Pergamon

Pattern Recognition, Vol. 29, No. 3, pp. 511–518, 1996
Elsevier Science Ltd
Pattern Recognition Society
Printed in Great Britain.
0031-3203/96 $15.00 + .00

0031-3203(94)00091-7

FAST CLUSTERING ALGORITHMS FOR VECTOR QUANTIZATION

J. S. PAN,†‡ F. R. McINNES† and M. A. JACK†

† Centre for Communication Interface Research, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, U.K.

‡ Department of Electronic Engineering, Kaohsiung Institute of Technology, Taiwan

(*Received* 9 *December* 1994; *in revised form* 23 *May* 1995; *received for publication* 28 *June* 1995)

Abstract—Some fast clustering algorithms for vector quantization (VQ) based on the LBG recursive algorithm are presented and compared. Experimental results in comparison to the conventional vector-quantization (VQ) clustering algorithm with speech data demonstrate that the best approach will save more than 99% in the number of multiplications, as well as considerable saving in the number of additions. The increase in the number of comparisons is moderate. An improved absolute error inequality (AEI) criterion for Euclidean distortion measure is also proposed and utilized in the VQ clustering algorithm.

VQ Partial distortion search Absolute error inequality LBG algorithm
Triangular inequality elimination Nearest neighbour

1. INTRODUCTION

Vector quantization[1] (VQ) is a very efficient approach to data compression. The encoder of VQ encodes a given set of k-dimensional data vectors $X = \{X_j | X_j \in R^k;\ j = 1, \ldots, T\}$ with a much smaller set of codewords $C = \{C_i | C_i \in R^k;\ i = 1, \ldots, N\}$ $(N \ll T)$. Only the index i is sent to the decoder. The decoder has the same codebook as the encoder and decoding is operated by table look-up procedure. The performance of data compressing depends on a good codebook of representative vectors.

The LBG algorithm[2] is an efficient VQ clustering algorithm. This algorithm is based either on a known probabilistic model or on a long training sequence of data. The main idea of this algorithm is the iterative application of a codebook modification operation where a distortion measure D is used to compute the cost $D(X_j, C_i)$ of reproducing the data vector X_j as the codeword C_i. Usually the Euclidean distortion measure is used to compute the cost. The iteration is terminated if the average distortion $D(X, C)$ converges. The iterative procedure is time consuming and it is difficult to apply the VQ clustering procedure for a real time operation.

The computational complexity of the LBG algorithm can be significantly reduced if an efficient codeword search algorithm is applied to the partitioning of the data vectors. Many fast algorithms have been proposed to increase the speed of codeword search.[3–12] Fischer and Patrick[3] presented a preprocessing algorithm to reorder the design sample such that many distance computations could be eliminated. Fukunaga and Narendra[4] proposed a branch and bound (BAB) algorithm for computing some nearest neighbours. The BAB algorithm is a tree search algorithm using a hierarchical decomposition of the sample set of known patterns. They used the criterion of triangular inequality to develop two rules to eliminate the distance computation in the tree classifier. Kamgar-Parsi and Narendra[5] added another two rules to the BAB algorithm to improve the computation time. Niemann and Goppert[6] combined these four rules into one and used a hierarchical partition of a pattern sample algorithm to obtain more efficient computation time. Jiang and Zhang[7] developed a more efficient BAB tree search algorithm for finding the nearest neighbour to a new data vector in the codebook. These efficient search methods are not suitable to apply to the VQ clustering algorithm due to the overhead of the pre-processing.

Bei and Gray[8] proposed the partial distortion search (PDS) algorithm to reduce computational complexity. PDS is a simple and efficient codeword search algorithm which has no extra storage or preprocessing requirements. Cheng *et al.*[9] proposed the minimax method to obtain the tentative match and improve the search efficiency. Vidal[10] presented the approximating and eliminating search algorithm (AESA), in which the computation time is approximately constant for codeword search in a large codebook size. AESA is a very efficient algorithm to reduce multiplication operations for large codebook size, but it needs many comparison operations. Soleymoni and Morgera[11] proposed the absolute error inequality (AEI) elimination criterion to improve the speed of VQ search. Chen

and Pan[12] applied the triangular inequality elimination (TIE) on VQ-based recognition of isolated words, taking advantage of the high correlation characteristics between data vectors of adjacent speech frames. In this paper, we improve our previous work[13] in clustering algorithms by developing the improved AEI criterion.

The structure of this paper is as follows. Section 2 describes the detail of the LBG algorithm, tentative match approaches and codeword elimination criteria, which include the previous vector candidate and previous partitioned centre, partial distortion search (PDS), hypercube approach, absolute error inequality (AEI) criterion, improved absolute error inequality criterion, triangular inequality elimination (TIE) criteria and codebook reorder method. The improved absolute error inequality criterion is also proved in this section. Section 3 lists the fast VQ clustering algorithms which combine the tentative match approaches and codeword elimination criteria described in Section 2. Experimental results are discussed in Section 4. Summary and conclusions are given in the last section.

The cepstrum[14,15] of a signal is defined as the Fourier transform of the log of the signal spectrum. Cepstrum coefficients are used as the test features in the clustering experiments because they are commonly used in speech coding, speech synthesis, speech recognition and speaker recognition. The test materials for these experiments consist of 200 words recorded from one male speaker. The speech is sampled at a rate of 16 kHz and 13-dimensional cepstrum coefficients with pre-emphasis value 0.98 are computed over 20 ms-wide frames with a 5 ms frame shift. A total of 20, 030 analysed frames are used in the VQ clustering experiments. Normally, this data set size is used to train up to 1024 codewords. Here, we use it in the VQ clustering experiments for eight and 1024 codewords.

2. RELATED METHODS

2.1. *LBG algorithm*

All of the fast VQ clustering algorithms described in this paper are based on the LBG algorithm.[2] This algorithm starts by assigning all the training data vectors to a single cluster and proceeds by binary splitting until the desired number of clusters is achieved. After each splitting of the clusters there is an iterative procedure in which the cluster centroids are re-estimated and the data vectors are re-classified until the average distortion between the centroids and their classified vectors converges. The classification at each stage uses the full-search algorithm to find the nearest centroid to each vector. The detail algorithm based on unknown distribution is as follows:

Step 1. Set $m = 1$. Calculate centroid $C_1 = \frac{1}{T}\sum_{j=1}^{T} X_j$, where T is the total number of data vectors.
Step 2. Divide each centroid C_i into two close vectors $C_{2i-1} = C_i*(1+\delta)$ and $C_{2i} = C_i*(1-\delta)$, $1 \le i \le m$. Here, δ is a small fixed perturbation scalar. Let $m = 2m$. Set $n = 0$, here n is the iterative times.
Step 3. Find the nearest neighbour to each data vector. Put X_j in the partitioned set P_i if C_i is the nearest neighbour to X_j.
Step 4. After obtaining the partitioned sets $P = (P_i; 1 \le i \le m)$, set $n = n + 1$. Calculate the overall average distortion $D_n = \frac{1}{T}\sum_{i=1}^{m}\sum_{j=1}^{T_i} D(X_j^{(i)}, C^i)$, where $P_i = \{X_1^{(i)}, X_2^{(i)}, \ldots, X_{T_i}^{(i)}\}$.
Step 5. Find centroids of all disjoint partitioned sets P_i by $C_i = \frac{1}{T_i}\sum_{j=1}^{T_i} X_j^{(i)}$.
Step 6. If $(D_{n-1} - D_n)/D_n > \varepsilon$, go to step 3; otherwise go to step 7. Here, ε is a small distortion threshold.
Step 7. If $m = N$, then take the codebook C_i as the final codebook; otherwise, go to step 2. Here, N is the codebook size.

2.2. *Previous vector candidate and previous partitioned centre*

In the VQ clustering procedure, speech data has the property that the classification result for the present vector is usually the same as or close to the classified result of the previous vector.[12] Moreover, most of the vectors which are re-estimated in a full-search actually remain in the same partitioned set as for the previous re-estimation. With binary codeword splitting, the most probable partition to which data vectors belong can be chosen from the separated centres of the partitioned set. The previous vector candidate and previous partitioned centre can be used as tentative matches in the VQ clustering algorithm. Figure 1 illustrates the relationship between the number of codewords and the probability that data vectors remain in the same partitioned set after re-estimation in full-search. For the fixed data vectors the more codewords being generated, the larger the probability that the data vectors belong to the same (previous) partitioned set. The probability is up to 0.949 for 1024 codewords. These results are average across the re-estimation and re-classification iterations when $\delta = 0.01$ and $\varepsilon = 0.005$.

2.3. *Partial distortion search*

Given one codeword C_t and the test vector X in k-dimensional space, the distortion of the Euclidean metric can be expressed as follows:

$$D(X, C_t) = \sum_{i=1}^{k} (x^i - c_t^i)^2, \tag{1}$$

where $C_t = \{c_t^1, c_t^2, \ldots, c_t^k\}$ and $X = \{x^1, x^2, \ldots, x^k\}$.

Given the current minimum distortion:

$$D(X, C_t) = D_{\min}, \tag{2}$$

$$\text{if} \quad \sum_{i=1}^{s} (x^i - c_j^i)^2 \ge D_{\min}, \tag{3}$$

$$\text{then} \quad D(X, C_j) \ge D(X, C_t), \tag{4}$$

where $s \le k$.

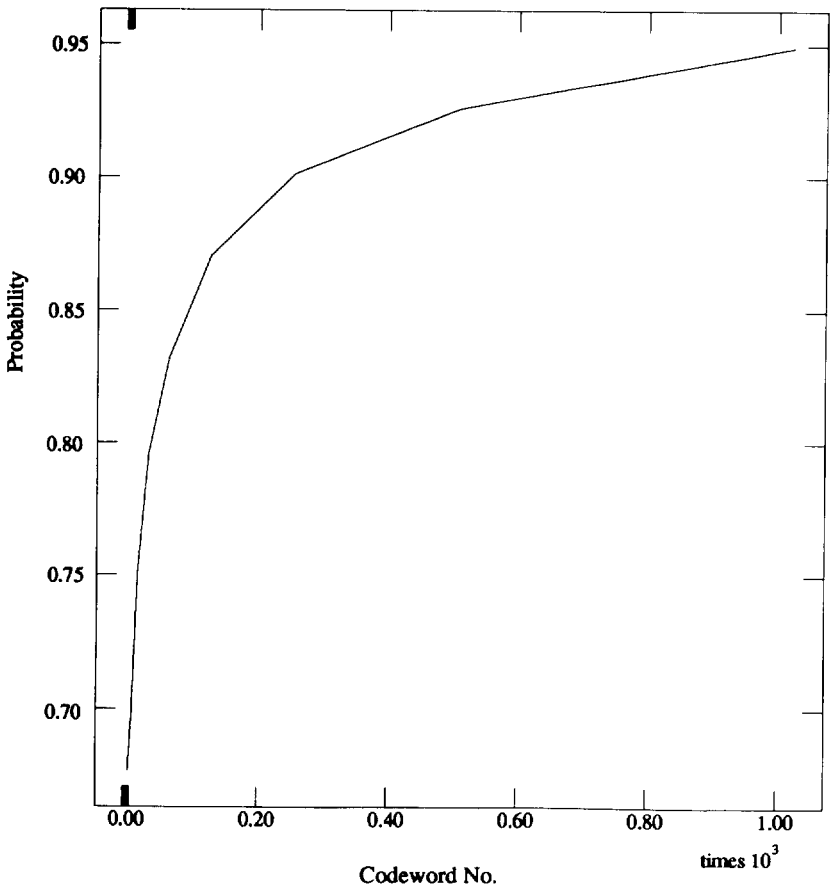


Fig. 1. Relationship between the number of codewords and the probability of the data vectors belonging to the previous partitioned set.

The efficiency of PDS[8] derives from elimination of an unfinished distortion computation if its partial accumulated distortion is larger than the current minimum distortion.

2.4. *Hypercube approach*

The hypercube approach is a well known premature method and it is quite efficient if the difference for any coefficient is generally larger than the difference of the other coefficients, such as the first coefficient of cepstrum coefficients. Assume equation (2) already exists:

$$\text{if} \quad |x^i - c_j^i| \geq \sqrt{D_{\min}}, \quad 1 \leq i \leq k, \tag{5}$$

then C_j will not be the nearest neighbour to X.

This approach is quite straightforward. There is no multiplication operation required for the test of the hypercube approach.

2.5. *Absolute error inequality criterion*

The absolute error inequality (AEI)[11] criterion is the mathematical relationship between the city block metric (or L_1) and the Euclidean metric (or L_2). Assume C_t is the current nearest neighbour to X:

$$\text{if} \quad \sum_{i=1}^{s} |x^i - c_j^i| \geq \sqrt{kD_{\min}}, \tag{6}$$

$$\text{then} \quad \sum_{i=1}^{k} (x^i - c_j^i)^2 \geq D_{\min}, \tag{7}$$

where $s \leq k$.

This means C_j will not be the nearest neighbour to X if equation (6) is satisfied. This criterion can be performed by comparing the first dimension-difference of the test vector and codeword with the right-hand side of equation (6). If equation (6) is not satisfied for $s = 1$, then check this criterion for higher s. This criterion is checked by increasing s until $s = k$ or the criterion is satisfied. Figure 2 shows the statistics for the elimination probability of AEI at each feature dimension. The previous partitioned centre is used as the initial codeword in this experiment. For 1024 codewords, 61.6% of impossible codeword matches will be eliminated by using AEI in the first dimension where only 0.5% codewords cannot be eliminated.

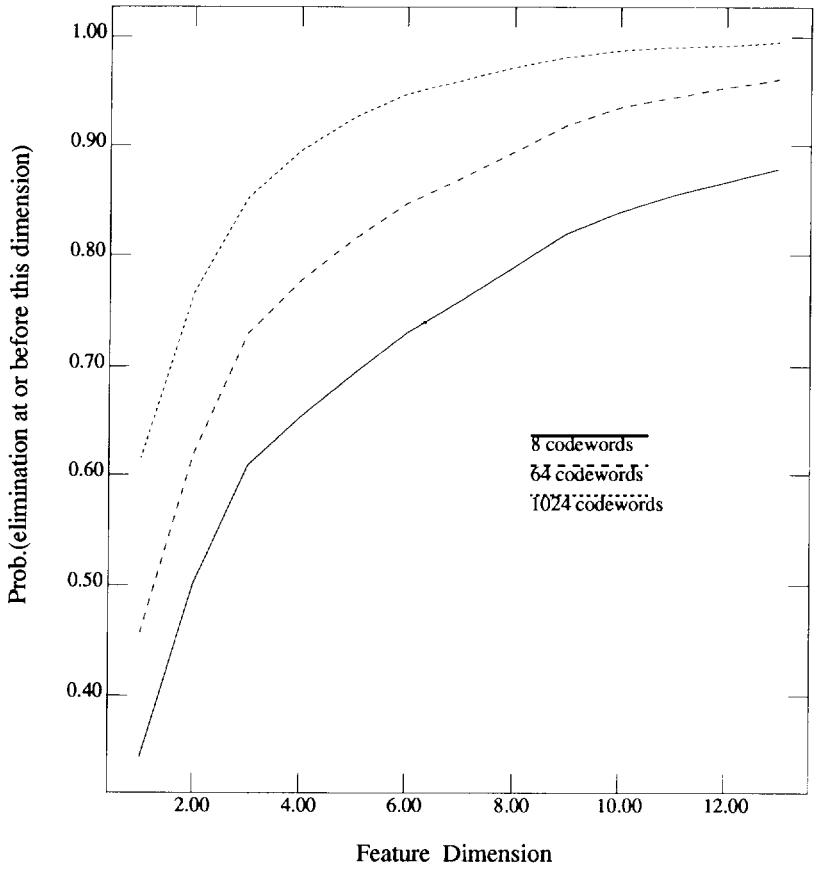


Fig. 2. The elimination probability of AEI of each feature dimension.

2.6. *Improved absolute error inequality criterion*

The AEI is not efficient compared with the hypercube approach by setting s to 1. Based on this phenomenon, an improved AEI criterion is presented here:

$$\text{if} \quad \sum_{i=1}^{s} |x^i - c_j^i| \geq \sqrt{hD_{\min}}, \tag{8}$$

$$\text{then} \quad \sum_{i=1}^{k} (x^i - c_j^i)^2 \geq D_{\min}, \tag{9}$$

where $s \leq h \leq k$.

The improved AEI criterion can be proved as follows:

Given $a_i \geq 0 \; \forall i$,

$$\begin{aligned} 0 &\leq \sum_{i=1}^{h} \left(a_i - \sum_{j=1}^{h} \frac{a_j}{h} \right)^2 \\ &= \sum_{i=1}^{h} a_i^2 - \frac{2}{h} \sum_{i=1}^{h} a_i \sum_{j=1}^{h} a_j + \sum_{i=1}^{h} \left(\sum_{j=1}^{h} \frac{a_j}{h} \right)^2 \\ &= \sum_{i=1}^{h} a_i^2 - \frac{1}{h} \left(\sum_{i=1}^{h} a^i \right)^2. \end{aligned}$$

Hence, $\sum_{i=1}^{h} a_i^2 \geq \frac{1}{h} \left(\sum_{i=1}^{h} a_i \right)^2$.

Hence, if $\sum_{i=1}^{h} a_i \geq \sqrt{hD_{\min}}$,

then $\sum_{i=1}^{h} a_i^2 \geq \frac{1}{h} \left(\sum_{i=1}^{h} a_i \right)^2 \geq \frac{1}{h} \left(\sqrt{hD_{\min}} \right)^2 = D_{\min}$.

Set $a_i = |x^i - c_j^i|$.

Hence if $\sum_{i=1}^{s} |x^i - c_j^i| \geq \sqrt{hD_{\min}}$,

then $\sum_{i=1}^{k} |x^i - c_j^i|^2 \geq D_{\min}$,

where $s \leq h \leq k$.

This criterion is similar to AEI, with h instead of k. Since $h \leq k$, this criterion provides a tighter bound than AEI. It is flexible by adapting s and h from 1 to k. For $h = 1$, it is the same as the hypercube approach.

2.7. *Triangular inequality elimination criterion*

Triangular inequality elimination[16] is an efficient method for codeword search. Let V be the set of data vectors and C be the set of codewords and x, y belong to the set V. On V, a distortion measure is defined as a mapping d: $V \times V \rightarrow R$, which is assumed to fulfill the

metric properties:

$$d(x, y) \geq 0; \quad d(x, y) = 0 \quad \text{iff } x = y \tag{10}$$

$$d(x, y) = d(y, x) \tag{11}$$

$$d(x, y) + d(y, z) \geq d(x, z) \tag{12}$$

Let C_1, C_2, C_3 be three different codewords and t be a test vector, then the following three criteria are obtained.

- Criterion 1.

Given the triangular inequality:

$$d(t, C_2) + d(t, C_1) \geq d(C_1, C_2); \tag{13}$$

$$\text{if} \quad d(C_1, C_2) \geq 2d(t, C_1), \tag{14}$$

$$\text{then } d(t, C_2) \geq d(t, C_1). \tag{15}$$

- Criterion 2.

Given the triangular inequality:

$$d(C_3, C_2) \leq d(t, C_2) + d(t, C_3); \tag{16}$$

$$\text{if} \quad d(C_3, C_2) \geq d(t, C_1) + d(t, C_2), \tag{17}$$

$$\text{then} \quad d(t, C_1) \leq d(t, C_3). \tag{18}$$

- Criterion 3.

$$\text{Assume} \quad d(t, C_1) \leq d(t, C_2).$$

$$\text{Given} \quad d(C_3, C_2) \geq d(t, C_2) - d(t, C_3); \tag{19}$$

$$\text{if} \quad d(C_3, C_2) \leq d(t, C_2) - d(t, C_1), \tag{20}$$

$$\text{then} \quad d(t, C_1) \leq d(t, C_3). \tag{21}$$

Criteria 2 and 3 can be merged to one criterion only, i.e.

$$\text{if} \quad d(t, C_1) \leq |d(C_3, C_2) - d(t, C_2)|, \tag{22}$$

$$\text{then} \quad d(t, C_1) \leq d(t, C_3). \tag{23}$$

To use Criterion 1, these distortions between all pairs of codewords are calculated in advance. If equation (14) is met, then the computation of $d(t, C_2)$ can be omitted if $d(t, C_1)$ has already been computed. Criterion 1 can be modified for a square error distortion measure. In the codeword searching system, a table is made to store the one-fourth of square distortion between codewords, i.e. store the value of $d^2(C_i, C_j)/4$, for $i = 1, 2, \ldots, N$; $j = 1, 2, \ldots, N$. Here, N is the number of codewords. The overhead of Criterion 1 is to establish the distortion table in which $N(N-1)k/2$ multiplications and $N(N-1)(2k-1)/2$ additions are needed. The physical meaning of Criteria 2 and 3 can be described as follows.

If the codeword C_i, $i \neq 1, 2$, does not locate between the two concentric circles (or in general hyperspheres) centred on C_2 with radii $d(t, C_2) \pm d(t, C_1)$, the computation of its distortion to the sample can be omitted, i.e. if $d(C_i, C_2) > d(t, C_2) + d(t, C_1)$ or $d(C_i, C_2) < d(t, C_2) - d(t, C_1)$, then eliminate the computation of C_i. For the special case $d(t, C_1) = d(t, C_2)$, Criterion 3 is in vain and Criterion 2 reduces to Criterion 1. Since Criteria 2 and 3 will induce square root computation, it is simple and efficient to use Criterion 1 only. Figure 3 illustrates the elimination probability using TIE combined with the previous partitioned centre in the VQ clustering procedure. For 1024 codewords, the elimination probability is 0.949.

2.8. *Codebook reorder method*

The codebook reorder method[16] is to reorder the codewords so as to increase the search efficiency. For the speech encoding, it chooses the nearest codeword of the previous frame as a tentative match to encode the present frame. From training data, calculate the probability of these codewords to be encoded and arrange these codewords in order of decreasing probability. The codeword search is operated from the most probable codeword to the least probable. It is simple and efficient to create a state table where these elements are indices of codewords and arranged in increasing order of distortion between the most probable codeword and the other codewords. In the VQ clustering procedure, the previous vector candidate or previous partitioned centre can be chosen as the most probable codeword so as to create the state table. The computational complexity is $O(N^2 \log_2 N)$ using Heapsort[17] to establish the state table.

3. FAST CLUSTERING ALGORITHMS

3.1. *APV-type clustering algorithm*

An efficient clustering algorithm must include two key elements, i.e. a good tentative match and a powerful codeword elimination criterion. We proposed the previous vector candidate as the tentative match with AEI and PDS to improve the conventional clustering algorithm. This algorithm is called the APV-type algorithm. It is described as follows.

Step 1. Set $m = 1$. Calculate centroid $C_1 = \frac{1}{T}\sum_{j=1}^{T} X_j$, where T is the total number of data vectors.
Step 2. Divide each centroid C_i into two close vectors $C_{2i-1} = C_i*(1+\delta)$ and $C_{2i} = C_i*(1-\delta)$, $1 \leq i \leq m$. Here, δ is a small fixed perturbation scalar. Let $m = 2m$. Set $n = 0$, here n is the iterative times.
Step 3. Compute the nearest neighbour for the first data vector X_1. For data vector X_j, use the nearest neighbour of X_{j-1} (previous vector candidate) as a tentative match and apply AEI with a PDS to find the nearest neighbour to each data vector. Place X_j in the partitioned set P_i if C_i is the nearest neighbour to X_j.
Step 4. After obtaining the partitioned sets $P = (P_i; 1 \leq i \leq m)$, set $n = n + 1$. Calculate the overall average distortion $D_n = \frac{1}{T}\sum_{i=1}^{m}\sum_{j=1}^{T_i} D(X_j^{(i)}, C_i)$, where $P_i = \{X_1^{(i)}, X_2^{(i)}, \ldots, X_{T_i}^{(i)}\}$.
Step 5. Find centroids of all disjoint partitioned sets P_i by $C_i = \frac{1}{T_i}\sum_{j=1}^{T_i} X_j^{(i)}$.
Step 6. If $(D_{n-1} - D_n)/D_n > \varepsilon$, go to step 3; otherwise go to step 7. Here, ε is a small distortion threshold.

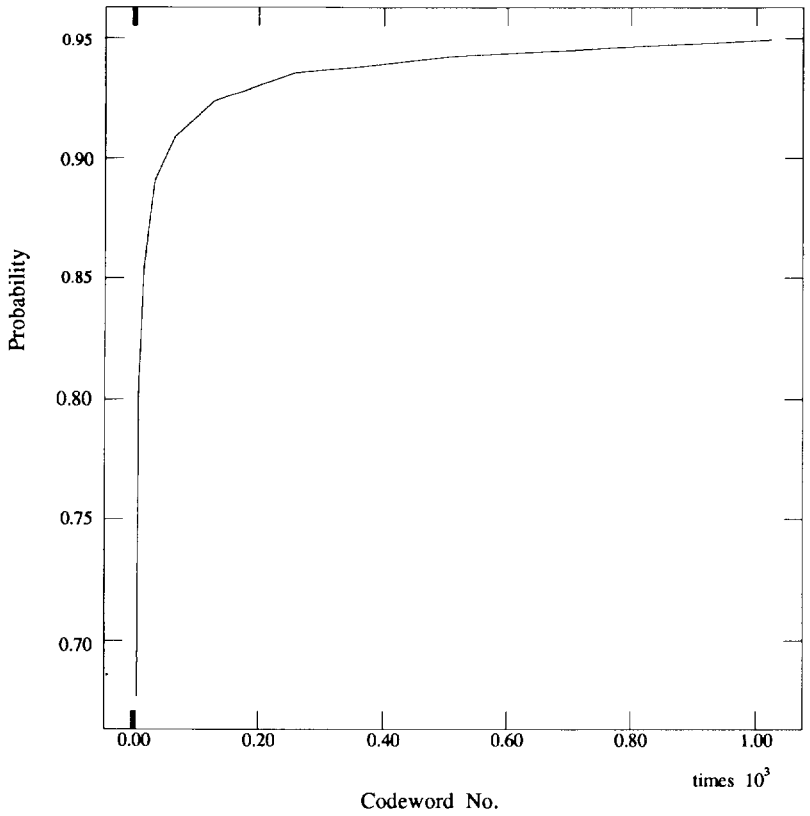


Fig. 3. Relationship between the number of codewords and the elimination probability using TIE.

Step 7. If $m = N$, then take the codebook C_i as the final codebook; otherwise, go to step 2. Here, N is the codebook size.

3.2. *APC-type clustering algorithm*

We find the previous vector candidate is a very efficient tentative match for word recognition.[12] It is not powerful compared with the previous partitioned centre in the clustering algorithm, because some adjacent data vectors are uncorrelated. We modify this clustering algorithm using a previous partitioned centre as a tentative match with AEI and PDS elimination criteria. This algorithm is referred to as an APC-type algorithm and is depicted as follows.

Step 1. Set $m = 1$. Calculate centroid $C_1 = \frac{1}{T}\sum_{j=1}^{T} X_j$, where T is the total number of data vectors.
Step 2. Divide each centroid C_i into two close vectors $C_{2i-1} = C_i*(1+\delta)$ and $C_{2i} = C_i*(1-\delta)$, $1 \le i \le m$. Here, δ is a small fixed perturbation scalar. Let $m = 2m$. Set $n = 0$, here n is the iterative times.
Step 3. For each data vector X_j, set $D_{\min} = \mathrm{MIN}(D(X_j, C_{2i-1}), D(X_j, C_{2i}))$, C_{2i-1} and C_{2i} are split from C_i associated to the partitioned set P_i to which X_j previously belonged. Choose C_{2i-1} or C_{2i} as the previous partitioned centre which is the nearest neighbour to X_j.
Step 4. Use the previous partitioned centre as a tentative match and apply AEI with PDS to find the nearest neighbour to each data vector. Place X_j in the partitioned set P_i if C_i is the nearest neighbour to X_j.
Step 5. After obtaining the partitioned sets $P = \{P_i; 1 \le i \le m\}$, set $n = n + 1$. Calculate the overall average distortion $D_n = \frac{1}{T}\sum_{i=1}^{m}\sum_{j=1}^{T_i} D(X_j^{(i)}, C_i)$, where $P_i = \{X_1^{(i)}, X_2^{(i)}, \ldots, X_{T_i}^{(i)}\}$.
Step 6. Find centroids of all disjoint partitioned sets P_i by $C_i = \frac{1}{T_i}\sum_{j=1}^{T_i} X_j^{(i)}$.
Step 7. If $(D_{n-1} - D_n)/D_n > \varepsilon$, take C_i as the previous partitioned centre for each $X_j \in P_i$ and go to step 4; otherwise go to step 8. Here, ε is a small distortion threshold.
Step 8. If $m = N$, then take the codebook C_i as the final codebook; otherwise, go to step 2. Here, N is the codebook size.

3.3. *APCH-type clustering algorithm*

The hypercube approach provides the tighter bound than AEI for $s = 1$. The APC-type algorithm can be further improved by adding the hypercube approach to step 4, which is known as the APCH-type algorithm. Use the previous partitioned centre as a tentative match. Check equation (5) to eliminate impossible codeword match. Apply AEI to eliminate the code-

word which cannot be eliminated using the hypercube approach. A PDS scheme is used for the codeword which cannot be eliminated using the hypercube approach and AEI criterion.

3.4. *IPC-type clustering algorithm*

The hypercube approach and AEI criterion are the special cases of the improved AEI criterion. Here, the improved AEI criterion is adopted to increase the efficiency for the clustering algorithm in Step 4. This algorithm is referred to as the IPC-type algorithm. By applying equation (8), this criterion can be separated into several sections. For 13-dimensional cepstrum coefficients, we can separate the improved AEI criterion into four sections. These four sections are set to $h = 1$ to check the first dimension-difference, $h = 4$ for the sum from the first dimension-difference to the fourth, $h = 9$ for the sum from the first dimension-difference to the ninth and $h = 13$ for the sum of all dimension-differences. A PDS scheme is used for the codeword, which cannot be eliminated using improved AEI criterion.

3.5. *TPC-type, ATPC-type and TPCR-type clustering algorithms*

Triangular inequality elimination (TIE) criteria can also be applied to Step 4 of the clustering algorithm. TPC-type is the clustering algorithm combining previous partitioned centres, TIE and PDS. The ATPC-type algorithm is the addition of AEI to the TPC-type algorithm, i.e. if the codeword cannot be eliminated using TIE, then apply AEI and PDS. The TPCR-type algorithm is the addition of a codebook reorder method to the TPC-type algorithm, i.e. reorder the codewords in increasing order of distortion between the previous partitioned centre and these codewords before applying TIE.

4. EXPERIMENTS AND RESULTS

The test materials used in the VQ clustering experiments are described in the first section. To verify these fast algorithms, the mathematical operations (multiplications, comparisons and additions) are used to calculate the computational efficiency. The experiments are carried out by setting the small fixed perturbation scalar to 0.01 and the small distortion threshold to 0.005. Ten approaches are compared in the VQ clustering procedure. The conventional exhaustive method is referred to as the CVT-type. P- and T-types are approaches using PDS and TIE in codebook design. The TPC-type is the algorithm using the previous partitioned centre as the most probable matching with TIE and PDS to reduce the clustering time. The TPCR-type is the TPC-type with the codebook reorder method. It is known as the APC-type if the previous partitioned centre is used as the tentative match with AEI and PDS to accelerate the clustering speed.

Table 1. Computational complexity of VQ clustering for eight codewords ($\times 10^6$)

Method	Mul.	Cmp.	Add.	Sum	Saving in mul. (%)
IPC	5.72	2.92	18.0	26.6	73.5
APCH	5.82	3.57	18.5	27.9	73.1
APC	5.88	6.87	25.0	37.8	72.8
APV	5.90	8.53	26.1	40.5	72.7
ATPC	5.87	5.87	21.3	33.0	72.8
TPC	5.63	2.30	15.0	22.9	73.9
TPCR	5.63	2.30	15.0	22.9	73.9
P	14.0	13.8	30.9	58.7	54.3
T	16.2	2.31	35.7	54.2	33.3
CVT	21.6	1.36	46.0	69.0	0

Table 2. Computational complexity of VQ clustering for 1024 codewords ($\times 10^6$)

Method	Mul.	Cmp.	Add.	Sum	Saving in mul. (%)
IPC	33.0	427	594	1054	99.2
APCH	33.8	481	663	1178	99.2
APC	34.1	680	1082	1796	99.2
APV	58.6	812	1323	2194	98.6
ATPC	105	463	465	1033	97.4
TPC	143	366	283	792	96.5
TPCR	143	539	282	964	96.5
P	777	777	1259	2813	81.1
T	1887	453	3649	5989	54.1
CVT	4109	315	7922	12346	0

The previous vector candidate instead of the previous partitioned centre in the APC-type is called the APV-type. The ATPC-type is the algorithm combining TIE, AEI, PDS and the previous partitioned centre. The APCH-type is the addition of the hypercube approach to the APC-type. The IPC-type is the algorithm combining the previous partitioned centre, improved AEI and PDS.

The experimental results for eight codewords and 1024 codewords are shown in Tables 1 and 2. For general processor architecture, the multiplication operation is more expensive than the comparison operation and addition operation. It is better to use the IPC-type algorithm for large codebook size and the TPC-type algorithm or TPCR-type algorithm for small codebook size. Tables 1 and 2 also illustrate the total mathematical operation number. In terms of the total number of operations, the TPC-type outperforms all the above algorithms. It needs extra computation time to generate the distortion table for the TIE approach, which is why the total number of multiplications in the ATPC-, TPC- and TPCR-types are larger than the IPC-, APC-, APV- and APCH-types for 1024 codewords. The codebook reorder method is not very efficient in the VQ clustering algorithm owing to the overhead of sorting procedure. In small codebook size, the TPCR-type is excellent. It is not superior compared with the IPC-, APCH-, APC-, APV-, ATPC-

and TPC-types for large codebook size. For the other codebook size, between eight codewords and 1024 codewords or more, these fast VQ clustering algorithms are also very efficient in computation.

5. SUMMARY AND CONCLUSIONS

In this paper, some fast VQ clustering algorithms are proposed and compared based on the number of multiplications, comparisons and additions. Among these approaches, using the previous partitioned centre as the tentative match with improved AEI and PDS is the most suitable approach for computer architectures in which the complexity of comparisons is negligible with respect to that of multiplications. For processor architectures such as those based on the Harvard architecture in which comparisons are computationally expensive, the combination of the previous partitioned centre, TIE and PDS outperforms the other algorithms.

For the tentative match approach, the previous partitioned centre provides a better approach than the previous vector candidate in the VQ clustering procedure.

The improved AEI criterion is presented which provides a tighter bound than the AEI criterion. For $s = 1$, it is the same as the hypercube approach. For $s = k$, it is the same as the AEI criterion. It provides the advanced criterion by adapting the parameters s and h from 1 to k. More than 40% of the total number of mathematical operations are saved comparing IPC with the APC approach and more than 10% comparing IPC with the APCH approach for 1024 codewords in the VQ clustering algorithm.

REFERENCES

1. R. M. Gray, Vector quantization, *IEEE Mag.* 4–29 (1984).
2. Y. Linde, A. Buzo and R. M. Gray, An algorithm for vector quantizer design, *IEEE Trans. Commun.* **COM-28**(1), 84–95 (1980).
3. F. P. Fischer and E. A. Patrick, A preprocessing algorithm for nearest neighbour design rules, *Proc. Natl. Electron. Conf.* 481–485 (1970).
4. K. Fukunaga and P. M. Narendra, A branch and bound algorithm for computing k-nearest neighbours, *IEEE Trans. Comput.* **C-24**, 750–753 (1975).
5. B. Kamgar-Parsi and L. N. Kanal, An improved branch and bound algorithm for computing k-nearest neighbour, *Pattern Recognition Lett.* **3**, 7–12 (1985).
6. H. Niemann and R. Goppert, An efficient branch-and-bound nearest neighbour classifier, *Pattern Recognition Lett.* **7**, 67–72 (1988).
7. Q. Jiang and W. Zhang, An improved method for finding nearest neighbours, *Pattern Recognition Lett.* **14**, 531–535 (1993).
8. C. Bei and R. M. Gray, An improvement of the minimum distortion encoding algorithm for vector quantization, *IEEE Trans. Commun.* **COM-33**(10), 1132–1133 (1985).
9. D. Y. Cheng, A. Gersho, B. Ramamurthi and Y. Shoham, Fast search algorithms for vector quantization and pattern matching, *IEEE ICASSP* 9.11.1–9.11.4 (1984).
10. E. Vidal, An algorithm for finding nearest neighbours in (approximately) constant average time, *Pattern Recognition Lett.* **4**, 145–157 (1986).
11. M. R. Soleymoni and S. D. Morgera, A high-speed algorithm for vector quantization, *IEEE ICASSP* 1946–1948 (1987).
12. S. H. Chen and J. S. Pan, Fast search algorithm for VQ-based recognition of isolated word, *IEE Proc. I* **136**(6), 391–396 (1989).
13. J. S. Pan, F. R. McInnes and M. A. Jack, Comparison of fast VQ training algorithms, *Aust. Int. Conf. Speech Sci. Technol.* (SST-94), 106–111 (1994).
14. L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Prentice-Hall Press, New York (1978).
15. J. R. Deller, J. G. Proakis and J. H. L. Hansen, *Discrete-Time Processing of Speech Signals*. Macmillan Press, New York (1993).
16. J. S. Pan, Fast speaker independent isolated word recognition system, MS thesis, Department of Communication Engineering, Chiao Tung University, Taiwan (1988).
17. W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, U.K. (1986).

About the Author—J. S. PAN received the B.S. degree in electronic engineering from the Taiwan Institute of Technology, Taiwan, in 1986 and the M.S. degree in communication engineering from the Chiao Tung University, Taiwan, in 1988. He is currently a Ph.D. candidate in the University of Edinburgh, U.K.

About the Author—F.R. McINNES received the B.A. degree in mathematics from the University of Cambridge in 1983 and a Ph.D. for research in automatic speech recognition from the University of Edinburgh in 1988. He is currently a Research Fellow at the University of Edinburgh's Centre for Communication Interface Research (CCIR).

About the Author—M.A. JACK is Professor of Electronic Systems at the University of Edinburgh and is Director of the new Centre for Communication Interface Research at the University. His research interests include work on spoken language engineering, particularly spoken dialogues for automated telephone services. He leads a team of 25 researchers and five Ph.D. students in CCIR. An author of over 150 scientific papers and five textbooks, Professor Jack is a Fellow of the Institution of Electrical Engineers and a Fellow of the Institute of Acoustics.