Sequential feature extraction for waveform recognition

by W. J. STEINGRANDT and S. S. YAU

Northwestern University
Evanston, Illinois

INTRODUCTION

Many practical waveform recognition problems involve a sequential structure in time. One obvious example is speech. The information in speech can be assumed to be transmitted sequentially through a phonetic structure. Other examples are seismograms, radar signals, or television signals. We will take advantage of this sequential structure to develop a means of feature extraction and recognition for waveforms. The results will be applied to speech recognition.

An unsupervised learning (or clustering) algorithm will be applied as a form of data reduction for waveform recognition. This technique will be called sequential feature extraction. The use of sequential feature extraction allows us to represent a given waveform as a sequence of symbols $a_1, \ldots, a_k$ from a finite set $A = \{a_1, \ldots, a_M\}$. This method of data reduction has the advantage of preserving the sequential structure of the waveform. The problem of waveform recognition can be transformed into a vector recognition problem by expanding the waveform using orthogonal functions. However, in this case the sequential structure is masked because the expansion operates on the waveform as a whole. Data reduction can also be carried out by time sampling and storing the samples as a vector. In this case the dimension of the vector is usually large. The data produced by sequential feature extraction is more compact. We will formalize the concept of sequential feature extraction and develop a performance criterion for the resulting structure. An unsupervised learning algorithm, which will optimize this structure with respect to the performance criterion, is presented. This algorithm, which can be applied to waveform recognition as well as vector recognition, represents an improvement over existing clustering algorithms in many respects. This method will allow unbounded strings of sample patterns for learning. The samples are presented to the algorithm one at a time so that the storage of large numbers of patterns is unnecessary.

The assumption of known probability measures is extremely difficult to justify in most practical cases. This assumption has been made in a number of papers, but no such assumption is made here. That is, the requirement for convergence is only that the measures be smooth in some sense. Braverman’s algorithm has been shown to have these advantages. However, he assumes that there are only two clusters, which, after a suitable transformation, can be strictly separated by a hyperplane. These assumptions are too restrictive for the practical applications considered in this work. In the clustering algorithm to be presented here, any number of clusters is allowed, the form of the separating surfaces is not as restricted, and strict separability of the clusters is not assumed. This algorithm is considerably more general than existing clustering algorithms in that it applies to time varying as well as time invariant patterns.

We will assume that the waveform is vector valued, i.e., $x(t)$ is in a set $\Omega = \{x(t) \mid \| \dot{x}(t) \| < M, \text{ all } t \in [0, T_x]\}$, where $\dot{x}(t)$ is the componentwise time derivative of $x(t)$. It is assumed that each pattern class has some unknown probability measure on this set.

A unified model for waveform recognition and vector recognition will be presented. It will be shown that the recognition of a vector pattern can be considered as a special case of waveform recognition. This will be done by observing that the pattern space of $n$-vectors $v$ is isomorphic to the space of all constant functions $x(t) = v$.

Recognition of real functions of time will be possible by defining a transformation to the space $\Omega$ or by assuming that $x(t)$ is one-dimensional. The problem of waveform recognition will be carried out in the space $\Omega$, where the dimension of $x(t)$ is most likely greater than one.

The experiments on speech will show an interesting relationship between the sequential features and the
standard linguistic phonetic structure for English. A recognition algorithm using sequential machines will be given that will accept symbol strings \( a_{v1}, \ldots, a_{v3} \) to classify spoken words.

### SEQUENTIAL FEATURE EXTRACTION

Figure 1 shows the process that is assumed to produce the vector waveform \( x(t) \). It is emphasized that this model may not represent an actual physical process as described. It is included as a means of demonstrating the assumptions about the sequential structure on \( \Omega \). In the figure it is assumed that there is some state of nature or intelligence such that pattern class \( i \) is present. The pattern classes are represented by the symbols \( u_i, i = 1, \ldots, R \). There exists a second set of symbols \( A = \{ a_{1}, \ldots, a_{M} \} \) called the *phoneme set*. Each \( a_i \) is called a *phoneme* (while the terminology is suggestive of speech and language, there may be little relation to the speech recognition problem). The second step converts \( u_i \) into a finite sequence of phonemes \( a_{v1}, \ldots, a_{v7} \), where \( v_i \) is the index of the \( i \)th phoneme in the sequence. The process of encoding \( u_i \) into \( a_{v1}, \ldots, a_{v7} \) is most likely unknown and is probably nondeterministic. That is, the sequence generated by a given \( u_i \) may not be unique.

Each sequence is then assumed to go through an encoding process into a real waveform \( w(t) \in W \), where \( W \) is the set of all continuously differentiable real waveforms such that \( w(t) \) and the time duration are bounded. This process is also most likely nondeterministic. For the most part, this encoding process is unknown; but some assumptions can be made. It is assumed that there is some unique behavior of \( w(t) \) for each \( a_i \). As each \( a_{v_i} \) from \( a_{v1}, \ldots, a_{v7} \) is applied to the encoder, the behavior of \( w(t) \) changes in some manner. This behavior is detected by using a transformation to a vector function of time \( x(t) \in \Omega \). This transformation can be considered to be described by some differential equation of the form

\[
\dot{x}(t) = f[x(t), w(t)],
\]

where \( f: \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n \) is a bounded continuous function. The explicit form for this equation may not be known, but the system that it describes is assumed to be determined from the physical process producing \( w(t) \). If this differential equation is properly chosen, then the value of \( x(t) \) at any time \( t \) is some pertinent measure of the recent past behavior of \( w(t) \).

We will shortly present a clustering algorithm on \( \Omega \) which is a generalization of the usual concept of clustering. Clustering for the time invariant case will first be reviewed. It is assumed that there exists a metric \( \rho \) that measures the similarity between patterns, where the patterns are assumed to be fixed points in \( \mathbb{R}^n \). \( \rho \) is such that the average intra-class distance is small, while the average inter-class distance is large. The method of cluster centers used by Ball and Hall is used to detect the clusters. It is assumed that the number of clusters is fixed, say at \( M \), and there are \( s_i \in \mathbb{R}^n, i = 1, \ldots, M \), such that each \( s_i \) has minimum mean distance to the points in its respective cluster. These \( s_i \) can be found by minimizing the performance criterion

\[
E_x \min_i \rho(s_i, x),
\]

where the expectation is with respect to the probability measure on \( \mathbb{R}^n \).

These assumptions will now be generalized for patterns that are time varying. Here the phonemes \( a_i \) play the part of the pattern class for the time invariant case. That is, the time invariant pattern classes are assumed to be the same as the time varying case except that the phoneme sequence producing the vector is always of length one, and \( x(t) \) is the constant function.

We will describe the general case in more detail. Here, as before, it is assumed that there is a similarity metric \( \rho \) on \( \mathbb{R}^n \). This metric measures the similarity of the behavior of \( w(t) \) at any given time \( t \) to that at any other time \( t_2 \). This is done by measuring the distance \( \rho[x(t_1), x(t_2)] \), where it is understood that \( x(t) \) and \( w(t) \) satisfy (1). The assumption is that (1) and \( \rho \) are such that if \( a_i \) was applied to the waveform encoder both at time \( t_1 \) and \( t_2 \), then \( \rho[x(t_1), x(t_2)] \) is small. On the other hand, if \( a_i \) was applied during \( t_1 \) and \( a_j \) during \( t_2 \), then \( \rho[x(t_1), x(t_2)] \) is large for \( i \neq j \).

In other words, each \( a_i \) produces behavior in \( w(t) \) such that the corresponding values for \( x(t) \) tend to cluster in distinct regions of \( \mathbb{R}^n \). Thus, the \( a_i \) are represented by clusters in \( \mathbb{R}^n \). It is assumed that each \( a_i \) has a cluster center \( s_i \) associated with it. This implies that for each \( a_i \) there is a point \( s_i \in \mathbb{R}^n \) such that when \( a_i \) is applied to the waveform encoder, the function \( x(t) \) tends to pass close to \( s_i \).

It will also be assumed that \( x(t) \) spends most of its time in those regions that are close to the \( s_i \). In other words, the more important features of \( w(t) \) are of longer duration. The example shown in Figure 2 illustrates the foregoing assumptions. The figure shows the action of \( x(t) \) under the application of \( a_1, a_2, a_3 \) to the

![Figure 1-Assumed process producing pattern waveforms](image-url)
encoder. In the figure the width of the path is inversely proportional to $||\hat{x}(t)||$.

This model is necessarily somewhat vague because we are unwilling to make assumptions about the probability measures on $\Omega$. If such assumptions were made, then a more formal definition of a cluster might be possible. For most practical problems such as speech recognition, these types of assumptions cannot be made.

Assuming $\rho$ and the $s_i$ were known, they could be used to reconstruct an estimate of the sequence $a_1, \ldots, a_k$ for an unknown waveform $x(t)$ in the following manner. Referring to Figure 3, each of the quantities $\rho[s_i, x(t)]$, $i = 1, \ldots, M$ are continuously calculated and the minimum continuously indicated. That is, suppose there exist times $t_i = 0, t_2, \ldots, t_k + 1 = T_x$ such that $\rho[s_i, x(t)] \leq \rho[s_j, x(t)]$ for all $j \neq i$ and all $t \in [t_i, t_{i+1}]$, then it is assumed that the phoneme sequence most likely to have produced $x(t)$ is $a_1, \ldots, a_k$.

Note that no adjacent phonemes in the sequence are ever the same. It is also apparent that the output sequence is independent of time scale changes in $x(t)$.

If $\rho$ and (1) are fixed, then for a given set of the $s_i$, $i = 1, \ldots, M$ there is a transformation defined by Figure 3. This transformation will be called $T_x: \Omega \rightarrow P$, where $P$ is the set of all finite sequences of symbols from $A$, $s = (s_1', \ldots, s_M')'$ and the prime of a matrix denotes its transpose.

The pair $(A, T_x)$ defines a sequential structure on $\Omega$. This sequential structure is extracted by the transformation $T_x$ defined in Fig. 3. Thus, the terminology sequential feature extraction has been used.

This definition of sequential feature extraction is unique in that it puts sequential structures in waveform recognition on a more formal basis. Gazdag has suggested a somewhat similar structure in what he calls machine events. His method involves linear discriminant functions, and he gives no method for determination of the structure.

The objective of the learning algorithm will be to determine $(A, T_x)$ by determining the composite vector $s$. The differential equation in (1) and $\rho$ are assumed to be determined from a study of the physical process producing $w(t)$. It is obvious that any random choice for $s$ will define a sequential structure. The learning algorithm will be required to find that $s$ which is optimum with respect to some performance function. This performance function is generalized from that mentioned previously for time invariant patterns.

Based on the previous discussion, the performance function for this case is

$$E_x C(s, x) = E_x \left( \frac{1}{T_x} \int_0^{T_x} \left[ \min_i \rho[s_i, x(t)] \right] dt \right),$$

where $C(s, x)$ is a function called the confidence function for a given waveform $x(t)$. The smaller $C(s, x)$ is for a given $x(t)$, the more confidence, on the average, can be placed in the resulting sequence of phonemes. Taking the statistical expectation over the entire population $\Omega$ gives us the performance function.

The object of the learning rule will be to find an $s^*$ such that $E_x C(s^*, x)$ is at least a local minimum for
It is obvious from (2) that direct evaluation of the performance function is not possible because the probability measures are not known. Using stochastic approximation, it can be shown that if a learning rule of the form

\[ s^{n+1} = s^n - a_n \nabla C(s^n, x^n) \]  

(3)
is used, then under certain conditions the sequence \([s^n]\) converges almost surely to a saddle point or local optimum \(s^*\), where \(s^n\) is the value for \(s\) at the \(n\)th stage of learning, \(x^n\) is the \(n\)th sample waveform, and \(a_n\) is a sequence of scalars satisfying certain convergence conditions. Note that \(x^n\) is unlabeled, i.e., no pattern class information is used in the learning rule.

It can easily be seen that if \(x(t) = v\), and \(T_x = 1\), then the performance function in (2) reduces to that for the time invariant case.

We are now in a position to calculate \(\nabla C(s, x)\) for a given pattern \(x(t)\).

Define

\[ A(S_i) = \{ x \in R^n | p(s_i, x) < p(s_j, x), \quad \text{all}\ j \neq i \} \]  

(4)

Each region \(A(S_i)\) corresponds to a phoneme \(a_i\). For each \(x(t)\), the sequence \(a_1, \ldots, a_k\) is simply a list of the regions \(A(S_i)\) through which \(x(t)\) passes. The \(t_1, \ldots, t_k\) are then the times at which \(x(t)\) passes from one region to the next. Using this, we can write

\[ C(s, x) = \sum_{i=1}^{k} \int_{t_i}^{t_{i+1}} \rho[x(t)] dt \]  

(5)

Taking the gradient and canceling terms we have

\[ \nabla_x C(s, x) = \sum_{i=1}^{k} \int_{t_i}^{t_{i+1}} \nabla_x \rho[x(t)] dt \]  

(6)

where \(\nabla_x\) is the gradient with respect to \(s_j\). It is also understood that the integral of a vector function is meant to be the vector of integrals of each of the individual components. The learning rule in (3) becomes

\[ s_{n+1} = s_n - a_n \sum_{i=1}^{k} \int_{t_i}^{t_{i+1}} \nabla_v \rho[s_{n+1}, x_n(t)] dt \]  

(7)

where

\[ \sum_{n=1}^{\infty} a_n = \infty, \quad \sum_{n=1}^{\infty} a_n^2 < \infty, \]  

(8)

\(x_n(t)\) is the \(n\)th sample waveform, and \(s_n^*\) is the value of \(s_j\) at the \(n\)th step of learning. An equivalent form is

\[ s_{j+n+1} = s_{j+n} - a_n \int_{t_i}^{t_{i+1}} x_i[x_n(t)] \nabla_v \rho[s_{j+n}, x_n(t)] dt, \]  

(9)

where \(x_i\) is the characteristic function of \(A(S_i)\).

\textbf{Example 1} Assume that \(\rho\) is the squared euclidean metric, i.e.,

\[ \rho(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2. \]

The learning rule in this case becomes

\[ s_{j+n+1} = s_{j+n} - a_n \frac{1}{T_x} \int_{0}^{T_x} x_i[x_n(t)] [s_{j+n} - x_n(t)] dt. \]  

(10)

\textbf{AUTOMATIC SPEECH RECOGNITION}

The automatic recognition of speech has received much attention since the advent of the digital computer. Most of the previous work in speech recognition has made use of the phonetic structure of speech. Almost all of these studies use the standard linguistic phonetic structure. Here we investigate the applicability of sequential feature extraction to the speech recognition problem. A sequential structure will be developed using a limited vocabulary. It will be seen that the resulting structure is related to the standard English phonetic structure. Because of this relationship to speech, we will refer to sequential feature extraction as a \textit{machine phonetic structure}.

In order to represent the speech waveform \(w(t)\) as a vector function of time we will use the common method of a bank of bandpass filters. In the experiments 50 filters were spaced from 250 to 7000 Hz. Each filter was envelope detected and sampled by an A/D converter and multiplexer. Therefore, \(x(t)\) is a 50 dimensional vector function of time.

Kabriskyn\textsuperscript{18} has shown that a neuron network similar to that found in the brain is capable of performing correlation calculations. Based on this we assume that the similarity metric defined by

\[ \rho(x, y) = \left(1 - \frac{x^T \cdot y}{\|x\| \cdot \|y\|}\right) = \frac{1}{2} \left( \frac{x}{\|x\|} - \frac{y}{\|y\|}\right)^2 \]  

(11)
is valid for speech sounds. Note that \(\rho(ax, by) = \rho(x, y)\) for all \(a, b, x, y\), i.e., the metric \(\rho\) is invariant to amplitude changes in the signal. Using this metric we have the following learning rule.

\[ s_{j+n+1} = s_{j+n} - a_n \Delta_i, \]  

(12)

where

\[ \Delta_i = \frac{1}{\|s_{j+n}\|^2} \left( \frac{1}{\|s_{j+n}\|^2} \|s_{j+n}^* - s_{j+n}\| - 1 \right) \frac{1}{T_x} \int_{0}^{T_x} x_i[x_n(t)] x_n(t) dt \]  

(13)
where

\[ \mathbf{x}_n(t) = \frac{1}{\| \mathbf{x}_n(t) \|^2} \mathbf{x}_n(t), \]  

(14)

and \( I \) is the \( n \times n \) identity matrix.

If we normalize \( \mathbf{x}(t) \) as part of the preprocessing and normalize each \( s_i \) after each step of learning, then we can write the learning rule as

\[ s_i^{n+1} = s_i^n - a_n [s_i^n s_i^n - I] T_{n-1} \int_0^{T_n} \mathbf{x}(t) \mathbf{x}(t) \, dt \]  

(15)

This rule was used to develop the phonetic structure presented in the next section on the experimental results.

**MACHINE PHONETIC STRUCTURE**  
**EXPERIMENTAL RESULTS**

This section describes the results of experiments using the data acquisition equipment previously described. The basic goals of the experiments were:

1. Test convergence of the algorithm  
2. Determine effects of local optimums  
3. Provide output for use in speech recognition  
4. Determine relationship to the standard linguistic phonetic structure, if any.

There were two sets of data used for the tests. One set consisted of 50 utterances each of the words "one", "four", "oaf", "fern", "were". These words were chosen because they contained a small number of sounds with an unvoiced as well as voiced sounds. One speaker was used for all utterances. It was found that the speaker's voice had enough variation to adequately test the algorithm. If the algorithm had been tested with many speakers, the variance would have been much larger. This would have lengthened the convergence times beyond what was necessary for a sufficient test.

The larger data set consisted of 40 utterances of each of the ten digits "one", "two", ..., "nine", "oh". These were all spoken by the same person. These words contain a wide variety of sounds: voiced, unvoiced, diphthongs, plosives, etc. This set was used to give a somewhat more severe test of convergence and to provide data for speech recognition. We will now consider the four goals of the experiments separately.

**Convergence:** Many runs with the small data set were made. Different starting points were chosen, and other conditions were varied. In all cases the algorithm showed a strong convergence.

Because there was only a finite set of samples, the convergence properties in (8) were academic. In order to better determine convergence, the sequence \( \{a_n\} \) was chosen to be constant over many steps of learning. If convergence was apparent under these conditions, then convergence under decreasing step increments can be assumed.

Figure 4 shows an example of the convergence of \( C(s, x) \) using the large data set. Due to the variance of the data, a direct plot of \( C(s, x) \) at each step of learning shows very little. The individual points for \( C(s, x) \) are so scattered that convergence is difficult to see. Figure 4 shows the plot after data smoothing. The solid curve represents averages of ten successive values of \( C(s, x) \). The dotted line represents further data smoothing. It can be seen that the performance function is not improved at each step but is improving over many samples. In order to demonstrate that the components of \( s^n \) were converging as well as \( C(s, x) \), the plot in Figure 5 was made. This is a plot of the tenth component of \( s_n \).
The Standard Phonetic Structure: The output strings from sample words were inspected for similarities to standard phonetic spellings. It was found that the two structures were similar in many respects. A one-to-one correspondence could be made between certain standard phonemes and machine phonemes. This was particularly true for consonants such as [s] or [ʃ].

Local Optimums: It was found that there definitely was more than one optimum. By choosing different starting points, the algorithm converged to different optimums. To see this, examine Figure 6. This is a plot of the smoothed data for two runs with the small data set. Each learning run was made with the same data except that the starting points were different. It can be seen from the figure that the initial point one converged to a local optimum that was not as good as that for the initial point two. We can be fairly certain that the first point will never converge to the second, since more than twice the number of learning steps were run for point one than for point two.

RECOGNITION OF PHONEME STRINGS

In this section we present a means of classifying the phoneme strings that are produced as a result of sequential feature extraction. For completeness, we shall restate the recognition problem here. There is a set of symbols $A = \{a_1, \cdots, a_M\}$ called phonemes. The pattern space $P$ is the set of all finite sequences of symbols from $A$. A typical pattern from $P$ will be denoted either by the sequence $a_{i_1}, \cdots, a_{i_t}$ or by $q$. There are $R$ pattern classes, each class has some characteristics associated with its sequences that differentiate it from the other classes.

If one were to use a Bayes decision procedure, the following would be needed. According to decision theory, in order to minimize the probability of error, the discriminant functions

$$g_i(q) = p(q \mid i) p(i), \quad i = 1, \cdots, R \quad (16)$$

are needed, where $q \in P$, $p(q \mid i)$ is the probability of $q$ given pattern class $i$, and $p(i)$ is the a priori probability of class $i$. If it can be assumed that the $p(i)$ are all equal, then they can be dropped from (16). The problem is then to estimate $p(q \mid i)$ for all $q$ and $i$. It is obvious that even if the length of the strings is bounded, the estimation of all the probabilities in (16) is an almost impossible task for a phoneme set of any size. For example, if there are ten phonemes and the strings are assumed to be no longer than length 5, then the number of probabilities is greater than $5^5$. The amount of data to estimate these probabilities is too large to obtain practically. Therefore, a Bayes decision procedure for this case is impractical. A decision procedure that does not require the estimation of all possible probabilities will have to be found.

In order to motivate the development that follows, we will outline the basic approach used for this recognition problem. A concept of the storage of prototype pattern strings is extended to what is called a generalized prototype string. This concept will be used in the pattern recognition problem as follows. The generalized prototype string will be defined as a truncated Markov chain. $R$ of these Markov chains are defined. These Markov chains produce finite strings of symbols in $P$. The probability measures $p(q \mid i)$ on these strings are assumed to approximate the probability measures for each of the $R$ pattern classes. These generalized prototype strings are used to define sequential machines for
recognition. These sequential machines will accept an unknown string \( q \) and calculate \( p(q \mid i) \). This will then be used to classify \( q \) according to (16).

The need for generalized prototype strings comes from the fact that the intra-class variance of the strings is large. If this variance is small, then a straightforward method of recognition exists. This method would be to store the most common output strings for each class. Each unknown pattern string \( q \) would then be matched against the stored strings. If there is a match with one of the stored strings for class \( i \), then \( q \) will be put in pattern class \( i \). If, however, the strings within a pattern class show a large variance, too many strings will have to be stored in order to recognize a reasonable number of patterns. To reduce the storage requirements in this case, the following concept of a generalized prototype string has been formulated.

In order to simplify notation, we will work only with the indices of the strings and omit the symbols \( a \). In other words, if we have a string \( a_{i_1}, \ldots, a_{i_k} \), then we will describe this string as \( \sigma_i \), \( \ldots \), \( \sigma_k \). This will cause no confusion.

If there are \( M \) phonemes in \( A \), then the possible indices for the symbols \( a_i \) run from 1 to \( M \). Assume that there is a new symbol \( a_{M+1} \) that represents string termination. That is, using the notation introduced above, each string is of the form \( \sigma_i, \ldots, \sigma_k, M+1 \). This will be useful when the truncated Markov chains are defined.

Suppose we have a prototype string \( n_{i_1}, \ldots, n_{i_k}, n_{m+1} = M+1 \). This string will be used to define a Markov chain that terminates at \( M+1 \) appears. To do this, assume that there exist probabilities \( p(i), i = 1, \ldots, m \), and \( p(j \mid k), j = k+1, \ldots, m+1 \). These probabilities, along with the sequence defined above, can now be used to define a Markov chain. This chain will produce subsequences of \( n_{i_1}, \ldots, n_{m+1} \). If \( n_{i_1}, \ldots, n_{i_k}, n_{m+1} \) is such a subsequence, then, using the above probabilities, the Markov property allows us to write

\[
p(n_{i_1}, \ldots, n_{i_k}, n_{m+1}) = p(i_1)p(m+1 \mid i_2) \prod_{j=2}^{k} p(i_j \mid i_{j-1}),
\]  

(17)

where \( p(n_{i_1}, \ldots, n_{i_k}, n_{m+1}) \) is the probability that this subsequence occurs, \( p(i_1) \) is the probability that index \( i_1 \) is the first index in the subsequence, and \( p(i_j \mid i_{j-1}) \) is the probability that index \( i_j \) follows \( i_{j-1} \). Note that if at any time \( i_j = m+1 \), the string terminates. Also note that the subsequence preserves the order of the original sequence. That is, \( p(j \mid k) = 0 \) for \( j \leq k \).

In accordance with the above discussion we have the following definition.

**Definition 4.** A sequence \( n_1, \ldots, n_m, n_{m+1} = M+1 \) together with the probabilities \( p(i), p(j \mid i), j = i + 1, \ldots, m+1, i = 1, \ldots, m \) is called a generalized prototype string \( S \). The string \( S \) is said to be generated by \( n_1, \ldots, n_m, n_{m+1} \).

**Definition 5.** The range of a generalized prototype string \( S \) is that set of subsequences \( Q \) such that a subsequence \( n_{i_1}, \ldots, n_{i_k}, n_{m+1} \) is in \( Q \) if and only if \( p(n_{i_1}, \ldots, n_{i_k}, n_{m+1}) > 0 \).

Thus, the generalized prototype string is actually a probability measure on \( P \). Suppose that \( \sigma_i, \ldots, \sigma_k, M+1 \) is a string in \( P \). If this sequence is not in the range of \( S \), then there is a subsequence \( n_{i_1}, \ldots, n_{i_k}, n_{m+1} \) such that \( \sigma_i = n_j \) for all \( j \). The probability measure on \( P \) is defined in the following manner.

**Definition 6.** If \( \sigma_1, \ldots, \sigma_k \) is a sequence in \( P \), then define

\[
p(\sigma_1, \ldots, \sigma_k) = 0, \text{ if } \sigma_1, \ldots, \sigma_k \text{ is not in the range of } S,
\]

\[
= p(n_{i_1}, \ldots, n_{i_k}, n_{m+1}), \text{ if } \sigma_1, \ldots, \sigma_k \text{ is in the range of } S, \quad (18)
\]

where \( n_{i_1}, \ldots, n_{i_k}, n_{m+1} \) is such that \( \sigma_j = n_j \) for all \( j \), and \( p(n_{i_1}, \ldots, n_{i_k}, n_{m+1}) \) is defined in (17). The probability in (18) will be called the measure associated with \( S \).

It can now be seen that the usual notion of a prototype string is a special case of the generalized prototype string. If \( p(1) = 1 \), and \( p(i \mid i-1) = 1 \) for all \( i \), then the resulting range of \( S \) consists only of the string \( n_1, \ldots, n_{m+1} \). This corresponds to the method described at the beginning of this section.

The following approach to the recognition problem will now be taken. Each pattern class \( i \) has a probability measure \( p(q \mid i) \) associated with it. It is assumed that there exist generalized prototype strings \( S_i, i = 1, \ldots, R \), such that \( p(q \mid i) \) is the measure associated with \( S_i \). In other words, we are using the concept of the generalized prototype string to approximate the measure \( p(q \mid i) \). The learning procedure will require that each \( S_i \) be determined. A recognition procedure must also be developed that will allow each of the \( p(q \mid i) \) to be evaluated for an unknown pattern. To simplify notation, write \( p_i(q) = p(q \mid i) \).

The calculation of \( p_i(q) \) associated with each \( S_i \) can be implemented by the use of sequential machines. It was shown that the measure on the range of \( S_i \) was the result of a truncated Markov chain. Let \( n_{i_1}, \ldots, n_{i_m}, M+1 \) be a string that generates \( S_i \). Assume for the moment that this string and the associated prob-
abilities have been completely determined. The sequential machine that implements \( S_i \) contains \( m_i + 2 \) states. These states are labeled \((0), 1, 2, \ldots, m_i, m_i + 1\), where \((0)\) is the reset or power on state and \( m_i + 1 \) is the terminal state. In other words, each machine state \( j \) is associated with the corresponding term \( m_j \), for \( j = 1, \ldots, m_i + 1 \). The \((0)\) state corresponds to the start of the sequence. Each state transition is defined in the following manner.

Let \( p_j(j), p_j(k | j), k = j + 1, \ldots, m_i + 1, j = 1, \ldots, m_i \) be the probabilities used to define \( S_i \). Suppose the sequential machine \( M_i \) is in state \((0)\). If \( p_j(j) \neq 0 \) define a transition to state \( j \) for input symbol \( n_j \). If \( p_j(k) = 0 \) for some \( k \), there is no transition from \((0)\) to state \( k \). Now suppose the sequential machine is in state \( k \). If \( p_j(k | j) \neq 0 \), define a state transition from state \( k \) to state \( j \) for input \( n_j \). Continue for all such states from 1 to \( m_i \). This process completely defines the sequential machines \( M_i \).

**Definition 7.** The sequential machine \( M_i \) is said to accept a sequence \( \sigma_1, \ldots, \sigma_h \) if this sequence is contained in the range of \( S_i \).

We are now in a position to see how the above sequential machines \( M_i \) can calculate the probabilities \( p_i(q) \). Recall that each state transition was defined using one of the probabilities \( p_j(j) \), or \( p_j(k | j) \). Thus, each state transition has an associated probability. If machine \( M_i \) accepts pattern string \( q \), then there is a sequence of state transitions leading to the final state \( m_i + 1 \). Each state transition has an associated probability. If the product of all these probabilities is formed, then it is seen that the result is the product in (17). But, it has been seen that this is the desired probability \( p_i(q) \) from Definition 6.

Therefore, the recognition procedure is as follows. The unknown string \( q \) is applied to all the sequential machines \( M_i, i = 1, \ldots, R \). If none of the machines accept \( q \), then it is rejected as unrecognizable. For each machine \( M_i \) that accepts \( q \) the probability \( p_i(q) \) is calculated in the following manner. An accumulator register is initialized to the value 1 at the start of \( q \). As each state transition is made during the application of \( q \) the probability associated with that transition is multiplied by the contents of the accumulator, and the result is stored back in the accumulator. After the machine reaches the final state \( m_i + 1 \), the desired probability \( p_i(q) \) is in the accumulator. These calculated probabilities are then used to classify \( q \) in the sense of (16).

Two examples will now be given that will clarify the above development.

**Example 2.** Suppose there are three phonemes in \( A \).
The measure associated with \( S_1 \) is then seen to be
\[
p_1(1234) = .5 \times .3 \times .9 \times 1.0 = .135
\]
\[
p_1(124) = .5 \times .3 \times .1 = .015
\]
\[
p_1(134) = .5 \times .6 \times 1.0 = .3
\]
\[
p_1(14) = .5 \times .1 = .05
\]
\[
p_1(24) = .5 \times .1 = .05
\]
\[
p_1(234) = .5 \times .9 \times 1.0 = .45
\]
In the same manner, the range for \( S_2 \) is
\[
234
2314
\]
The measure associated with \( S_2 \) is
\[
p_2(234) = 1.0 \times 1.0 \times .1 = .1
\]
\[
p_2(2314) = 1.0 \times 1.0 \times .9 \times 1.0 = .9
\]
Note that the ranges for \( S_1 \) and \( S_2 \) overlap in that 234 is common to both. But 234 will be put in class 1 since \( p_1(234) > p_2(234) \).

There is a subtle point about the application of the Markov chain. While the strings in the range of a generalized prototype string are assumed to be produced as a result of a Markov process, the strings themselves are not Markov. The next example will illustrate this point.

**Example 3.** Consider the generalized prototype string shown in Fig. 8 (Sequential machines will be used to define the \( S_i \) from this point on since the notation is more compact). The range of this generalized prototype string is
\[
1324
134
234
2324
\]

![Figure 8—Example of prototype string whose range is not Markov](From the collection of the Computer History Museum (www.computerhistory.org))
Unfortunately, the method for learning the $S_i$ is not as formal as the preceding. The prototype strings were determined from the tables of sample strings. For example, consider the sample strings in Table I. Each string has been listed in the table so that each column contains only one phoneme. The strings have been arranged so that the order of the phonemes is unchanged and the number of columns is minimized. These tables were formed by a manual procedure, and at present this procedure cannot be written as a sequence of steps. Once these tables have been determined, the prototype sequences can be defined. Each of the columns in the tables contains a distinct phoneme. These phonemes are taken to be the prototype string that generates $S_i$ for each class $i$. This is best described through the use of an example. Consider the table for “one” in Table II. For this case

$A = \{1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22\}$

The prototype string that generates $S_1$ for the class “one” is $10, 22, 20, 17, 16, 11, 17, 11, 1, 15, 5, 3, 1, 15, 3$.

Crude estimates of the probabilities can also be made by counting the transitions between states as defined by rows in the table. Using these probabilities and the above prototype string, we have the generalized prototype string for “one” in Figure 9.

A computer program was written that simulated the entire recognition system. The program accepted isolated words, computed the phoneme strings, and implemented the sequential machines. The sequences in the training set were recognized using the sequential machines. Using this method, the recognition rate was one hundred percent for the 250 patterns in the training set.

In order to further demonstrate the power of the algorithm, the recognition was run using a restricted system. The association of probabilities with state transitions was removed. Recall that the calculated probabilities for an input pattern string were used only if more than one machine accepted the string. In order to provide a more severe test for the concept of using sequential machines, the use of the calculated probabilities was dispensed with. In this case, only one pattern from the 250 was accepted by more than one machine. One sample pattern for “three” was accepted by both the machine for “three” and “two”. Thus, under these circumstances, the sequential machines performed most satisfactorily.

It was desirable to continue to simplify the algorithm in order to see when the performance began to degenerate. Therefore, the following additional simplification was tried. The assumption that each input string had phoneme $M+1$ in the terminal position was dropped. Using the same definition of acceptance, this implies that a sequential machine will accept an input string even if the machine is not driven to its final state. It is stressed here that these changes in the algorithm were not made to try to improve performance, but were made to try to see how far the algorithm could be degraded and still achieve good results.

Recognition was attempted using the strings without the terminal symbol described above. The sequential
machines with no probabilities were used. If more than one machine accepted a pattern, then it was assumed to be an error. Under these conditions, the error rate was 4%. That is, there were ten error patterns out of the 250 presented to the system. The confusion matrix for this test is given by

<table>
<thead>
<tr>
<th>decision class</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>22</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>20</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This algorithm represents considerable simplification of the full algorithm. Under these conditions the error rate was still low. We can conclude that the potential of the complete algorithm is such that further work is highly desirable.

The conclusions that can be made from this section are as follows. It has been seen that sequential feature extraction has considerable utility for use in the two stage recognition procedure presented here. The structure produces phoneme strings that can easily be used to design the sequential machine for the second stage of recognition. The recognition results in the second stage were encouraging. The initial motivation for development of the second stage recognition was to demonstrate the capabilities of the machine phonetic structure. However, the experimental results indicate that this has potential for solving recognition problems independent of the sequential feature extraction.

CONCLUSIONS

In this study we have presented a means of detecting sequential structures in waveforms for recognition. This process is called sequential feature extraction. The waveforms were assumed to be produced by a random process that was unknown. A learning algorithm that automatically generated a structure for sequential feature extraction was presented. This learning rule was unsupervised, and was shown to be a generalization of previously unsupervised learning rules for the time invariant case.

It was shown that sequential feature extraction could be considered as a transformation $T$, from $W$ to the set of all finite sequences of symbols from a set $A$ called the phoneme set. A structure on $T$, was developed so that the transformation was dependent on a set of parameters $s$. This allowed us to find an $s$ that was optimum with respect to a performance function.

This algorithm was applied to a problem in speech recognition. Experimental results were given that showed interesting relationships between the standard phonetic structure and the structure developed by sequential feature extraction. It was concluded that the automatically developed structure was related to the linguistic structure, but that there were significant differences due to the continuously time changing character of speech.

A new concept called the generalized prototype string was presented. This was a generalization to the probabilistic case of the method of storage of prototype strings. Each generalized prototype string was seen to be a means of approximating the probability measures on $P$. Once the generalized prototype strings were found, it was possible to design sequential machines for recognition. These sequential machines were seen to implement the calculation of estimates of the individual pattern class probabilities $p_i(q)$, where $q \in P$. Using these probabilities, the pattern could be classified according to Bayes decision theory.

The sequential feature extraction presented in this study represents a new approach to waveform recognition and unsupervised learning. For the case of speech, it was seen that sequential feature extraction was related to a phonetic structure. While phonetic structures are not new, the concept of using unsupervised learning to automatically develop a phonetic structure is new. The sequential structure in speech or other types of waveforms can be detected by using this algorithm. Because the algorithm is automatic, there is no bias due to previous results from linguistics. This is a particular advantage if the algorithm is to be applied to applications other than speech.

The restriction of the unsupervised learning algorithm to the time invariant case showed that the algorithm had advantages over current methods. No knowledge of the probability measures is required, strict separa-
bility of clusters is not required, the class of allowed metrics is large, and there is no requirement that the sample patterns be stored for processing since the algorithm will accept patterns one at a time.

REFERENCES

1 B P LATHI
   Signals, systems and communication
   Wiley New York 1965

2 S C FRALICK
   Learning to recognize patterns without a teacher
   IEEE Trans Inf Th Vol 13 pp 57-64 January 1964

3 D P COOPER P W COOPER
   Adaptive pattern recognition without supervision
   Proc IEEE International Convention 1964

4 E A PATRICK J C HANCOCK
   Nonsupervised sequential classification and recognition of patterns
   IEEE Trans on Inf Th Vol 12 July 1966

5 C G HILBORN D G LAINIOLIS
   Optimal unsupervised learning multategory dependent hypothesis pattern recognition
   IEEE Trans Inf Th Vol 14 May 1968

6 E M BRAVERMAN
   The method of potential functions in the problem of training machines to recognize patterns without a teacher
   Automation and Remote Control Vol 27 October 1966

7 G H BALL D J HALL
   ISODATA—An iterative method of multivariate analysis and pattern classification

International Communications Conference Philadelphia Pennsylvania June 1966

8 J GAZDAG
   A method of decoding speech
   University of Illinois AD 641 132 June 1966

9 K W OTTEN
   Simulation and evaluation of phonetic speech recognition techniques—Vol III Acoustical characteristics of speech sounds systematically arranged in the form of tables
   NCR Company AD 601 422 March 1964

10 I LEHISTE
   Acoustical characteristics of selected English consonants
   Int J Am Linguistics Vol 30 July 1964

11 W F MEEKER A L NELSON P B SCOTT
   Voice to teletype code converter research program. Part II—Experimental verification of a method to recognize phonetic sounds

12 K W OTTEN
   Simulation and evaluation of phonetic speech recognition techniques Vol II—Segmentation of continuous speech into phonemes
   NCR Company AD 601 423 March 1964

13 J L FLANNAGAN
   Speech analysis, synthesis and perception
   Springer-Verlag New York 1965

14 M KABRISKY
   A proposed model for visual information processing in the human brain
   University of Illinois Press Urbana Illinois 1966

15 J S KENYON T A KNOTT
   A pronouncing dictionary of American English
   G & C Merriam Springfield Massachusetts 1953