A Modified *K*-Means Clustering Algorithm for Use in Isolated Work Recognition

JAY G. WILPON, MEMBER, IEEE, AND LAWRENCE R. RABINER, FELLOW, IEEE

Abstract—**Studies of isolated word recognition systems have shown that a set of carefully chosen templates can be used to bring the performance of speaker-independent systems up to that of systems trained to the individual speaker. The earliest work in this area used a sophisticated set of pattern recognition algorithms in a human-interactive mode to create the set of templates (multiple patterns) for each word in the vocabulary. Not only was this procedure time consuming but it was impossible to reproduce exactly because it was highly dependent on decisions made by the experimenter. Subsequent work led to an automatic clustering procedure which, given only a set of clustering parameters, clustered patterns with the same performance as the previously developed supervised algorithms. The one drawback of the automatic procedure was that the specification of the input parameter set was found to be somewhat dependent on the vocabulary type and size of population to be clustered. Since a naive user of such a statistical clustering algorithm could not be expected, in general, to know how to choose the word clustering parameters, even this automatic clustering algorithm was not appropriate for a completely general word recognition system. It is the purpose of this paper to present a clustering algorithm based on a standard *K*-means approach which requires no user parameter specification. Experimental data show that this new algorithm performs as well or better than the previously used clustering techniques when tested as part of a speaker-independent isolated word recognition system.**

I. INTRODUCTION

PATTERN recognition techniques have been widely used in all aspects of speech recognition. However, the area which has depended most on pattern recognition techniques is that of pattern clustering to derive a set of speaker-independent templates for isolated word recognition. The problem here is straightforward. We are given a set of N word patterns, where each pattern is a single utterance of one particular word in the vocabulary, spoken (in general) by N different talkers, and our task is to cluster the N patterns into M clusters such that within each cluster the word patterns are highly similar. In this manner we can represent the N-pattern training set by M templates, where each template is derived from the patterns within each cluster. Typically, for $N = 100$ patterns of a word, about $M = 10$–12 clusters are needed for high recognition accuracy [1].

The earliest pattern clustering algorithms for isolated words were semiautomatic [1], [2]. They used some fairly sophisticated pattern recognition techniques (e.g., ISODATA, chainmap, *K*-means), but relied on manual intervention to guide the clustering. This procedure, although quite successful in clustering the data, was unacceptable for widespread use because of its lack of repeatability of the results and because of the inordinate time required to cluster data (primarily due to the decisions made by the human observer).

The next step was to develop an automatic clustering procedure which needed no manual intervention and which could cluster the word patterns with the same performance as achieved by the semiautomatic procedures. The result was the UWA (unsupervised without averaging) algorithm [3]. The philosophy here was a simple one, namely, home in on the largest cluster (via an iterative procedure), find all word patterns "close" to the center of this cluster, eliminate them from the training set, and recluster the remaining patterns.

The UWA algorithm created template sets that were shown to yield recognition accuracies as high as those obtained through manual clustering techniques on several word vocabularies [3]–[5]. However, there were still some inherent problems with implementing the UWA algorithm. First, a distance threshold had to be provided by the user so as to define closeness. The way in which this distance threshold was chosen was not well defined. Different words generally had different thresholds. Different populations of talkers also often had different thresholds for the same word. To this end, a set of convergence criteria was developed whereby the clustering algorithm itself would determine whether its given threshold was too high or too low (according to the cluster occupancy statistics) and would modify the threshold appropriately. The rules that were implemented involved introducing several extra parameters dealing with the number of clusters and cluster sizes. After using the clustering algorithms on different data sets over a long period of time, a better understanding was developed of which parameter values worked well and which ones did not. These procedures led to the development of a set of convergence criteria (and all associated thresholds) for the UWA clustering algorithm. However, it was found that when changes were made in population size and vocabulary type, some of these criteria were not always correct. Further study was done for several different data sets to refine the convergence criteria. However, we have not been able to adequately define a convergence criterion that would work well in all situations.

A second problem with the UWA clustering procedure was that coverage of the entire training set was not guaranteed. Hence, depending on the distance threshold for clustering, the set of M clusters often covered only 85–90

Manuscript received February 16, 1984; revised November 14, 1984.
The authors are with AT&T Bell Laboratories, Murray Hill, NJ 07974.

0096-3518/85/0600-0587$01.00 © 1985 IEEE

percent of the training patterns for the case of 100 training patterns, when M was large (i.e., 12) [2], and significantly fewer patterns when M was small (i.e., 6 or less) [1]. The training patterns not included in the clustering coverage were the so-called "outlier" patterns whose minimum distance to a cluster center exceeded the distance threshold. Such outlier patterns generally do contain useful speech information which was totally lost in the UWA procedure.

As a result of these drawbacks in the UWA procedure, a clustering algorithm based on the conventional K-means iteration has been developed [6]. We call this clustering procedure the modified K-means (MKM) algorithm. The philosophy of the MKM algorithm is essentially that used in standard vector quantization (VQ) codebook designs [7]–[10] based on an idea originally due to Lloyd [11]. As such, the general ideas behind the clustering procedure are well known and it is only the details of using the K-means loop on entire word patterns, each consisting of a temporal sequence of vectors rather than single vectors, which distinguishes the MKM algorithm from more traditional K-means procedures. The idea of quantizing short sequences of LPC vectors (e.g., 5–10 vectors per sequence) has been used recently in the totally different context of low bit rate coding [12]–[13]. However, there are significant differences between these proposals and the present method. These differences, which will be discussed in more detail later in this paper, include the method of computing a cluster center (i.e., using a minimax or pseudoaverage instead of a generalized centroid) and the splitting procedure used to initialize the K-means iteration.

The rationale for developing the MKM algorithm was as follows. Recent work on using the K-means iteration to cluster LPC vectors for designing VQ codebooks [7] has led to high performance systems for speech coding and recognition [8]–[10], [14], [15]. The philosophy of iteratively refining clusters and cluster centers such that all training patterns are included in some cluster is appealing since it eliminates the need for using artificial distance thresholds, etc., to decide whether or not to include a training pattern in a given cluster. Thus, based on its success in VQ codebook design, and on its desirable property of including all patterns in some cluster, the K-means iteration appeared to be a natural for use in a word clustering procedure.

The MKM algorithm is intended for clustering single word patterns from a wide variety of talkers with different ways of saying words (i.e., different accents, pronunciations, etc.). Hence, it is intended for use as a speaker-independent word clustering procedure. Technically, it could equally well be applied to speaker-dependent recognition systems, but, in general, there has not been the need for sophisticated clustering procedures in this case since a single talker is generally highly consistent in his pronunciation of single words (this is, in fact, the basis for the statistical pattern recognition paradigm).

The organization of this paper is as follows. In Section II-A, a review of the UWA clustering algorithm is given. In Section II-B, the MKM algorithm is defined. Section II-C describes two alternative methods for computing cluster centers. In Section II-D, the concept of creating an average cluster center is reviewed. In Section III, recognition results are presented comparing the UWA and MKM algorithms. Finally in Section IV a discussion of the results is given.

II. Unsupervised Algorithms for Clustering Word Data

Following the development in Levinson *et al.* [2], we assume that we are given a finite set Ω of N observations

$$\Omega = \{x_1, x_2, \cdots, x_N\} \tag{1}$$

where each observation x_i is a pattern representing a replication of one specific spoken word. Each pattern has an inherent duration (e.g., x_i is n_i frames long), and each frame of the pattern is some measured set of features. For the recognition system we use, the feature set is the set of $(p+1)$ autocorrelation coefficients ($p=8$) [16], [17].

Since it is intended that the clustering of the N observations be based entirely on distance (similarity) data (as is done in the actual recognition system), a distance $\delta(x_i,x_j)$ between patterns x_i and x_j is defined as

$$\delta(x_i,x_j) = \frac{1}{n_i} \sum_{k=1}^{n_i} d(k,w(k),i,j) \tag{2}$$

where the local frame distance $d(k,w(k),i,j)$ is the log likelihood distance proposed by Itakura [17] between the kth frame of x_i and the $w(k)$th frame of x_j, i.e.,

$$d(k,w(k),i,j) = \log \left[\frac{(\boldsymbol{a}^j_{w(k)})' R^i_k (\boldsymbol{a}^j_{w(k)})}{(\boldsymbol{a}^i_k)' R^i_k (\boldsymbol{a}^i_k)} \right] \tag{3}$$

where $\boldsymbol{a}^i_k$ is the vector of LPC coefficients of the kth frame of pattern i, R^i_k is the matrix of autocorrelation coefficients of the kth frame of pattern i, and $'$ denotes vector transpose. The function $w(k)$ is the warping function obtained from a dynamic time warp match (DTW) of pattern j to pattern i which minimizes d_{ij} over a constrained set of possible $w(k)$ [2], [18], [19].

From the initial set of N patterns an $N \times N$ distance matrix $\hat{D}$ can be defined with entry $\hat{\delta}(x_i,x_j)$ defined as

$$\hat{\delta}(x_i,x_j) = \frac{\delta(x_i,x_j) + \delta(x_j,x_i)}{2}. \tag{4}$$

Equation (4) yields a symmetric distance matrix requiring storage for only $N(N-1)/2$ terms (since $\hat{\delta}(x_i,x_i) = 0$ all i). Since we are generally using very large N (from 100–1000), the storage saved by symmetrizing the distance is not insignificant. Also, since there are no distinguished tokens, an asymmetric distance (as is the case for the Itakura distance measure) is unreasonable.

The purpose of the clustering is to represent the set Ω as the union of M disjoint clusters $\{\lambda_i, i = 1, 2, \cdots, M\}$ such that

$$\Omega = \bigcup_{i=1}^{M} \lambda_i. \tag{5}$$

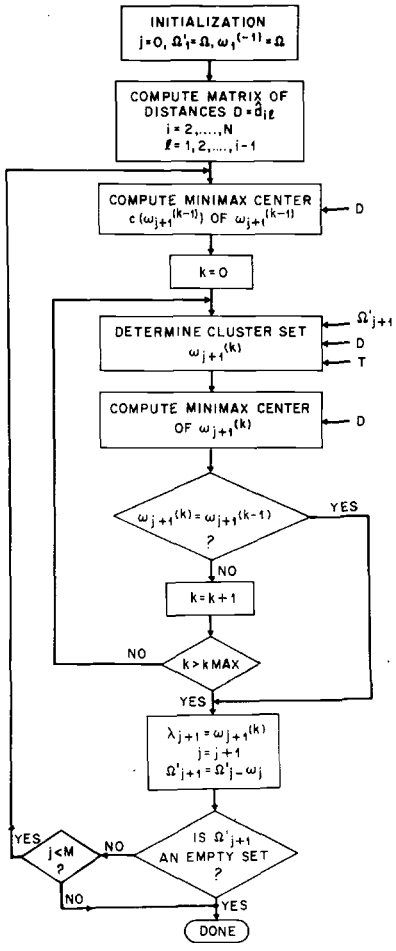


Fig. 1. Flow diagram of the UWA clustering procedure.

The total number of clusters M need not be known or specified *a priori*. We denote the center or prototype of cluster ω_i as $c(\lambda_i)$ and we note that $c(\lambda_i)$ need *not* be a member of ω_i.

A. Unsupervised Clustering Without Averaging (UWA)

A flow diagram of the UWA clustering algorithm is given in Fig. 1. The notation we use is as follows. We assume that the raw data x_i, $1 \le i \le N$ is to be clustered into clusters, λ_j, $1 \le j \le M$. The partial coverage set Ω_j, which includes all patterns in the first j clusters, is defined as

$$\Omega_j = \bigcup_{i=1}^{j} \lambda_i = \Omega_{j-1} + \lambda_j \tag{6}$$

and the partial (remaining) observation set Ω'_{j+1} is defined as the observations (patterns) *not* included in the first j clusters, i.e.,

$$\Omega'_{j+1} = \Omega - \bigcup_{i=1}^{j} \lambda_i = \Omega - \Omega_j = \Omega'_j - \lambda_j \tag{7a}$$

$$= \{x'_1, x'_2, \cdots, x'_{q(j)}\} \tag{7b}$$

where x'_i is an element of Ω'_{j+1} and $q(j)$ is the number of patterns that remain to be clustered after the first j clusters have been formed. (By definition, $q(0) = N$.)

We denote the cluster index by j, and the iteration index as k. We define $\omega_j^{(k)}$ as the set of patterns in cluster j at the kth iteration, and $c(S)$ is the minimax center of the set S, i.e.,

$$c(S) = x_{i^*} \in S \quad \text{such that} \quad \max_m \hat{\delta}(x_{i^*}, x_m) \le \min_i \max_m \hat{\delta}(x_i, x_m), \tag{8}$$

i.e., the minimax center of the set S is the pattern in S whose maximum distance to any other pattern in S is minimum.

With the above definitions, the UWA clustering algorithm works as follows.

1) Initialization $j = 0$, $k = 0$, $\Omega'_1 = \Omega$, $\omega_1^{(-1)} = \Omega$. Compute matrix of distances $D = \hat{\delta}(x_i, x_m)$, $1 \le i, m \le N$.

2) Determine $c(\omega_{j+1}^{(k-1)})$, the minimax center of $\omega_{j+1}^{(k-1)}$. Since all distances of any pair of patterns in Ω are precomputed and stored in D, minimax computations are especially simple to implement.

3) Determine elements of the cluster set $\omega_{j+1}^{(k)}$ as

$$\omega_{j+1}^{(k)} = \bigcup_{l \in \Omega'_{j+1}} x'_l \quad \text{such that} \quad \hat{\delta}\,(c(\omega_{j+1}^{(k-1)}), x'_l) \le T,$$

i.e., all patterns in Ω'_{j+1}, that are within a distance T of $c(\omega_{j+1}^{(k-1)})$.

4) Determine $c(\omega_{j+1}^{(k)})$, the new minimax center of cluster $\omega_{j+1}^{(k)}$.

5) Check if $\omega_{j+1}^{(k)} = \omega_{j+1}^{(k-1)}$, i.e., the cluster is unchanged since the last iteration. If this is the case, convergence is obtained and $\lambda_{j+1} = \omega_{j+1}^{(k)}$, j is incremented, and the new partial observation set Ω'_{j+1} is generated as

$$\Omega'_{j+1} = \Omega'_j - \lambda_j.$$

This set is checked to guarantee that it is not an empty set. If not, and if $j < M$, the algorithm sets $k = 0$, $\omega_{j+1}^{(-1)} = \Omega'_{j+1}$, and goes back to step 2. Otherwise, if $j \ge M$, or if $\Omega'_{j+1} = \phi$, the clustering is over.

If no local convergence is obtained, k is incremented and checked against the maximum iteration count. If this count is exceeded, k is decremented and the algorithm proceeds as if convergence was obtained. If not, the algorithm continues at step 3.

It can be shown that prominent, distinct clusters will be readily found by this procedure since the cluster sets at consecutive iterations will be identical. However, for highly overlapping data, as the cluster center changes, so does the cluster composition, causing the need for several iterations. These iterations are reminiscent of the merge and split phases of ISODATA [20]. Theoretically, the UWA algorithm can lead to geometrically overlapping clusters; practically this is irrelevant since each training pattern is assigned to a unique cluster.

The user-supplied inputs to the UWA method are the lower half-matrix of distances D, the number of observations N, the distance threshold T, and the maximum iteration count $K_{\max}$. Initial values of T are chosen based on theoretical estimates of LPC distances [17], [21], however

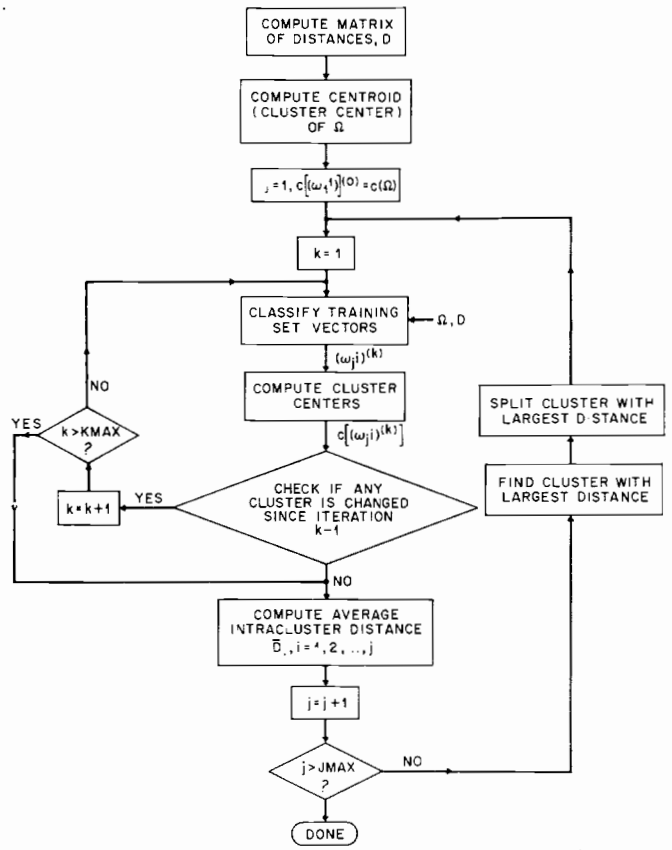


Fig. 2. Flow diagram of the MKM clustering procedure.

the algorithm itself, based on a user supplied set of parameters, can modify T to increase (lower T) or decrease (raise T) the total number of clusters required to represent the observation set. Typically, a maximum iteration count of 10 is used. For most clusters only 2 or 3 iterations are required to obtain a stable result.

B. The Modified K-Means Algorithm (MKM)

A flow diagram of the MKM algorithm is given in Fig. 2. We denote the ith cluster of a j cluster solution, at the kth iteration, as $(\omega_j^i)^{(k)}$, where $i = 1, 2, \cdots, j$, and $k = 0, 1, \cdots, K_{\max}$ (where $K_{\max}$ is a maximum iteration count). Values of j go from 1 (single cluster) to $J_{\max}$ (a maximum cluster count). The MKM clustering proceeds as follows.

1) Initialization $j = 1$, $k = 1$, $i = 1$. Compute matrix of distances $D = \hat{\delta}(x_l, x_m)$, $1 \le l, m \le N$.

2) Compute cluster center $c(\Omega)$ of Ω, the entire training set. The cluster center may be computed as either a minimax center, or a pseudoaverage center (see Section II-C for a discussion of these computations).

3) Set $(\omega_1^1)^{(0)} = \Omega$, $c[(\omega_1^1)^{(0)}] = c(\Omega)$.

4) Classify each pattern, x_l of Ω, as belonging to one of the clusters $(\Omega_j^i)^{(k)}$, $i = 1, 2, \cdots, j$ by choosing the cluster which has minimum distance from the pattern x_l to the cluster center $c[(\omega_j^i)^{(k)}]$, i.e.,

$$x_l \in (\omega_j^i)^{(k)} \text{ iff } \hat{\delta}(x_l, c[(\omega_j^i)^{(k)}]) \le \hat{\delta}(x_l, c[(\omega_j^{\hat{i}})^{(k)}])$$

$$\text{for all } \hat{i} \ne i.$$

5) For each resulting cluster $(\omega_j^i)^{(k)}$, $i = 1, 2, \cdots, j$, compute a cluster center $c[(\omega_j^i)^{(k)}]$, again using either a minimax or a pseudoaverage center.

6) A convergence check is made to see if any cluster has changed from the previous iteration (i.e., if $(\omega_j^i)^{(k)} = (\omega_j^i)^{(k-1)}$, $i = 1, 2, \cdots, j$). If any cluster has changed, the iteration counter k is incremented and checked against a maximum value. If the iteration count does not exceed the maximum allowed value, the K-means iteration (steps 4–5) is repeated.

7) Compute $\bar{D}_i$, the average intracluster distance, for all clusters, as

$$\bar{D}_i = \frac{1}{|(\omega_j^i)|} \sum_{\substack{l=1 \\ l \in (\omega_j^i)}}^{N} \hat{\delta}(x_l, c[(\omega_j^i)])$$

where $|(\omega_j^i)|$ is the number of patterns of Ω included in cluster (ω_j^i) at the end of the K-means iteration.

8) When convergence is obtained (i.e., all clusters remain unchanged from the previous iteration, or the maximum iteration count is exceeded), the clusters $(\omega_j^i)^{(k)}$ are saved as the best j cluster solution, the cluster count index j is incremented and checked against the maximum cluster size $J_{\max}$. If the count is exceeded, then the procedure is terminated and final cluster centers are created for all clusters (i.e., $j = 1, 2, \cdots, J_{\max}$, $i = 1, 2, \cdots, j$).

9) If another stage of the MKM is to be run, the algorithm first finds the cluster $(\lambda_{j-1}^{i'})$ with the largest intracluster cluster distance, i.e.,

$$\bar{D}_{i'} \ge \bar{D}_i \qquad i \ne i'.$$

This largest cluster is split into two by finding the indexes l_1, l_2 of the pair of patterns in $(\lambda_{j-1}^{i'})$ such that the distance between these patterns is maximized, i.e.,

$$\hat{\delta}(x_{l_1}, x_{l_2}) \ge \hat{\delta}(x_{l_3}, x_{l_4}) \qquad \text{for all } \; l_3, l_4 \in \lambda_{j-1}^{i'}.$$

The new initial cluster center for the ith cluster is set to the pattern x_{l_1}, and the new initial cluster center for the jth cluster (i.e., the new one added at this iteration) is the pattern x_{l_2}. All other initial cluster centers $c[(\omega_j^i)^{(0)}]$ are chosen as the cluster centers $c[(\omega_{j-1}^i)]$ at the end of the previous K-means iteration. The iteration counter is reset to $k = 1$.

10) The K-means iteration proceeds as before (steps 4–5).

A final template set is then created by either using the center points of the computed clusters, or by averaging (after time alignment) all patterns within a cluster as defined in Section II-D. One feature of the MKM algorithm is that it computes an optimal clustering, with 100 percent coverage, for all clustering sizes from one cluster per word to $J_{\max}$ clusters per word. In this manner a reference set for isolated word recognition can easily be created with a different number of templates for each word in the vocabulary. This may be of practical value for problems where some vocabulary words exhibit a lot of variability and, therefore, need many templates to characterize them, and

where other vocabulary words are not so variable and can be well represented by a small number of templates.

C. Computation of a Cluster Center

Two different techniques were used to compute the cluster centers needed by the MKM algorithm. The first one was to compute a minimax center point. That is, choose as cluster center, that point for which the maximum distance to all other points in the cluster is minimum over all points in the clusters. If we denote the set ω with J patterns as $\omega = \{x_1, x_2, \cdots, x_J\}$ then the minimax center is the pattern x_{l^*} such that

$$\max_{1\le l\le J} \hat{\delta}(x_{l^*},x_l) \le \min_{1\le m\le J} \max_{1\le l\le J} \hat{\delta}(x_m,x_l). \tag{9}$$

The minimax center has the property that it is a pattern of the training set. Hence, the distance from all training set patterns to any such minimax center is easily computed by looking up its value in the distance matrix D.

The second technique used to compute a cluster center is based on finding the pattern within the cluster which has the largest population of patterns whose distance falls within some designated threshold. This is done in the following way. For each cluster ω we first compute the average distance d and standard deviation σ_d between all points as

$$\bar{d} = \frac{1}{(J)(J-1)} \sum_{l=1}^{J} \sum_{\substack{m=1 \\ m\neq l}}^{J} \hat{\delta}(x_l,x_m) \tag{10a}$$

$$\sigma_d = \left[\frac{1}{J(J-1)} \sum_{l=1}^{J} \sum_{\substack{m=1 \\ m\neq l}}^{J} \hat{\delta}^2(x_l,x_m) - (\bar{d})^2\right]^{1/2}. \tag{10b}$$

A count c_l for the lth pattern is obtained by computing the number of patterns in ω whose distance to pattern x_l is less than the empirically chosen threshold

$$T = \bar{d} + 0.5\sigma_d.$$

The pattern x_l with the largest count is chosen as the cluster pseudocenter. If a tie existed between two patterns then the pattern chosen as the cluster center was the one with the smallest average distance to *all* points in the cluster. The resulting cluster center, using this technique, is called the pseudoaverage center. Again it can be seen that the pseudoaverage center is an actual pattern of Ω; hence computation of distances in the K-means iteration again involves table lookups.

D. Averaging Techniques Used to Obtain Cluster Centers

For both algorithms of Section II, a final cluster center (obtained after convergence) was obtained via a true time-alignment, averaging technique. One important consideration in obtaining these final cluster centers was the manner in which this averaging was carried out [3]. To facilitate this discussion we consider two patterns x and y in the observation set Ω. Token x is assumed to consist of N_x frames of LPC features, and pattern y consists of N_y frames of features, where each frame is a set of $p + 1$ ($p = 8$ in our system) residual normalized autocorrelation coefficients. If we denote the ith frame of x (or y) as $x(i)$ [or $y(i)$], then we can represent x and y as the set of vectors

$$x = (x(1), x(2), \cdots, x(i), \cdots, x(N_x)), \tag{17a}$$

$$y = (y(1), y(2), \cdots, y(i), \cdots, y(N_y)) \tag{17b}$$

where

$$x(\mathrm{i}) = (x_0(i), x_1(i), \cdots, x_p(i)) \tag{18}$$

and similarly for $y(i)$.

In order to average patterns x and y we must have a correspondence between frames of x and frames of y. For simplicity, we assume pattern y is being mapped to pattern x. As such a dynamic time warping procedure is used to give the mapping

$$x(i) \rightarrow y(k) = y(w(i)) \tag{19a}$$

or

$$k = w(i), \qquad i = 1, 2, \cdots, N_x, \tag{19b}$$

i.e., the ith frame of pattern x corresponds to the $k = w(i)$th frame of pattern y. As such when we average patterns x and y we produce pattern z

$$z = (z(1), z(2), \cdots, z(N_x)) \tag{20}$$

where

$$z(i) = \tfrac{1}{2}[x(i)+y_k(w(i))]. \tag{21}$$

The addition of time aligned patterns of (21) is a vector addition, i.e., each component of the vector is added independently. When we average Q patterns of Ω, we successively warp each of the patterns to the estimated center of the cluster and then average the time registered patterns (of course we normalize by $1/Q$ in this case).

E. Computational Complexity

The computational complexity of the MKM algorithm is readily determined by examining each of the steps in the procedure of Section II-B. The initialization (step 1) in which the distance matrix $D = \hat{\delta}(x_l,x_m)$, $1 \le l, m \le N$ is computed, requires N^2 DTW computations, where each DTW requires about $L^2/3$ distance computations (where L is the average number of frames in a word), and a distance computation for the likelihood ratio distance requires $(p + 1) = 9$ multiplications and additions. Thus, the total computation for obtaining the distance matrix D, is

$$C_{\mathrm{INIT}} = \frac{N^2L^2(p+1)}{3} *, +. \tag{22}$$

This computation needs to be performed once for any clustering procedure, and must be considered as fixed overhead. The cost of symmetrizing D is insignificant.

The computation involved in step 2, for determining the

cluster center of the entire set, is essentially N^2 distance table lookups and distance comparisons for the minimax center, and approximately N^2 multiplications and additions for the pseudocenter approach. Since this computation is only performed once (on the entire set), the computation here is not significant, even for N on the order of 1000. The computation involved in the initialization of the clusters and cluster centers of step 3 is insignificant. The heart of the computation of the MKM algorithm is in step 4 (the classification step), and step 5 (the recalculation of cluster centers). For classification of the N training patterns to one of J clusters, a total of NJ distance lookups and NJ distance comparisons is required. This computation is insignificant, even for large values of N and J. For determining new cluster centers via the minimax procedure, only table lookups and distance comparisons are required. Again the computation is insignificant. For determining cluster centers via the pseudoaveraging procedure, on average about (N^2/J) multiplications and additions are required. Thus, the computation for the MKM iteration is

$$C_{MM} = (N^2/J) \quad \text{table lookups and comparisons} \quad (23a)$$

$$C_{PA} = (N^2/J) \quad \text{multiplications and additions} \quad (23b)$$

where C_{MM} is the minimax center computation, and C_{PA} is the pseudoaverage center computation. Since the K-means loop is iterated (until convergence), the computation (C_{MM} or C_{PA}) is multiplied by the number of iterations to convergence. The computation for the convergence check (step 6) is insignificant.

The computation for steps 7–10, in which the average intracluster distances are computed (step 7), an overall termination check is made (step 8), the largest cluster is split (step 9), and the K-means loop continues (step 10) essentially involves only table lookups and comparisons of distances. Hence, the computation here is again negligible.

The creation of the final reference set, from the clustering of the N training patterns into J clusters, involves N DTW's. Although this computation is not negligible, it again is done only once.

To illustrate the speed of the MKM procedure for clustering a set of $N = 100$ training patterns into $J = 12$ clusters, on a Data General MV8000 computer (0.5 MIPS machine), the computation of distances required about 1 h, the K-means iterations for solutions with 1–12 clusters required about 2 min, and the computation of the final reference set (the centroids of each cluster) took 30 s. Thus, the fixed overhead in computing the distance matrix totally dominated the computation of the MKM procedure.

III. Isolated Word Recognition Using the MKM Algorithm

The evaluation of the MKM algorithm involved clustering a set of speech data consisting of 4786 isolated patterns from a digits vocabulary. The training patterns (and a subsequent set of 6883 independent testing patterns) were recorded in a field trial at a regional telephone switching office. About 1660 talkers provided the training and testing data, and, in general, there was essentially no overlap in talkers between the training set and the testing set. All training and test recordings were made under identical telephone recording conditions which included highly variable background, highly variable transmission, and highly variable signal levels. For more information about the details of the recording conditions, the reader is referred to [22].

TABLE I
Number of Tokens for Each Digit Used in the Training and Testing Phases for Evaluating the MKM Algorithm

	Training Set	Testing Set
0	271	276
1	259	374
2	675	1013
3	606	808
4	580	813
5	489	788
6	592	911
7	443	621
8	454	730
9	414	549
TOTAL	4786	6883

TABLE II
Recognition Results for Telephone Digits Database Using Several Different Sets of Templates

Cluster Center	Autocorrelation Averaging	KNN Rule	MKM Recognition Accuracy (%)	UWA Recognition Accuracy (%)
Minimax	Yes	1	81.5	81.1
		2	83.7	82.5
		3	83.9	82.1
Pseudo-Average	Yes	1	81.9	81.7
		2	83.9	83.6
		3	84.2	83.2
Minimax	No	1	76.9	74.9
		2	81.5	79.8
		3	81.9	80.5
Pseudo-Average	No	1	78.1	75.8
		2	81.6	79.6
		3	81.8	80.1

Table I shows the distribution of the training and testing patterns amongst the 10 digits. Clustering was performed using the UWA and the MKM algorithms. The two procedures of Section II-C were used to determine the cluster centers. Templates were created both with and without the final averaging technique. In all cases a 30 template per word reference set was created. The recognition system used was the LPC-based isolated word recognition system developed in our laboratory [1], [6].

The results of a set of recognition runs in which the 6883 test digits were recognized using different sets of templates are given in Table II which shows error rate for each of the different training conditions. Given in this table are results for several values of the K-nearest neighbor decision rule in which the recognition is based on the best K distances for each word (rather than the best single distance as is conventionally the case for a nearest neighbor recognizer). Experience has shown that values of K of 2

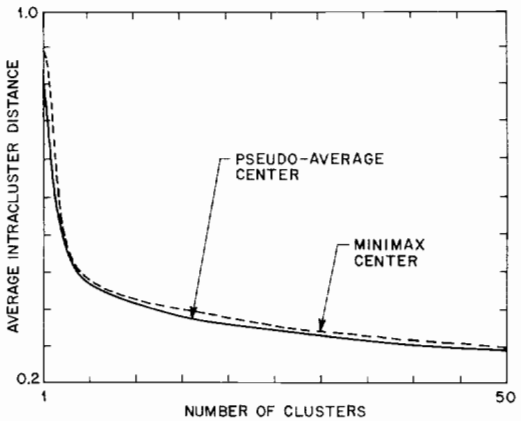


Fig. 3. Plot showing average distortion measure as a function of the number of clusters used per word for the minimax center (dotted-line) and the pseudoaverage center (solid line).

or 3 provide real performance improvements when using 10 or more templates for each vocabulary word [1].

The results of Table II show the following.

1) The MKM recognition accuracy was always greater than or equal to the UWA recognition accuracy. (The improvement in accuracy ranged from a low of 0.2 percent to a high of 2.3 percent across different training conditions. With about 7000 test digits, a difference of 0.8 percent is significant at the 95 percent confidence level; hence, the improvement of 1 percent for the best training condition is marginally significant.)

2) The pseudoaverage cluster center was always marginally better than the minimax cluster center in terms of recognition performance. As a further check on the utility of the pseudoaverage (as opposed to the minimax average), the average intracluster distance was computed for all cluster sets ranging from 1 to 50 clusters using both the minimax and pseudoaveraged cluster centers. The results, plotted in Fig. 3, show that, for all cluster sizes, the average intracluster distortion from the pseudoaverage cluster center simulations was slightly smaller than that computed using a minimax cluster center. This result indicates that the pseudoaverage cluster center led to slightly tighter clusters than those obtained from the minimax center.

3) The final clusters obtained via autocorrelation averaging (after time-alignment to the minimax center) always form a template set with significantly higher performance (as much as 6.3 percent) than when the final clusters were obtained as the minimax center. For single vector clustering problems, i.e., the VQ codebook design problem, this is exactly the way in which cluster centroids are found [9]; hence, there is a good theoretical basis for averaging patterns within a cluster to determine a cluster center.

On the basis of the results given above, a second experiment was conducted on a smaller set of digits. In this case 100 talkers (50 male, 50 female) each spoke each digit two times over a local, dialed-up telephone line. One pattern of each digit, for each talker, was used to give a 100 pattern training set; the other pattern was used to give a 1000 pattern independent test set. The UWA clustering was used to give a set of 12 templates per digit; similarly the MKM was used to give optimal clustering with from 1 to 12 clusters per digit, i.e., the one cluster per digit set used all 100 training patterns, as did the two cluster set, etc.

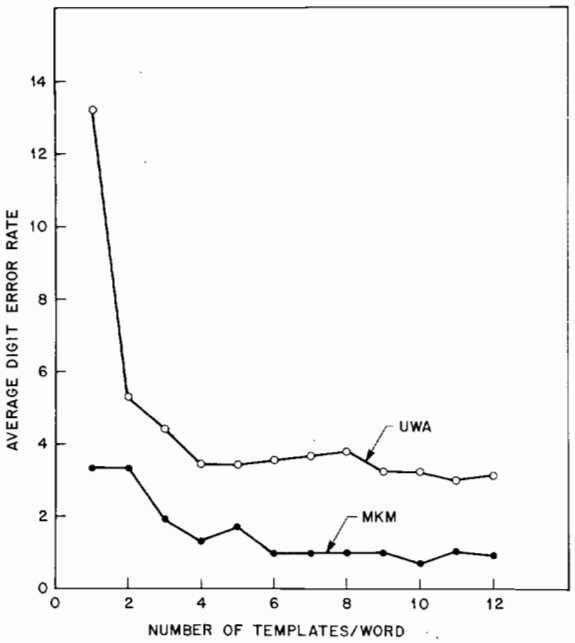


Fig. 4. Average digit error rate versus the number of templates per word for a laboratory database of 1000 digits with clusters generated by both the UWA and MKM clustering procedures.

A recognition test was performed on the independent 1000 digit test set and recognition error rates were measured, using both template sets, with the number of templates varying from 1 to 12. (Of course, for the UWA templates, the smaller template sets are *always* subsets of the entire training set.) The results of the recognition tests are shown plotted in Fig. 4. Since the data were laboratory data, the average error rates were considerably lower than for the field telephone recordings discussed previously.

It can be seen from Fig. 4 that for one template per word, the full coverage MKM template yielded a 10 percent lower error rate than the partial coverage UWA template. Even for larger template sets, the MKM templates provided about 2 percent reduction in error rate over that obtained using the UWA templates. Such an improvement in performance is highly significant at the 99 percent confidence level.

IV. Discussion

The main purpose of this paper was to develop a fully automatic, "idiot proof," word clustering procedure based on the highly successful *K*-means iteration used in VQ codebook design and other related areas. Such a procedure was developed in this paper. On the surface the resulting algorithm appears to be a trivial application of known technology. However, the successful adaptation of the *K*-means iteration to word clustering required solving some small but important practical problems such as how to obtain cluster centers (centroids) at each stage of the iteration so that computation would not be prohibitive, how

to split clusters to advance from one size solution to the next, and finally how to create the final cluster representations, i.e., the word templates from patterns within the cluster.

The success of our algorithm has been demonstrated via simulation on two sets of isolated digit data. On one set (highly divergent data samples for both training and testing) we showed that the MKM clustering led to a recognition system whose performance exceeded that of a previous clustering by a small amount. For this data set the diversity of background signal and noise levels, the variability of transmission conditions, and the diversity of talkers mediated against any possibility of large improvements in system performance. However, for a better behaved database of laboratory recordings, the superiority of the MKM clustering over previous clustering procedures was clearly shown.

The importance of the results lies primarily in the simplicity of the resulting MKM clustering procedure, and the resulting recognition performance. The fact that the MKM has a strong theoretical foundation in the K-means iterative procedure also makes this clustering procedure an attractive one.

V. Summary

The purpose of this investigation was to develop an automatic clustering algorithm, which could be implemented by any user with a minimal amount of knowledge about clustering procedures and provide template sets as accurate as those created by other clustering algorithms. This goal led to the development of the modified K-means (MKM) clustering algorithm. Recognition results were presented indicating that the MKM algorithm performed as well as, or better than, the well-established UWA clustering algorithm. The results also indicate that a pseudoaverage cluster center provides slightly better performance than a minimax center. Finally, we have again shown the advantages of averaging the autocorrelation vectors within a given cluster to obtain a final template.

References

[1] L. R. Rabiner, S. E. Levinson, A. E. Rosenberg, and J. G. Wilpon, "Speaker independent recognition of isolated word using clustering techniques," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 336–349, Aug. 1979.

[2] S. E. Levinson, L. R. Rabiner, A. E. Rosenberg, J. G. Wilpon, "Interactive clustering techniques for selecting speaker independent reference templates for isolated word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 134–141, 1979.

[3] L. R. Rabiner and J. G. Wilpon, "Considerations in applying clustering techniques to speaker independent word recognition," *J. Acoust. Soc. Amer.*, vol. 66, pp. 663–673, Sept. 1979.

[4] ——, "Speaker independent isolated word recognition for a moderate size (54 word) vocabulary," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 583–587, Dec. 1979.

[5] J. G. Wilpon, L. R. Rabiner, and A. F. Bergh, "Speaker independent isolated word recognition using a 129 word airline vocabulary," *J. Acoust. Soc. Amer.*, vol. 72, pp. 390–396, Aug. 1982.

[6] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Probability and Statistics*, Berkeley, CA, 1967, vol. 1, pp. 281–296.

[7] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84–95, Jan. 1980.

[8] B. Juang, D. Wong, and A. H. Gray, J., "Distortion performance of vector quantization for LPC voice coding," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 294–303, Apr. 1982.

[9] A. Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel, "Speech coding based upon vector quantization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 562–74, Oct. 1980.

[10] D. Wong, B. Juang, and A. H. Gray, Jr., "An 800 bit/second vector quantization LPC vocoder," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 770–80, Oct. 1982.

[11] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 129–137, Mar. 1982.

[12] S. Roucos, R. M. Schwartz, and J. Makhoul, "A segment vocoder at 150 b/s," in *Proc. ICASSP '83*, Apr. 1983, pp. 61–64.

[13] D. Y. Wong, B. H. Juang, and D. Y. Cheng, "Very low data rate speech compression with LPC vector and matrix quantization," in *Proc. ICASSP '83*, Apr. 1983, pp. 65–68.

[14] A. Buzo, H. Martinez, and C. Rivera, "Discrete utterance recognition based upon source coding techniques," in *Proc. ICASSP'82*, May 1982, pp. 539–542.

[15] J. E. Shore and D. Burton, "Discrete utterance speech recognition without time alignment," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 473–491, July 1983.

[16] J. D. Markel and A. H. Gray Jr., *Linear Prediction of Speech*. New York: Springer-Verlag, 1976.

[17] F. Itakura, "Minimum prediction residual applied to speech recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 67–72, Feb. 1975.

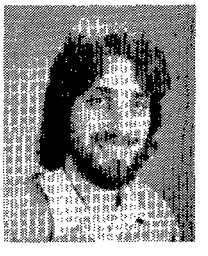
[18] L. R. Rabiner, A. E. Rosenberg, and S. E. Levinson, "Considerations in dynamic time warping algorithms for discrete word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 26, pp. 575–582, 1978.

[19] H. Sakoe, and S. Chiba, "A dynamic programming approach to continuous speech recognition," in *Proc. Int. Congress on Acoust.*, Budapest, Hungary, 1971, paper 20C-13.

[20] G. H. Ball and D. J. Hall, "Isodata—An iterative method of multivariate analysis and pattern classification," in *Proc. IFIPS Congress*, 1965.

[21] J. M. Tribolet, L. R. Rabiner, and M. M. Sondhi, "Statistical properties of an LPC distance measure," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 550–558, Oct. 1979.

[22] J. G. Wilpon, and L. R. Rabiner, "On the recognition of isolated digits from a large telephone customer population," *Bell Syst. Tech. J.*, vol. 62, part I, pp. 1977–2000, Sept. 1983.



Jay G. Wilpon (M'84) was born in Newark, NJ on February 28, 1955. He received the B.S. and A.B. degrees (cum laude) in mathematics and economics, respectively, from Lafayette College, Easton, PA, in 1977, and the M.S. degree in electrical engineering/computer science from Stevens Institute of Technology, Hoboken, NJ, in 1982.

Since June 1977 he has been with the Acoustics Research Department at Bell Laboratories, Murray Hill, NJ, where he is a member of the Technical Staff. He has been engaged in speech communications research and is presently concentrating on problems in isolated and connected word speech recognition. He has published extensively in this field and has been awarded several patents. His current interests lie in training procedures for both speaker-dependent and speaker-independent recognition systems, speech detection algorithms, and determining the viability of implementing speech recognition systems for general usage.

Lawrence R. Rabiner (S'62–M'67–SM'75–F'75), for a photograph and biography, see this issue, p. 560.