J. Inst. Maths Applics (1979) **24**, 59–70

Nearest Neighbour Searches and the Curse of Dimensionality

R. B. MARIMONT

Office of Biometry and Epidemiology, OD
National Institute of Neurological
and Communicative Diseases and Stroke

AND

Mathematical Research Branch, National Institute of Arthritis,
Metabolism and Digestive Diseases, National Institutes of Health, Bethesda,
Maryland 20014

AND

M. B. SHAPIRO

Division of Computer Research and Technology,
National Institutes of Health,
Bethesda, Maryland 20014

[Received 1 February 1978 and in revised form 13 December 1978]

It is shown that efficiency of cut-off algorithms for nearest neighbour searches depends on the ratio of variance in a lower bound space B to variance in the original space L. The usual choice of a one dimensional B space fails for a high dimensional L space because this ratio is then low. If the dimensionality of B is about half that of L the equivalent of no more than 75% of the full distance computations need be done, independent of the dimensionality of L space. If the variance in B space can be increased by either sorting or the principal components transformation performance is appreciably better.

1. Introduction

THE NEAREST NEIGHBOUR problem arises in many computer applications. "Nearness" may refer to propinquity in physical space—for example, airplanes in a traffic pattern. Such problems involve at most three spatial dimensions, and time, for real time applications. "Nearness" may also refer to similarity in some important respect—for example, in a physiological sense, men are "nearer" to apes than they are to ants. Problems involving pattern recognition, classification, matching, and information retrieval share the concept of a real feature space, with coordinates and a metric defined so that points geometrically "near" to each other in that space represent objects similar to each other. The dimensionality of the feature space may be very high.

Assume that we have established such a space, and have located within it a number of points, or vectors, each representing some known entity. These points are known as prototypes, and we shall refer to them collectively as the library. Given a new point

0020-2932/79/010059+12 $02.00/0

(the query point) in the space, we wish to find which of the prototypes is its nearest neighbour. For example, the prototypes could be known diseases, and the query point the appropriate numbers representing the condition of a patient with an undiagnosed disease. The nearest neighbour would then represent the disease whose signs, symptoms etc. most closely match the patient's in some sense, and would therefore be a good first guess at a diagnosis.

The brute force approach to finding the nearest neighbour of a query point is to compute its distance to each of the prototypes, and to select the smallest one. It is obvious that if the library is large, and the space is of high dimensionality, this can be a lengthy computation. If there are many query points to be identified and a rapid response is needed, the brute force method is impractical.

There have been recent striking advances in algorithms for searching moderately large files of modest dimensionality (between 2 and 8). There are the cut-off methods of Burkhard & Keller (1973) and Friedman, Baskett & Shustek (1975), and the recursive methods of Fukunaga & Narendra (1975), and Friedman, Bentley & Finkel (1975) (k–d trees). Since Friedman is first author of two of the above references, we shall avoid repetition of triple names by referring to them by their second authors, Baskett, and Bentley, respectively. Baskett's method, sometimes referred to as a one dimensional projection, is called by Bentley the sorting algorithm. Baskett and Bentley present both theoretical analyses and experimental results. In addition Bentley presents a direct comparison of the one dimension projection and k–d tree methods.

It is difficult to estimate the effectiveness of these algorithms for high dimensional data. For example the approximations used in the derivations in Bentley and Baskett are invalid for high dimensional spaces. We will follow Baskett and use d for the dimensionality of the space, N for the number of points, and k for the number of nearest neighbours sought. Since the advantage in all these algorithms lies in reducing the number of distance computations necessary for finding the nearest neighbours, a figure of merit is the expected number of distance computations divided by the number of points in the library. We will use E to denote this reduction ratio. For the brute force method, obviously, $E = 1$. From the abstract of Baskett, we have the value of

$$E = \frac{1}{\sqrt{\pi}} [kd\Gamma(d/2)]^{1/d} 2^{(1-1/d)} N^{-1/d}. \tag{1}$$

From Bentley equation (3), for $k = 1$

$$E = \frac{1}{N} 2^d \quad \text{(independent of } N\text{)}. \tag{2}$$

Evaluation of equation (1) for $N = 100$, leads to the nonsensical result of $E > 1$ for $d > 12$. For equation (2), $E > 1$ for $N < 2^d$. Since both of these equations agree reasonably well with the experimental results, for $2 \leqslant d \leqslant 8$, it seems likely that the errors are caused by assumptions invalid for higher d.

One can estimate roughly the behaviour for high dimensionality by extrapolating from the change in performance from $d = 2$ to $d = 8$. The time per query (for $N \cong 8200$), for the k–d tree method, increases by a factor of **31** as d increases from 2 to

8. It seems likely, from this number that for substantially higher values of d, even the k–d procedure would be ineffective in reducing computing time. The data of Fukunaga and Narendra is not as complete as Bentley's and the dependence on dimensionality cannot be estimated.

For Burkhard's method, studies by Shapiro (1977) showed that for a ten dimensional space, only about 16% of the library could be cut-off (with one reference point). For the best coordinate method, Baskett shows a cut-off of approximately 40% for 8 dimensions, and a rapid reduction for higher d.

What one may conclude from available data is that the behaviour of these algorithms at dimensionalities higher than 8 is not known, but that the degradation between $d = 2$ and $d = 8$ suggests that for substantially higher values of d, such algorithms would be ineffective.

This paper suggests a method for which the reduction ratio E is largely independent of d, and which therefore may be useful in regions of d in which known algorithms deteriorate. Since this method requires assessing each record for at least a partial distance computation it is not suggested for applications where the file is so large that retrieval from external storage is the limiting factor. It is applicable to those problems in which d is large and the library file fits into fast memory, so that the number of distance computations is the limiting factor.

We shall analyse the reasons for the ineffectiveness of the sorting algorithm and present a method whereby, in the worst case, computing time can be reduced by approximately 25% over brute force methods, independent of the dimensionality of the space.

2. The General Problem

The purpose of using a cut-off criterion is that for some subset of the library vectors, one need compute only a simple lower bound for the distance to the query vector, rather than the distance itself. (We use Euclidean distance throughout.) To facilitate the discussion, we will use the following symbols.

$\{x_i\}$, $i = 1, 2, \ldots, n$, the set of fixed (library) column vectors.
X the matrix of the x_i $(d \times n)$.
X^* the conjugate transpose of x. Since our matrices are real, X^* is the transpose of X.
L the d dimensional space in which the vectors x_i lie.
q the query vector.
Δ_i the Euclidean distance from q to $x_i = \|x_i - q\|$.
Δ_o the distance from q to its nearest neighbour; Δ_a the current estimate of Δ_o.
B a space (the image space) into which L is mapped by the transformation $i(x)$.
$i(x_i)$ the image of x_i in B.
δ_i the Euclidean distance from $i(q)$ to $i(x_i) = \|i(q) - i(x_i)\|$, i.e. the distance between the images of x_i and q.

The transformation $i(x)$ must possess the property that

$$\|i(x) - i(y)\| \leqslant \|x - y\| \quad \text{for all } x, y, \quad \text{in } L.$$

That is each distance $\delta(i(x), i(y))$ in B is a lower bound for the corresponding distance in L. Examples of such lower bound transformations are: (1) Projections—B is a subspace of L. (2) Norm transformations—B is the real number line, e.g. $i(x_i) = \|x_i - z\|$, where z is an arbitrary reference vector. Assume that for a given q, all lower bound distance $\{\delta_i\}$, and one actual distance, Δ_a have been computed. Then for all x_i such that $\delta_i \geqslant \Delta_a$, we have $\Delta_i \geqslant \Delta_a$, since $\Delta_i \geqslant \delta_i$; i.e. the x_i are further from q than the vector x_a. (In practice, Δ_a is the current shortest distance, rather than the first.) Denote by c the number of vectors x_i such that $\delta_i > \Delta_a$. Then the maximum number of full distance computations is $N-c$, since c vectors have been cut-off by use of the lower bound distance or cut-off criterion. For each vector that is cut-off, only δ_i rather than Δ_i has been computed. Time saved is approximately proportional to the difference in computing time for distances in L and B space, multiplied by c, the number of cut-off vectors. It is therefore desirable (1) that distances in B be easy to compute, and (2) that c, the number of vectors cut-off, be as large as possible. As is often the case, these two desiderata cannot be simultaneously attained.

3. One Dimensional B Space and the Curse of Dimensionality

The simplest of all distance computations occurs in a one dimensional space, where if coordinate axes are suitably chosen, $\delta(x, y) = |x_1 - y_1|$, i.e., the distance is the absolute value of the algebraic difference of the single coordinates of the two vectors. Also, the records may be stored in sort on the coordinate chosen, so that no sorting of the δ_i is necessary. For these reasons, two commonly used cut-off methods choose B, the lower bound space, to be one dimensional. The mapping functions are

1. $i(x) = \|x - z\|$ where z is an arbitrary vector, called the reference vector. This is used by Burkhard & Keller (1973). It is a lower bound transformation since by the triangular inequality

$$\delta(i(x), i(y)) = |i(x) - i(y)| = \big|\|x - z\| - \|y - z\|\big| \leqslant \|x - y\|.$$

2. $i(x) = x_k$, where x_k is the coordinate in that dimension of L in which the density of x_{ki} in the neighbourhood of q_k is a minimum. B is thus a one dimensional subspace of L, lying along the kth coordinate axis. This method is used by Friedman *et al.* (1975). Here

$$\delta(i(x), i(y)) = |x_k - y_k| \leqslant \left(\sum_{j=1}^{d} (x_j - y_j)^2\right)^{1/2} = \|x - y\|.$$

For both of these mapping functions, the number of vectors cut-off drops rapidly as the dimensionality of L increases.

We will show that this "curse of dimensionality" is not an affliction of the cut-off method in general, but of the choice of B as a one dimensional space, and that if B is of a higher dimensionality (usually about half of d), we can always cut-off about half of the library. Because as the number of dimensions increases the distance computations become longer, and also if subspace dimensionality exceeds 1, the δ_i must be sorted, we save less time per vector cut-off, but we can always reduce computing time substantially, regardless of the dimensions of L.

4. Distance and Variance

Perhaps the best way to understand the curse of dimensionality in this problem is to consider the procedure geometrically. Consider B space, (now of arbitrary dimensionality), the set $i(x_i)$, the images of x_i in B, and $i(q)$, the image of q. By the nature of our mapping, the distances between image vectors are less than those between library vectors. However, for a useful cut-off method, image distances must be comparable to library distances, since only those vectors whose image distances exceed *some* library distance can be cut-off. Draw a hypersphere in B, of radius Δ_o (the distance in L of some point x_o from q) about $i(q)$. The vectors cut-off are those which lie on or outside the hypersphere, or alternately, the vectors for which distances in L must be computed lie within it.

It is known that the mean square distance between column vectors of a matrix is equal to twice the sum of the row variances of the elements, where the row variance,

$$v_i = \frac{1}{N}\sum_{j=1}^{N}(x_{ij}-\overline{x_i})^2,$$

and $\overline{x_i}$, the mean value of the row elements, is given by

$$\overline{x_i} = \frac{1}{N}\sum_{j=1}^{N} x_{ij}.$$

Thus,

$$S = 2N^2V, \tag{3}$$

where S is the sum of the pair wise squared Euclidean distances between the columns of X, and V, the sum of the row variances, will be called the matrix variance.

Consider now the "scaling" of distances involved in Baskett's method. Assume that variances of coordinates are equal in all dimensions. If we denote by V_B the variance in B space, which in this case is one dimension of L, and V_L the total variance, then

$$V_B = \frac{V_L}{d}. \tag{4}$$

Or denoting by S_B and S_L the sums of the squared distances in B and L, we have

$$\frac{S_B}{S_L} = \frac{1}{d}, \tag{5}$$

or that distances in B are very roughly $1/\sqrt{d}$ times those in L, if B is one dimensional and the variances of the coordinates of L are the same for each dimension. The equal variances case gives the worst results with a cut-off procedure. If variances are not equal, and the axis having the largest variance is chosen as B space, then V_B/V_L and S_B/S_L are larger than $1/d$. (This is why the equal variance case is worst.) This distance compression in B space makes obvious the reason for the "curse of dimensionality"; the image vectors are so close together that any hypersphere of radius Δ_o will include most of them, and so few can be cut-off.

This simple analysis leads to the general concept of distance scaling for any choice of B, i.e. from equations (1) and (3)

$$\frac{\overline{\delta_i^2}}{\overline{\Delta_i^2}} = \frac{V_B}{V_L}. \tag{6}$$

Since c, the number of vectors cut-off is an increasing function of $\overline{\delta_i^2}/\Delta_i^2$, it will be an increasing function of V_B/V_L.

Having seen in a general way the defects of mapping a high dimensional L space into a 1 dimensional B space, let us now improve the procedure.

5. Computing the Reduction Ratio for the Standard Case

Consider the total computing done. Let N be the number of points in the library, and let c be the number of points cut-off. Let t_L and t_B be the times for one distance computation in L and B space, and t_{L-B} the time for a distance computation in L space when the distance in B space is known. (In some cases, the computation in B space is part of the computation in L space.) Let t_M be the computing time for mapping a vector from L to B, s be the number of query vectors to be examined, and $h(N)$ be the time to sort N items. Then the computing time for each query vector q is given by

$$T = Nt_B + (N-c)t_{L-B} + h(N) + t_M + (N/s)t_M \tag{7}$$

where $h(N) = 0$ if B is one dimensional. If $s \gg N \gg 1$, then the last two terms of equation (7) can be omitted. We shall assume that this condition holds—a very large query set and a large library, so that we can ignore the expense of mapping L onto B.

If E is the ratio of computing time of this procedure to that of the brute force method, then

$$E = (Nt_B + (N-c)t_{L-B} + h(N))/Nt_L. \tag{8}$$

Let $F = (N-c)/N$. F is the fraction of distances computed. Then

$$E = \frac{t_B + Ft_{L-B}}{t_L} + \frac{h(N)}{Nt_L}. \tag{9}$$

Now assume that B is of dimension r, and that t_B, t_L, t_{L-B} are proportional to r, d, and $d-r$ respectively. Then, writing E and F as functions of r, we have

$$E = \frac{r + (d-r)F(r)}{d} + \frac{h(N)}{Nt_L}. \tag{10}$$

By making assumptions about the distributions of query and library sets, we can compute distributions of δ_i and Δ_o, which allow us to evaluate $F(r)$ in equation (10). The quantity $h(N)$ will be estimated later. We will assume that the query and library sets have the same distributions. Then the distribution of distances between points in the query and library sets is the same as that between points in the library set.

If the elements of a $d \times n$ matrix X are independently and normally distributed with equal means and variance = 0·5, then the distance squared between any two columns is the sum of d squared variates independently and normally distributed with variance = 1, and therefore has the chi-squared distribution with d degrees of freedom. The density function of chi-square is given by Morrison (1976),

$$f(\chi^2) = \frac{2^{-d/2}}{\Gamma(d/2)} (\chi^2)^{(d-2)/2} \exp(-\chi^2/2). \tag{11}$$

To find the distribution of Δ_o, note that if $H(x)$ is the distribution function of any variate, then the distribution of the smallest value, Δ_o, in a sample of size n is given by (see Hogg & Craig, 1978)

$$P(\Delta_o > k) = (1 - H(k))^n. \tag{12}$$

Setting $P = 0{\cdot}5$ to find the median value of Δ_o, which we will denote by Δ_{om}, we find that

$$H(\Delta_{om}) = 1 - 2^{-1/n}. \tag{13}$$

The solid curves of Fig. 1 show distributions of distances for standard libraries in 5, 10, 20, 30, and 40 dimensions. The dashed lines show equation (13) for $n = 100$, 1000, and 10000, and the solid line shows $F = 0{\cdot}5$, the median. As an illustration, the intersection of the curves $d = 40$ and $n = 1000$ gives the median value, approximately 4·1, of the smallest distance in a sample of 1000 distances for standard libraries in 40 dimensions. The ordinate of 4·1 for each curve then gives $F(r)$ of equation (10) for the corresponding value of r. For example, $F(30) = 0{\cdot}02$, $F(20) = 0{\cdot}3$, and $F(10) = 0{\cdot}9$ (the latter cannot of course be read from Fig. 1).

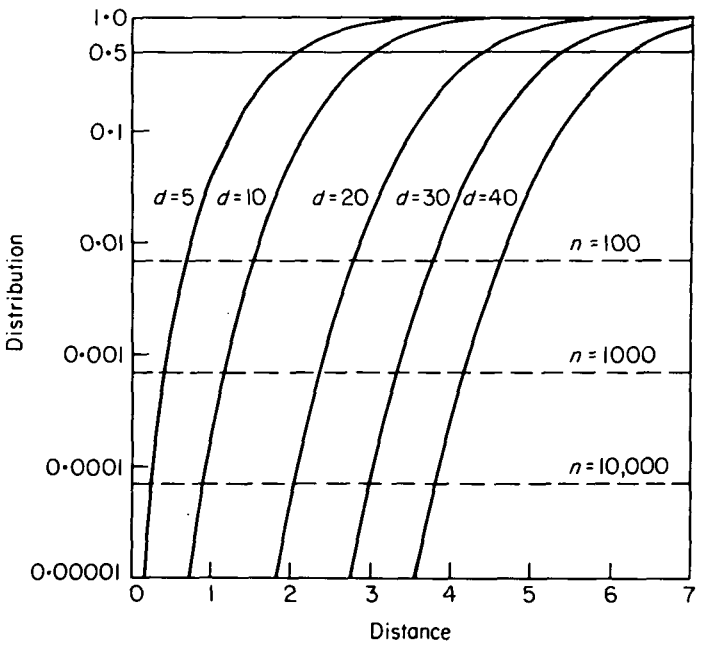

FIG. 1. Standard libraries in d dimensions. Solid lines show distribution of distances. Dashed lines show probability that a distance is equal to the median value of the smallest distance in a sample of size n. See text.

Computation of E (equation 10) requires also an estimate of $h(N)/Nt_L$, the sorting correction if the dimension of $B > 1$. This will depend on the characteristics of the computer, and the algorithm used—it is obviously not necessary to sort all the δ_i, but rather to find $\delta_i < \Delta_o$. However, assuming a complete sort, which has computing time

of order $n \log n$, the correction factor is approximately $(\log n)/(z+2)d$ when z is the ratio of multiplication to addition times. Knuth (1973) uses a value of 5 for his mythical computer MIX. For large computers, z may be as low as 2. We have used 2 for a worst case estimate. Figs. 2 and 3 show the reduction ratio, E (equation 10) a function of r, for libraries of 100, 1000, and 10000 records, in spaces of 20 and 40 dimensions.

For clarity we will describe graphically the computation for one point of a 40 dimensional curve (Fig. 3), the point $r = 30$, on the curve $n = 100$. Refer to Fig. 1. Since $d = 40$, from the intersection of $d = 40$ and $n = 100$ we read $\Delta_{om} = 4{\cdot}5$. From the curve $d = 30$ we read $F(4{\cdot}5) = 0{\cdot}1$, which is the value of $F(r)$ in equation (10). Adding the sorting correction, $(\log n)/4d$, results in the value of $E = 0{\cdot}8$.

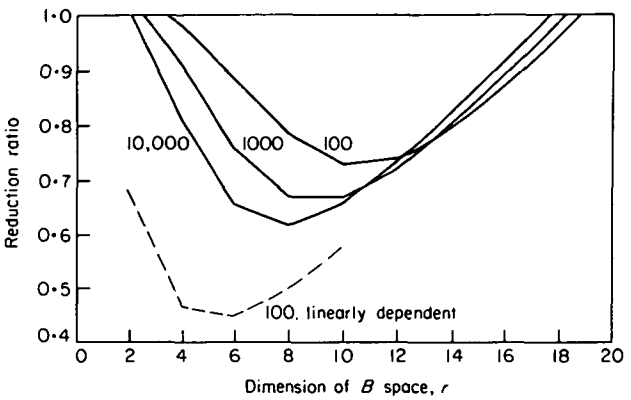

FIG. 2. Reduction ratio (E of equation 10) for libraries of 20 dimensions and 3 values of n. Solid lines show E for standard libraries. Dashed line shows E for linearly dependent library, $n = 100$, optimized by SVD method.

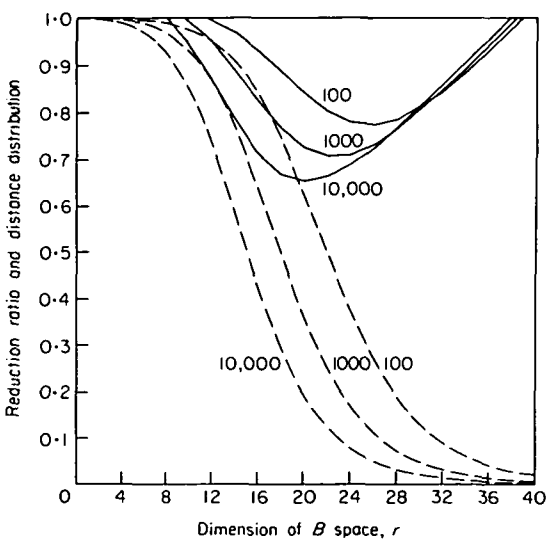

FIG. 3. Libraries of 40 dimensions and 3 values of N. Solid lines show E, reduction ratio. Dashed lines show F, fraction of full distance computations. See equation (10), text.

Performance depends very little on dimensionality—note how little difference there is between the two sets of curves in Figs. 2 and 3. Optimum values of r are in the neighbourhood of $d/2$, with reduction ratios of 0·6 to 75. The fraction of full distances computed, $F(r)$ in equation (10) is also shown in Fig. 3. $F(r)$ at the optimum is in the range of 0·15–0·30.

6. More General Cases—Use of the Singular Value Decomposition

The case of independent equal variance normally distributed coordinates in each dimension (which we will call the standard case) is a worst case for all normal distributions. We will show that the cases of unequal variances, or linearly dependent coordinates, or a combination of both, result in better performance.

Consider a matrix Y, with unequal row variances. Let the rows be ordered so that $V_1 \geqslant V_2 \geqslant \ldots V_d$, where V_i denotes the variance of the ith row of Y. Then for any r, $V_B(Y) \geqslant rV_L/d$, the equal variance case. Therefore $F(V)$, equation (10), would be less than that for the equal variance case, and E also less.

A matrix with equal row variances and linearly dependent rows can be transformed to yield a matrix with unequal row variances and linearly independent rows by means of the singular value decomposition which we will now discuss. Originally devised by Eckart & Young in 1936, the method has evolved in various disciplines under different names. It is called principal components analysis in statistics (Morrison, 1976), the singular value decomposition (SVD) in matrix analysis (Stewart, 1973), and the Karhunen-Loeve expansion in communication theory. For a complete discussion, the reader is referred to these sources. We will present the results in terms of matrix analysis.

Let $\bar{x}$ be the vector of row means of X. Let $A = X - \bar{x}u^*$, $u = (1, 1, 1 \ldots 1)$. Each row of A then has mean zero. There exists an orthogonal matrix W, such that $G = W^*A$ is row orthogonal, and such that the matrix consisting of the first r rows of G is the best choice for B space. The existence of W is guaranteed by the SVD theorem, which states that if A is an $m \times n$ (real) matrix, then there exist unitary (orthogonal) matrices U and W, and a diagonal matrix P such that

$$A^* = UPW^*, \tag{14}$$

where $P = \text{diag}\,(\sigma_1, \sigma_2, \ldots, \sigma_m)$ and $\sigma_1 \geqslant \sigma_2 \geqslant \ldots \geqslant \sigma_m \geqslant 0$. The quantities σ_i are called the singular values of A, and the rth column of W is the singular vector of A corresponding to σ_r. The singular values of A are the positive square roots of the eigenvalues of A^*A, and the singular vectors are the eigenvectors of A^*A. The variance of the rth row of $G = W^*A$ is given by σ_r^2/n.

Only the matrix W^*, rather than the complete SVD decomposition, is needed. If a routine for the eigenvectors of a symmetric matrix is available, W^* can be computed directly. Remember, however, that row means of A must equal 0.

The value of $\phi = V_B/V_L$, for the rotated matrix G, for a given r is

$$\phi_G = \frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{d} \sigma_i^2}. \tag{15}$$

This is a useful formulation, since the denominator,

$$\sum_{i=1}^{d} \sigma_i^2 = \sum_{j=1}^{n} \sum_{i=1}^{d} g_{ij}^2 = \sum_{j=1}^{n} \sum_{i=1}^{d} a_{ij}^2,$$

can be computed directly from A. Therefore only the r largest eigenvalues need be computed in order to evaluate ϕ.

7. Example of the Use of the SVD

The distribution of distances for non-standard cases is not easy to compute. However, computer simulations can be useful. If each row of the matrix X has a normal distribution with $v = 0{\cdot}5$, then intercolumnar distances squared should have the chi-square distribution mentioned above. Since distance distributions for standard matrices of 100 vectors in 5, 10, and 20 dimensions differed from the theoretical values by less than 1%, simulations were used as a valid way to examine non-standard cases.

To illustrate the SVD method, a matrix $G_{20\times100}$ (equation 14) was generated as follows. A standard matrix $M_{20\times100}$ was partitioned horizontally into 4 equal submatrices, M_i, $i = 1, 2, 3, 4$. The linearly dependent matrix $M1$ was generated by adding $0{\cdot}5M_1$ to M_2, M_1 to M_3, and $2M_1$ to M_4. $M1$ was then scaled to yield A, with $\mu = 0$, and $v_i = 0{\cdot}5$ for each row. G was computed from the SVD of A (equation 14), as above. The total variance of G is of course 10. However, for each value of r, ϕ is larger than r/d, the value for the standard case. For example, for $r = 10$ ($r/d = 0{\cdot}5$), ϕ was 0·884. Values of equation (10) for $r = 2, 4, 6, 8, 10$ were computed by generating the distance distributions and values of Δ_{om} for G. The results are shown in the dashed curve of Fig. 2. The optimum value of E is only 0·45, for $r = 6$, compared with 0·73 at $r = 10$, for the standard case. The variance fraction, ϕ, at the optimum $\cong 0{\cdot}7$.

It is clear that the advantage gained from the SVD rotation depends on the inequality of variance of G, the transformed matrix. This in turn depends on the degree of linear dependence of the rows of A. Obviously, if the rows of A are nearly orthogonal little will be gained by the transformation. As noted in equation (15) ϕ for any r can be evaluated by finding the r largest singular values of A.

Knowledge of ϕ, however, does not lead directly to the best choice of r. For the standard cases, $\phi = r/d$. From Fig. 2, $d = 20$, for $n = 100$, 1000, and 10 000, we have ϕ, at optimum E, $\cong 0{\cdot}55$, 0·5, and 0·4 respectively. From Fig. 3, $d = 40$, we have in the same order, $\phi \cong 0{\cdot}65$, 0·55, and 0·5. For the unequal variance case, as we have seen, $\phi \cong 0{\cdot}7$. Since the optimum choice of r will always be determined by the structure of the particular library, the best procedure might be to start with r corresponding to $\phi \cong 0{\cdot}55$, and try a few neighbouring r values in order to locate the optimum.

We have used principal components analysis here to optimize our cut-off procedure, but it is worth noting that its original purpose, to reduce the dimensionality of a set of data, may be applicable to the high dimensional data problems we are dealing with. It is known that dimensionality may be substantially reduced with good approximation to interpoint distances if some rows of the library matrix are not linearly independent. The independence of the library matrix rows should be examined—it may be that a restructuring of the whole file to a lower

dimensional space is feasible.

The principal component rotation can be used, with $r = 1$, to yield a one dimensional subspace = the space of v_1, the singular vector of A corresponding to the largest principal value. The variance of the elements of the first row of the rotated matrix W^*A is given by σ_1^2/n, where σ_1 is the largest principal value.

8. Summary

1. We have shown that a cut-off procedure may be described as a mapping of a $d \times n$ data matrix X in L space onto a lower bound space B (the image space), and the use of the lower bound property to substitute distance computations in image space for the lengthier distance computations in the library space.
2. The effectiveness of the procedure depends on two factors:
   (a) The computing advantage of image distances over object distances (ease).
   (b) The ratio of image distances to object distances (scale).
3. Decreasing the dimensionality of B increases ease, but reduces scale. In particular, the usual choice of unity as the dimension of B reduces scale so drastically as to render the procedure almost useless for $d > 10$.
4. Since scale is an increasing function of the variance in B space, increasing that variance raises effectiveness. Two methods are:
   (a) Choosing for B space the r rows of greatest variance of the matrix X. This is simple but not optimum.
   (b) Rotating axes by principal components and then choosing the r rows of largest variance. This is more difficult, necessitating computation of the singular vectors of X, but may effect substantial improvement.
5. Simulations based on matrices whose coordinates are normally distributed showed a fairly flat optimum in the neighbourhood of 0·5 for the ratio of dimensionality of B to L space.

These results are presented as an illustration of the principles involved, and a guide to further exploration by others, rather than as a complete analysis. The problem parameters and computing resources determine the details of the procedures, and will of course be different for each problem or installation. It seems likely however, that the methods shown here will minimize the "curse of dimensionality" in high dimensional nearest neighbour searches.

All computations and drawings were done using MLAB (Knott, 1972), an interactive system which runs on the DEC-10 or DEC-20 computer systems. MLAB is available for public distribution. Inquiries may be directed to Gary Knott, National Institutes of Health, Bethesda, MD, 20014, U.S.A.

References

BURKHARD, W. A. & KELLER, R. M. 1973 Some approaches to best-match file searching. *Communs ACM*, **16** (4), 230–236.

ECKART, C. & YOUNG, G. 1936 The approximation of one matrix by another of lower rank. *Psychometrika* **1** (3), 211–218.

FRIEDMAN, J. H., BASKETT, F. & SHUSTEK, L. J. 1975 An algorithm for finding nearest neighbours. *IEEE Trans. Comput.* **C-24** (10), 1000–1006

FRIEDMAN, J. H., BENTLEY, J. L. & FINKEL, R. A. 1975 An algorithm for finding best matches in logarithmic time. *Stanford University Computer Science Report STAN-CS-75-482.*

FUKUNAGA, K. & NARENDRA, P. M. 1975 A branch and bound algorithm for computing k–nearest neighbours. *IEEE Trans. Comput.* **C24,** 750–753.

HOGG, R. V. & CRAIG, A. G. 1978 *Introduction to Mathematical Statistics.* New York: Macmillan.

KNOTT, G. D. & REECE, D. K. 1972 MLAB: A Civilized Curve-Fitting System. *Proceedings of the ONLINE '72 International Conference.* **1,** Brunel University, 497–526.

KNUTH, D. E. 1973 *The Art of Computer Programming.* 2nd Ed., Vol. 11. Mass: Addison-Wesley.

MORRISON, D. F. 1976 *Multivariate Statistical Methods.* 2nd Ed. New York: McGraw-Hill.

SHAPIRO, M. B. 1977 The choice of reference points in best match file searching. *Communs ACM,* **20** (5), 339–343.

STEWART, G. W. 1973 *Introduction to Matrix Computations.* New York: Academic Press.