AN EFFICIENT APPROXIMATION-ELIMINATION ALGORITHM FOR FAST NEAREST-NEIGHBOR SEARCH

V. Ramasubramanian and K. K. Paliwal
Computer Systems and Communication Group
Tata Institute of Fundamental Research
Homi Bhabha Road, Bombay – 400 005, India

Abstract

In this paper, we present an efficient algorithm for fast nearest-neighbour search in multidimensional space under a so called approximation-elimination framework. The algorithm is based on a new approximation procedure which selects codevectors for distance computation in the close proximity of the test vector and eliminates codevectors using the triangle inequality based elimination. The algorithm is studied in the context of vector quantization of speech and compared with related algorithms proposed earlier. It is shown to be more efficient in terms of reducing the main search complexity, overhead costs and storage.

1. Introduction

Nearest neighbour search consists in finding the closest point to a query point among N points in K-dimensional space. This search is widely used in several areas such as pattern classification, nonparametric density estimation and data compression using vector quantization. Reducing the complexity of nearest-neighbour search is of considerable interest in these areas, and particularly in vector quantization encoding where the complexity increases exponentially with dimension K [1]. In this paper, we discuss fast nearest-neighbour search in the context of vector quantization. Vector quantization encoding is the minimum-distortion quantization of a vector $\mathbf{x} = (x_1, \ldots, x_K)$, (referred to as the *test vector*) using a given set of N K-dimensional *codevectors* called the *codebook* $\mathbf{C} = \{\mathbf{c}_j\}_{j=1,\ldots,N}$, of size N, under some distance measure $d(\mathbf{x}, \mathbf{y})$. This involves finding the nearest-neighbour of $\mathbf{x}$ in $\mathbf{C}$, given by $q(\mathbf{x}) = \mathbf{c}_k : d(\mathbf{x}, \mathbf{c}_k) \leq d(\mathbf{x}, \mathbf{c}_j),\ j = 1, \ldots, N$, which requires N vector distance computations $d(\mathbf{x}, \mathbf{c}_j)$ using the exhaustive full-search.

One of the common approaches to fast nearest-neighbour search is the use of inexpensive elimination rules to eliminate codevectors which cannot be nearer to the testvector $\mathbf{x}$ than the current nearest-neighbour $\mathbf{c}_n$. Among these, the triangle inequality based elimination rule, which is applicable when the distance measure is a metric is a popular one [4]. If $d_n = d(\mathbf{x}, \mathbf{c}_n)$ is the current-nearest neighbour distance, then the triangle inequality based elimination rule rejects $\mathbf{c}_j$ if $d(\mathbf{c}_j, \mathbf{a}) > d(\mathbf{x}, \mathbf{a}) + d_n = r''$ or $d(\mathbf{c}_j, \mathbf{a}) < d(\mathbf{x}, \mathbf{a}) - d_n = r'$, where $\mathbf{a}$ is some point in $\mathcal{R}^K$. This elimination requires the N codevector - anchor point distances $\{d(\mathbf{c}_j, \mathbf{a})\}_{j=1,\ldots,N}$, and the test vector - anchor point distance $d(\mathbf{x}, \mathbf{a})$. The efficiency of this triangle-inequality elimination increases with the use of more distinct anchor points, $\{\mathbf{a}_m\}_{m=1,\ldots,M}$, the codevectors retained after elimination being those present inside the intersection volume of the multiple hyperannulus corresponding to $\{\mathbf{a}_m\}_{m=1,\ldots,M}$.

Under this triangle-inequality based elimination in vector space, the general issue of optimal number and location of multiple anchor points is important and has been addressed in a distance coordinate representation framework. Here a vector in K-dimensional space is represented uniquely by its $K+1$ Euclidean distances (termed the distance coordinates of the vector) to some $K+1$ fixed points [2]. Based on this formulation, efficient search algorithms namely, the Fixed Anchor Points – Approximation Elimination Search Algorithm (FAP-AESA) and Incremental Fixed Anchor Points – Approximation Elimination Search Algorithm (IFAP-AESA) were proposed. In the approximation-elimination search framework, in addition to using efficient elimination rules to reject codevectors without distance computation, explicit importance is given to the sequence in which the codevectors are examined as determined approximately by their relative closeness to the test vector. These algorithms were shown to be more efficient fixed point equivalents of the Approximation Elimination Search Algorithm (AESA) proposed earlier [4] for search in vector space. In comparison to the AESA algorithm which has a very high $O(N^2)$ storage complexity, these algorithms use only $O(N)$ storage while achieving complexity reductions closely comparable to AESA.

0-7803-0532-9/92 $3.00

1.1 Distance coordinate representation

In the distance coordinate representation, a vector $\mathbf{x}$ in $\mathcal{R}^K$ is uniquely represented by its $K+1$ distances $\mathbf{d}^x = \{d_m^x = d(\mathbf{x}, \mathbf{a}_m)\}_{m=0,\ldots,K}$, (referred to as the *(spherical) distance coordinates* of $\mathbf{x}$) from $K+1$ fixed points $\{\mathbf{a}_m\}_{m=0,\ldots,K}$ (called the anchor points, henceforth) which are such that the the K difference vectors $\{(\mathbf{a}_m - \mathbf{a}_0)^T\}_{m=1,\ldots,K}$ are linearly independent [2]. When either $M < K+1$, or $M = K+1$ but the anchor points do not satisfy the linear independent condition, the M distances of $\mathbf{x}$ from the M anchor points will not locate $\mathbf{x}$ uniquely. These anchor points are referred as 'sub-optimal'. An example of the anchor point configuration used in [2] which satisfies the optimal conditions for unique point location is: $\mathbf{a}_0$ at the origin and $(\mathbf{a}_1, \ldots, \mathbf{a}_K)$ along the coordinate axes at equal distances from the origin. The N codevectors $\{\mathbf{c}_j\}_{j=1,\ldots,N}$ are represented by their $K+1$ distance coordinates $\mathbf{d}^{c_j} = \{d_m^j = d(\mathbf{c}_j, \mathbf{a}_m)\}_{m=0,\ldots,K}$ using some specific optimal anchor point configuration. Given a test vector $\mathbf{x}$, its $K+1$ distance coordinates $\mathbf{d}^x$ are also computed from the given optimal $K+1$ anchor points.

In the Fixed Anchor Point - Approximation Elimination Search Algorithm (FAP-AESA), the $K+1$ fixed anchor points are used all at the same time for approximation and elimination. FAP-AESA uses the approximation criterion as in Vidal [4] for selecting candidate codevectors close to the test vector $\mathbf{x}$. The main computational overhead in FAP-AESA is the full approximation that has to be carried out in the first step. In order to reduce this overhead cost incurred due to the use of all anchor points, the Incremental FAP-AESA (IFAP-AESA) [2] uses the anchor points incrementally as in AESA with the main difference that only a prespecified number of anchor points from an anchor point configuration is used. The performance of IFAP-AESA was found to be comparable to that of FAP-AESA for optimal $K+1$ anchor points while having lower overhead costs. However, given optimal $K+1$ fixed anchor points, this is actually a trade-off in efficiency of search since use of less than $K+1$ anchor points in the early stages of search in IFAP-AESA corresponds to poorer approximation-elimination efficiency than FAP-AESA. Therefore, it is an interesting and important issue to find other alternate approximation procedures, which employ all $K+1$ anchor points with the advantage of exploiting the optimal point location property, but, have lower approximation costs.

In this paper, we propose and study a search algorithm termed 'Spherical-GRid Intersection Detection Search' (S-GRIDS) which is based on a new approximation procedure, termed the grid-intersection detection search (GRIDS). This locates candidate codevectors in the proximity of the test vector with very low overhead approximation costs, while requiring some additional storage in comparison to FAP-AESA.

2. Grid intersection detection search (GRIDS)

2.1 The grid structure

Given $\mathbf{C} = \{\mathbf{c}_j\}_{j=1,\ldots,N}$, and M anchor points $\{\mathbf{a}_0, \ldots, \mathbf{a}_{M-1}\}$, for any anchor point $\mathbf{a}_m$, the distances $\{d_m^j = d(\mathbf{c}_j, \mathbf{a}_m)\}_{j=1,\ldots,N}$ are the distances of the N codevectors $\{\mathbf{c}_j\}_{j=1,\ldots,N}$ from $\mathbf{a}_m$. Let the index sequence $I_m : \{k_1, k_2, \ldots, k_N\}_m$ be a permutation of $\{1, \ldots, N\}$, such that $d_m^{k_1} \leq d_m^{k_2} \leq \ldots \leq d_m^{k_N}$. An index in this list $\{k_1, k_2, \ldots, k_N\}_m$ is denoted as k_i^m, with the corresponding 'codevector - anchor point' distance $d_m^{k_i} = d(\mathbf{c}_{k_i}, \mathbf{a}_m)$. For M anchor points $\mathbf{a}_0, \ldots, \mathbf{a}_{M-1}$, the M index lists $\{\{k_i^m\}_{i=1,\ldots,N}\}_{m=0,\ldots,M-1}$, specifying the ordering of the codevector according to their distances from each of the anchor points, represents an implicit spherical grid structure formed by the M sets of N concentric spherical shells, each set centered at an anchor point. A codevector $\mathbf{c}_j$ is located at the intersection of M shells, (one from each anchor point shell set), whose indices are such that $\{k_i^m = j\}_{m=0,\ldots,M-1}$. Under optimal anchor point configuration satisfying unique point location conditions, the intersection of $M = K+1$ shells will be a point of 0-dimensionality and the location of a codevector $\mathbf{c}_j$ is represented uniquely in the index list $I_m, m = 0, \ldots, K$, by the set of indices $\{k_i^m : k_i^m = j\}_{m=0,\ldots,K}$.

Given a test vector $\mathbf{x}$, its M distance coordinates are $\mathbf{d}^x = \{d_m^x\}_{m=0,\ldots,M-1}$; d_m^x represents the 'test-vector shell' centered at anchor point $\mathbf{a}_m$. Let the index k_l^m in the index list of anchor point $\mathbf{a}_m$ be such that $d_m^{k_l} < d_m^x \leq d_m^{k_l+1}$, i.e., (k_l^m, k_r^m) (using $r = l+1$). The indices of (k_l^m, k_r^m) in I_m are henceforth referred as (l_m, r_m). Given $\{\{k_i^m\}_{i=1,\ldots,N}\}_{m=0,\ldots,M-1}$ which specify the distance ordering $\{d_m^{k_i}\}_{i=1,\ldots,N}$ for each anchor point, $(l_m, r_m), m = 0, \ldots, M-1$ are determined by M binary searches.

Let the nearest-neighbour of $\mathbf{x}$ be $\mathbf{c}_k$. The distance coordinates of $\mathbf{c}_k$, $\{d_m^k\}_{m=0,\ldots,M-1}$, will then be small perturbations of the distance coordinates of $\mathbf{x}$, $\{d_m^x\}_{m=0,\ldots,M-1}$. Therefore, the codevector shell of $\mathbf{c}_k$ will be located close to the test-vector shell in the concentric shell set of each anchor point. Accordingly, the indices representing the codevector-shell of $\mathbf{c}_k$ will be present in the close proximity of the indices $(k_l^m, k_r^m), m = 0, \ldots, M-1$. Therefore, a codevector close to $\mathbf{x}$ will have its set of M indices in the proximity of the indices $(l_m, r_m), m = 0, \ldots, M-1$. In order to find a codevector $\mathbf{c}_j$ from the ordered index list, it is sufficient to find the M indices, one

corresponding to each anchor point list $\{k_i^m\}_{i=0,\ldots,N}$, such that $\{k_i^m = j\}_{m=0,\ldots,M-1}$. In particular, given $(l_m, r_m), m = 0, \ldots, M-1$ (corresponding to the two codevector-shells adjacent to the test vector shell), the index list $\{k_i^m\}_{i=1,\ldots,N}$ can be examined on either side of $(l_m, r_m), m = 0, \ldots, M-1$ to obtain the M indices of a codevector located close to the test vector $\mathbf{x}$. The approximation procedure for locating candidate codevectors in the close proximity of the test vector involves considering the issue of how the indices are examined starting from (and around) (l_m, r_m) and how the codevector indices occurring M times in the M index lists $\{k_1^m, k_2^m, \ldots, k_N^m\}_{m=0,\ldots,M-1}$ are determined efficiently.

2.2 Intersection detection

Given M lists of indices, such as for instance, the indices $\{\ldots, k_{l-2}^m, k_{l-1}^m, k_l^m, k_{l+1}^m, k_{l+2}^m, \ldots\}_{m=0,\ldots,M-1}$ around (k_l^m, k_{l+1}^m), the indices common to them can be obtained by an multi-set intersection procedure based on obtaining the 'intesection count' of the indices in the sets. This procedure is based on the simple observation that an index common to the M lists has to occur M times in the collection of the M lists. Thus, it suffices to keep count of the number of times each index has occurred in the M lists and the index which has a count value of M is the common index. This multi-set intersection count is carried out as follows: initialize a counter $C[1 \ldots N]$ of size N to zero and scan through the indices of the M lists, using each index k_i^m as a pointer to the counter array and increment the count value at that location, i.e., $C[k_i^m]$, by one; the index k_i^m for which $C[k_i^m] = M$, is detected as a common index.

2.3 Expansion search procedure

Based on this principle of finding a common index in M lists, the approximation procedure for locating codevectors in the proximity of the test vector essentially involves considering the indices in the index list I_m around $(l_m, r_m), m = 0, \ldots, M-1$ in some order, for use as the index to the intersection counter. This is done as follows: Cycle m through 0 to $M-1$, and for each m, the index whose corresponding shell is nearest to the test vector shell along that distance coordinate is chosen, i.e., l_m is chosen if $|d_m^{k_l} - d_m^x| \leq |d_m^{k_r} - d_m^x|$, else, r_m is chosen and let $i = k_l^m$ or k_r^m accordingly; increment $C[i]$ by one. If $C[i] \neq M$, l_m is set to $l_m - 1$ (if l_m was chosen for incrementing) and r_m is set to $r_m + 1$ (if r_m was chosen) and continue the intersection detection search to consider l_m or r_m from next list. If $C[i] = M$ after an increment step in the above cyclic expansion, the codevector $\mathbf{c}_i$ is said to have been detected and it is selected for computing the distance with the test vector $\mathbf{x}$. If $d(\mathbf{c}_i, \mathbf{x}) \leq d_n$, where d_n is the current nearest-neighbour distance (initially set to a large number), then the current nearest-neighbour $\mathbf{c}_n$ is updated to $\mathbf{c}_i$ and an elimination is carried out using the triangle-inequality elimination with the M anchor points as $\mathbf{C} = \mathbf{C} - \{\mathbf{c}_j : d_m^{c_j} < d_m^x - d_n$ or $d_m^{c_j} > d_m^x + d_n$ for $m = 0, \ldots, M-1\}$. This can be realized in the grid structure by the truncation of the index lists using the hyperannulus bounds of the current nearest-neighbour ball and pruning the grid with the removal of codevector indices rejected by the truncation.

Subsequent to elimination, the search continues to find the next candidate codevector from the remaining indices in the index lists. An anchor point index list gets exhausted on one or both sides during the expansion procedure or after the elimination steps and the list is removed from the search. The search terminates when the codevector set to be searched is empty, i.e., when all the anchor point index lists are exhausted.

2.4 Approximation efficiency

While the above procedure selects a codevector by the common occurrence of its index from among the M index lists, the issue of the optimality of the anchor point configuration determines how close these selected candidate codevectors are to the test vector. For $M = K + 1$ and optimal anchor point configuration, a codevector is located uniquely as a 0-dimensionality point intersection in the grid and is represented by the index list $\{k_i^m = j\}_{m=0,\ldots,K}$. Thus, finding the indices $\{k_i^m : k_i^m = j\}_{m=0,\ldots,K}$, in the proximity of the indices (l_m, r_m) will amount to detecting a codevector as a 0-dimensionality intersection in the close proximity of the test vector in the spherical grid structure. However, for $M < K + 1$, finding M indices $\{k_i^m : k_i^m = j\}_{m=0,\ldots,M-1}$ in the proximity of the indices (l_m, r_m) can amount to locating a codevector which is actually at a large distance from the test vector, but whose M sub-optimal distance coordinates, individually, are in the close proximity of the test vector distance coordinates.

3. Sethi's 3-point algorithm [3]

The S-GRIDS algorithm proposed here is also a correct generalization and improvement over Sethi's algorithm [3] which is referred here as the '3-point algorithm' as it uses 3 reference (anchor) points. In terms of the search described above, the 3-point algorithm is sub-optimal in three ways: i) the search terminates with the first candidate codevector as the nearest neighbour to the test vector. and hence, does not ensure optimal solution, ii) the number of anchor points used for search is only 3, i.e., it corresponds to the use of $M = 3$ in the GRIDS search. From the

approximation efficiency considerations (sec. 2.4), the 3-point algorithm can be seen to be acceptably suboptimal only for the planar ($K = 2$) case but highly (and unacceptably) suboptimal for higher dimensions with gross inaccuracies in the nearest-neighbour determination, iii) no emphasis is given to the location of the 3 anchor points used.

4. Experiments and results

In Table I, we compare the performance of S-GRIDS algorithm with other related algorithms — AESA [4], FAP-AESA [2], and Sethi's 3-point algorithm [3] in the context of vector quantization of speech waveform data. Here, the performance is compared for $K = 8$ and $N = 1024$ in terms of average and maximum number of codevectors checked ($\overline{nc}$, $\widehat{nc}$), search optimality, storage and approximation overhead cost. The search optimality is given in terms of SNR and percentage error (%ERR) in nearest-neighbour classification corresponding to the final nearest-neighbour determined by the search. The following main points can be observed from this table: i) S-GRIDS offers complexity reductions comparable to AESA while having only $O(N)$ storage complexity as against the exhorbitant $O(N^2)$ storage complexity of AESA, ii) S-GRIDS has significantly lower approximation overhead cost than FAP-AESA while using all the $K + 1$ anchor points in achieving comparable complexity reduction. This, however, requires increased storage to maintain the ordered distance coordinates and index lists, iii) Sethi's 3-point algorithm is extremely sub-optimal in its nearest-neighbour solution: it has a very low SNR of 1.39 in comparison to the full-search SNR of 12.51 and high percentage error of 66.1%), iv) S-GRIDS can be seen to have the best overall performance among the various algorithms, when all factors such as average, worst-case complexity, approximation overhead cost and storage are taken into account. Its excellent overall complexity reduction over the exhaustive full-search can be noted.

5. Conclusions

In this paper, we have proposed an approximation elimination algorithm using the distance coordinate formulation and a new approximation-elimination procedure termed grid intersection detection search (GRIDS). This procedure examines codevectors in the close proximity of the test vector in a spherical grid structure created by the ordered distance coordinates of the codevectors from multiple anchor points. This procedure is shown to be an efficient alternative to related algorithms proposed in the literature earlier such as the AESA [4] and FAP-AESA [2] achieving comparable complexity reductions while incurring very low approximation costs. The proposed algorithm has also been shown to be a correct generalization and improvement over the 3-point algorithm proposed earlier. Based on the distance coordinate representation, the 3-point algorithm is shown to be acceptably suboptimal only for the planar case but highly (and unacceptably) suboptimal for higher dimensions with gross inaccuracies in the nearest-neighbour determination.

References

[1] J. Makhoul, S. Roucos and H. Gish, "Vector quantization in speech coding", *Proc. IEEE*, vol. 73, pp. 1555–1588, Nov. 1985.

[2] V. Ramasubramanian and K. K. Paliwal, "An efficient approximation-elimination algorithm for fast nearest-neighbor search based on a spherical distance coordinate formulation", *Signal Processing V: Theories and Applications*, EUSIPCO-90.

[3] I. K. Sethi, "A fast algorithm for recognizing nearest neighbors", *IEEE Trans. Syst. Man, Cybern.*, vol. SMC-11, pp. 245-249, Mar. 1981.

[4] E. Vidal, "An algorithm for finding nearest neighbors in (approximately) constant average time complexity", *Pattern Recognition Letters*, No. 4, pp. 145–157, 1986.

Table I - Comparison of S-GRIDS with related algorithms — AESA, FAP-AESA, Sethi's 3-pt algorithm
Dimension $K = 8$, Codebook size $N = 1024$; Data: 5000 vectors of speech waveform

Algorithm	$\overline{M}$	$\widehat{M}$	$\overline{nc}$	$\widehat{nc}$	SNR	%ERR	Approx-cost	Storage	(bytes)
FULL-SEARCH	—	—	1024	1024	12.51	0	—	—	—
AESA	8.9	80	8.9	80	12.51	0	185	$4N(N-1)/2$	2095104
FAP-AESA	9	9	4.7	83	12.51	0	1057	$4N(K+1)$	36864
S-GRIDS	9	9	3.9	87	12.51	0	129	$6N(K+1)+2N$	57344
SETHI'S 3-Pt ALGORITHM	3	3	3	3	1.39	66.1	13.4	$4(3N)+2(4N)$	20480

($\overline{M}$, $\widehat{M}$): (average, maximum) number of anchor points used