CMU-CS-78-110

A PROBLEM IN MULTIVARIATE STATISTICS: ALGORITHM, DATA STRUCTURE AND APPLICATIONS

Jon Louis Bentley and Michael Ian Shamos
Departments of Computer Science and Mathematics

April 1978

DEPARTMENT
of
COMPUTER SCIENCE

Carnegie-Mellon University

CMU-CS-78-110

A PROBLEM IN MULTIVARIATE STATISTICS: ALGORITHM, DATA STRUCTURE AND APPLICATIONS

Jon Louis Bentley and Michael Ian Shamos
Departments of Computer Science and Mathematics

April 1978

Abstract

We investigate problems and applications associated with computing the empirical cumulative distribution function of N points in k-dimensional space and employ a multidimensional divide-and-conquer technique that gives rise to a compact data structure for geometric and statistical search problems. We are able to show how to compute a large number of important statistical quantities much faster than was previously possible.

This research was supported by the Office of Naval Research under Contract N00014-76-C-0370.

A PROBLEM IN MULTIVARIATE STATISTICS: ALGORITHM, DATA STRUCTURE AND APPLICATIONS

1. INTRODUCTION

The computational complexity of statistical procedures has just begun to be investigated [Gonzalez 77] [Shamos 77] but proves to be a rich source of new theoretical problems and algorithm design questions. In this paper we conduct an exhaustive analysis of an important computational problem in statistics involving a novel data structure that is a direct outgrowth of the divide-and-conquer algorithm used to solve the problem. We establish lower bounds on computation time, relate the problem to some searching and combinatorial questions, and present a variety of applications.

A multivariate statistical sample of N observations on each of k variables can be regarded conveniently as a set of N points in Euclidean k-space. We say that a point $X = (x_1,...,x_k)$ dominates point Y, written $X \geq Y$, iff $x_i \geq y_i$ for all i. That is, X dominates Y iff it is greater than or equal to Y in all coordinates. The dominance relation is easily seen to define a partial order on vectors. We assume for simplicity that all N points are distinct, but this will not affect the asymptotic running times of our algorithms. The rank r(Z) of a point Z (not necessarily a sample point) is the number of sample points dominated by Z. The normalized rank, r(Z)/N, which is the fraction of points dominated by Z, is better known as the empirical cumulative distribution function (ECDF) and arises in a host of statistical applications. (The use of the word "rank" in this context is suggestive but is a slight misnomer because the points are not linearly ordered. A different definition of rank is given in [Yao 74] but is unrelated to the ECDF). We now distinguish two computational problems:

1. (All points ECDF) Given N points in k-dimensional space, find the number of points dominated by each.

2. (ECDF search) Given N points in k-dimensional space, with preprocessing allowed, find r(Z) for a new arbitrary point Z (without adding Z to the data structure).

The all-points ECDF problem is the crucial step in computing the statistics associated with the Hoeffding, Kolmogorov-Smirnov (K-S), and Cramer-Von Mises tests [Hollander 73] [Hajek 67]. (These applications are discussed below.) It includes the vector maxima problem of [Kung, et al. 75] as a special case, since a maximum (in their parlance) is defined as a vector that is not dominated by any other. The reflection $Z \leftarrow -Z$ transforms a maximum into a vector whose rank is zero, so the ECDF problem can be used to find maxima.

In econometrics, it is common to represent the yield of a combination of investments as a point in multidimensional space, and one is interested in strategies that are not dominated by any others. By this view, one step in the selection of an investment portfolio is an EDCF problem.

The ECDF generalizes the notion of inversions of a permutation. Consider a two-dimensional set (x_i,y_i), such that the x_i are in increasing order. Projecting the points on the y-axis and reading them in increasing y-order induces a permutation $\pi_1,...,\pi_N$ of $\{1,...,N\}$. Point i is dominated by point j iff $i<j$ (that is, $x_i<x_j$) and $\pi_i<\pi_j$. (See Figure 1).

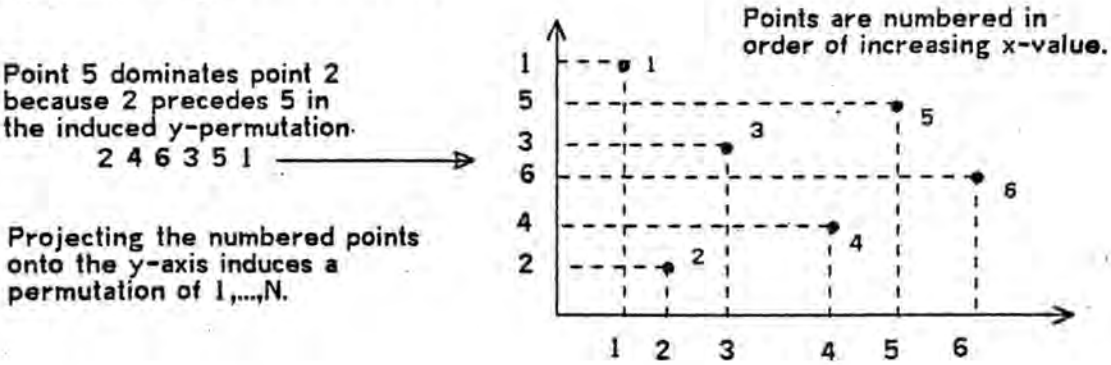


Figure 1: The connection between domination and inversions.

The number of points dominated by (x_i,y_i) is the number of inversions of π in which i participates. This shows that the ECDF is fundamentally a discrete problem and is based only on the ranks of the coordinates, not on their actual values. The generalization to higher dimensions is now clear. Let $P_1,...,P_s$ be a collection of permutations of 1,...,N. Two integers i,j with $1 \leq i,j, \leq N$ will be said to form a s-inversion iff $i < j$ but i follows j in each permutation. For example, the pair (2,5) is a 4-inversion in the following set:

4 5 3 1 2 6 5 6 1 2 4 3 6 3 1 4 5 2 5 3 1 4 6 2

(Because 2 follows 5 in each permutation).

Determining the number of k-inversions in which each integer is involved is equivalent to a (k+1)-dimensional ECDF problem.

Finding the number of elements dominated by each member of a general partial order must take $O(N^2)$ time in the worst case but the dominance relation we have defined is a partial order of a special type -- it is induced by the lexicographic product of k linearly ordered sets. This structure will lead to a fast algorithm. This algebraic view leads to interesting combinatorial questions, such as characterizing those posets that are isomorphic to a set of points in k-space under the dominance ordering.

The ECDF is, in a very powerful sense, an excellent estimator of the underlying population CDF, which we often wish to determine. In order to be able to use this function we must be able to compute its value r(Z) at an arbitrary point Z. A typical application is the multivariate Kolmogorov-Smirnov procedure for testing the hypothesis that two samples have come from the same underlying distribution. The test statistic K is equal to the maximum absolute difference between the ECDFs of the two samples. To compute K it suffices to evaluate the ECDF of sample A at each of the points in sample B, and vice-versa. This entails a search for each point of B to determine how many points in A it dominates. Below we discuss a number of search algorithms that illustrate various time-space tradeoffs and concentrate on one that runs in $O(\log^2 N)$ time (in two dimensions) and requires only O(N log N) space and preprocessing time.

2. ECDF Algorithms

It is easy to solve the all-points ECDF problem in $O(kN^2)$ time by comparing each of the N points against every other point to determine how many it dominates. While in a general partial order of N-vectors this would also be a trivial lower bound on the time needed to compute the number of vectors dominated by each, we have seen that the dominance ordering is of special form. In the present case we will use this structure to improve on the naive algorithm.

We employ a multidimensional divide-and-conquer scheme similar to the one described in [Bentley 76a] and [Bentley 76b], which at each level of recursion reduces the dimension of the problem by one and the number of points by a factor of two.

Theorem 1: The all-points ECDF problem can be solved in $O(N \log^{k-1} N)$ time in the worst case.

Algorithm:

1. Let P be a hyperplane, normal to one of the coordinate axes, that divides the collection of points into two subsets A and B, each containing N/2

points. Such a plane can be found in O(N) time by choosing P to pass through a point that has median first coordinate.

2. Recursively solve the all-points problem on subsets A and B. That is, for each point in A we find the number of points in A that it dominates, and similarly for B. If T(N,k) is the time required to solve the entire problem, then the solution of these two subproblems can be accomplished in time 2T(N/2,k).

3. Without loss of generality, let A be the set of points whose first coordinate does not exceed that of any point of B. Note that the ECDF values obtained for set A in the recursive subproblem solution are the correct final values, since P was constructed so that no point of A can dominate any point of B. It remains only to update the B values to reflect the number of points in A that are dominated.

4. We now observe that each point of B already dominates each point of A in at least one coordinate, namely, the coordinate whose axis is normal to the dividing plane P. This coordinate can thus be removed from further consideration in forming the corrected solution for B. The coordinate can be eliminated without changing any dominance relations by merely projecting all of the points onto P, which is a subspace of one lower dimension. This projection can be accomplished in O(N) time if pointers are used instead of copying lists of coordinates. (Otherwise, O(kN) time would be required.) The projected subproblem can be solved in time T(N,k-1). Note that the "subproblem" is of a somewhat special form, as we are only interested in learning for each point of B the number of points in A that it dominates.

5. Combining the subproblem solutions obtained in steps 2 and 4 can be accomplished in O(N) time, so the recurrence relation for T is

$$T(N,k) = 2T(N/2,k) + T(N,k-1) + O(N) \quad .$$

By sorting the points in advance on each coordinate we may make use of the fact that $T(N,2) = O(N \log N)$ [Shamos 77] to obtain

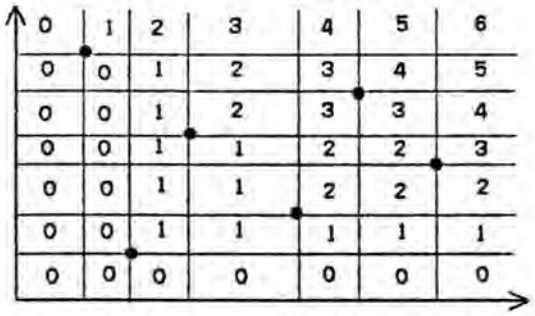
$$T(N,k) = O(N \log^{k-1} N) + O(kN \log N) \quad .$$

In unusual circumstances, the number of dimensions may greatly exceed the number of sample points, N, in which case the above recursion is inefficient because its effort is concentrated on reducing the number of points in the subproblems. If $k > N^4$, the method of [Yao 74], which explicitly constructs the matrix of the partial order, can be used to compute the ECDF in $O(kN^2/\log N)$ time. If only the vector maxima are desired, a slight modification of the above algorithm achieves the $O(N \log^{k-2} N)$ performance attained (for k>2) in [Kung 75]. It has been shown [Bentley 77a] [Bentley 77b] that this modified maxima algorithm runs in O(N) expected time for a very wide class of input distributions.

3. ECDF Searching

Once the all-points problem has been solved, we are in a position to arrange the solution into a data structure that will make it easy to determine the number of points dominated by a new point X. The most simple algorithm requires no preprocessing at all and operates by comparing X to each of the N k-dimensional sample points. This obvious approach requires O(kN) initialization time, O(kN) query time, and O(kN) storage. It is somewhat surprising that the query time can be significantly reduced with no asymptotic increase in the storage used. The method of k-d trees [Bentley 75] achieves $O(kN^{1-1/k})$ search time after O(kN log N) preprocessing, but still needs only O(kN) storage [Lee 77].

It is possible to perform ECDF searching extremely rapidly if sufficient storage and preprocessing time are available. The method is based on the fact that there exist k-dimensional rectangular parallelepipeds within which the number of dominated points remains constant. We may easily see why this is true. Consider some point Z (not in the original set) for which r(Z) = 5 and imagine moving Z parallel to some coordinate axis. (Refer to Figure 2.)



Horizontal and vertical lines are drawn through each sample point. Within each of the resulting rectangles the number of dominated points (and hence the ECDF) is constant.

Figure 2: The ECDF is constant within rectangular regions.

The value of r(Z) cannot change until Z passes the projection of some point of the original set on that axis. This is true of each coordinate. If we were to construct hyperplanes normal to the coordinate axes at each sample point (a total of Nk hyperplanes), these would divide space into $(N+1)^k$ rectangular regions within each of which the function r is constant. Such a structure can readily be queried in O(k log N) time by a binary search along each coordinate, so we have

Theorem 2: ECDF searching can be performed in O(k log N) time, with $O(N^k + kN \log N)$ storage and preprocessing time.

The storage used by this procedure is prohibitive. We propose a data structure and search algorithm that nearly achieves O(log N) search time, but

which uses less than quadratic storage. The data structure is "isomorphic" to the all-points ECDF algorithm in the sense that it is a tree structure having a branch corresponding to each recursive call in the algorithm. The storage required by the search procedure and the time used by the all-points algorithm are described by exactly the same recurrence relation. Furthermore, the data structure for searching can be built conveniently during the solution of the all-points problem at no asymptotic increase in running time.

Let us first treat the two-dimensional case (refer to Figure 3). The dividing line L is the two-dimensional instance of the hyperplane P described in the all-points algorithm, and is used to define the first test. Let A be the set of N/2 points to the left of line L and let B be the set of points to the right. Given a new point Z we want to determine the number of points (in both A and B) that it dominates. In a single comparison against L we can determine whether Z lies in A or in B. If Z lies in A (the diagram on the left in Figure 3) it cannot possibly dominate any point of B, so we may confine our attention to a subproblem of half the size of the original. The recurrence describing this situation is just

$$T(N) = T(N/2) + 1 \ .$$

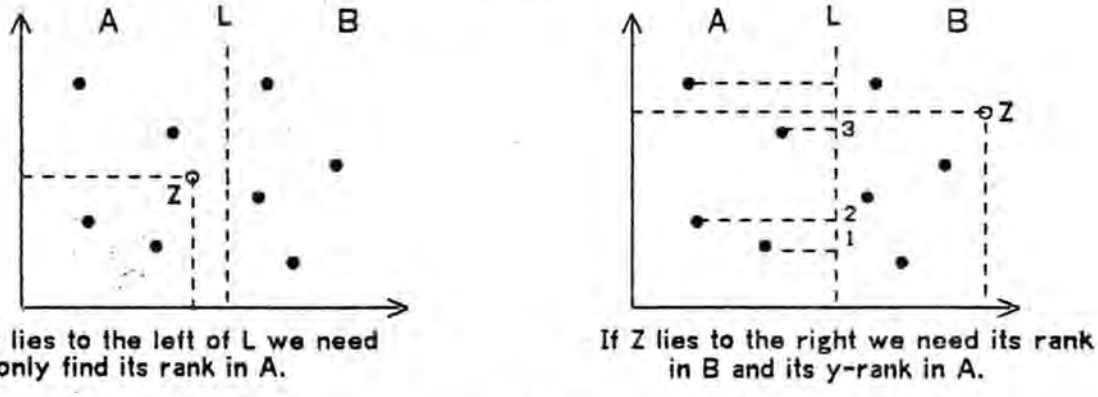


Figure 3: The two cases of ECDF searching in the plane.

If we learn from the first comparison that Z lies in B then the problem is only slightly more complicated (the right diagram in Figure 3). We must find the number of points in B that are dominated by Z, which can be done in time T(N/2). We then add to that the number of points in A dominated by Z. Since, however, the x-coordinate of Z is known to be greater than that of point of A, this number is merely the number of points of A that lie below Z. If we project the points of A onto L and sort them in advance (as part of the preprocessing) we will be able to locate Z in this ordering in O(log N) time by binary search. Thus the recurrence that results when Z is in B is

$$T(N) = T(N/2) + O(\log N) \ .$$

It is immediate that $T(N) = O(\log^2 N)$, even if the second case arises after each comparison.

The generalization to k dimensions is completely straightforward. The line L is replaced by a hyperplane and the sorted list by a (k-1)-dimensional ECDF search structure. The search time is given by the recurrence

$$T(N,k) = T(N/2,k) + T(N,k-1), \qquad T(N,1) = O(\log N),$$

of which the solution is $T(N,k) = O(\log^k N)$. The storage requirement of this algorithm is easy to analyze in view of its recursive structure. In two dimensions we need to store two data structures on N/2 points and one linear list of length N/2. Thus, $S(N,2) = 2S(N/2,2) + O(N) = O(N \log N)$. In k dimensions, we have

$$S(N,k) = 2S(N/2,k) + S(N/2,k-1) = O(N \log^{k-1} N) .$$

The preprocessing time is described by precisely the same relation, giving

Theorem 3: ECDF searching can be accomplished in $O(\log^k N)$ time, using $O(N \log^{k-1} N)$ storage and $O(N \log^{k-1} N)$ time for preprocessing.

4. Applications

We now will present some new applications of the ECDF algorithms and elaborate on some of those presented in the introduction.

4.1 Range Queries

An inconvenient but common type of geometric search problem is the range query [Knuth 73]. Given a set of N points in the plane, with preprocessing allowed, how many lie in the rectangle defined by $a \le x \le b$ and $c \le y \le d$? Within the framework of the ECDF problem, range searching becomes elementary. In two dimensions, (see Figure 4) we may find the number of points lying in rectangle ABCD by computing

$$ECDF(A) - [\, ECDF(B) + ECDF(C) \,] + ECDF(D) .$$

This is a simple application of the combinatorial principle of inclusion-exclusion. We are thus able to respond to a range query by evaluating the distribution function at four points.

In k dimensions we need to evaluate the ECDF at 2^k points, but this still only requires $O(\log^k N)$ time and $O(N \log^{k-1} N)$ storage (or, $O(2^k \log N)$ time and $O(N^k)$ storage.) The range query example provides a link between the empirical distribution function and the empirical density function. The fraction of a set contained in a plane region F is a consistent estimator of the probability content of that region (the probability density integrated over F) [Loftsgaarden 65].

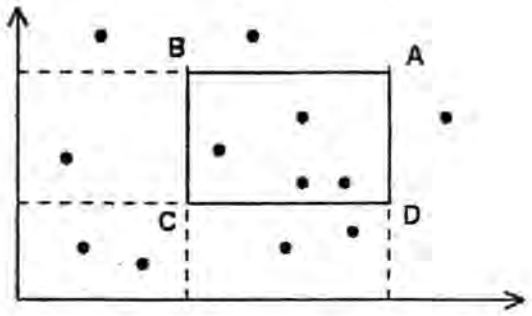


Figure 4: Range searching as an ECDF problem.

4.2 Kolmogorov-Smirnov Statistic

Because of its intimate relation to the ECDF, we point out an anomaly between the K-S one- and two-sample tests. The K-S one-sample statistic is the maximum deviation between the ECDF of a finite point set and a given hypothetical CDF (which we assume can be evaluated at a single point in constant time). A linear-time one-sample K-S algorithm in one dimension has been given by [Gonzalez 77]; it makes use of the fact that any CDF must be a monotonic function. While the situation in higher dimensions is unclear, the ECDF algorithm of this paper can be used to compute the K-S statistic in $O(N \log^{k-1} N)$ time. The K-S two-sample statistic is the maximum deviation between the ECDFs of two given finite point sets.

Theorem 4: The Kolmogorov two-sample statistic must take O(N log N) time to compute, in the worst case.

Proof: The K-S two-sample statistic is zero iff the point sets are identical. Set equality is shown to require O(N log N) comparisons in [Reingold 72].

Acknowledgements

We wish to thank Jerome Friedman of the Stanford Linear Accelerator and Lawrence Rafsky of the Chase Manhattan Bank for concentrated and helpful discussions. The algorithms described here were implemented by Elizabeth Rentmeesters at Carnegie-Mellon University and at SLAC.

Summary

We see that the empirical cumulative distribution function, a ubiquitous

quantity in statistical analysis, can be computed quickly at the given sample points and can be evaluated quickly at other points. The data structure for ECDF searching was arrived at directly from the ECDF algorithm itself. The problems we have considered impinge on many others in different applications areas, all of which may be solved by the techniques developed here.

References

[Bentley 75] Bentley, J.L. Multidimensional Binary Search Trees Used for Associative Searching. CACM 18,9(1975), pp. 509-517.

[Bentley 76a] Bentley, J.L. Divide and Conquer Algorithms for Closest Point Problems in Multidimensional Space. PhD. Thesis. Univ. of North Carolina at Chapel Hill (1976).

[Bentley 76b] Bentley, J.L. and Shamos, M.I. Divide-and-Conquer in Multidimensional Space. Proc. Eighth Annual ACM Symposium on Theory of Computing. Hershey, PA (1976), pp. 220-230.

[Bentley 77a] Bentley, J.L. and Shamos, M.I. Divide and Conquer for Linear Expected Time. Information Processing Letters, to appear.

[Bentley 77b] Bentley, J.L, Kung, H.T., Schkolnick, M., and Thompson, C. On the Average Number of Maxima in a Set of Vectors, this Proceedings.

[Gonzalez 77] Gonzalez, T., Sahni, S., and Franta, W.R. An Efficient Algorithm for the Kolmogorov and Lilliefors Tests. ACM TOMS March, 1977.

[Hajek 67] Hajek, J. and Sidak, Z. Theory of Rank Tests. Academic Press (1967). 297 pp.

[Hollander 73] Hollander, M. and Wolfe, D. Nonparametric Statistical Methods. Wiley(1973). 503pp.

[Knuth 73] Knuth, D.E. The Art of Computer Programming. Volume III: Sorting and Searching. Addison-Wesley (1973).

[Kung 75] Kung, H.T., Luccio, F., and Preparata, F.P. On Finding the Maxima of a Set of Vectors. JACM 22(1975), 469-476.

[Lee 77] Lee, D.T. and Wong, C.K. Worst-Case Analysis for Region and Partial Region Searches in Multidimensional Binary Search Trees and Quad Trees. IBM T.J. Watson Research Center Preprint (1977).

[Loftsgaarden 65] Loftsgaarden, D.O. and Quesenberry, C.P. A Nonparametric Estimate of a Multivariate Density Function. Ann. Math. Stat. 36(1965), pp. 1049-1051.

[Reingold 72] Reingold, E.M. On the Optimality of Some Set Algorithms. JACM 19,4(1972), pp. 649-659.

[Shamos 77] Shamos, M.I. Geometry and Statistics: Problems at the Interface. In Algorithms and Complexity, J.F. Traub, ed. Academic Press (1977).

[Yao 74] Yao, A. C.-C. and Yao, F.F. On Computing the Rank Function of a Set of Vectors. University of Illinois Computer Science Department Report UIUCDCS-R-75-699 (1975).

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* A PROBLEM IN MULTIVARIATE STATISTICS: ALGORITHM, DATA STRUCTURE AND APPLICATIONS		5. TYPE OF REPORT & PERIOD COVERED Interim 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Jon Louis Bentley and Michael Ian Shamos		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0370
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Depts. of Computer Science and Mathematics Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217		12. REPORT DATE April 1978 13. NUMBER OF PAGES 12
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)* We investigate problems and applications associated with computing the empirical cumulative distribution function of N points in k-dimensional space and employ a multidimensional divide-and-conquer technique that gives rise to a compact data structure for geometric and statistical search problems. We are able to show how to compute a large number of important statistical quantities much faster than was previously possible.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*