COMPARISON OF FAST VQ TRAINING ALGORITHMS

J. S. Pan†‡, F. R. McInnes†and M. A. Jack†

†Centre for Communication Interface Research, University of Edinburgh, UK
‡Department of Electronic Engineering, Kaohsiung Institute of Technology, Taiwan

ABSTRACT - Some fast approaches to VQ training for the LBG recursive algorithm are presented and compared. The computational efficiency is based on the number of multiplications, comparisons, additions, and the sum of these three mathematical operations. Experimental results in comparison to conventional VQ training algorithms 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.

INTRODUCTION

Vector quantization (VQ) (Gray, 1984) is a very efficient approach to data compression. The encoder of VQ encodes a given set of k-dimensional data vectors $X=\{X_j|X_j \in R^k; j = 1,\dots,T\}$ with a much smaller set of codewords $C=\{C_i|C_i \in R^k; i = 1,\dots,N\}(N \ll T)$. 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 (Linde et al., 1980) is an efficient VQ training 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 training procedure for 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. Bei and Gray (1985) 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. Vidal (1986) presented the approximating and eliminating search algorithm (AESA) in which the computation time is approximately constant for codeword search in a large codebook size. Soleymoni and Morgera (1987) proposed the absolute error inequality (AEI) elimination to improve the speed of VQ search. Chen and Pan (1989) 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, previous vector candidate and previous partitioned centre, absolute error inequality (AEI) elimination, partial distortion search (PDS) and 2-level partial distortion search, hypercube approach, triangular inequality elimination (TIE) and codebook reorder method are described and applied to VQ training algorithm.

The test materials for these experiments consist of two hundred words recorded from one male speaker. The speech is sampled at a rate of 16 kHz and 13-dimensional cepstrum coefficients are computed over 20 ms-wide frames with a 5 ms frame shift. A total of 20,030 analyzed frames are used in the VQ training experiments.

FAST VQ TRAINING ALGORITHMS

Previous vector candidate and previous partitioned centre

In the VQ training procedure, speech data has the property that the present vector to be classified is usually the same as or close to the classified result of the previous vector. 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 training algorithm. Fig. 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 is the probability that the data vectors belong to the same (previous) partitioned set. The probability is up to 0.949 for 1024 codewords.

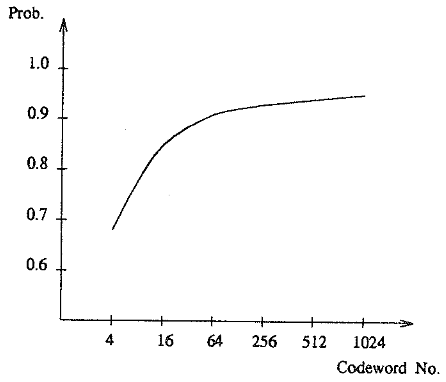

Figure 1: Relationship between the number of codewords and the probability of the data vectors belonging to the previous partitioned set

Absolute error inequality

The absolute error inequality is the mathematical relationship between the city block metric (or l_1) and the Euclidean metric (or l_2). Given one codeword C_t and the square error distortion $D_{min} = D(X_j, C_t)$,

$$\text{if} \qquad \sum_{p=1}^{h} | X_j^p - C_t^p | > \sqrt{kD_{min}}, \tag{1}$$

$$\text{then} \qquad \sum_{p=1}^{k} (X_j^p - C_t^p)^2 > D_{min}, \tag{2}$$

where k is the dimension of the data vector and $h \leq k$.

This means C_t will not be the nearest neighbour to X_j if Eq. 1 is satisfied. The 2-level partial distortion search is the standard PDS which is inserted in the l_1 metric distortion calculation and the square error distortion calculation. Fig. 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 (shown in dimension 14). The distortion of the surviving codewords is calculated by PDS.

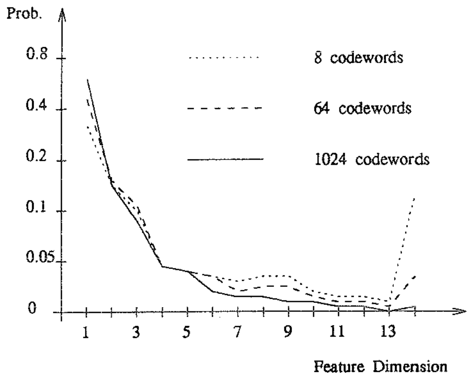

Figure 2: the elimination probability of AEI at each feature dimension

Hypercube approach

Consider an input vector X_j and let the smallest distortion found before checking C_i be D_{min}; then C_i cannot be a better match than the previous one if

$$|X_j^p - C_i^p| > \sqrt{D_{min}} \quad \text{for} \qquad 1 \le p \le k \tag{3}$$

For cepstrum coefficients, the energy compacts in the first coefficient. Normally, the absolute difference of the first coefficients is larger than the difference of the other coefficients. It is simple and efficient to check the difference of the first coefficient only for the hypercube approach.

Triangular inequality elimination

Triangular inequality elimination 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: $V \times V \to R$, which is assumed to fulfill the metric properties:

$$d(x,y) \ge 0; d(x,y) = 0 \ \text{iff} \ \ x = y \tag{4}$$

$$d(x,y) = d(y,x) \tag{5}$$

$$d(x,y) + d(y,z) \ge d(x,z) \tag{6}$$

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

- Criterion 1:
  Given the triangular inequality

$$d(t,C_2) + d(t,C_1) \ge d(C_1,C_2); \tag{7}$$

$$\text{if} \qquad d(C_1,C_2) \ge 2.d(t,C_1), \tag{8}$$

$$\text{then} \qquad d(t,C_2) \ge d(t,C_1). \tag{9}$$

- Criterion 2:
  Given the triangular inequality

$$d(C_3, C_2) \leq d(t, C_2) + d(t, C_3); \tag{10}$$

$$\text{if} \quad d(C_3, C_2) \geq d(t, C_1) + d(t, C_2), \tag{11}$$

$$\text{then} \quad d(t, C_1) \leq d(t, C_3). \tag{12}$$

- Criterion 3:
  Assume $d(t, C_1) \leq d(t, C_2)$.

$$\text{Given} \quad d(C_3, C_2) \geq d(t, C_2) - d(t, C_3); \tag{13}$$

$$\text{if} \quad d(C_3, C_2) \leq d(t, C_2) - d(t, C_1), \tag{14}$$

$$\text{then} \quad d(t, C_1) \leq d(t, C_3). \tag{15}$$

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

$$\text{if} \quad d(t, C_1) \leq |d(C_3, C_2) - d(t, C_2)|, \tag{16}$$

$$\text{then} \quad d(t, C_1) \leq d(t, C_3). \tag{17}$$

To use Criterion 1, these distortions between all pairs of codewords are calculated in advance. If Eq. 8 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 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 $i = 1, 2, ..., N; j = 1, 2, ..., N$. Here N is the number of codewords. The overhead of criterion 1 is to establish distortion table in which $N(N-1)k/2$ multiplications and $N(N-1)(2k-1)/2$ additions are needed. The physical meaning of Criterion 2 and 3 can be described as following :

If the codeword $C_i, i \neq 1, 2$, does not locate between the two concentric circles centered on C_2 with radii $d(t, C_2) \pm d(t, C_1)$, the computation of its distortion to the test 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 Criterion 2 and 3 will induce square root computation, it is simple and efficient to use Criterion 1 only. Fig. 3 illustrates the elimination probability using TIE combined with previous partitioned centre in VQ training procedure. For 1024 codewords, the elimination probability is 0.949.

Codebook reorder method

The codebook reorder method is to reorder these codewords so as to increase the search efficiency. For the speech encoding, it chooses the nearest codeword of the previous frame as tentative match to encode the present frame. From training data, calculate the probability of these codewords to be encoded and arrange these codewords in the 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 the increasing order of distortion between the most probable codeword and the other codewords. In the VQ training procedure, the previous vector candidate or previous partitioned centre can be chosen as the most probable codeword so as to create the state table. $N(N-1)(N-2)/2$ comparisons are operated to establish the state table using bubble sort method.

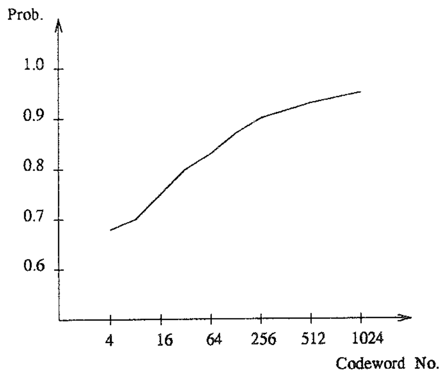

Figure 3: Relationship between the number of codewords and the elimination probability using TIE

EXPERIMENTAL RESULTS

To verify these fast algorithms, the mathematical operations (multiplications, comparisons, and additions) are used to calculate the computational efficiency. Nine approaches are compared in VQ training procedure. The conventional exhaustive method is referred to as CVT-type. P-type and T-type are approaches using PDS and TIE in codebook design. TPC-type is the algorithm using previous partitioned centre as the most probable matching with TIE and PDS to reduce the training time. TPCR-type is the TPC-type with codebook reorder method. It is called APC-type if the previous partitioned centre is used as the tentative match with AEI and 2-level PDS to accelerate the training speed. The previous vector candidate instead of previous partitioned centre in APC-type is called APV-type. APTC-type is the algorithm combined TIE, AEI, PDS and previous partitioned centre. APCH-type is the addition of hypercube approach to APC-type.

The experimental results for 8 codewords and 1024 codewords are shown in Table 1 and Table 2. For general processor architecture, the multiplication operation is more expensive than the comparison operation and addition operation. It is better to use APCH-type algorithm for large codebook size and TPC-type algorithm or TPCR-type algorithm for small codebook size. Table 1 and Table 2 also illustrate the total mathematical operation number. In terms of the total number of operations, TPC-type outperforms all of the above algorithms. It needs extra computation time to generate the distortion table for TIE approach that is why the total number of multiplications in ATPC-type and TPCR-type are larger than APC-type, APV-type and APCH-type for 1024 codewords. The codebook reorder method is not very efficient in VQ training algorithm owing to the overhead of sorting procedure. In small codebook size, TPCR-type is excellent. It is not superior compared with APCH-type, APC-type, APV-type, ATPC-type and TPC-type for large codebook size.

CONCLUSIONS

In this paper, some fast VQ training 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 hypercube approach, AEI and 2-level PDS is the most suitable for computer architectures in which the complexity of comparisons is negligible with respect to that of multiplications. It outperforms the other algorithms if the combination of previous partitioned centre, TIE and PDS is used for the processor architectures

method	mul.	cmp.	add.	sum	saving in mul.
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 1: computational complexity of VQ training for 8 codewords ($\times 10^6$)

method	mul.	cmp.	add.	sum	saving in mul.
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	5191	282	5616	96.5 %
P	777	777	1259	2813	81.1 %
T	1887	453	3649	5989	54.1 %
CVT	4109	315	7922	12346	0 %

Table 2: computational complexity of VQ training for 1024 codewords ($\times 10^6$)

such as those based on the Harvard architecture in which comparisons are computationally expensive.

REFERENCES

Linde, Y., Buzo, A. and Gray, R. M (1980) An Algorithm for Vector Quantizer Design, *IEEE Trans. on Communications*, COM-28(1), 84–95

Gray, R. M. (1984) Vector Quantization, *IEEE Magazine*, 4–29.

Bei, C. and Gray, R. M. (1985) An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization, *IEEE Trans. on Communications*, COM-33(10), 1132–1133.

Vidal, E. (1986) An Algorithm for Finding Nearest Neighbours in (Approximately) Constant Average Time, *Pattern Recogn. Lett.*, 4(3), 145–157.

Soleymoni, M. R. and Morgera, S. D. (1987) A High-Speed Algorithm for Vector Quantization, *IEEE ICASSP*, 1946–1948.

Chen, S. H. and Pan, J. S. (1989) Fast Search Algorithm for VQ-based Recognition of Isolated Word, *IEE Proceedings-I*, 136(6), 391-396.