperplane in the N dimensional space. Let this plane be defined by HP(B,b) where B is the parameter number and b is the value of the parameter. $P_i(f_{iB} < b)$ is the probability that pattern i has feature values less than b for feature B.

$$P_i(f_{iB} < b) = \sum_{k=0}^{b-1} p(k,f_{iB})$$

and

$$P_i(f_{iB} \geq b) \quad \sum_{k=b}^{b_{max}(f_{iB})} \quad p(k,f_{iB})$$

$b_{max}(f_{iB})$ is the maximum value permitted for f_{iB}. We define the entropy, S, of the hyperplane HP(B,b) as

$$S = -\sum_{i=1}^{M} P_i(f_{iB} \geq b) \ln P_i(f_{iB} \geq b)$$

$$-\sum_{i=1}^{M} P_i(f_{iB} < b) \ln P_i(f_{iB} < b)$$

The hyperplane that produces the minimal entropy, S, which is the maximum information H, is defined to be the "best" hyperplane for discriminating between different patternclasses. This is the hyperplane that would be used at the node in the signature tree. Hyperplanes that yield the minimal entropy tend to split a minimal number of class clusters and those classes that are split have a minimal number of elements split from like elements. Trivial hyperplanes that separate no classes but do yield minimal entropy values must be thrown out: The criterion for hyperplane acceptance is that clusters centers must lie on both sides of the hyperplane. A simple verbal statement of the entropy minimization argument is as follows: The goal is to separate a minimal number of like points in feature space with hyperplanes that separate different points.

We have just defined the entropy of a single node in a tree. But what about the entire tree? Is the entropy of the entire tree simply the sum of the node entropies?

The answer is no. Moreover, minimizing the entropy of entire tree is not necessarily the thing to do. Minimizing the entropy of the entire tree would tend to produce long skinny trees. But short fat trees can be just as accurate as the skinny ones and in addition have a shorter mean traversal time. So entropy extremization for the entire tree and entropy extremization at nodes are operationally dealt with independently.

For the sake of completeness, a recursive formula is given for the entropy of a tree branch. From this the entropy' of the entire tree may be generated.

Let the level of tree nodes be labeled so that the root node level is 1; the next node level is 2, etc. Choose a node at level n. $S_{node}(n,i)$ is the entropy of this node for pattern class i. $S^k_{branch}(n,i)$ is the total of the node entropy for pattern class i, $S_{node}(n,i)$, and all other contributions from the nodes below that node that are on the k branch, where superscript k indicates that the branch that contains the node n is the right or left branch from the node at n - 1.

$$S^k_{branch}(n) = \sum_i S^k_{branch}(n,i) = \sum_i S_{node}(n,i) + \sum_i P(f_{iB} < b,n)\, S^0_{branch}(n+1,i)$$

$$+ \sum_i P(f_{iB} \geq b,n)\, S^1_{branch}(n+1,i)$$

Returning to the topic of entropy extremization, note that hyperplanes that lie between a maximal number of different pattern classes tend maximize tree entropy. The entire hyperplane selection process could thus be based nn an entropy mini-max principle. However, we shall not pursue this possibility now.

2.4 Tree Shape, Accuracy and Speed: If there are M pattern classes to be separated, then the smallest binary tree that can produce this separation has M-1 nodes, regardless of **shape.**

Proof: It is required that exactly M pointers to group identification lists be produced by a tree of binary nodes. Let there be M binary nodes. Each node has one input channel and two output channels. All but one of the nodes have their input channels connected to other nodes, so m-1 of the 2M output channels are used for internode connections. This means that **m+1** channels are available for pointers to group identification lists. Hence **m+1** = M. Thus the smallest number of binary nodes that can separate M groups is M-1 regardless of the way they are interconnected.

If the tree has one terminal node, then the tree is a linear list (and it is quite "skinny"). The mean number of nodes traversed before locating a specific pattern class is, therefore, (M-1)/2.

If every tree node points to another node wherever possible, then the resulting tree is as **"fat"** as possible. If $M = 2^x$, where **M is** the number of classes to be separated, then there are M/2 terminal nodes, and the number of nodes traversed in locating a pattern is exactly x.

All trees have shapes between the above two extremes (fat and skinny). If $\bar{T}$ is the mean number of nodes traversed, then $\ln_2 M \leq \bar{T} \leq M/2$ for any binary tree using the minimal number of nodes.

It is conjectured that maximizing tree entropy (as distinct from node entropy) as mentioned at the end of section 2.3, minimizes $\bar{T}$, the mean number of tree nodes that need to be traversed to find a specific pattern (provided that the maximization proceeds under the constraint of utilizing only those hyperplanes that produce a minimal number of nodes). It may be possible to rigorously prove. this in some simple way but to do so is not attempted here.

The plausibility argument goes as follows: Minimizing tree entropy tends to create skinny trees and maximizing tree entropy tends to create fat ones. We have just seen that fat trees have smaller $\bar{T}$ values than skinny ones. Therefore, maximizing tree entropy tends to minimize $\bar{T}$.

In summary, the shape of the tree has no relation to its discriminatory power but doesaffect the mean number of nodes that must be traversed in pattern classification.

2.5 Sizes of Trees: In section 2.4, we found that the smallest number of nodes in a binary tree with M terminal branches (pointing to M pattern classes) is M-1. Achieving this minimal tree size is the desired goal but not always attainable. The actual tree size is determined by the feature space. If different classes do not tend to occupy different portions of feature space then discrimination is difficult and the tree size is large.

Figure 1 shows two examples of patterns in a two dimensional feature space.

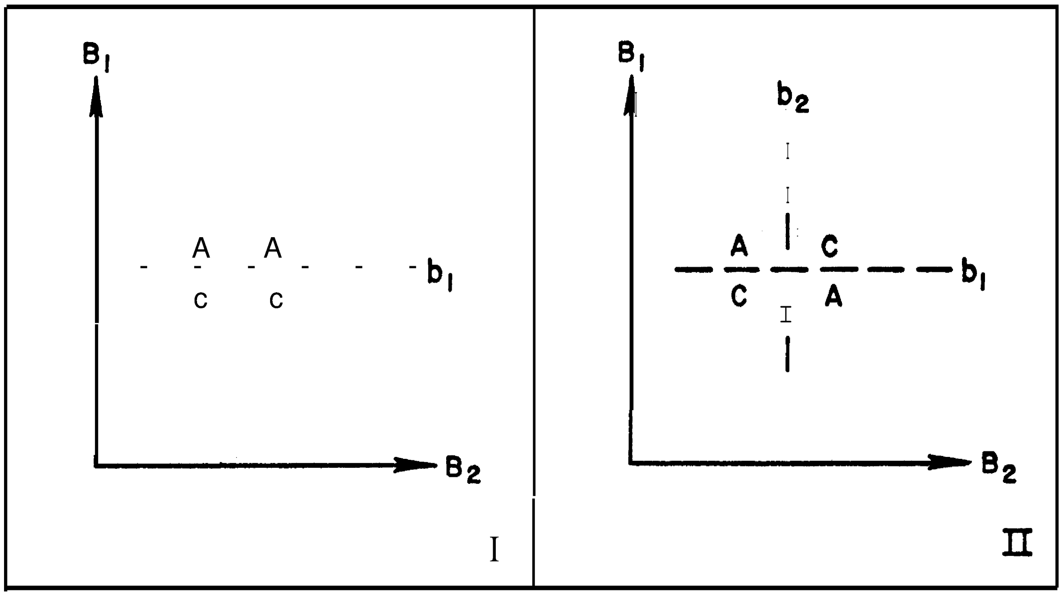

Fig. 1. Two hyperplanes are available, b_1 and b_2. In case I, bl is sufficient to discriminate between patterns A and B. But in case II both bl and b2 must be used.

For the sake of illusration, we assume that only two **hyperplanes** are available, $H(B_1,b_1)$ and $H(B_2,b_2)$. The lexicon contains four pattern elements, two of class A and two of class C. In **case** I of Figure 1, a single node tree using $H(B_1,b_1)$, is all that is needed to separate class A from class C. But in case II, both **hyperplanes** are needed. The corresponding trees are shown in Figure 2.

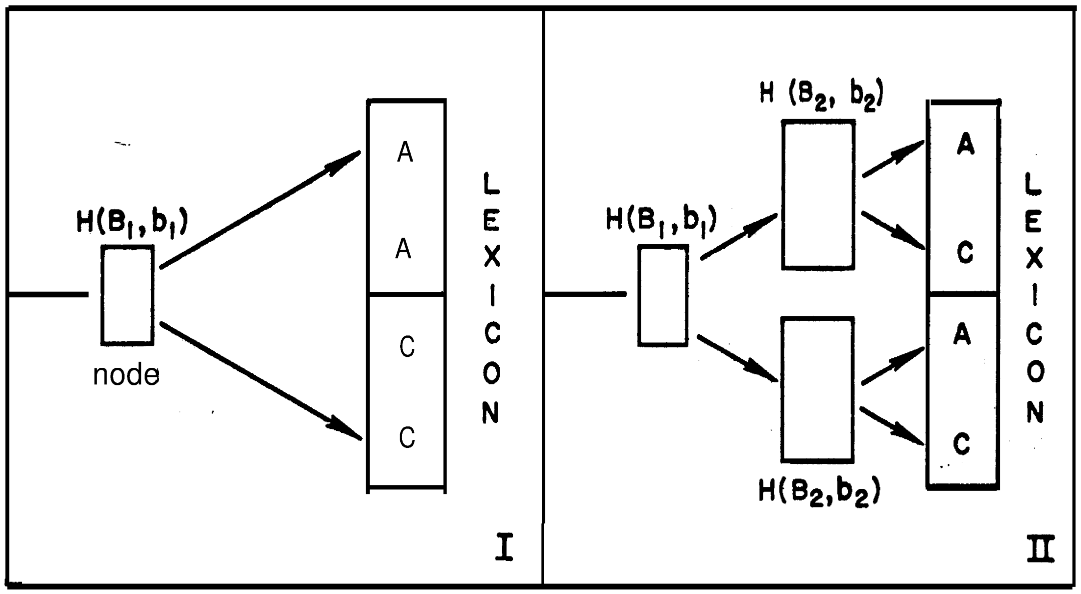

Fig. 2. Tree structures and lexicons used to separate patterns A and C as shown in Figure 1.

The point of the illustrations in Figure 1 and Figure 2 is that the feature space representation of patterns directly determines the tree size. If we assume that **none of the n** pattern classes exist such that like class elements are nearest neighbors to each other in the feature **space, then** it is possible that the number of hyperplanes required for total discrimination could be as large as n-l. But n-l is an upper limit.

Example: A lexicon of 1000 utterances containing only 22 different words could require as many as 999 hyperplanes. On the other extreme, it could achieve total discrimination with only 21 hyperplanes.

In general, the maximum number of signature tables possible is x where:

1) x is one less than the number of samples in the lexicon, or

2) $x = 2^b$ where b is the number of bits used to represent a point in feature space.

Case 1 applies if the number of samples is less than 2^b; otherwise case 2 applies. For example, if there are 10 feature values, each ranging between 0 and 7, thus using 3 bits, then the number of bits, b, used to represent a point in feature space is 30. If there are 10^4 sample patterns in the lexicon then case 1 applies because $2^{30} > 10^4$.

Conjecture: The effectiveness of the feature space in classifying patterns is reflected in the size of its optimal signature trees: the smaller the trees, the better the space.

This can be understood as follows. Let the total volume of feature space be partitioned first into n subvolumes and then into m subvolumes where m> n. On the average, a randomly located feature point in the m space will be closer to a partition than a randomly located feature point in n space. Thus any "noise", fluctuations, in the location of the feature point is more likely to carry it across a partition boundary in m space than in n space. If the subvolumes are created by the hyperplanes of the signature tree, then the larger subvolumes created by smaller trees are clearly less susceptible to error producing noise. This fact gives rise to the following heuristic:

The primary goal of the signature tree method is to guide the construction of a feature space that minimizes the tree size, and, if possible, produces a tree with only M-1 nodes where M is the number of pattern classes.

3. DATA BASES AND LEXICONS

The data base is a computer file containing relatively unrefined data compared to a lexicon which is a more compact version of the same file. The lexicon exists explicitly for the purpose of identifying patterns and is usually constructed after some sort of preparatory analysis of the data base.

3.1 Data Base: In the research on speech reported here, the data base is a one-dimensional array containing M utterances, for a total of 25*M words of array storage.

The first word in each 25 word block is reserved for a pattern label which is identification number or symbol. (This is sometimes called the "key" by other writers.)

The next 16 words contain digitized amplitude and zero crossing values for an utterance.(3) Amplitude and zero crossings are recorded every 10 milliseconds with averaging over the 10 millisecond intervals by a hardware preprocessor. A typical utterance is "How are you?" and lasts about one second. However, most utterances can still be recognized when averaging is increased to cover much longer time intervals. For instance, an entire utterance regardless of its length can be scaled to fit into 32 time units for our work with no apparent loss of key features. There are three zero crossing and three amplitude measurements per time unit, each using three bits, for a total of*18 bits. PDP-10 computer words have 36 bits. Consequently, the entire utterance is stored in 16 computer words.

The remaining 8 words in each 25 word block are used for storage of global features of the utterance. Global features are measured as required during growth of a signature tree and stored within a byte field of one of the 8 words.

3.2 Lexicons: The lexicon used for identification of utterances could be the entire data base, or it could consist of the utterance label and one or more of the eight words of global feature values; or it could be nothing more than the utterance label. The reason that the lexicon can be nothing more than pattern labels is that the information for discrimination is contained in the signature tree structure. Significant savings in storage can result from using a lexicon of utterance labels and the appropriate signature tree, in place of the full data base. For example, let there be a list of 22 words each spoken 4 times for a total of 88 utterances. Let n be the number of data base words per pattern and M be the number of classes. In this case M = 22. Let the mean number of sample patterns per class be m. In this example, m = **4.** It is possible to reduce the storage requirements from n*m*m for the full data base to **(M-1)*2+M.** The (M-1)*2 term comes from the fact that each of the (M-1) nodes in the signature tree uses <u>two</u> computer words. The M term is the number of computer words required by the lexicon of utterance labels. The storage reduction is **M(n*m-3)+2.** If we set m = 25, then the storage requirement could be reduced from 2200 to 64, which is impressive. Achieving this maximum reduction depends upon the feature extractors and the hyperplanes they generate. But the maximum reduction noted in this example is frequently attained in the applications reported in section 6.

The savings-in storage is one of the most interesting and potentially useful aspects of the signature tree method especially for large vocabularies and/or a large number of different speakers.

4. FEATURE EXTRACTORS

A feature is defined to be anything in a pattern that can be measured. A feature extractor is a subprogram that measures a feature and returns an integer value. (The restriction to integer feature values is not a universal convention; it is used here because the feature values are used for array indexing more than anything else.)

For example, let the pattern be a two-channel time series. The features could include: number of maxima and minima; slopes greater or smaller than **X**; areas under curves; ratios of areas; distances between adjacent minima; second derivatives; moving averages; fourier power spectra averaged over a given frequency range: autocorrelations; cross correlations; the kitchen sink. Feature extractors would return scaled integer values for each of these features.

5. SAMUEL'S SIGNATURE TABLES

The signature table technique is a perceptron-like[2] pattern classification method. It is based on a sequence of table look-up operations where each table is a multi-dimensional array. An individual signature table can be viewed as an n dimensional hyperplane in an m dimensional feature space, where n is **equal** to the number of input parameters to the table and m is the total number of parameters available to all the tables. The array indexes for any given table are specified by previous tables or by feature measurements performed directly on a pattern.

Samuel uses signature tables to evaluate the relative merit of various board positions in the game of checkers. A small set of board parameters, e.g., 12, are used as input variables. A typical arrangement might have three first level tables, each table having four input parameters, with each input parameter coming from a board feature measurement (such as "piece count"). Each level one table would contain integer values that would serve to index one or more of the tables at the next level. At "level two", one or two tables would operate in the same way as level one tables except that their input would come from level one tables. Their output would index yet another table which would contain the **"score"**, where the score gives the relative value of the board pattern being examined.

Initially, the signature tables are empty. They are filled during the **"learning"** stages of the program. The data used in learning can be discarded once it has been processed since the number of times a pattern projects onto a hyperplane is recorded in a cell on the hyperplane at the point of incidence. For this method to have any hope of success, the different training patterns projected on a hyperplane must tend to fall on different portions of the hyperplane. Otherwise, when an unidentified pattern falls on a point on a hyperplane, no information could be gained to help identify **it.** When the user does not possess a preconceived notion for the appropriate hyperplanes, and he usually does not, then the real problem becomes one of finding appropriate groupings of input parameters, It was precisely this problem that led to the signature <u>tree</u> system. The signature table method provides no means of automatically connecting individual tables dynamically. And not having this flexibility severely limits the application of the technique to research problems. This shortcoming is eliminated in the signature <u>tree</u> technique developed in this paper.

6. APPLICATION TO SPEECH HECOGNITION

Moderate success has been attained in the application of the signature tree method to speech recognition. An accuracy of around 90% was obtained for the twenty-two different utterances used in the training runs. For such a short list of utterances this accuracy is not particularly good, by itself; but it becomes interesting when the speed, "learning" properties, and flexibility of the program are considered.

Speed: The through-put identification rate is five to ten times faster than the excellent programs of **Astrahan**(3); and **Vicens** and **Reddy.**(4) (However, their programs identify word lists three to five times longer and with higher accuracy than this one. If the present program were used on the longer lists, its speed advantage would be somewhat diminished.)

Learning: An improvement in the correct identification rate of nearly 40% is observed as the number of examples of each utterance increases from one to four. This is probably the most outstanding success of the program.

Flexibility: New hypotheses and changes in data base formats can be entered in only a few minutes, the time it takes to type in a subroutine. This should increase the fabrication rate of good feature extractors and ultimately lead to superior speech recognition.

A few of the SAIL procedures used to assist in feature extraction are as follows (their names give the flavor of their intended function):

TOTAL ENERGI_PER_TIME UNIT, FIND MAJOR **CHANGES_IN_TOTAL_ENERGY, ALL_CH_ARE_ZERO, ALL_CH_HAVE_ENERGY, FIND_MINIMUM_IN_ENERGI**_PER-CH, MOVING AVERAGE, **ENERGI_BETWEEN** MINIMA, TOTAL_E_AROUND-MAX-PEAK, **FIND_RANGE_OF**_VOWEL, **COUNT_MAXIMA,** FORM-CHANNEL-RATIOS, COMPACT-UTTERANCE, and GENERATE-TEMPLATE.

The entire program is written in PDP-10 SAIL. In the actual runs, to avoid disc storage problems, the entire data base of 10 speakers was seldom used; most experiments were done with the five pronunciations of the utterance list by Lee Erman. This required **DATA_BASE** core array storage of 2750 thirty-six bit computer words.

A typical machine "learning" experiment is carried out as follows: The first 550 **DATA_BASE** words, which contains the first 22 utterances, is processed. Various feature extractors are applied in the order determined by the user through a list prepared with the STOPGAP text editor. When a feature value is found that serves to separate one or more of the 22 utterances from the rest, then a tree node is created and the number of the feature extractor is stored in the node: Also stored in the node are pointers to the top and the bottom of the **DATA_BASE sublists** created by the feature value (see Figure 3).

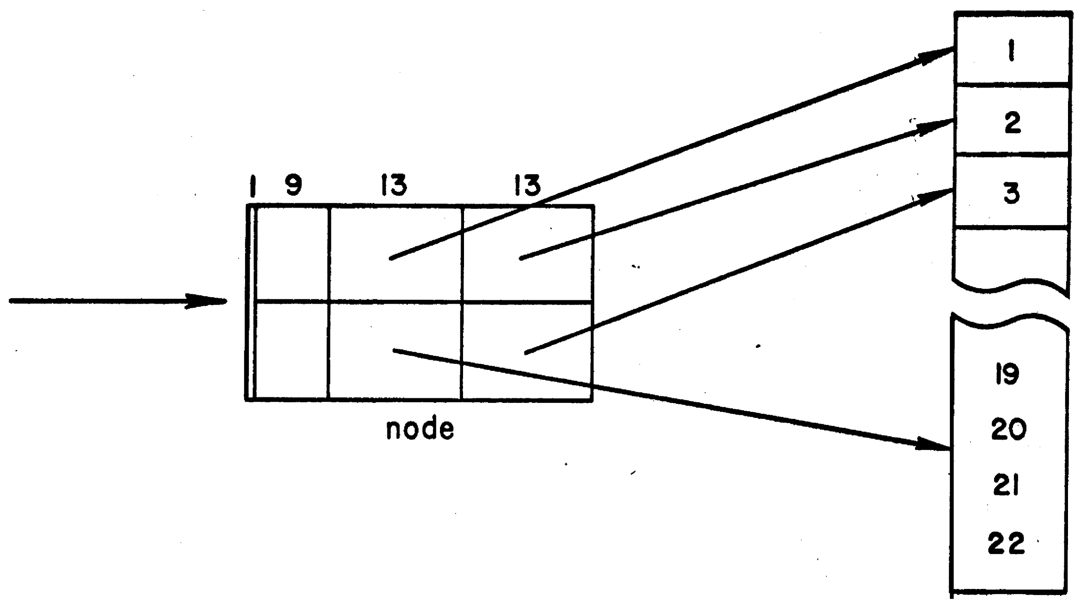

Fig. 3. Terminal tree node that separates utterances one and two from utterances three through twenty-two.

Since there is only one data point for each utterance class, finding hyperplanes that lay between different utterance classes is virtually assured. In practice, the resulting signature trees nearly always have only 21 nodes (as would be predicted). When tested on utterances 89 through 110, which is the fifth set stored from 2225 through 2750 in DATA BASE, the scores are usually 50% correct. When the system is further trained with utterances 1 through 44, a 21 node tree usually results and its scores are centered on 68% correct when tested on utterances 89 through 110. Learning on utterances 1 through 66 gives 77% correct, and learning on utterances 1 through 88 gives 86% correct. This is always accomplished with fewer than 30 nodes per tree.

Plotted in Figure 4 are average scores for the tree method and the absolute minimum error method.'

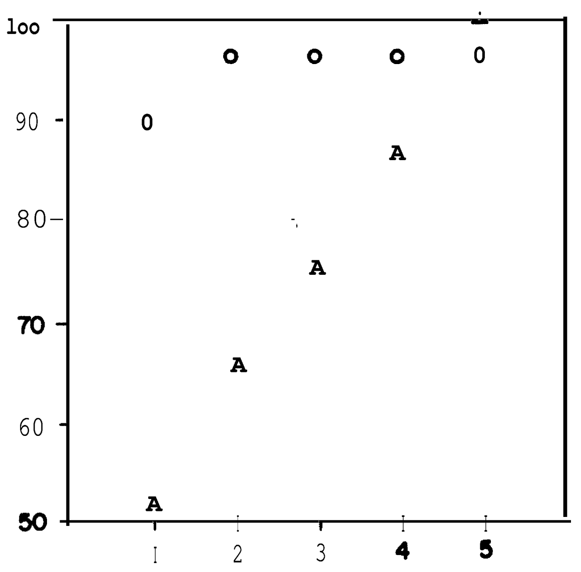

Fig. 4. Average percent correct plotted vs. the number of training samples for each utterance. Absolute minimum errors are "O"'s; trees are" "',.

The 100% score for the tree method on sample set 5 is a consequence of the fact that the same utterances were used in training as were used in identifying the utterances.

As a point of comparison, the mean feature values were tabulated for the first 22 utterances determined, and utterances 89 through 110 were then processed as unknown utterances, and predictions were made using the least absolute error criterion. Scores ranges from 85% to 95% for several sets of feature extractors. Similarly mean values were tabulated for 44, 66, 88 and 110 utterances and then tested on utterances 89 through 110. The result is shown in Figure 4. The minimum absolute error approach was more accurate simply because it incorporated more redundancy. But the speed advantage of the tree method stood out clearly; the minimum absolute error approach took between 20 and 30 times longer in the identification phase than the tree method did.

The values plotted in Figure 4 are averages obtained from ordering the trial feature extractors differently and/or omitting certain feature extractors. However, these results represent optimal overall performance combinations of features.

Table 1 is computer output which shows the 22 utterances given in the data base along with response of the program for a particular run. The program correctly identified 19 of the 22 utterances, it used a tree with 22 nodes (signature tables); 88 utterances were used in growing the trees (four examples of each utterance); a maximum of 400 hyperplanes were available to the program from which it selected 22 to grow the tree. The four columns of numbers show which utterances were put in the same hypercell by the tree hyperplanes. For example, the cell that is pointed to by the key features of utterance 89 in the data base, which happens to be the "make" in column one, is a cell containing four **l's.** Consequently, utterance **"l"**, which is the numeric identifier for "make", is predicted and "make" is written in column two.

If the cell contained more than one utterance class, e.g. 1 1 1 2, then the utterance which occurs the greatest number of times is predicted; in this example, utterance 1 would be predicted. However, a cell with more than one utterance class will occur only when there are no hyperplanes to separate the classes. In the case shown in Table 1, all the cells (there are 23 of them) contain only one utterance class each. Furthermore, only utterance 22 failed to lie totally in one cell.

Table1

```
                 THE  TOTAL  NUMBER OF SIGNATURE TABLES  =22

  THE TREE  STRUCTURE AND DATA BASE  ARE  FROM FILE TREE0
  THE  NUMBER  OF  UTTERENCES  IN THE TRAINING SAMPLE=88
  THE  BEGINNING  UTTERENCE NUMBER IS 89
  THE  ENDING UTTERENCE  NUMBER  IS 110
  THE  MAXIMUM  DEPTH PERMITTED IN TRAINING  =  400

 GIVEN UTTERENCE        UT  PREDICTED
    KAKE                    MAKE                  1    1    1    1

    UNITE                   UNITE                 2    2    2    2

    DELETE                  DELETE                3    3    3    3

    ONE                     ONE                   4    4    4    4

   EIGHT                    EIGHT                 5    5    5    5

    THREE                   GELETE                3    3    3    3

    WHOLE                   WHOLE                 7    7    7    7

    GCTAL                   OCTAL                 8    8    8    8

    EXCHANGE                EXCHANGE              9    9    9    9

    CORE                    CURE                 10   18   10   10

    MULTIPLY                MULTIPLY             11   11   11   11

    SIX                     SIX                  12   12   12   12

    SUBTRACT                UNITE                 2    2    2    2

    SCALE                   MAKE                  1    1    1    1

    DIRECTIVE               DIRECTIVE            15   15   15   15

    OUTPUT                  OUTPUT               16   16   16   16

    INTERSECT               INTERSECT            17   17   17   17

    REGISTER                REGISTER             18   18   18   18

    FINE,  THANK  YOU       FINE,  THANK  YOU    19   19   19   19

    GOOD  MORNING           GOOD  MORNING        20   20   20   20

    HOW  ARE  YOU           HOW  ARE  YOU        21   21   21   21

    EXCUSE  ME              EXCUSE  ME           22   22   22
```

A tree that contains only one utterance class per cell will correctly identify all utterances used in the training set. If the tree of this run had been used to identify utterances 1 through 110, it would have missed only 3 utterances, the same number it missed on utterances 89 through 110.

Great effort was expended optimizing the speed of the program and with good success. For example, to grow the tree for this run, which required testing some 400 hyperplanes and 88 utterances, required only one minute PDP-10 compute time. The identification of the twenty-two utterances, 89 through 110, required about 10 seconds.

Many variations on this basic scheme were tried; one in particular yielded 91% correct prediction scores. The trick here was simply to simultaneously employ three or more signature trees grown on the same training data but forced to use different hyperplanes. This provided something of a "redundancy" check; it helped to eliminate the effects of "noisy" measurements.

Figure 5 shows a typical utterance that has been compacted for storage in the DATA_BASE. There are six variables A_1, Z_1, A_2, Z_2, A_3, Z_3, each of which can assume eight values between 0 and 7.

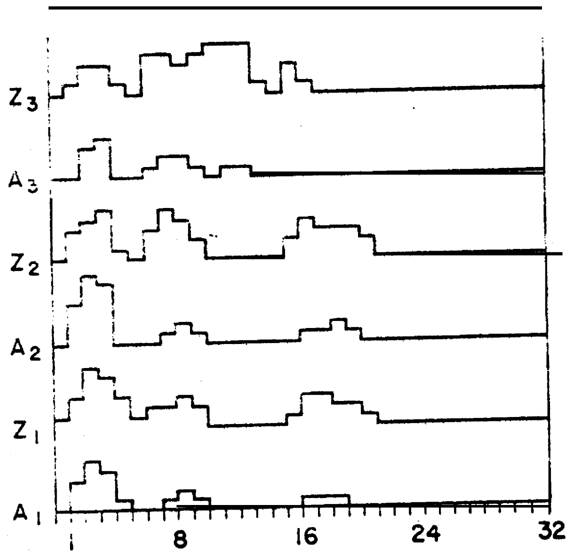

Fig. 5. Amplitude and zero crossing versus time for each of the three speech preprocessing filters for the utterance "register".

No normalization was attempted. The original array which typically contained **x=200** time units was reduced to a 32 time unit array by averaging over x/32 + 1 time units at once and storing the result in a single time unit and repeating 32 times. Compaction of the speech data in this fashion, as crude as it is, preserved enough information for interword discrimination.

CONCLUSIONS

When the number of labeled patterns available to train an automatic recognition system is small, the predictive accuracy of a signature tree system is poor; a minimum error criterion using as many features as possible can be much more accurate. When enough data is available to establish probability distributions of feature values for pattern classes, the signature tree method allows a large increase in speed of the recognition system over the minimum error approach with little degradation of accuracy.

The signature tree method selects a minimal set of feature extractors from a large set supplied by the user. In this fashion, the man-machine unit "learns" which features are the most useful in pattern discrimination. For particular types of patterns, e.g., those arising in speech or vision research, general "front end" feature extractors will probably be discovered that will greatly increase the overall power of the signature tree method.

Presently under development are two major additions to the signature tree heuristics repertoire: a linear discriminant preprocessor which forms linear combinations of feature extractors,and a technique for copying noisy patterns into both of the sublists created by a feature value. Forming linear combinations of input parameters increases the number of hyperplanes available for creating tree nodes. As the system stands, it only uses hyperplanes perpendicular to the coordinate axes. Linear combinations of features will create "**skew**" hyperplanes. The other addition, that of duplicating patterns so to make them appear on both branches from a node, is expected to greatly reduce errors resulting from "noise" (random fluctuations) in parameters.

The spectacular learning rate (a 40% increase in the number of correct identifications in the course of increasing the number of exemplary samples per utterances from one to four) is somewhat artificial. By utilizing a combination of the minimum error criterion and the signature tree technique (instead of simply using the signature tree), a much higher score could have been obtained at the outset, thus lessening the range for possible improvement. In this case, a shift in emphasis from score optimization to speed optimization would be appropriate. Using the minimum error criterion slows the identification rate significantly but improves accuracy. Increasing the number of samples in the training set allows the signature trees to function more accurately, thus allowing the trees to replace the minimum error criterion, which results in an overall increase in speed. So, the "learning" aspect of the signature tree method can be applied to optimize speed or accuracy depending on how it is used.

Automatic Control Systems[5] theory is closely related to the signature tree method. In particular, the control systems field has produced a number of algorithms for generating "decision surfaces" in hyperdimensional feature spaces, e.g. linear, polynomial and statistical discriminant functions. While the signature tree heuristics presented here offer little help in generating binary discriminant functions, control theory offers little help in combining a large number of binary decisions. However, an ideal total system is possible with the binary discriminant functions serving to supply trial hyperplanes to a signature tree generator. A symbiotic union of automatic control systems with signature trees seems likely and is being investigated by the author.

ACKNOWLEDGEMENTS

The author would like to thank A. L. Samuel for encouragement, guidance and inspiration for this work on signature trees. Enlightening conversation about automatic speech recognition with Raj Reddy is also greatly appreciated. Gratitude is expressed to John McCarthy and Les Earnest for providing a free and stimulating environment for this research.

BIBLIOGRAPHY

1. Samuel, A. L., Some Studies in Machine Learning Using the Game of Checkers. II - Recent Progress, IBM J of R&D. 3, #6, 601-617, Nov. 1967.

2. Minsky, Marvin and Seymour Papert, Perceptions, The M.I.T. Press, 1969.

3. Astrahan, M. M., Speech Analysis by Clustering or the Hyperphoneme Method, A. I. Memo 124, Stanford Artificial Intelligence Project.

4. Vicens, P., Aspects of Speech Recognition by Computer, A. I. Memo 85, Stanford Artificial Intelligence Project.

5. Fu, Kinq-sun, "Learning Control Systems - Review and Outlook", IEEE TRANS. Automatic Control, Vol. AC-15, No. 2, p. 210-221, April 1970.